Announcing ATARI software

from the authors of An Invitation to Programming

exciting games and educational programs for kids, teenagers and adults featuring sound and color graphics.

available on guaranteed-to-load cassettes at fine computer dealers in your area or, write us directly for descriptive materials

Program Design, Inc.
Department CA
11 Idar Court
Greenwich, CT 06830

203-661-8799

SOFTWARE for the ATARI 800* and ATARI 400* from QUALITY SOFTWARE

STARBASE HYPERION™

By Don Ursem

Become absorbed in this intriguing, original space simulation of war in the far future. Use strategy to defend a front line Star Fortress against invasion forces of an alien empire. You create, deploy, and command a fleet of various classes of space ships, while managing limited resources including power generators, shields and probes. Real time responses are sometimes required to take advantage of special tactical opportunities. Use of color, sound, and special graphics add to the enjoyment of this program. At least 24K of RAM is required.

On Cassette — $19.95 On Diskette — $22.95

NAME THAT SONG

By Jerry White

Here is great entertainment for everyone! Two players listen while the Atari starts playing a tune. As soon as a player thinks he knows the name of the song, he presses his assigned key or joystick button. There are two ways to play. The first way requires you to type in the name of the song. Optionally, you can play multiple choice, where the computer asks you to select the title from four possibilities. The standard version requires 24K of RAM (32K on diskette) and has over 150 songs on it. You also get a 16K version that has more than 85 songs. The instructions explain how you can add songs to the program, if you wish. Written in BASIC.

On Cassette — $14.95 On Diskette — $17.95

QS FORTH

By James Albanese

Want to go beyond BASIC? The remarkably efficient FORTH programming language may be just for you. We have taken the popular fig-FORTH model from the FORTH Interest Group and expanded it for use with the Atari Personal Computer. Best of all we have written substantial documentation, packaged in a three ring binder, that includes a tutorial introduction to FORTH and numerous examples. QS FORTH is a disk based system that requires at least 24K of RAM and at least one disk drive. Five modules that may be loaded separately from disk are the fig-FORTH kernel, extensions to standard fig-FORTH, an on-screen editor, an I/O module that accesses Atari's operating system, and a FORTH assembler.

Diskette and Manual — $79.95 Manual Only — $39.95

FOR OUR COMPLETE LINE OF ATARI SOFTWARE
PLEASE WRITE FOR OUR CATALOG

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

ASK FOR QUALITY SOFTWARE products at your favorite computer store. If necessary you may order directly from us. MasterCard and Visa cardholders may place orders by calling us at (213) 344-6599. Or mail your check or bankcard number to the address above. California residents add 6% sales tax. *Shipping Charges:* Within North America orders must include $1.50 for shipping and handling. Outside North America the charge for airmail shipping and handling is $5.00. Pay in U.S. currency.

*Indicates trademarks of Atari.

```
70 X=X+1:Z=Z+1
80 FOR W=1 TO 50:NEXT W
90 NEXT AZ
100 IF Z=91 THEN Z=193:X=0:Y=2:GOTO 30
110 IF Z=219 THEN Z=225:X=0:Y=4:GOTO 30
120 IF Z=219 THEN Z=225:X=0:Y=6:GOTO 30
130 FOR W=1 TO 500:NEXT W
140 POSITION 2,9:PRINT #6; "COLOR
    STATEMENT"
150 PRINT #6; " GRAPHICS!"
160 FOR W=1 TO 1000:NEXT W
```

If you are trying to figure out how we got all those alphabet characters using PLOT and COLOR statements, read on.

As any intermediate programmer can tell you, one cannot plot points in modes 1 and 2. you get absolutely nothing displayed if you try it. Of course the stumbling beginner might think the reason one gets nothing is that one did not enter a COLOR statement. Sandwiching COLOR 1 between the lines and trying again, he discovers that he has plotted an "!" instead of a point. "Pixel-head!" he chides himself. "You can't use PLOT in modes 1 and 2!" He notes this in his reference manual and ranks himself a step closer to intermediate programmer, missing the opportunity to discover more hidden graphics.

The Atari will plot a *character* in modes 1 and 2 at whatever position the programmer commands. The nature and color of that character are determined by a single COLOR statement. Using the COLOR Statement Graphics Chart, you can display any Atari keyboard character (POKE 756 for lower case) by using the associated COLOR statement, then plotting X,Y coordinates to place it at the desired position on the screen.

Once again, SETCOLOR 0 to 3 or POKE 709 to 711 can be used to color each individual character, *including* lower case characters which are normally limited to only two colors. (Note: these SETCOLOR's and POKE's work only when using GR. 1 or 2 + *16*.) Again, redefined characters may be used and this time manipulated arithmetically. Game-writers, rejoice!

While the PRINT #6; approach displays numbers, punctuation and arithmetic signs, the COLOR/PLOT technique allows access to upper and lower case letters as well. Preference for one method over the other will vary from user to user and application to application, as you will see once you have tried them a few times.

Table A. "HIDDEN GRAPHICS" CHART

To Get Character	Press CTRL + Key	Character	(Default)	SE.	POKE
0	P	normal	yellow	0	708
1	Q	"hidden"	green	1	709
2	R	inverse, normal	blue	2	710
3	S	inverse, "hidden"	red	3	711
4	T				
5	U				
56	V				
7	W				
8	X				
9	Y				
:	Z				
!	A				
"	B				
#	C				
$	D				
%	E				
&	F				
'	G				
(	H				
)	I				
*	J				
+	K				
,	L				
–	M				
.	N				
/	O				
[	;				
@	.				

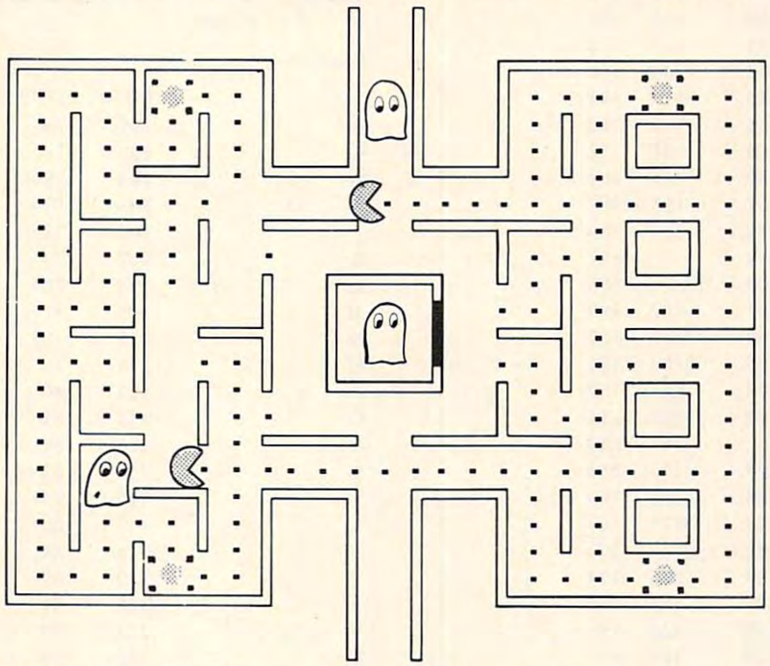
Other Color*		Press Keys
^	green	ESC then DELETE · BACK S
	red	ESC then CTRL + DELETE · BACK S
<	green	ESC then CTRL + !–
	red	ESC then SHIFT + DELETE · BACK S
>	green	ESC then CTRL + —
	red	ESC then CLR · SET · TAB
=	green	ESC then CTRL + |≐
	red	ESC then SHIFT + INSERT ·>
?	green	ESC then CTRL + —*
	red	ESC then SHIFT + CLR · SET · TAB
–	green	ESC then CLR · SET · TAB
	red	ESC then CTRL + INSERT ·>
;	green	ESC then ESC
]	red	ESC then CTRL + ".2

* greens manipulated by SE.1 and POKE 709
reds manipulated by SE.3 and POKE 711

Table B. COLOR STATEMENT GRAPHICS CHART

		Use Color Number			
To Get	SE.	0	1	2	3
Character	POKE	708	709	710	711
		green	yellow	red	blue
!		1	33	129	161
"		2	34	130	162
#		3	35	131	163
!		4	36	132	164
%		5	37	133	165
&		6	38	134	166
'		7	39	135	167
(		8	40	136	168
)		9	41	137	169
*		10	42	138	170
+		11	43	139	171
,		12	44	140	172
–		13	45	141	173
.		14	46	142	174
/		15	47	143	175
0		16	48	144	176

GHOST HUNTER™



THE ULTIMATE GAME OF HIDE AND SEEK.

In the beginning... there was Star Raiders.™

Then... came Missile Command.™

Now... Arcade PLUS introduces Ghost Hunter,™ the first in a new generation of software for the Atari® 400/800™ personal computers.

Your mission in Ghost Hunter™ is simple—rid the mansion on Huckleberry Hill of ghosts... before they get you!

Ghost Hunter™ begins where most computer games end!

- 51 game variations!
- One or two player game... you can play alone or head-to-head against another player!
- Choose from 16 different "floor plans" or let the computer randomly choose for you!
- Special features allow you to "Build-A-Hunter" and "Hide-Instantly"!
- Fast paced, 10 color action with 4-channel music and simulated sound effects!

Ghost Hunter™ is available on cassette or disk for Atari® 400/800™ computers with 16K minimum memory from your local Atari computer dealer. Or send $29.95 (cassette)/$34.95 (disk) + $2.50 postage and handling (CA residents please add 6% sales tax).

arcade PLUS

5276 Hollister Avenue Suite 208 Santa Barbara, CA 93111 (805) 683-2305

1	17	49	145	177
2	18	50	146	178
3	19	51	147	179
4	20	52	148	180
5	21	53	149	181
6	22	54	150	182
7	23	55	151	183
8	24	56	152	184
9	25	57	153	185
:	26	58	154	186
;	27	59	*	187
<	28	60	*	188
=	29	61	157	189
>	30	62	158	190
?	31	63	159	191
@	96	224	*	192
[	91	123	219	251
]	93	*	221	253
\	92	124	220	252
^	94	126	222	254
–	95	127	223	255
A	97	65	225	193
B	98	66	226	194
C	99	67	227	195
D	100	68	228	196
E	101	69	229	197

Table C. COLOR STATEMENT GRAPHICS CHART (Cont.)

Character	Color Number			
	green	yellow	red	blue
F	102	70	230	198
G	103	71	231	199
H	104	72	232	200
I	105	73	233	201
J	106	74	234	202
K	107	75	235	203
L	108	76	236	204
M	109	77	237	205
N	110	78	238	206
O	111	79	239	207
P	112	80	240	208
Q	113	81	241	209
R	114	82	242	210
S	115	83	243	211
T	116	84	244	212
U	117	85	245	213
V	118	86	246	214
W	119	87	247	215
X	120	88	248	216
Y	121	89	249	217
Z	122	90	250	218

*Writing color statements that would logically appear in these positions displays nothing on the screen.

©

Atari Program Listings

COMPUTE! is starting a new, standardized Atari program listing format. All the editing and cursor-control characters are spelled out (e.g., CLEAR for clear screen) and surrounded by brackets.

Other characters, such as CTRL-T, the "ball" character, will be listed as the "normal" character within brackets: {T}. A series of identical control characters will be indicated by a number within the brackets, e.g. 5 DOWN for five cursor downs and 12 R for twelve CTRL-R's. Two control characters, { =} and {–} should be shifted. Any reverse-field text will be enclosed in vertical lines, l like this l. (Press the Atari logo key ( ) for each vertical line.) We expect that this convention will permit easy, unambiguous program typing.

*Indicates trademarks of Atari

String Art

Craig Maiman
Spring Valley, NY

If you always wanted a program which generates beautiful mathematical patterns, now you have it. This program serves very well to impress friends (and yourself!) with the graphics capabilities of the Atari 400/800.

The program actually generates two lissajous figures that are TWISTed relative to each other, it then draws lines between them. The variables that determine the shape of the lissajous are FREQUENCY RATIO and PHASE. Since a lissajous pattern is generated by two signals perpendicular to each other, as on an oscilloscope, we can specify their frequency ratio to obtain many different figures. Changing the phase makes the pattern seem to rotate in 3-D space. An example would be a circle which has a frequency ratio of one-to-one and a phase of 0 degrees. If you now change the phase to 45 degrees it will look like a tilted ellipse. Another variable which can be controlled is DISPLACEMENT: this variable determines the vertical separation of the two lissajous patterns. It can be varied between 0 and 95.

Here are some numbers to generate nice patterns:

FREQUENCY RATIO: 1,1
PHASE: 40
TWIST: 135
DISPLACEMENT: 0
FREQUENCY RATIO: 2,1
PHASE: 0
TWIST: 60
DISPLACEMENT: 0
FREQUENCY RATIO: 1,2
PHASE: 45
TWIST: 120
DISPLACEMENT: 70

Hints

1. Good numbers for the FREQUENCY RATIO are various combinations of 1,2, and 3. Higher numbers tend to make messy pictures.

2. For PHASE, 0 and 90 are good numbers, but try numbers in between also.

3. The TWIST can be varied from –180 to 180. Try them all if you want.

4. DISPLACEMENT can vary between 0 and 95. When you get near 90 all the pictures start looking the same. For starters, I would recommend using 0 then try 20, 30, etc.

Now to reveal the secrets of the program:

Lines	Description
20-40	Screen initialization (Put String... in inverse video)
50-80	Prompts for input (Put these in inverse video)
90	Tests for illegal displacement
130-140	Initializes screen for GRAPHICS 8
160-170	generates first lissajous
180-190	generates second lissajous
200	plots and connects the 2 lissajous
210-220	tests for key touch

```
10 REM STRING ART ! BY CRAIG MAIMAN, JUL
Y, 1981
20 GRAPHICS 0:SETCOLOR 1,3,10:SETCOLOR 2
,3,0
30 SETCOLOR 4,6,0:H=95:POKE 752,1
40 ? "        | STRING ART PROGRAM |"
50 ? :? :? "|   INPUT FREQUENCY RATIO |"
;:INPUT A,B
60 ? :? :? "|   INPUT PHASE |";:INPUT PH

70 ? :? :? "|   INPUT TWIST |";:INPUT TW

80 ? :? :? "|   INPUT DISPLACEMENT |";:I
NPUT DI
90 IF DI>95 THEN 80
100 ? :? :? "|   WHEN PICTURE IS DONE HI
T |"
110 ? "|   ANY KEY TO CONTINUE      |"
120 FOR D=0 TO 680:NEXT D
130 DEG :GRAPHICS 24:SETCOLOR 1,3,10
140 SETCOLOR 2,3,0:SETCOLOR 4,6,0:COLOR
1
150 FOR ANG=0 TO 360 STEP 5
160 X1=H*SIN(A*ANG)+159
170 Y1=(H-DI)*COS(B*ANG+PH)+96-DI
180 X2=H*SIN(A*(ANG+TW))+159
190 Y2=(H-DI)*COS(B*(ANG+PH+TW))+96+DI
200 PLOT X1,Y1:DRAWTO X2,Y2:NEXT ANG
210 IF PEEK(764)<>255 THEN POKE 764,255:
GOTO 20
220 GOTO 210
```

©

Billiard Bounce

Kevin and Priscilla Laws
Carlisle, PA

This program was written initially to provide a graphic demonstration of two lessons on Billiard Ball Mathematics presented by Harold Jacobs in his delightful book entitled *Mathematics: A Human Endeavor* (Freeman, San Francisco, 1970). Once the program was entered, we discovered that we could spend hours watching wonderful patterns unfold before us. Floor designs, Navajo rugs, smooth and nubby fabrics can all be designed with a simple change of two inputs.

In the program, the path of a "billiard ball" is traced on a "table" with a horizontal length of 160 pixels and a vertical width of 96 pixels. The user inputs the horizontal and vertical distance the ball moves during each program step. (These inputs determine the angle at which the ball moves.) A background color and trace color are chosen at random during each run to prevent viewers from becoming tired of the color scheme.

When the program is run the Atari prints:

Angle Horizontal, Vertical?

The user then enters two numbers separated by a comma, such as 1,2 and presses the return key. Users will quickly discover that integers lead to fairly smooth patterns, large numbers to rapidly unfolding patterns, and decimal fractions to jagged lines. Some entries a novice user might like to try:

3.14159, 3.14159	**(leads eventually to a waffle iron)**
3.3, 7.7	**(looks like a woven curtain)**
6.2, 6.3	**(a folksy looking fabric)**
4.5, 6.3	**(an indian rug)**
3.4, 5.5	**(a greek design)**
2, 9	**(bedsprings!)**

If a particularly interesting pattern appears before the design is complete it can be studied by pressing the "CNTRL" key and the "1" key simultaneously. Hitting these two keys again will allow the design process to continue.

The use of the GRAPHICS 7 + 16 mode allows the program to fit easily in an 8K Atari.

```
10 PRINT "Angle Horizontal, Vertical":IN
PUT X1,Y1
20 GRAPHICS 7+16
30 COLR1=INT(RND(0)*15)+1:SETCOLOR 4,COL
R1,10
40 COLR2=INT(RND(0)*15)+1:SETCOLOR 0,COL
R2,5
50 IF COLR1=COLR2 THEN 40
60 X=X-X1:Y=Y-Y1
70 IF ABS(Y)>95 THEN Y=95
80 IF ABS(X)>159 THEN X=159:GOTO 100
90 GOTO 60
100 COLOR 1:PLOT ABS(X),ABS(Y)
110 DRAWTO ABS(X),ABS(Y)
120 X=X-X1:Y=Y-Y1
130 IF ABS(X)>159 THEN X=159
140 IF ABS(Y)>95 THEN Y=95
150 GOTO 110
```

©

Blinking Characters

Frank C. Jones
Silver Spring, MD

The inverse video (⁄ʌ\ ) key on the Atari 400/800 computer allows messages to be displayed in inverse video for special emphasis or eye-catching effects. Another, sometimes even more dramatic, method of catching the viewer's eye is to have the message flash on and off, or blink. There is no simple command in Atari BASIC to produce this effect, but the key to producing it lies in the register, maintained by the operating system, called CHACT (Dec-755, Hex-2F3). If bit 1 in this register is set true, inverse video characters are displayed in inverse video; if it is set to zero they are displayed normally. However if bit 0 is set true, these characters are displayed as blank spaces (inverse video or normal blanks depending on bit 1).

Looking For A Faster Solution

With this information we can write a program that will produce blinking characters on the screen. (Program 1). The trouble with this approach is that our BASIC program is completely preoccupied with timing loops and toggling bit 0 of CHACT. If we try to incorporate this method in a program that does anything else, the timing problem gets very difficult, if not downright impossible. What we really want is a routine that will sit in the background and toggle bit 0 of CHACT on a regular basis without interfering with any BASIC program that might be running at the time. Fortunately the Atari has in it the resources we need to do just this.

The Atari operating system maintains five separate timers which are incremented or decremented during every vertical blank period (the period between successive TV picture frames during which the screen is dark). One of these, called CDTMV2 (Hex21A) is a two byte down counter that can be set to any value between 1 and 65535. Every 60th of a second, during vertical blank, the operating system reduces this number by one and, when it counts to zero, initiates a subroutine jump to the address that it finds in the two byte vector CDTMA2 and returns to the operating system, waiting for the next time the counter counts down to zero.

Program 2 achieves this result by poking a machine language program into memory starting at page 6 (Dec1536,Hex 0600) and transferring control to it via the USR function. Since the operation of this program is obvious, we will spend no time discussing it. Rather, we will turn our attention to the assembly language version of the program that does all of the work, Program 3.

Lines 20-30: Identifies the hex locations of the names times and registers.

Line 50: Starts the program assembly at location hex 0600 (decimal 1536).

Lines 60-130: Initialize jump vector and start timer.

Line 60: Pops one number off the stack. This is required by the USR function; the routine will work without this step, but you will get an Error-9 on return to BASIC.

Lines 70-100: Stores the address of the routine that begins on line 140 in the subroutine jump vector CDTMA2.

Line 110-120: Stores a number in the timer, CDTMV2, to get it going; the number itself is arbitrary.

Line 130: Return from initializing subroutine.

Line 140: Start of subroutine that does the blinking; load the register CHACT into the accumulator.

Line 150: AND the accumulator with the number one, turns off all bits except bit 0.

Line 160: EOR — exclusive OR the accumulator with the number one; reverses the state (toggles) of bit zero.

Line 170: Stores the result back into the register CHACT.

Line 180-190: Resets the timer CDTMV2 at 30 (½ sec.).

Line 200: Return from blink subroutine.

This program, in machine language, is contained in the string of numbers in the DATA statement, line 50, of the BASIC in Program 2. A few of the numbers are readily identified and can be changed by the user to obtain different effects. First of all, the last #30 in the list (the 29th number) is the number that is loaded into the countdown timer each cycle. It determines the delay time between each jump to the toggling routine and hence the blink frequency. Since the counter is decremented every 1/60 of a second, loading 30 in the timer causes the characters to be on for ½ second and blank for another ½ second. This

ALTERNATE REALITY SOFTWARE

Presents for the Atari

THE I CHING

THE ANCIENT CHINESE BOOK OF DIVINATION

- The complete text of the world's oldest book on disk
- 40K program
- 73 disk files (155,000 bytes)
- Occupying 1211 disk sectors
- High Resolution Graphics
- Animation
- Music and Sound Effects
- Instructional text material
- Calculates and Displays Hexagrams
- Displays Judgement, Image, Moving Lines for primary & secondary Hexagrams
- $44.95

order from:

Alternate Reality Software
2111 W. Arapahoe Drive
Littleton, Colorado 80120

Dealer inquiries invited
Atari is a T.M. of Atari, Inc.

VIEW FROM THE TOP
(TRS-80 16K Atari 24K)
A presidential simulation in which you as president must solve the whole array of problems now facing our country. $24.95

WORKERS COMPENSATION RETRO RATING
(TRS-80 8K Atari 8K)
This is an insurance utility to aid agents and brokers in the rating of retrospective rating programs. $19.95

MURDER-MURDER-MURDER
(TRS-80 16K Atari 24K)
In this simulation you as a well known detective must solve the mystery. Depending on how you proceed with your investigation, the solution may change. $21.95

SPACE EMPIRES
(TRS)-80 J6K Atari 16K)
With this fantasy you must colonize as many planets as you can in 20 years. You will face many problems and hazards but the rewards are great — your own empire. $15.95

The above programs are available on cassette only.

Send check, money order or see your local dealer.

C.E.L. PROGRAMS
4040 Winterpark Dr.
Liverpool, NY 13088

TRS-80 is a trademark of the Tandy Corporation
Atari is a trademark of Atari, Inc.

Atari is a reg. ™ of Atari, Inc.

UPGRADE YOUR ATARI®

REPORT CARD

STUDENT *Basic A+*

PARENT *Optimized Systems Software*

SUBJECTS

ADVANCED I/O .... A+
STRUCTURED PROGRAMMING .... A+
PLAYER/MISSILE GRAPHICS .... A+

DEPORTMENT

EASE OF USE .... A+
FLEXIBILITY .... A+
COMPATIBILITY .... A+

THE ONLY LOGICAL UPGRADE FOR ATARI BASIC!

GRADUATE TO BASIC A+ FROM THE AUTHORS OF ATARI BASIC!

BASIC A+

BASIC A+ will rate an A+ from any Atari user! Upward compatible with Atari Basic, it adds statements and features that enhance the Atari 800's real power, flexibility, and ease of use: Superior I/O features for business and other applications. Additional file manipulation commands. Significant help in program development and debug. Structured programming aids. And MORE! A partial list of the enhancements of BASIC A+ includes:

RPUT/RGET (record I/O) **BPUT/BGET** (binary I/O) **ERASE**
PRINT USING SET TAB INPUT"..." DIR PROTECT RENAME
TRACE WHILE...ENDWHILE IF...ELSE...ENDIF
SUPERB PLAYER/MISSILE GRAPHICS

BASIC A+ requires a disk and 32K bytes of RAM. Since no cartridge is used, BASIC A+ will take advantage of all the RAM (48K bytes) in a maximum Atari 800 system (recommended) $80

OS/A+

Completely compatible with Atari's DOS (version 2), but with an advanced, command-driven console processor. Simple. Flexible. Powerful. With an easy-to-use **BATCH** capability. OS/A+ **INCLUDES** all the following utilities (and more):

EASMD (Editor/**AS**se**M**bler/**D**ebug) is our upgraded all-in-one assembly language development package for the 6502 microprocessor. The editor — with such features as FIND and REPLACE — can even edit BASIC A+ programs. The assembler can include multiple source files in a single assembly.

DUPDSK and FORMAT create master or slave disks. Make sector by sector copies of any OS/A+ disk.

COPY a simple, single file copy utility.

All of this power is included in our OS/A+ package $80

For the utmost in capability and flexibility, our combination system — **BASIC A+** and **OS/A+** — is available $150

SOFTWARE IN A CLASS BY ITSELF
AVAILABLE NOW THROUGH LOCAL DEALERS

Optimized Systems Software, Inc.
10379 Lansdale Ave., Cupertino, CA 95014,
(408) 446-3099

number may be changed by the user to be anything between 1 and 255 to make the blink frequency anything between 1/30 of a second to 8½ seconds.

See For Yourself

The two ones in the list, the 22nd and 24th numbers are the ones that are AND'ed and EOR'ed against the contents of CHACT. Changing the first one to 23 leaves inverse video on during the blanking. If both ones are changed to threes, inverse video is on when the characters are on, but the blanks are normal. Changing both ones to twos causes the characters to alternate between normal and inverse video. Finally putting fours in place of the ones produces an effect that you will just have to see for yourself; I still haven't figured out a use for this one.

Once the BASIC program has been entered and run, it may be erased by typing NEW, (RETURN), and entering a new program and the flashing will continue (the flashing will stop during I/O to or from a disk or cassette since stage two vertical blank processes are suspended during I/O operations). System Reset will stop the flashing and bring back inverse video; however, merely typing A = USR(1536) (RETURN) will reinstate the flashing characters without having to reload and run the BASIC program.

This program may be added on to almost any other BASIC program to produce the flashing characters and should add some new twists to your special effects.

Program 1.

```
10 CHACT=755
20 DELAY=200
30 PRINT "!HELLO!"
40 FOR I=1 TO DELAY:NEXT I
50 POKE CHACT,0
60 FOR I=1 TO DELAY:NEXT I
70 POKE CHACT,1
80 GOTO 40
90 END
```

Program 2.

```
10 FOR I=1536 TO 1536+32
20 READ B:POKE I,B:NEXT I
30 A=USR(1536)
40 END
50 DATA 104,169,17,141,40,2,169,6,141,41
,2,169,30,141,26,2,96,173,243,2,41,1,73,
1,141,243,2,169,30,141,26,2,96
```

Program 3.

```
            10 ;CHARACTER BLINK ROUTINE
02F3        20 CHACT      =   $2F3
021A        30 CDTMV2     =   $21A
0228        40 CDTMA2     =   $228
0000        50            *=  $0600
0600 68     60 INITIATE   PLA
0601 A911   70            LDA #BLINK&$00
FF
0603 8D2802 80            STA CDTMA2
0606 A906   90            LDA #BLINK/256

0608 8D2902 0100          STA CDTMA2+1
060B A91E   0110          LDA #30
060D 8D1A02 0120          STA CDTMV2
0610 60     0130          RTS
0611 ADF302 0140 BLINK    LDA CHACT
0614 2901   0150          AND #1
0616 4901   0160          EOR #1
0618 8DF302 0170          STA CHACT
061B A91E   0180          LDA #30
061D 8D1A02 0190          STA CDTMV2
0620 60     0200          RTS
```

©

Make Your Atari Keyboard A Little Friendlier

Ric Mears
San Francisco

If you've ever been typing on the Atari computer and hit the inverse video key instead of the shift key, cleared the screen inadvertently, or locked the keyboard into character graphics mode when you only meant to move the cursor right one space, then this article is for you.

Although I do use assembly language to tame the Atari down a bit, don't worry if you don't know a thing about programming in assembly because I've included a version of my code written in BASIC. And for those of you who are old pros at tickling the insides of your machines, you'll find a listing of the original source code with notes on how it works.

This code does two main things. First, it speeds up the initial delay you encounter while waiting for the auto-repeat to engage. Normally there is a pause of about 3/4 of a second after pressing a key before it starts to repeat. This seems a bit too long to me. It gets to be aggravating, especially when moving the cursor around the screen for editing. You can adjust this speed to suit yourself, or even shut the auto-repeat off altogether if you like.

The Right Pinkie's Burden

The second main thing this code does is to give you an audible signal when you press the inverse video key, the clearscreen key, caps/lower key, or the shift-delete key. The burden of finding all these keys plus a number of others is dumped completely on the right hand's little pinky, making it all too easy to make a mistake. For example, when typing a long mathematical formula, it has been quite frustrating to me when, instead of hitting the final right parenthesis, I'd accidentally press the clear-screen, leaving me to start all over. This happened to me so many times that I decided to do something about it. Now, instead of the screen going blank, I get the sound of five quick keyclicks letting me know my little pinky is off target. If I really want the screen cleared, I just hold the key down a bit longer than usual for the key to be activated. This short wait seems a small price to pay compared to the time that can be lost in correcting mistakes.

Whether you know assembly or not, if you have a disk drive and DOS 2, you can create an "AUTORUN" file so that, when you turn your computer on, the keyboard will behave itself automatically. If you're using DOS 1 or a cassette only system, you'll have to run the BASIC program each time you power up. Of course, those of you without DOS 2 running the assembler cartridge will have to load in the necessary object code after powering up.

It is possible to "tap the computer on the shoulder"...

Cassette Users

A special note to anyone using a cassette, with or without disk drives: If you want to use your cassette in addition to taming your keyboard down, you'll have to switch the two values at the beginning of the BASIC version of the program. I'll try to explain:

There are 256 bytes of memory (referred to as Page 6) available for use any way you see fit located at address 600 hex, or 1536 decimal. The computer's operating system doesn't use them, nor does any typical BASIC program. Still, many BASIC programs, with a need for high-speed assembly language subroutines, *do* use Page 6. In my own programming, I frequently make use of this area of memory and wanted to put the keyboard code someplace else which was safe from unfriendly programs. There are 131 bytes of memory similar to Page 6, located at address 3FD hex, 1021 decimal, unused for anything but conversations between the computer and the cassette. Since I rarely use my cassette, I appropriated this area of memory to hold the new keyboard code. Since this forfeits your ability to use the cassette, you may wish to forfeit use of Page 6 instead by switching the two previously mentioned values at the beginning of the program. The initialization code can pretty much go anywhere, since it's only used once when the program is run, but the keyboard code has to have a permanent place to reside. If neither Page 6 nor the cassette buffer will work for you, you'll have to find some other place for the code. But when you do, you'll only have to change the values at lines 190 and 200 to the new addresses where you want the code placed since it is position independent.

Setting Up

For those of you using a cassette only, or those of you with DOS 1, just run the BASIC program that follows. I suggest first saving a copy of the program before running it though, since, as with any assembly program, if you make a mistake, your computer may very well "lock-up," requiring you to turn it off and back on. Hopefully you will find this program useful and a timesaver. Once it has been run, it is no longer needed in memory and is transparent to normal operations of the computer.

This paragraph is for those of you using DOS 2. Run the BASIC program that follows and see how your keyboard acts afterwards. Then, call up the DOS menu and type "K" for the binary save option. If you already have an AUTORUN file on your disk, respond with "D1: AUTORUN,SYS/A, 3FD, 47F". If you don't already have an AUTORUN file on the disk leave the "/A" off the filename. This will save or append the cassette buffer area containing the new keyboard instructions. Again type "K", and now respond with "D1: AUTORUN-.SYS/A, 600, 60B, 601". Don't forget the "/A". This will append the initialization code along with the proper initialization address. That's it Now, whenever you turn the computer on with this AUTORUN file present, your keyboard will be ready to go automatically.

Changing The Auto-repeat Speed

As I mentioned before, the delay before the auto-repeat engages can be changed, and I've shortened it to suit my typing habits. Decreasing the second number in the data list at line 520, which is now 15, will decrease the length of time before the auto-repeat starts. I doubt you will want to decrease it, but increasing this number will slow the repeat down if you end up typing things like "LLISSTT." You can turn the auto-repeat off altogether by changing this number to a zero. If you should want to disable the new keyboard characteristics (to use the cassette for example) just press the System Reset key. It'll restore the keyboard to normal.

After careful study of the operating system source code, I regret to report that two desirable changes are not feasible. It does not look possible to change the speed of the auto-repeat once it has begun. Nor does it look feasible to shut off the keyclick with software. The keyclick got on my nerves when I first got my computer and, for what it's worth, I dicsovered that the bottom cover of the computer is easily removed (at the expense of voiding your warrenty) and the speaker can be unplugged. However, you lose use of the warning bell or buzzer, so I chose to solder a 200 ohm resister in series with the speaker to soften the volume of the keyclick without losing the bell.

How It Works

At the heart of the Atari computer is a 6502 microprocessor chip. It's always running some kind of program and always in the only language it understands, 6502 machine code. Whenever you use the BASIC cartridge, you're really running a machine language program called "BASIC," which waits for you to tell it to do something, such as to run a program you've typed into memory with its assistance. This program then "interprets" your instructions and sends the 6502 off to execute various machine code subroutines which collectively, and in the proper sequence, accomplish what you told the BASIC "interpreter" you wanted to have done. So, any time your computer is on, the 6502 is always executing machine code instructions.

Interruptions

It is possible to "tap the computer on the shoulder," asking it to stop what it's doing and take care of something else. This is called an *interrupt*. Regardless of whether the computer was busy drawing on the screen, trying to figure out a BASIC program, or whatever, it'll take note of what it was doing so that it can pick up again where it left off after it finishes taking care of the interruption.

There are all kinds of things that can cause an interrupt. One of them is pressing a key on the keyboard. Whenever a key is pressed, the 6502 gets "tapped on the shoulder," and it goes to a subroutine located in the operating system ROM. This subroutine does some checking around and finds out that a key was pressed. Then it looks at memory location 208 hex, which is called VKEYBD, short for Keyboard Vector. Now, *vector* is just a fancy word for *pointer* and normally, this memory location contains the starting address of another subroutine in ROM which takes care of saving information about the particular key that was pressed so that, at some time later on, yet another subroutine can do something with it (like print the letter on the screen). The initialization code at the end of the assembly language listing changes the value contained in the keyboard vector VKEYBD so that the 6502 will follow *our instructions* on what to do when a key is pressed. In fact, we'll tell the 6502 to figure out if the key pressed is one of those that our little pinky hits by mistake so often.

Most of the assembly code is fairly self-explanatory, but a few things are worth discussing. At line 500 of the assembly language listing a check is done first to make sure that the computer wasn't interrupted while it was in the middle of doing something where timing is critical, such as talking to the disk drive. Lines 610-650 take care of keybounce, caused by the vibration of the switch contacts inside the keys.

It is also worth noting that this whole routine is executed only in response to a key being pressed. Holding a key down for the auto-repeat does not generate any further interrupts, and repetition of the key is accomplished at a later time during another type of interrupt, the Vertical Blank period. This interrupt occurs sixty times a second during the time between pictures on your TV screen. The instructions executed during this interrupt are located in an area of the operating system that does not appear to be bypassable. This is also where the keyclicks are generated, making it impossible to shut them off or change the auto-repeat speed itself.

May you never know the agony of delete (or clearscreen) again!

Although it is possible to intercept the vertical blank interrupt, it does not look feasible to replace the operating system ROM routines which take care of these things. If anyone has information to the contrary I'd be very interested in hearing from you. The reason the initial repeat speed can be altered is that the auto-repeat timer, SRTIMR, is initialized during the key-pressed interrupt.

At line 810 bits 6 and 7 of the key code are set to zero since these bits indicate the shift or control key was pressed which is no importance in the case of these two keys. The keycodes themselves have no relation to the ASCII or ATASCII codes. Only the engineers who designed the keyboard could tell you why they created it the way they did, (like why the little pinky got stuck with so many keys to deal with). Anyway, if you want to alter the program to warn you about your own problem keys, you can easily determine their codes with the following one-line BASIC program:

```
10 PRINT PEEK(53769): GOTO 10
```

At lines 1210-1260 you may notice a slightly curious bit of code. VCOUNT is a hardware register which can be read to determine the current scan line being drawn on the TV screen. This value will be zero, sixty times each second, and provides a quick and easy way to obtain a do-nothing loop without tying up any memory locations for counting. Of course, you might wonder why I didn't just use one of the system timers. There are two big reasons why not. One is that I wanted my routine to be as transparent to normal computer operations as possible and tying up a timer could conflict with another program. But a bigger reason is that the system timers are themselves maintained during an interrupt cycle, the old vertical blank period.

Thus, whenever pressing a key causes an interrupt, all other low priority interrupts are temporarily ignored, meaning that the system timers stop. They remain frozen during the time that this program code is being executed and resume counting afterwards. If having the timers lose accuracy is a problem for your application, I suggest adding a little more code which would make an appropriate correction to the timer values. I didn't do this because there is no way the extra code would fit into the cassette buffer. It is being packed to the very last byte already.

Finally, at lines 1400-1430 of the assembly listing, bit 3 of the memory-mapped hardware register named SKSTAT (Serial Port/Keyboard Status) is selected with a logical AND command. This bit indicates whether the last key pressed is still depressed, and is the determining factor as to whether one of those problem keys is actually wanted. May you never know the agony of delete (or clearscreen) again!

Program 1.

```
100 REM . MAKE YOUR KEYBOARD FRIENDLIER
110 REM .     BASIC Program Version
120 REM .
130 REM .       Ric Mears  8/15/81
140 REM
150 REM If you still want to use your
160 REM cassette, switch these two values:
170 REM (but you'll lose use of Page 6)
180 REM
190 CODE=1021
200 INIT=1536
210 REM
220 REM Poke the new code into place:
230 REM
240 FOR I=CODE TO I+130
250 READ OPCODE:POKE I,OPCODE
260 NEXT I
270 REM
280 REM Poke the init code into place:
290 REM
300 FOR I=INIT TO I+11
310 READ OPCODE:POKE I,OPCODE
320 NEXT I
330 REM
340 REM Fix the absolute address in case
350 REM the user has selected different
360 REM locations for the code.
370 REM
380 HIGH=INT(CODE/256)
390 LOW=CODE-HIGH*256
400 POKE INIT+2,LOW
410 POKE INIT+7,HIGH
```

```
420 REM
430 REM Now reset the the keyboard vector:
440 REM
450 INIT=USR(INIT)
460 PRINT :PRINT "ALL DONE"
470 REM
480 REM Here's the assembled program
490 REM as data:
500 REM
510 DATA 165, 66, 208, 81, 138, 72, 152, 72
520 DATA 160, 15, 173, 9, 210, 205, 242, 2
530 DATA 208, 5, 173, 241, 2, 208, 55, 173
540 DATA 9, 210, 201, 159, 208, 10, 173, 255
550 DATA 2, 73, 255, 141, 255, 2, 176, 38
560 DATA 201, 116, 240, 43, 201, 118, 240, 39
570 DATA 201, 182, 240, 35, 41, 63, 201, 39
580 DATA 240, 29, 201, 60, 240, 25, 173, 9
590 DATA 210, 141, 252, 2, 141, 242, 2, 169
600 DATA 3, 141, 241, 2, 133, 77, 140, 43
610 DATA 2, 104, 168, 104, 170, 104, 64, 160
620 DATA 5, 32, 216, 252, 162, 8, 142, 31
630 DATA 208, 162, 75, 173, 11, 212, 208, 251
640 DATA 202, 208, 248, 136, 208, 235, 160, 255
650 DATA 173, 11, 212, 208, 251, 136, 208, 248
660 DATA 160, 40, 173, 15, 210, 41, 4, 240
670 DATA 189, 208, 203
680 REM
690 REM Here's the initialization
700 REM code as data:
710 REM
720 DATA 104, 169, 253, 141, 8, 2, 169, 3
730 DATA 141, 9, 2, 96
740 END
750 REM End of BASIC program listing
```

Program 2.

```
0100 ; MAKE YOUR KEYBOARD FRIENDLIER
0110 ;      Assembler Source Code
0120 ;
0130 ;          Ric Mears  8/15/81
0140 ;
0150 ; EQUATES
0160 ;
0170 VKEYBD    = $208     ;Keyboard vector
0180 KBCODE    = $D209    ;Pokey register
0190 CONSOL    = $D01F    ;Speaker
0200 VCOUNT    = $D40B    ;Scan-line counter
0210 SKSTAT    = $D20F    ;Keyboard status
0220 CLICK     = $FCD8    ;OS keyclick routine
0230 CH        = $2FC     ;Current key pressed
0240 CH1       = $2F2     ;Last key pressed
0250 KEYDEL    = $2F1     ;Keybounce counter
0260 SSFLAG    = $2FF     ;Cntrl-1 start-stop
0270 ATTRACT   = $4D      ;Attract mode flag
0280 CRITIC    = $42      ;Critical code flag
0290 SRTIMR    = $22B     ;Auto-Repeat timer
0300 CNTRL1    = $9F
0310 CLEAR1    = 118      ;Internal
0320 CLEAR2    = 182      ; codes
0330 INVERSE   = 39       ;  for these
0340 CAPSLWR   = 60       ;   problem
0350 DELETE    = 116      ;    keys
0360 ;
0370 ;
0380 ; ENTRY POINT
0390 ;
0400 ; Whenever a key is pressed, an
0410 ; interrupt sends the 6502 here:
0420 ;
0430  *=$3FD             ;Or any safe place
0440 ;
```


```
0450 ; If the computer was executing
0460 ; critical code, then forget
0470 ; about the keyboard interrupt:
0480 ;
0490 NEWPROCEDURE
0500  LDA CRITIC
0510  BNE EXIT
0520 ;
0530  TXA               ;OS has already
0540  PHA               ; saved register A,
0550  TYA               ;  must also
0560  PHA               ;   save X & Y
0570 ;
0580 ;
0590  LDY #$F           ;Set new Auto-
0600 ;                  ; repeat speed
0610  LDA KBCODE
0620  CMP CH1           ;Same as last key?
0630  BNE NEWKEY
0640  LDA KEYDEL        ;If KEYDEL > 0
0650  BNE OUT           ; ignore as bounce
0660 NEWKEY
0670  LDA KBCODE
0680  CMP #CNTRL1       ;Take care of
0690  BNE NOTCTRL1      ; Control-1
0700  LDA SSFLAG        ;  stall flag
0710  EOR #$FF
0720  STA SSFLAG
0730  BCS OUT
0740 NOTCTRL1
0750  CMP #DELETE       ;Check for
0760  BEQ ALERT         ; aggravating
0770  CMP #CLEAR1       ;    keys
0780  BEQ ALERT
0790  CMP #CLEAR2
0800  BEQ ALERT
0810  AND #$3F          ;Strip off shift
0820  CMP #INVERSE      ; & cntl bits
0830  BEQ ALERT         ;  since these
0840  CMP #CAPSLWR      ;   keys are
0850  BEQ ALERT         ;    unique
0860 ;
0870 ;
0880 ; This point reached if a regular key
0890 ; or the typist wants the special key
0900 ;
0910 NOCHANGE
0920  LDA KBCODE
0930  STA CH            ;Pass the
0940  STA CH1           ; letter on
0950  LDA #3
0960  STA KEYDEL        ;Set debounce
0970  STA ATTRACT       ;Reset Attract flag
0980 OUT
0990  STY SRTIMR        ;Set auto-repeat
1000 ;                  ;    speed
1010  PLA
1020  TAY               ;Restore
1030  PLA               ; registers
1040  TAX
1050 EXIT
1060  PLA               ;And return from
1070  RTI               ; the interrupt.
1080 ;
1090 ; ALERT ROUTINE
1100 ;
1110 ; A problem key has been pressed
1120 ; so do the special signal.
1130 ;
1140 ALERT
1150  LDY #5            ;For 5 clicks
1160 LOOP5
1170  JSR CLICK
1180  LDX #8            ;Turn speaker
1190  STX CONSOL        ; back off
1200 ;
1210  LDX #75           ;For stall length
1220 WAIT
1230  LDA VCOUNT        ;Scan line count
1240  BNE WAIT          ; (extra delay)
1250  DEX
1260  BNE WAIT
1270  DEY
1280  BNE LOOP5         ;5 clicks yet?
1290 ;
1300  LDY #255
1310 WAITAGAIN
1320  LDA VCOUNT        ;Stall a moment
1330  BNE WAITAGAIN     ; after sounding
1340  DEY               ;  the alert
1350  BNE WAITAGAIN
1360 ;
1370  LDY #40           ;Slower initial
1380 ;                  ; repeat for these
1390 ;
1400  LDA SKSTAT        ;Are they still
1410  AND #4            ; holding the
1420  BEQ NOCHANGE      ;  key down?
1430  BNE OUT
1440 ;
1450 ;
1460      *= $600
1470 ;
1480 ; INIT POINT
1490 ;
1500 ; Resets keyboard vector so that
1510 ; whenever a key-pressed interrupt
1520 ; occurs, the 6502 will go the new
1530 ; routine.
1540 ;
1550 INIT
1560  PLA
1570  LDA #NEWPROCEDURE & $00FF
1580  STA VKEYBD
1590  LDA #NEWPROCEDURE / 256
1600  STA VKEYBD+1
1610  RTS
1620 ;    End of Assembly Program Listing
```

©

Adding High Speed Vertical Positioning To P/M Graphics

David H. Markley
Reynoldsburg, OH

By now many of you have been playing the "Aliens from Outer Space" program I described in **COMPUTE!** #11, and have been able to experiment with an actual game program incorporating the advanced player/missile graphics of the Atari computer. As you may have observed, player images can be moved horizontally across the playfield quite easily just by placing the player's horizontal coordinate (0-120) into its associated horizontal position register. Vertical positioning with the P/M graphics however is somewhat more difficult. Since the player's vertical position on the playfield inversely corresponds to its position within the image buffer, it is necessary to relocate each byte of the image up or down within the buffer to produce vertical movement. For example, if we move the player's image to higher address locations within the image buffer, the player will appear to move downward on the playfield.

A BASIC routine can be written using PEEKs and POKEs to move the player within the image buffer, but for most applications this method is too slow. An alternative, however, is to use a small, general purpose vertical positioning routine written in 6502 assembly code which can be called by BASIC's USR instruction.

The vertical positioning routine shown in Program 1 is relatively simple, but provides the user with a flexible and easy method of handling P/M graphics within a BASIC program. This program not only provides a valuable tool to use with player missile graphics, but for those of you who have not used assembly language routines with BASIC, it will provide some insight into this area as well. The routine is called by a BASIC statement similar to below:

```
DUMMY = USR(VP,IMAGE,LAST LOCATION,
        NEW LOCATION)
```

The variable to the left of the equal sign called "DUMMY" is used by some machine language subroutines as a target for a value returned by the program. The vertical positioning routine however, does not return a usable value, but the DUMMY variable is still required to satisfy Atari's USR format requirements. Within the parentheses of the command are four arguments. The first argument, VP, is the transfer address to the VP routine which has been placed into a free area of memory. Loading of the VP routine into memory will be described later with a program application example. Following the transfer address argument (which, by the way, is also required for any USR routine called by BASIC) are three arguments which are passed to the VP routine.

Before either step can be executed, the routine must first look at the image's data structure and get the image size parameter.

These arguments are the address of the image's data structure, the address of the image's current position in the P/M image buffers, and the address of its new position. Each image requires a small data structure which provides the VP routine with a pattern of the actual image which it will vertically reposition. An example of a typical image data structure is shown in Figure 1. The first byte of data provides the VP routine with the image's size in bytes. The second and following bytes are used to form a bit map pattern of the image as it would appear in the P/M image buffers.

The next two arguments contained in the USR command tell the VP routine the image's current and new positions. These arguments are actual memory addresses into the image buffers, therefore care must be taken to assure that they do not access another area of memory by mistake.

Routine Operation

The program operation begins with an initialization step in which the three arguments passed to it by the USR command are removed from the processor's stack and placed into an area in page zero where they can be more easily used. You may have noticed that a total of seven bytes are popped off the stack during this operation. This is because the USR command always places a one byte argument count onto the stack followed by the arguments themselves. The arguments are always two bytes in length.

Once the initialization task is complete, the routine is ready to begin its intended task of moving the player image. Basically the operation is performed in two steps. The image data is first removed from its current location and then copied to its new location. Before either step can be executed, the routine must first look at the image's data structure and get the image size parameter. This value tells the routine how large an image it must handle and thus determines the number of bytes it must remove and restore. To remove an image from its current location, the routine simply goes to the current location address and writes zeroes into an X number of memory locations indicated by the size parameter. Replacement of the image is done by copying from the image's data structure an X number of bytes, also determined by the size parameter, to the image buffer starting at the address specified by the new position argument.

In some cases it may not be desirable to have the VP routine perform both the delete and restore functions. One example would be if the player image is to be removed from the viewing field and not restored at a new location. This can be handled by using the following routine call:

```
DUMMY = USR(VP,IMAGE,CURRENT
        LOCATION,0)
```

The zero in the new location field tells the VP routine not to attempt to restore the image. Likewise, the delete function can be disabled by placing a zero in the current location field.

Let's Have Some Fun

Now that we have looked at the Player/Missile Vertical Positioner routine, let's put it to work. The following game will show you how to load the player images and VP routine into memory and how to use the routine in other ways besides vertical positioning.

This game which I call "Island Jumper" involves the cooperation of two characters named Crash Coleman and Deadeye Dan. Crash is the pilot of a reliable (but not so stable) airplane, the "Leaping Lucy." Crach has only had one flying lesson, but has courageously volunteered to make this flight so that you can see the VP routine in action. Although he has successfully managed to get the Leaping Lucy off the ground, he seems to be having some trouble keeping her in level flight. Our other daredevil of the sky, Deadeye Dan, will attempt, with your help, to jump out of Crash's airplane and land on Target Island. Since the ground seems to be a bit unstable from Dan's point of view, he is having difficulty figuring out when to jump and asks that you help him by pulling back on your joystick controller when you think he's on target.

Dan will make a total of five jumps each time you play the game. He will try to land on top of a sand dune located on the left side of the island. If he makes the jump on Crash's first pass over the island and lands on the dune with both feet, you get 30 points. If you don't give Dan the signal to jump during the first pass, Crash will continue to fly over the island until a jump is made. Each additional pass will deduct eight points from Dan's maximum obtainable score.

Dan can also land in the area between the sand dune and the palm tree, but you will only receive a maximum of 15 points for the jump. At the completion of the game, the computer will give you both a final score the last game played and the highest score for all games played since the last RUN command was entered. To play another game, press the button on the joystick controller.

The data for the VP routine and the player data structures is read from data statements and POKEd into memory by lines 110 thru 310 of the program. It is loaded into memory page 6 (starting at address 1536) which is a 256 byte area in memory that Atari has reserved for user binary data and machine language routines. Once the data structures and VP routine are loaded into memory, they are referenced in the BASIC program by variable names whose values have been set to the starting address of the data structure or VP routine they represent.

Program 1.

```
10 REM VERTICAL POSITIONER EXAMPLE
20 REM ISLAND JUMPER
30 REM COPYRIGHT (c) 1981
40 REM BY DAVID H. MARKLEY
50 GRAPHICS 2:POKE 752,1
60 SETCOLOR 4,9,4
70 ? #6:? #6:? #6:? #6;"      ISLAND"
80 ? #6:? #6;"      JUMPER"
90 ? ,"      BY"
100 ? :? ,"DAVID MARKLEY"
110 VP=1536
120 FOR G=0 TO 93
125 READ D
130 POKE VP+G,D
135 NEXT G
140 REM VERTICAL POSITIONER CODE
150 DATA 104,162,5,104,149,220,202,16,25
0,198,220,198,222,160,0,177,224,170
160 DATA 168,165,223,240,9,169,0,145,222
,136,208,249,138,168,165,221,240,7,177,2
24,145,220,136,208,249,96
170 REM AIRPLANE DATA
180 APIMG=VP+44
190 DATA 6,142,132,255,255,4,14
```

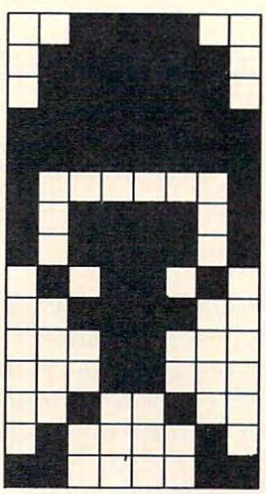
```
200 REM JUMPER DATA
210 JPIMG=APIMG+7
220 DATA 9,189,189,90,60,24,24,36,66,129

230 REM JUMPER AND CHUTE DATA
240 JSIMG=JPIMG+10
250 DATA 15,60,126,126,255,255,129,189,1
89,90,60,24,24,36,66,195
260 REM WAVING JUMPER
270 JWIMG=JSIMG+16
280 DATA 15,0,0,0,0,0,128,188,188,88,60,
26,25,37,66,195
290 REM DATA USED TO CLEAR MEMORY
300 CLEAR=JWIMG+16
310 DATA 255
320 FOR D=1 TO 300:NEXT D
330 GRAPHICS 5
340 SETCOLOR 2,9,2
350 SETCOLOR 4,8,6
360 I=PEEK(106)
365 X=I*256-1172
370 POKE X,112
371 POKE X+1,71
372 POKE X+2,96
373 POKE X+3,I-1
374 POKE X+4,112
375 POKE X+5,74
376 POKE X+6,160
377 POKE X+7,I-5
380 I=I-8
390 POKE 54279,I
400 J=I*256+513
410 POKE 559,46
420 POKE 53256,1
430 POKE 53277,3
440 POKE 704,56
450 POKE 705,12
460 D=USR(UP,CLEAR,J,0)
465 SLOPE=2
470 TOP=J+17
480 BOT=J+55
490 SETCOLOR 0,12,8
500 SETCOLOR 1,1,2
510 COLOR 2
520 PLOT 37,34:DRAWTO 42,34
530 PLOT 36,35:DRAWTO 49,35
540 PLOT 47,29:DRAWTO 47,34
550 COLOR 1
560 PLOT 43,30:DRAWTO 47,27
570 PLOT 51,30:DRAWTO 47,27
580 PLOT 47,27:DRAWTO 49,30
590 PLOT 47,27:DRAWTO 45,30
600 PLOT 46,27
610 HSCORE=0
620 LAPOS=0
630 APOS=J+70
640 I=-1
650 JUMP=5
660 SCORE=0
670 PNTS=30
680 JMP=0
690 SOUND 0,31,4,4
700 POKE 623,4
710 JSTOP=J+219
720 FOR G=20 TO 245 STEP 3
730 POKE 53248,G
740 D=USR(UP,APIMG,LAPOS,APOS)
750 IF JMP=0 AND G<180 AND STICK(0)<>15
THEN JMP=APOS+132:POKE 53249,G+4:IMG=JPI
MG:D=USR(UP,IMG,0,JMP)
760 LJMP=JMP
770 IF JMP=0 THEN 880
780 JMP=JMP+3
790 IF JMP<J+200 THEN HJMP=G+4:POKE 5324
9,HJMP:SOUND 1,G,10,8:GOTO 860
800 IMG=JSIMG
804 JMP=JMP-2
808 SOUND 1,0,0,0
810 IF HJMP>=122 AND HJMP<=126 THEN JSTO
P=J+208:GOTO 860
820 IF HJMP<120 OR HJMP>134 THEN 860
830 JSTOP=J+210
840 POKE 623,1
850 IF PNTS>15 THEN PNTS=15
860 IF JMP>JSTOP THEN 940
870 D=USR(UP,IMG,LJMP,JMP)
880 LAPOS=APOS
890 APOS=APOS+I
900 D=USR(UP,APIMG,LAPOS,APOS)
910 IF APOS>BOT THEN I=-SLOPE
920 IF APOS<TOP THEN I=SLOPE
930 NEXT G
940 IF JMP<J AND PNTS>9 THEN PNTS=PNTS-8
:GOTO 1220
950 IF JMP<J THEN 1220
970 IF HJMP<120 OR HJMP>134 THEN TONE=8:
GOTO 1010
980 SCORE=SCORE+PNTS
985 TONE=12
990 D=USR(UP,JWIMG,0,JMP-1)
1000 ? "SCORE ";SCORE:? :?
1010 FOR D=15 TO 0 STEP -1
1020 SOUND 1,12,TONE,D
1030 FOR I=1 TO 10:NEXT I
1040 NEXT D
1050 SOUND 0,0,0,0
1055 SOUND 1,0,0,0
1060 JUMP=JUMP-1
1070 IF JUMP<>0 THEN 1170
1080 IF SCORE>HSCORE THEN HSCORE=SCORE
1090 FOR I=1 TO 120
1100 IF I=1 THEN ? "HIGH SCORE ";HSCORE:
```

```
? :?
1110 IF I=60 THEN ? "FINAL SCORE ";SCORE
:? :?
1120 IF STRIG(0)=1 THEN 1150
1130 D=USR(UP,CLEAR,J,0)
1135 PRINT
1140 GOTO 630
1150 NEXT I
1160 GOTO 1090
1170 ? "JUMP ";6-JUMP:? :?
1180 FOR D=0 TO 250:NEXT D:?
1190 D=USR(UP,CLEAR,J,0)
1195 I=SLOPE
1200 IF RND(0)>0.5 THEN I=-SLOPE
1210 GOTO 670
1220 POKE 77,0
1225 GOTO 690
1230 END
```

Figure 1: Image Data Structure for the Player/Missile Vertical Positioner Routine.

Image Pattern



Byte Number	Byte Value
1	60
2	126
3	126
4	255
5	255
6	129
7	189
8	189
9	90
10	60
11	24
12	24
13	36
14	66
15	195

DATA 15,60,126,126,255,255,129,189,189,90,60,24,24,36,66;195

Program 2. Assembly language representation of the P/M Vertical Positioner Routine

```
10  ; P/M VERTICAL POSITIONER
20  NEW=220
30  CURRENT=222
40  IMAGE=224
50  START   PLA          ; REMOVE ARGUMENT
                          BYTE COUNT
60          LDX #5       ; REMOVE 6 BYTES
70  LP1     PLA          ; AND PLACE IN PAGE
                          ZERO
80          STA NEW,X
90          DEX
100         BPL LP1
110         DEC NEW
120         DEC LAST
130         LDY #0
140         LDA (IMAGE),Y ; GET IMAGE BYTE
                          COUNT
150         TAX
160         TAY
170         LDA LAST+1   ; IF ZERO DON'T DELETE
180         BEQ SKIPD
190 LP2     LDA #0       ; DELETE IMAGE
200         STA (LAST),Y
210 DEY
220         BNE LP2
230 SKIPD   TXA
240         TAY
250         LDA NEW+1    ;IF ZERO DON'T
                          RESTORE
260         BEQ SKIPR
270 LP3     LDA (IMAGE),Y ; COPY IMAGE DATA TO
                          NEW
280         STA (NEW),Y  ; ADDRESS
290         DEY
300         BNE LP3
310 SKIPR   RTS
```

©

A Poor Programmer's Word Processor

Frank Roberts
Ft. Wayne, IN

A few days ago I walked into a local computer store for new software for my Atari 800. I was informed — just as expected — that there wasn't much available yet, "but a lot was expected real soon." Well, being impatient, low on cash and desperately wanting something besides Star Raiders to play with I decided to tackle one of those "soon-to-be-available" projects myself. After all, didn't I really buy my Atari to have fun with? The following program is a primitive (but workable) method of justifying left and right margins for letters and texts — sort of a poor man's word processor.

Program 1.

```
1 REM ***** PSEUDO WORD PROCESSOR (D:WRI
TE.LET)
5 GRAPHICS 0
10 DIM A$(100),B$(100)
20 REM
30 ? :? "HOW MUCH MARGIN ";:INPUT MARGIN

40 ? :? "ENTER TEXT (IN DOUBLE LINES) "
50 ? :? "WHEN FINISHED, ENTER '999'"
60 ? :? "TO BEGIN, HIT RETURN ";:INPUT A
$
70 WIDTH=INT(80-(2*MARGIN)):POKE 83,INT(
WIDTH)/2+2
80 POKE 201,MARGIN-1
90 INPUT A$:IF A$="999" THEN 140
95 IF LEN(A$)=0 THEN 120
100 IF LEN(A$)=WIDTH+1 THEN B$=A$:GOTO 1
20
110 GOSUB 1000
120 LPRINT " ",B$:B$="":PRINT
130 GOTO 90
140 END
1000 REM *****SUBROUTINE: JUSTIFY RIGHT
MARGIN
1010 C=0
1020 FOR J=1 TO LEN(A$)
1030 IF LEN(B$)=WIDTH+1 THEN 1070
1040 C=C+1:B$(C)=A$(J)
1050 IF A$(J,J)=" " THEN C=C+1:B$(C,C)="
"
1060 NEXT J
1070 RETURN
```

Program 2.

```
20 OPEN #1,8,0,"D:FILE.LET"
50 ? :? "DO YOU WANT PRINTOUNT NOW ";:IN
PUT A$
80 PRINT #1;MARGIN
120 PRINT #1;B$:B$="":PRINT
160 IF A$="NO" THEN END
170 RUN "D:TYPE.LET"
```

Program 3.

```
1 REM *** FETCH TEXT AND PRINT (D:TYPE.L
ET)
10 DIM A$(100)
20 ? :? "WHEN READY, HIT RETURN ";:INPUT
 A$
30 OPEN #1,4,0,"D:FILE.LET"
40 TRAP 80
50 INPUT #1,MARGIN:POKE 201,MARGIN
60 INPUT #1,A$:LPRINT " ",A$
70 GOTO 60
80 CLOSE #1
90 ? :? "DESTROY FILE NOW (Y-N) ";:INPUT
 A$
100 IF A$(1,1)="N" THEN END
110 XIO 33,#1,0,0,"D:FILE.LET"
```

How It Works

Line 70 calculates the parameters of the text string according to the MARGIN selected and POKEs the right margin of the screen monitor to one-half the width. The latter is necessary to accomodate one full line of typing for each A$ to be printed (the screen is only 40 columns wide). The user enters two lines for each single line of text and types as close to the right margin as possible without hyphenating the last word in the middle of a syllable. The subroutine beginning at LINE 1000 counts the characters within each line of text and adjusts the length of the text by inserting the proper number of spaces into the string.

The program is designed to print directly to an AXIOM II printer, but with the following modifications (Program 2) a temporary file (D:FILE.LET) can be made which will allow storage of the text for future printout or multiple copy.

There is also a file retrieval program (Program 3; LINE 110 automatically deletes the unwanted file without going to DOS). ©

COMPUTE! Magazine

When we first started **COMPUTE!** magazine in the Fall of 1979, we made a commitment to build a strong, ongoing users resource. Since that beginning, we've provided thousands of personal computer owners with breakthroughs, innovations, practical software, and in-depth reviews of new and significant products. In every issue of **COMPUTE!**, you'll find dozens of pages showing you exactly how to make better use of your computer; articles which tutor the beginner and challenge the advanced.

We're your specialists, whether your interests lie in home applications, game development, educational insights, or better business planning. Here are just a few facts to tell you more about **COMPUTE!**:

' In every issue, you'll find clearly, cleanly printed software you can type right into your computer and use...

Here's a comparison of average machine specific articles in the last three issues of **COMPUTE!**, matched against the average number of articles in the "Big Four" ...the personal computer magazines with circulations of 100,000 or more:

Computer	COMPUTE!		"Big Four"	
Type	Articles	Programs	Articles	Programs
Atari	10.67	13.3	1.08	1.42
PET/CBM	11.33	11.0	1.5	2.33
Apple	4.0	6.0	3.58	3.17
OSI	2.0	4.0	.5	.5

(Numbers are average number of articles or programs in **COMPUTE!** issues 16, 17, and 18 compared with the average number of articles or programs in each of the same three issues of the four largest personal computing magazines.)

Subscribe to **COMPUTE!** today! We're the Resource...

Announcing Home and Educational COMPUTING!, the newest publication from the publishers of COMPUTE!

We've expanded the scope of our new magazine and, beginning in January, you'll find resourceful coverage and helpful information for the **VIC-20, the Atari 400, the TI 99/4A, the Radio Shack Color Computer, the Sinclair ZX-81,** and more. We'll be covering personal applications, education, new thresholds, the pocket computers, and a great deal more.

COMPUTE! Books

Our **First Book of Atari** is now shipping (if you've already ordered one it's on the way). If you haven't, check with your dealer, or order directly from us by using the convenient toll-free numbers below.

In the continental US, Call TOLL FREE 800-345-8112. In PA call 800-662-2444.

SUBSCRIPTION RATES

	COMPUTE! (monthly) Twelve issues	Home And Educational COMPUTING! (bimonthly) Six issues
US:		
One Year	$20.00	$10.00
Two Years	$36.00	
Three years	$54.00	
Canada	$25.00	$12.00
Europe, Air Delivery	$38.00	$18.00
Elsewhere, Surface	$25.00	$12.00
Air:		
Middle East, Central America and North Africa	$48.00	$24.00
South America, South Africa, Far East and Australia	$88.00	$24.00

All subscriptions from outside the US must be prepaid in US funds drawn on a US bank or international money order. Visa/MasterCard also accepted.

Please enter my subscription to: ☐ **COMPUTE!**
☐ 1 year ☐ 2 years ☐ 3 years
☐ **Home and Educational COMPUTING!**

Please send me ________ copy(s) of **COMPUTE!'s First Book of ATARI** at $12.95 each + $2.00 shipping/handling.

Name ______________________________

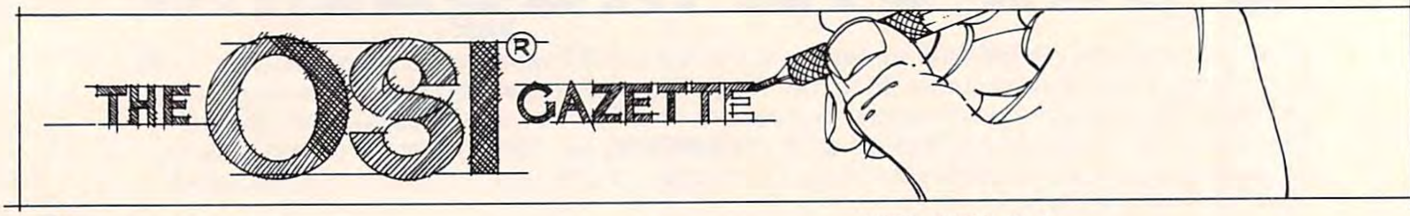
Address ______________________________

____

For fastest service in the US call the toll free number above, or send this coupon to: **COMPUTE!**, 515 Abbott Drive, Broomall, PA 19008.

Bill me orders (US only) are charged a $1.00 billing fee.

COMPUTE!
515 Abbott Drive
Broomall, PA 19008



Super Cursor V1.3

Frank Cohen
Pacific Palisades, CA

My biggest complaint about Ohio Scientific's Superboard II has been about the awful video output. It's almost ironic noting all the good things the Superboard has going for it: a nice keyboard; a powerful Microsoft BASIC in ROM; a dependable cassette interface; 8K of RAM; and many other functions. The irony comes into play when you turn on the Superboard and take a look at the 24 by 24 video. And it gets worse as you start to use BASIC to list programs the effective display size becomes 23 by 20.

In reading through *The First Book of OSI*, from Elcomp Publishing, I found that a company names Silver Spur Electronics, in Chino, California, sells detailed instructions to double the display size by adding several jumpers and a couple of I.C.'s to the board. The modified display yields an effective display size of about 26 lines of 48 characters (which can be enlarged if you don't want a border around the display).

After making the modifications, though, the BASIC in ROM still thinks the memory map of the video display is the same, and so it only uses half the screen. Included with the modification instructions is a software patch which will allow BASIC to utilize the whole display. However, that, too, gives you only a *very* simple cursor. Using other computers I

```
;SUPER CURSOR V1.3
;Written by Frank Cohen
;
;Cursor Routine for OSI Superboard II
;to suppliment Microsoft's Basic-in-ROM
;cursor functions.
;
;Note: This program works with Steven
;Chalfin's video modifications and needs
;to be changed to work with a Superboard's
;normal 24 by 24 video. At the end of this
;listing are the changes for 24 by 24 video.
;
;
;This program loads into 1E40-1FE7 hex
;which is the top of memory on an 8K
;Superboard II. It may be reassembled for other
;addresses if desired.
;
;Directions: Once loaded the following must
;be done to start Super Cursor-
; 1. Set the Zero page locations
; 2. Cold start BASIC limiting the memory size
;     to 7624 (dec.).  MEM SIZ? 7624
; 3. Poke the following-
;          POKE 538,64:POKE539,30
;At this point a solid white cursor should
;appear at the home position (upper left corner)
;If this happens you have successfully loaded
;Super Cursor V1.3. If not, try it again.
;
;Options:
; To turn off the scrolling function-
;   POKE 7861,128:POKE 7862,30
; To turn on the scrolling function-
;   POKE 7861,105:POKE 7862,31
; To change the cursor symbol-
;   POKE 8033,X  (where x is a graphics number)
;
;HOME LOCATION = D0CC (hex)
;Horizontal Boundary = 44   (2C hex)
;Verticle Boundary = 26     (1A hex)
;
;BASIC Commands-
; Clear Screen = PRINT CHR$(1)
; Home Cursor = PRINT CHR$(2)
;
;Zero Page Usage
;>MR 1 80       >33 80 06
00E0 CC               CURSLOC LOW;Cursor Location Low byte
00E1 D0               CURSLOC HI ;Cursor Location High byte
00E2 20               TEMP       ;Stores byte from cursor location
00E3 00               HL         ;Horizontal Location of Cursor
00E4 00               VL         ;Verticle Location of Cursor
00E5 00               SCURS LOW  ;16 Bit scratch pad register
00E6 00               SCURS HI   ;
                                 ;
                                 ;
1E40                             ;Start of Program
1E40 8D 02 02   Start    STA 0202        ; Save all register onto the
1E43 48                  PHA             ; the stack
1E44 8A                  TXA
1E45 48                  PHA
1E46 98                  TYA
1E47 48                  PHA
1E48 AD 02 02            LDA 0202
1E4B C9 5F      Menu     CMP $5F         ;Check key pressed for cursor
1E4D D0 03               BNE NDE         ;function
1E4F 20 BE 1F            JSR Left
1E52 C9 02      NDE      CMP $02
1E54 D0 03               BNE NHO
1E56 20 80 1E            JSR Home
1E59 C9 0D      NHO      CMP $0D
1E5B D0 03               BNE NCR
1E5D 20 95 1E            JSR CR
1E60 C9 0A      NCR      CMP $0A
1E62 D0 03               BNE NLF
1E64 20 AB 1E            JSR LF
1E67 C9 01      NLF      CMP $01
1E69 D0 03               BNE NCL
1E6B 20 C2 1E            JSR CLS
1E6E C9 00      NCL      CMP $00
1E70 F0 03               BEQ Exit
1E72 20 E8 1E            JSR Dispc
1E75 68         Exit     PLA             ;Restore all the resisters from
1E76 A8                  TAY             ;the stack
```

OSI TRS-80 COLOR-80 OSI

GALAXIAN - 4K - One of the fastest and finest arcade games ever written for the OSI, this one features rows of hard-hitting evasive dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. Specify system — A bargain at $9.95 OSI

LABYRINTH - 8K - This has a display background similar to MINOS as the action takes place in a realistic maze seen from ground level. This is, however, a real time monster hunt as you track down and shoot mobile monsters on foot. Checking out and testing this one was the most fun I've had in years! — $13.95. OSI

THE AARDVARK JOURNAL

FOR OSI USERS — This is a bi-monthly tutorial journal running only articles about OSI systems. Every issue contains programs customized for OSI, tutorials on how to use and modify the system, and reviews of OSI related products. In the last two years we have run articles like these!

1) A tutorial on Machine Code for BASIC programmers.
2) Complete listings of two word processors for BASIC IN ROM machines.
3) Moving the Directory off track 12.
4) Listings for 20 game programs for the OSI.
5) How to write high speed BASIC — and lots more —

Vol. 1 (1980) 6 back issues - $9.00

Vol. 2 (1981) 4 back issues and subscription for 2 additional issues - $9.00.

ADVENTURES!!!

For OSI, TRS-80, and COLOR-80. These Adventures are written in BASIC, are full featured, fast action, full plotted adventures that take 30-50 hours to play. (Adventures are interactive fantasies. It's like reading a book except that you are the main character as you give the computer commands like "Look in the Coffin" and "Light the torch".)

Adventures require 8K on an OSI and 16K on COLOR-80 and TRS-80. They sell for $14.95 each.

ESCAPE FROM MARS (by Rodger Olsen)

This ADVENTURE takes place on the RED PLANT. You'll have to explore a Martian city and deal with possibly hostile aliens to survive this one. A good first adventure.

PYRAMID (by Rodger Olsen)

This is our most challenging ADVENTURE. It is a treasure hunt in a pyramid full of problems. Exciting and tough!

TREK ADVENTURE (by Bob Retelle)

This one takes place aboard a familiar starship. The crew has left for good reasons - but they forgot to take you, and now you are in deep trouble.

DEATH SHIP (by Rodger Olsen)

Our first and original ADVENTURE, this one takes place aboard a cruise ship - but it ain't the Love Boat.

VAMPIRE CASTLE (by Mike Bassman)

This is a contest between you and old Drac - and it's getting a little dark outside. $14.95 each.

OSI NEW-NEW-NEW OSI
TINY COMPILER

The easy way to speed in your programs. The tiny compiler lets you write and debug your program in Basic and then automatically compiles a Machine Code version that runs from 50-150 times faster. The tiny compiler generates relocatable, native, transportable machine code that can be run on any 6502 system.

It does have some limitations. It is memory hungry — 8K is the minimum sized system that can run the Compiler. It also handles only a limited subset of Basic — about 20 keywords including FOR, NEXT, IF THEN, GOSUB, GOTO, RETURN, END, STOP, USR(X), PEEK, POKE, -, =, *, /, , <, > , Variable names A-Z, and Integer Numbers from 0-64K.

TINY COMPILER is written in Basic. It can be modified and augmented by the user. It comes with a 20 page manual.

TINY COMPILER — $19.95 on tape or disk OSI

SUPERDISK II

This disk contains a new BEXEC* that boots up with a numbered directory and which allows creation, deletion and renaming of files without calling other programs. It also contains a slight modification to BASIC to allow 14 character file names.

The disk contains a disk manager that contains a disk packer, a hex/dec calculator and several other utilities.

It also has a full screen editor (in machine code on C2P/C4)) that makes corrections a snap. We'll also toss in renumbering and program search programs — and sell the whole thing for —

SUPERDISK II $29.95 (5¼") OSI

BARE BOARDS FOR OSI C1P

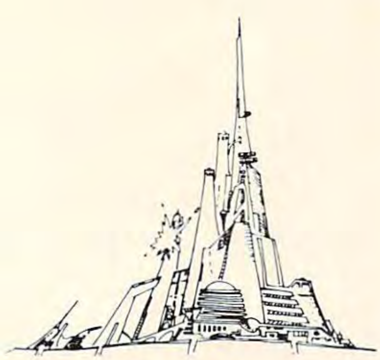
MEMORY BOARDS!!! — for the C1P — and they contain parallel ports!

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.

16K RAM BOARD FOR C1P — This one does not have a parallel port, but it does support 16K of 2114's. Bare Board $39.95.



WORD PROCESSING THE EASY WAY— WITH MAXI-PROS

This is a line-oriented word processor designed for the office that doesn't want to send every new girl out for training in how to type a letter.

It has automatic right and left margin justification and lets you vary the width and margins during printing. It has automatic pagination and automatic page numbering. It will print any text single, double or triple spaced and has text centering commands. It will make any number of multiple copies or chain files together to print an entire disk of data at one time.

MAXI-PROS has both global and line edit capability and the polled keyboard versions contain a corrected keyboard routine that make the OSI keyboard decode as a standard typewriter keyboard.

MAXI-PROS also has sophisticated file capabibilities. It can access a file for names and addresses, stop for inputs, and print form letters. It has file merging capabilities so that it can store and combine paragraphs and pages in any order.

Best of all, it is in BASIC (0S65D 51/4" or 8" disk) so that it can be easily adapted to any printer or printing job and so that it can be sold for a measly price.

MAXI-PROS — $39.95. Specify 5¼ or 8" disk.

SUPPORT ROMS FOR BASIC IN ROM MACHINES — C1S/C2S. This ROM adds line edit functions, software selectable scroll windows, bell support, choice of OSI or standard keyboard routines, two callable screen clears, and software support for 32-64 characters per line video. Has one character command to switch model 2 C1P from 24 to 48 character line. When installed in C2 or C4 (C2S) requires installation of additional chip. C1P requires only a jumper change. — $39.95

C1E/C2E similar to above but with extended machine code monitor. — $59.95 OSI

ARCADE GAMES FOR OSI, COLOR-80 AND TRS-80 (8K OSI, 16K TRS-80 AND COLOR-80)

TIMETREK - A REAL TIME, REAL GRAPHICS STARTRECK. See your torpedoes hit and watch your instruments work in real time. No more unrealistic scrolling displays! $14.95.

STARFIGHTER - This one man space war game pits you against spacecruisers, battlewagons, and one man fighters, you have the view from your cockpit window, a real time working instrument panel, and your wits. Another real time goody. $9.95

BATTLEFLEET - This grown up version of Battleship is the toughest thinking game available on OSI or 80 computers. There is no luck involved as you seek out the computers hidden fleet. A topographical toughie. $9.95

QUEST - A NEW IDEA IN ADVENTURE GAMES! Different from all the others, Quest is played on a computer generated mape of Alesia. Your job is to gather men and supplies by combbat, bargaining, exploration of ruins and temples and outright banditry. When your force is strong enough, you attack the Citadel of Moorlock in a life or death battle to the finish. Playable in 2 to 5 hours, this one is different every time.

16K COLOR-80 OR TRS-80 ONLY. $14.95

Please specify system on all orders

This is only a partial listing of what we have to offer. We offer over 120 games, ROMS, and data sheets for OSI systems and many games and utilities for COLOR-80 and TRS-80. Send $1.00 for our catalog.

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

OSI COLOR-80

found that I really liked being able to Home, or Clear Screen, or Line Feed, or Backspace the cursor. All these are not possible with the cursor program in the ROM.

Super Cursor solved my needs for an advanced cursor program. In addition to the above functions, it can actually Backspace the cursor (the BASIC in ROM version prints another underline), you can define what the cursor looks like by picking any of the graphics characters available, you can also scroll at the bottom of the display or wrap around to the top. All these functions are available in BASIC or you can use Super Cursor from a machine language program.

If you have not installed the video modifications for the larger display size you will need to modify several locations in Super Cursor. These modifications can be found in the listing after Super Cursor's machine language listing.

In operating Super Cursor, some steps must be taken to tell BASIC to use Super Cursor rather than its old cursor. First load Super Cursor into memory. If you have an assembler, you can reassemble it to fit anywhere in memory. It occupies approximately 425 bytes of memory. If you don't have an assembler, I would not advise trying to move Super Cursor as almost everything uses subroutines which need absolute addresses (you would have to renumber everything). Super Cursor, as it is listed, fits into the top portion of an 8K Superboard II.

Once loaded, it is necessary to set up the page zero memory vectors. There are seven bytes in all which must be set as follows:

00E0 CC D0 20 00 00 00 00

After you have completed this, you can cold-start BASIC. Be sure to limit BASIC's memory size to only 7624 bytes or else you will wipe out Super Cursor. To limit BASIC's memory, enter:

```
1E77 68                     PLA
1E78 AA                     TAX
1E79 68                     PLA
1E7A 4C 6C FF               JMP FF6C          ;Jump back to BASIC
1E7D EA EA EA               NOP               ;For future expansion
                                              ;
1E80 20 53 1F     Home      JSR TC            ;Home routine
1E83 A9 D0                  LDA $D0           ;Set Cursloc to DOCC
1E85 85 E1                  STA Cursloc Hi
1E87 A9 CC                  LDA $CC
1E89 85 E0                  STA Cursloc Lo
1E8B A9 00                  LDA $00           ;Set HL and VL to 00
1E8D 85 E4                  STA VL
1E8F 85 E3                  STA HL
1E91 20 60 1F               JSR SC
1E94 60                     RTS
                                              ;
1E95 20 53 1F     CR        JSR TC            ;Carrage Return
1E98 A9 00                  LDA $00           ;Subtract HL from Cursloc
1E9A 85 E6                  STA SCURS HI
1E9C A5 E3                  LDA HL
1E9E 85 E5                  STA SCURS LO
1EA0 20 B0 1F               JSR SBCC
1EA3 A9 00                  LDA $00
1EA5 85 E3                  STA HL
1EA7 20 5A 1F               JSR CT
1EAA 60                     RTS
                                              ;
1EAB A5 E4        LF        LDA VL            ;Line Feed
1EAD C9 19                  CMP $19           ;Check for Scroll
1EAF D0 0A                  BNE LFA
1EB1 20 53 1F               JSR TC            ;Carrage return and Scroll
1EB4 20 69 1F               JSR Scroll
1EB7 20 95 1E               JSR CR
1EBA 60                     RTS
1EBB 20 95 1E     LFA       JSR CR            ;No scroll
1EBE 20 27 1F               JSR DOWN
1EC1 60                     RTS
                                              ;
1EC2 A2 00        CLS       LDX $00           ;Clear Screen
1EC4 A9 20                  LDA $20           ;Set up
1EC6 9D 00 D0     CLA       STA D000,X
1EC9 9D 00 D1               STA D100,X
1ECC 9D 00 D2               STA D200,X
1ECF 9D 00 D3               STA D300,X
1ED2 9D 00 D4               STA D400,X
1ED5 9D 00 D5               STA D500,X
1ED8 9D 00 D6               STA D600,X
1EDB 9D 00 D7               STA D700,X
1EDE CA                     DEX
1EDF F0 03                  BEQ CLSE
1EE1 4C C6 1E               JMP CLA
1EE4 20 5A 1F     CLSE      JSR CT
1EE7 60                     RTS
                                              ;
1EE8 85 E2        DISPC     STA Tempreg       ;Display a character
1EEA A5 E3                  LDA HL
1EEC C9 2C                  CMP $2C
1EEE F0 04                  BEQ DISA          ;Check for a line overflow
1EF0 20 FB 1E               JSR Right
1EF3 60                     RTS
1EF4 20 95 1E               JSR CR            ;Carrage return and line feed
1EF7 20 AB 1E               JSR LF
1EFA 60                     RTS
                                              ;
1EFB 20 53 1F     RIGHT     JSR TC            ;Cursor Right
1EFE A5 E3                  LDA HL            ;Check for overflow
1F00 C9 2C                  CMP $2C
1F02 F0 10                  BEQ RA
1F04 E6 E3                  INC HL            ;Increment Cursor
1F06 A9 00                  LDA $00
1F08 85 E6                  STA SCURS HI
1F0A A9 01                  LDA $01
1F0C 85 E5                  STA SCURS LO
1F0E 20 A2 1F               JSR ADDC
1F11 4C 23 1F               JMP FRI
1F14 A9 00        RA        LDA $00           ;Subtract 2C from Cursor
1F16 85 E3                  STA HL
1F18 A9 00                  LDA $00
1F1A 85 E6                  STA SCURS HI
1F1C A9 2C                  LDA $2C
1F1E 85 E5                  STA SCURS LO
1F20 20 B0 1F               JSR SBCC
1F23 20 5A 1F     FRI       JSR CT
1F26 60
                                              ;
1F27 20 53 1F     DOWN      JSR TC            ;Cursor Down
1F2A A5 E4                  LDA VL            ;Check for overflow
1F2C C9 1A                  CMP $1A
1F2E F0 10                  BEQ DDN
1F30 E6 E4                  INC VL            ;Add 40 to Cursor
1F32 A9 00                  LDA $00
1F34 85 E6                  STA SCURS HI
1F36 A9 40                  LDA $40
1F38 85 E5                  STA SCURS LO
1F3A 20 A2 1F               JSR ADDC
```

7624, in response to the cold-start question: Mem Siz?

Now that you are running BASIC, all you have to do is to type POKE 538,64:POKE 539,30 and press ENTER. You should see the solid white cursor in the upper left (HOME) position of the display. If you hit the space bar, it should move. If it doesn't behave properly then go back into the Monitor Program and check to see if you entered Super Cursor correctly. It is quite easy to make a typing mistake with machine language programs.

If you don't want the cursor to scroll when it reaches the bottom of the screen, you can turn off the scroll function by typing: POKE 7861,128:POKE 7862,30. You can also turn on the scroll function by typing POKE 7861,105:POKE 7862,31. If you want to change the cursor symbol to some other graphics character, all you have to do is to type POKE 8033,x (where x is the graphics

```
1F3D 4C 4F 1F            JMP FDN
1F40 A9 00      DDN      LDA $00          ;Subtract 0640 from Cursor
1F42 85 E4               STA VL
1F44 A9 06               LDA $06
1F46 85 E6               STA SCURS HI
1F48 A9 40               LDA $40
1F4A 85 E5               STA SCURS LO
1F4C 20 B0 1F            JSR SBCC
1F4F 20 5A 1F   FDN      JSR CT
1F52 60                  RTS
                                          ;
1F53 A5 E2      TC       LDA TEMPREG      ;Temp reg. goes to Cursor location
1F55 A0 00               LDY $00
1F57 91 E0               STA (CURSLOC),Y
1F59 60                  RTS
                                          ;
1F5A A0 00      CT       LDY $00          ;Cursor location goes to Temp reg.
1F5C B1 E0               LDA (CURSLOC),Y
1F5E 85 E2               STA TEMPREG
                                          ;
1F60 A9 A1      SC       LDA $A1          ;Cursor symbol goes to Cursor location
1F62 A0 00               LDY $00
1F64 91 E0               STA (CURSLOC),Y
1F66 A9 00               LDA $00
1F68 60                  RTS
                                          ;
1F69 20 53 1F   SCROLL   JSR TC           ;Scroll display one
1F6C A9 20               LDA $20          ;Set up SCURS
1F6E 85 E2               STA TEMPREG
1E70 A9 00               LDA $00
1E72 85 E5               STA SCURS LO
1E74 A9 D0               LDA $D0
1E76 85 E6               STA SCURS HI
1F78 A2 00      SCRT     LDX $00          ;Scroll it
1F7A A0 40               LDY $40
1F7C B1 E5               LDA (SCURS),Y
1F7E 81 E5               STA (SCURS),X
1F80 A5 E5               LDA SCURS LO
1F82 18                  CLC
1F83 69 01               ADC $01
1F85 85 E5               STA SCURS LO
1F87 90 02               BCC SCAT
1F89 E6 E6               INC SCURS HI
1F8B A5 E6      SCAT     LDA SCURS HI
1F8D C9 D8               CMP $D8
1F8F F0 03               BEQ SCON
```

number). Normally, the cursor is equal to 161, which is a white box. If you want to Home the cursor type, PRINT CHR$(2). If you want to Clear the screen type PRINT CHR$(1).

Until I began to use the Home and Clear functions, I didn't realize what could be accomplished in a BASIC program. The following is a short program which tests the Random Number Generator of the Superboard's Microsoft BASIC. By running this program, you will see the screen being updated as though the program POKEs the display with the correct information. Actually, the use of the HOME function is all that is being utilized.

```
 10 REM RANDOM NUMBER
    GENERATOR TEST
          Remarks
 20 DIM A(9)
 30 PRINT CHR$(1),CHR$(2)
          Clear and Home
 40 POKE 8033,32
          Change the cursor to a space
 50 FOR I=1 TO 1000
 60 X=INT(RND(1)*10)
 70 A(X)=A(X)+1
 80 PRINT CHR$(2)
          Home the cursor
 90 FOR J=0 TO 9
100 PRINT J;"=";A(J)
110 NEXT J
120 PRINT"SAMPLE=";X
130 PRINT"I=";I
140 NEXT I
150 POKE 8033,161
          Restore cursor
160 END
```

As you can see by running this program, working with the Superboard II gets easier and easier with the help of an advanced cursor program like Super Cursor V1.3.


```
1F91 4C 78 1F             JMP SCRT
1F94 A2 40      SCON      LDY $40
1F96 A9 20      SCA       LDA $20          ;Blank bottom line
1F98 9D C0 D7             STA D7C0
1F9B CA                   DEY
1F9C D0 F8                BNE SCA
1F9E 60                   RTS
1F9F EA EA EA             NOP              ;for future expansion
                                           ;
1FA2 A5 E0      ADDC      LDA CURSLOC LO   ;Add SCURS to CURSLOC
1FA4 18                   CLC
1FA5 65 E5                ADC SCURS LO
1FA7 85 E0                STA CURSLOC LO
1FA9 A5 E1                LDA CURSLOC HI
1FAB 65 E6                ADC SCURS HI
1FAD 85 E1                STA CURSLOC HI
1FAF 60                   RTS
                                           ;
1FB0 A5 E0      SBCC      LDA CURSLOC LO   ;Subtract CURSLOC from SCURS
1FB2 38                   SEC
1FB3 E5 E5                SBC SCURS LO
1FB5 85 E0                STA CURSLOC LO
1FB7 A5 E1                LDA CURSLOC HI
1FB9 E5 E6                SBC SCURS HI
1FBB 85 E1                STA CURSLOC HI
1FBD 60                   RTS
                                           ;
1FBE 20 53 1F   LEFT      JSR TC           ;Cursor Left
1FC1 A5 E3                LDA HL           ;Check for overflow
1FC3 F0 10                BEQ LLE
1FC5 C6 E3                DEC HL           ;Add 01 to CURSLOC
1FC7 A9 00                LDA $00
1FC9 85 E6                STA SCURS HI
1FCB A9 01                LDA $01
1FCD 85 E5                STA SCURS LO
1FCF 20 B0 1F             JSR SBCC
1FD2 4C E4 1F             JMP LEFY
1FD5 A9 2C      LLE       LDA $2C          ;Add 2C to Cursor
1FD7 85 E3                STA HL
1FD9 A9 00                LDA $00
1FDB 85 E6                STA SCURS HI
1FDD A9 2C                LDA $2C
1FDF 85 E5                STA SCURS LO
1FE1 20 A2 1F             JSR ADDC
1FE4 20 5A 1F   LEFY      JSR CT
1FE7 60                   RTS
                                           ;
                                           ;
                                           ;Routines
1E40            START                      ;Start of Program
1E80            HOME                       ;Home cursor
1EFB            RIGHT                      ;Cursor Right
1F27            DOWN                       ;Cursor Down
1E95            CR                         ;Carriage Return
1EAB            LF                         ;Line Feed
1EC2            CLS                        ;Clear Screen
1EE8            DISPC                      ;Display a character
1F53            TC                         ;Temp reg. goes to display
1F5A            CT                         ;Cursor char. goes to temp reg.
1F60            SC                         ;Cursor symbol goes to disply
1F69            SCROLL                     ;Scroll display one line up
1FA2            ADDC                       ;Add SCURS to CURSLOC
1FB0            SBCC                       ;Subtract SCURS from CURSLOC
1FBE            LEFT                       ;Cursor Left
                                           ;
                                           ;End
```

Modifications to Super Cursor V1.3 for 24 by 24 Video

Zero page locations must be changed as below:

00E0	**85**	**Cursloc LO**
00E1	**D0**	**Cursloc HI**

Make the following changes to the main program:

1E88	**85**	**LDA $85**
1EAE	**17**	**CMP $19**
1EED	**17**	**CMP $17**
1F01	**17**	**CMP $17**
1F1D	**17**	**CMP $17**
1F2D	**17**	**CMP $17**
1F45	**0E**	**LDA $0E**
1F49	**02**	**LDA $20**
1F7B	**20**	**LDY $20**
1F8E	**D4**	**CMP $D4**
1F95	**20**	**LDY $20**
1FD6	**17**	**LDA $17**
1FDE	**17**	**LDA $17**

©

Memory Recall Test

V. Nasser
Birmingham, England

If a human subject is exposed to a set of random numbers or letters for a short span of time, the number of items recalled is generally about 7 ± 2 items. Theories abound as to the capacity for immediate memory. Obviously words/letters/numbers that are meaningful will be more likely to be remembered than meaningless items. Also, if numbers can be organized in a meaningful way, then the probability of accurate recall is greater. It is possible, with training, to increase one's immediate memory span by a considerable amount. Let me give a typical example. Exposed to the numbers: 162536496481 (and providing one recognizes that each pair of digits is the square of consecutive natural numbers from 4 to 9) then one only has to remember six "chunks," but will nevertheless seem to remember 12 numbers.

However, if the numbers are random, obviously one may not always be able to organize the digits in a meaningful way. My program works in the following way: it asks the subject how many numbers he or she wants to recall. When the subject enters the required items, the program will display the appropriate number of random numbers for a certain time. This exposure time incidentally, is a function of the number of numbers chosen. So that three numbers will be exposed for a much shorter time than ten numbers. After the exposure of random numbers, the screen is automatically cleared and the subject is asked to input the numbers in the sequence originally shown. The program will terminate upon the first erroneous digit input and give the correct answers.

So, having explained a bit about the psychology of immediate memory and presented a program to test your memory span, what use can this program be put to?

Under the influence of alcohol or sedative drugs the memory span drops in proportion to the amount consumed. Further, in certain conditions of brain damage it is not possible to remember more than two or three digits. Thus it has diagnostic possibilities. But more interesting, in my opinion, is the way the program encourages you to remember. If one starts at a low level, the initial successes create the automatic reinforcements necessary to increase one's memory span. This is remarkable since the numbers displayed are random and there is very little chance of organizing them in any meaningful way.

It can be used for any age group from nine years onwards. The program is both simple and absorbing. It can be adapted for any computer using BASIC, but was specifically written for the Superboard II.

```
1 FOR T=1 TO 32:PRINT:NEXT
2 PRINT "  MEMORY RECALL TEST"
3 PRINT:PRINT:PRINT
8 CLEAR
9 PRINT "  TYPE NUMBER OF NUMBERS TO BE ¬
      ¬RECALLED"
10 INPUT P
15 DIM A(P)
16 FOR I=1 TO 32:PRINT:NEXT
20 FOR N=1 TO P
25 A(N)=INT(RND(1)*10)
26 PRINT TAB(2);
30 PRINT A(N);:IF POS(1)>18 THEN PRINT:
      ¬PRINT
50 NEXT
55 FOR X=1 TO 500*N:NEXT X
56 FOR T=1 TO 32:PRINT:NEXT
60 FOR N=1 TO P
70 INPUT Y
80 IF Y<>A(N) THEN 100
85 NEXT N
90 PRINT " WELL DONE "
95 GOTO 8
100 PRINT " INCORRECT"
110 PRINT " THE CORRECT ANSWERS ARE"
120 FOR N=1 TO P
125 PRINT TAB(2);
130 PRINT A(N);:IF POS(1)>18 THEN PRINT:
      ¬PRINT
135 NEXT N
136 PRINT:PRINT:PRINT
140 GOTO 8
READY.
```

THE PET® GAZETTE

A Look At Superpet

The SuperPET looks about the same as an 8032 PET which is not surprising — it *is* an 8032 with two additional boards inside, 64K more RAM (necessary to hold the new, disk-based languages), and a new I/O system. Externally, it resembles an 8032 except for three things: 1. the logo reads "SuperPET" and, below, "SP9000"; 2. there is a set of stick-on key overlays for APL special characters; and 3. there are two, three-position switches attached to the side of the black base, below the numeric keypad.

SuperPET comes with four high-level languages, "Waterloo micro-" versions of: BASIC (40.5K), PASCAL (40.5K), FORTRAN (52.5K), and APL (64.75K). COBOL is expected soon. In addition, there is an extensive development system (nearly a high-level language) for work in 6809 machine language (an Assembler, Monitor, Linker, and Editor).

Availability

According to Commodore, a SuperPET ($1995) ordered today would arrive in about 45 days. Owners of 8032's could install a single-board upgrade for approximately $500. This upgrade is expected to be available early in 1982.

All the languages are on a single 8050 disk. This does not mean, though, that the SuperPET cannot be used with a 2040 disk system. Program 1 will redefine an 8050 as device #9. It should be linked to a power-off 2040. Turning on the 2040 leaves it as device #8. Program 2 will move the languages from an 8050 to a 2040. The value of F$ must be added to the program when transferring the final two programs (a library of utilities) on the disk, "&00,)".%80" and "7!4,)".%80". The internal quotes cannot, of course, be part of a filename — what's more, the characters are not what they seem and must be defined using CHR$ as shown in Program 3.

Program 1.

```
10 OPEN 15,8,15
20 PRINT #15,"M-W" CHR$(12)CHR$(00
    )CHR$(2)CHR$(9+32)CHR$(9+6
    4)
30 REM THE LAST TWO 9'S DEFINE DEV
    ICE #9.
40 REM FOR THE 2031 (SINGLE DRIVE)
    , USE 119 INSTEAD OF 12.
```

Program 2.

```
5 PRINT"{CLEAR}":CATALOGD0ONU9:IN
    PUT"{DOWN}FILE NAME";F$
7 SCRATCH(F$):POKE59464,0
10 OPEN1,9,8,"0:"+F$+",P,R":K=1:SO
    =59464
20 OPEN2,8,8,"0:"+F$+",P,W":B=255:
    Z=0
30 GET#1,A$:S=ST:B=B+K:IFB>254THEN
    B=Z:NB=NB+K:PRINT"{UP}{ERA
    ERASE END}BLOCK:";NB
35 IFA$=""THENA$=CHR$(Z)
40 PRINT#2,A$;:IFS=ZTHEN30
60 CLOSE1:CLOSE2:POKE 59467,0
70 PRINT"{CLEAR}TRANSFER COMPLETE{
    DOWN}":CATALOGD0
```

Program 3.

```
1 REM CREATES FILENAME FOR NON-ST
    ANDARD CHARACTERS IN SYSTE
    M LIBRARY FILENAME
3 F$=CHR$(102)+CHR$(112)+CHR$(112
    )+CHR$(108)
4 F$=F$+CHR$(105)+CHR$(98)+CHR$(4
    6)+CHR$(101)+CHR$(120)+CHR
    $(112)
5 REM ELIMINATE LINE FIVE
```

Program 3a.

```
1 REM FILENAME 7!4,)".%80
3 F$=CHR$(119)+CHR$(97)+CHR$(116)
    +CHR$(108)
4 F$=F$+CHR$(105)+CHR$(98)+CHR$(4
    6)+CHR$(101)+CHR$(120)+CHR
    $(112)
```

JINSAM

Data Manager selected by NASA, Kennedy Space Center

SAVE TIME! SAVE MONEY!
JINSAM saves taxpayers 100's of thousands of $

OFFICIALLY APPROVED BY COMMODORE

"Much more powerful than you can imagine!"
Robert Baker
Kilobaud Microcomputing

JINSAM is an integrated system. It makes it easy to use your information to its fullest. No more will hundreds of valuable hours be spent searching or analyzing needed information nor re-entering information for various reports.

JINSAM transforms your desk-top computer into the "state of the art" data processing machine with features and accessories found nowhere, even at 10 times the price. NASA, Kennedy Space Center selected JINSAM 8.0 and saved approximately $95,000 over other software/hardware costs. Riley County, Kansas also selected JINSAM 8.0 and saved approximately $90,000 over other software/hardware costs.

JINSAM is designed for you. It is forgiving. It has help commands for every option, available at the touch of a button. The amount of information you store, its structure and/or your hardware can change but your data won't have to be re-entered. Recovery utilites are included even for catastrophes, security passwords are built in for privacy, simple editing and entry includes auto recall, and deleting records is easy and the space is reclaimed. JINSAM includes TWO FREE accessories for reports and labels. You have unlimited report formats with summing and lined up decimals and the label printer prints up to 5 across – any size and even allows single envelopes or index cards.

JINSAM 1.0 allows fast and easy file handling, manipulation and report generation for any CBM computer with CBM 2040 disk drive. It features a menu for ease, has encrypted passwords, 3 deep sorts, .5 to 3 second recall.

JINSAM 4.0 for CBM 4000 series adds JINSORT, user accessible machine sort of 1000 records in 15 seconds; compaction/expansion of information, automatic list maintainance, unlimited number of fields, unlimited record length and much more.

JINSAM 8.0 for CBM 8000 series has all 4.0 features plus unlimited sort, horizontal format, and search by key or record number.

JINSAM 8.2, NEW FOR '82 expands 8.0 capabilities by adding information search by word, key or record number and machine language print, format and manipulation routines.

★ CUSTOM DATA FILES
★ CUSTOM REPORTS/LABELS
★ KEYED RANDOM ACCESS
★ FAST/EASY/MENU DRIVEN
★ MULTIPLE SEARCH KEYS
★ PRIVACY ACCESS CODES
★ WILD CARD SEARCH

JINSAM EXECUTIVE version (soon to be released) is our most powerful professional system for the CBM 8000 and 9000 series. Executive will have 8.2 extended features plus allow multiple users with in-use lockout protection, executive command files, automatic math relations, join, merge or link files, greatly increased record capacity and machine information search by word, as well as by key or record number and many, many more features.

There are currently 7 more interfacing modules – and more under development, including independent interfaces between JINSAM and business packages for your G/L, A/R, A/P needs. We announce the availability of modules and enhancements in JINSAM's quarterly newsletter.

WORDPROPACK - Intelligent interface for WordPro 3, 3+, 4, 4+, creates lists of information from JINSAM files. It allows up to 10 conditions based on each item of information. Produce individualized letters, report cards, special reports, checks, invoices, etc.

MULTI-LABEL - Prints multiple labels per record with up to 2 lines for messages and consecutive numbering. Produce inventory, caution labels, bulk mail labels, etc.

MATHPACK - global calculator/editor +, −, *, /, by another field or constant; null (remove contents) of a field or replace contents of a field with any word, number or phrase. Sum multiple fields in each record or running sum of single field in all records. Extract information or effect permanent change. Replace in the same field or place in a waiting field.

DESCRIPTIVE STATPACK - Determine MEAN, MEDIAN, MODE, STANDARD DEVIATION, VARIANCE, RANGE. Generate HISTOGRAMS from 1 to 25 steps, and produce Z-SCORE reports.

ADVANCED STATPACK - (You must also acquire DESCRIPTIVE STATPACK) Generate one, two or three way CROSSTABS (number of occurance) CHI SQUARE, LINEAR REGRESSION, with graphic representation and prediction, LINEAR CORRELATION and SIMPLE ANALYSIS OF VARIANCE.

CALCPACK - 2 way interface to VisiCalc or any user program. It lets you use VisiCalc for complex manipulation, editing, placing results in JINSAM for sorting, storing or moving data to WordPro as well as giving the ability for exchange with your own applications.

INTERAC - Interface which can read VisiCalc files, WordPro files and almost any sequential files to build JINSAM databases automatically. For example: You could "download" information on holerith cards to sequential files and INTERAC would place them into JINSAM files.

All accessories are accessed thru the JINSAM menu and require security password to gain entrance.

JINSAM gives you FREEDOM OF CHOICE. Start with JINSAM 1.0 and upgrade hardware and data at any time. Choose from accessories at any time. The JINSAM Newsletter brings the latest updates, user input and uses and keeps an eye on the future.

JINSAM stands alone by placing "a lot of potential computing power in one integrated program package" (Fred Klein, Ferson div. of Bausch & Lomb). "The JINSAM package is justification for buying a system no matter what the hardware, be it Vector or Commodore or whatever the system" (Larry Colvin, Micro Computer Systems). It is a "grandfather" in this young field. JINSAM EXECUTIVE will be the third generation in development. All JINSAM systems are sophisticated and flexible yet easy to use. JINSAM is saving its users valuable time and money in government, educational and research institutions, business and industry worldwide.

JINSAM is a Commodore approved product. See your local dealer for a demonstration.

JINSAM Data Manager

.... Additional Information and nearest dealer
.... Newsletter Subscription ($5 US/$8 Worldwide)
.... JINSAM Demo Disk ($15, plus shipping & tax)
.... User's Guide ($40.00, plus shipping & tax)

Please send to:

Name ______________________

Position ______________________

Company ______________________

Address ______________________

City, State, Zip ______________________

Phone (   ) ______________________

Computer, Disk ______________________

Application ______________________

JINI MICRO-SYSTEMS, Inc.
Box 274C • Kingsbridge Station
Riverdale, New York 10463
(212) 796-6200

Dealer Inquiry Welcome

Manuals

The computer comes with six manuals, one for each language plus the *System Overview: Commodore SuperPET*. They are large (the BASIC manual is 221 pages) and contain numerous example programs (a second disk includes some of them). The manuals may be purchased separately from Howard W. Sams & Co., Inc., 4300 West 62nd St., P.O. Box 7092, Indianapolis, IN 46206. A minor annoyance in this otherwise carefully planned documentation is the fact that the number *1* and the lowercase *l* are identical in the Assembler handbook. In general, however, great care has obviously been taken to thoroughly explain each language. The BASIC book, for example, could easily serve as a textbook for learning this version of the language.

The Software Philosophy

Perhaps one of the first questions which comes to the mind of a microcomputerist is: what is a 40K BASIC? Personal computers contain versions of BASIC which are usually 4 to 12K large. An advanced BASIC might reach 18K. What is added when BASIC is 40.5K?

The authors of *System Overview*: "These language interpreters have been designed specifically for educational use in the teaching of computer programming. The design of the interpreters features good error diagnosis and debugging capabilities which are useful in educational and other program-development environments." There are explicit, lengthy error messages, search and replace (from the Editor), a trace facility, and *structured programming*.

Briefly, structured programming is a kind of tightening up of the rules of a language. It eliminates programming shortcuts in an effort to make programs more readable and to make languages more easily learned. Loops, for example, are supposed to be indented so they can be *seen*:

```
10 LOOP
20      X=X+1
30      Y$=VALUE$(X)
40      IF X=5000 THEN QUIT
50 ENDLOOP
```

Multiple statements per line are discouraged, spaces are required between the IF and X in IF X THEN..., the keyword VALUE must be spelled out (it replaces STR$ and VAL), LOAD "FILENAME" must have the second quote, NEXT must have a variable, dir "disk/1" replaces cAd1, and so on. This elimination of shortcuts makes programs more easily debugged, more easily read, but it also makes them larger, slows typing them in, and slows execution times. Comparing the run time of the above with the non-structured equivalent: (FOR I=1 TO 5000: Y$ = STR$(I): NEXT) takes 54 seconds, the structured version takes 119 seconds.

Some abbreviations are permitted: l for LIST, ? for PRINT. Also, the language contains a DEL function for deleting lines, RENUM for renumbering, and A for automatic line numbering.

The BASIC

As might be expected, there are significant additions and some changes to the Microsoft BASIC which is standard on other PET/CBM computers. NEW becomes CLEAR. TI becomes TIME. ! can mean REM. Structured control statements (IF, ELSE, etc.) must not be followed with anything else on a line.

A number of new functions are implemented: CURSOR (i%) sets the cursor to the position on the screen defined by the argument. DATE$ holds the current date. EPS gives the smallest number that the computer can represent. INF gives the largest. FP(x) returns the fractional part of x. IP(x) gives the integer part of x. HEX(x$) will give the hexadecimal equivalent of the decimal argument (up to a value of 32737) and HEX$(x) goes the other way. IDX(a$,b$) returns the position at which b$ first occurs within a$. IO STATUS replaces ST. MOD(x,y) provides the modulus of x for the range y. ORD(s$) returns the position of the one-character s$ in the system's set of characters. PI is pi. RPT$(s$,n) gives a string which is s$ concatenated n times. STR$(a$,s,1) is MID$.

Changes or additions to BASIC statements include: CHAIN provides program overlaying with parameter passing (USE, like DATA, contains the list to be passed). FNEND permits multiple-line function definitions. GUESS...ADMIT...ENDGUESS establish a structure similar to: 10 INPUT A$ <> "YES and A$ <> "NO" THEN PRINT "ANSWER YES OR NO": GOTO 10. (The ADMIT statement replaces an IF THEN.)

ELSEIF, ENDIF, LOOP, ENDLOOP, UNTIL, ELSE, WHILE, UNTIL, and QUIT are all statements which replace various IF THEN and FOR NEXT loop types. They are aspects of "structured programming." An ON-RESUME/IGNORE-ENDON structure permits control over some error conditions from within a program. Zero division, EOF, pressing the STOP key, under- and overflow are among the conditions which can be trapped.

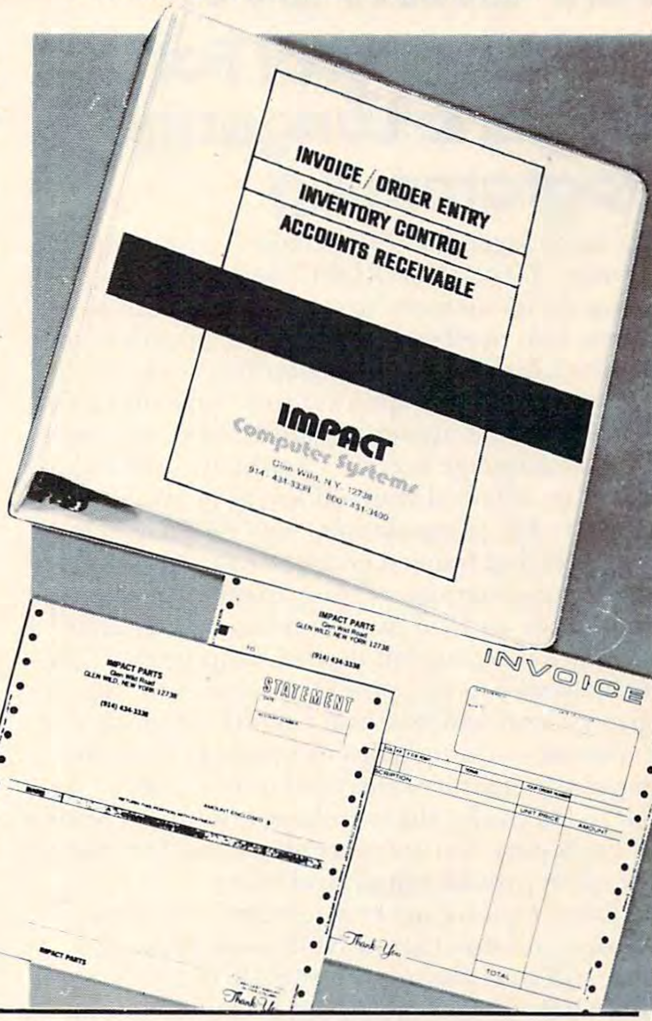
This brief summary merely hints at the wealth of software and hardware to be explored in the SuperPET. When asked what impresses them most about this machine, each industry expert answers differently. Some say that the introduction of a serious version of APL is the most significant aspect of the computer. Some say it is the built-in RS-232 interface. Some mention the multiple languages or the inherent ability to speak directly to mainframe computers or the massive bank-switched RAM. All seem to agree, however, that the new PET is super. ©

INVENTORY CONTROL PLUS*...

*INVOICING — ORDER ENTRY — ACCOUNTS RECEIVABLE

for CBM 8032 and 8050

***PARTRAC:** Three separate but interconnected computer programs. All are "on line" or in the computer at the same time. It is not necessary to change disks with PARTRAC!



INVOICING SYSTEM

a. Instant stock check on any part #.
b. Instant price check on any part #.
c. Instant customer status check.
d. Writes invoices and posts to accounts receivable automatically.
e. Writes credit memos and posts to accounts receivable automatically.
f. Writes daily totals reports.

INVENTORY SYSTEM

a. Keeps track of all information related to a part #. Prices, costs, quantities on order, in stock, back ordered and vendor.
b. Prints reports
   1.) "Reorder Advice Report"
   2.) "Price List Report"
   3.) "Complete Parts Inventory Report" with total cost of inventory. (Your accountant will like that.)

ACCOUNTS RECEIVABLE

a. Posts all open account sales.
b. Prints "Accounts Receivable Report" with aging function.
c. Prints "Statements".
d. Prints "List of Accounts Names".
e. Allows partial payments.

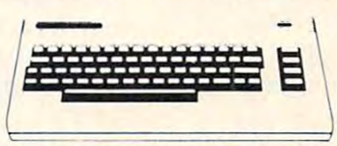
PARTRAC is fully expandable because of modular design. The basic system handles 8000 part #'s, 1000 accounts, 10,000 open invoices. This is expandable to about 20,000 part #'s, 3,000 accounts and 25,000 open invoices.

AVAILABLE AT YOUR LOCAL COMMODORE DEALER

GUARANTEED BEST PRICES!

CBM 8032
$1099

commodore



VIC 20 $259

4016	$799.00
4032	$999.99
8096	$1795.00
CBM 4022 Printer	$629.00
Tally 8024	$1699.00
CBM C2N Cassette Drive	$69.00
CBM 4040 Dual Disk Drive	$999.00
CBM 8050 Dual Disk Drive	$1349.00

Vic-TV Modual	$19.00
Vic Cassette	$69.00
Vic Disk Drive	$ Call
Vic 6 Pack program	$44.00

CBM Software

WordPro 3 Plus	$199.00
WordPro 4 Plus	$299.00
Commodore Tax Package	$399.00
Visicalc	$149.00
EBS Accts. Rec./Inventory Interactive Syst.	$595.00
BPI General Ledger	$329.00
OZZ Information System	$329.00
Dow Jones Portfolio	$129.00
Pascal	$239.00
Legal Time Accounting	$449.00
World Craft 80	$289.00
Word Check	$180.00
Create-A-Base	$180.00
Power	$89.00
Socket-2-Me	$20.00
Jinsam	Call

Printers

Epson MX-70	Call for Prices
Epson MX-80	
Epson MX-80 FT	
Diablo 630	
Nec 5530	$2495.00
TEC 1500 Starwriter 25cps	$1495.00
TEC 1500 Starwriter 45cps	$1995.00

IMPACT Computer Systems

IN N.Y. CALL
914-434-3338
(800) 431-3400
GLEN WILD, N. Y. 12738

FREE FREIGHT--SAME DAY SHIPPING

VISA MasterCard

TO ORDER: Phone orders invited (800 is for order desk only). Send check or money order and receive free shipping. New York residents add 7% sales tax. Add 3% for Visa or M.C. Equipment is subject to price change and availability without notice. Please call between 9 A.M. and 5 P.M.

SUPERMON:

A Primary Tool For Machine Language Programming

Here is the legendary Supermon — a version for Upgrade (3.0 or "New ROM") and 4.0 PETs, all keyboards, all memory sizes, 40 or 80 column screens. You need not know how to program in machine language (ML) to enter this program — or to use it. In fact, exploring with Supermon, you will find that the mysterious world of your computer's *own* language becomes gradually understandable. You will find yourself learning ML.

Many ML programmers with PET/CBM machines feel that Supermon is the essential tool for developing programs of short to medium length. All Upgrade and 4.0 machines have a "resident" monitor; a program within the computer's ROM which allows you to type SYS 1024 and see the registers, load and save and run ML programs, or see a memory dump (a list of numbers from the computer's memory cells.) But to program or analyze ML easily, disassembler, assembler, hunt, and single-step functions are all practical necessities. Supermon provides these and more.

Even if you've never assembled a single instruction and don't know NOP from ROL, this article will lead you step-by-step through the entry and SAVE of Supermon. And even if you do not plan to explore ML right now, you might consider putting this program into your library. If you ever decide to work a bit in ML, Supermon will prove invaluable.

How To Enter Supermon

1. Type in the BASIC program (Program 1). It is the same for all versions. Then save it normally by typing SAVE "CONTROL". This program will be used later to automatically find your memory size, transfer Supermon to the top, and report to you the SYS address you use to activate it.

2. Now the hard part: type SYS 1024 which enters you into the machine language monitor. You will see something like the following:

Figure 1.

```
B*
     PC  IRQ  SR AC XR YR SP
.;  0401 E455 32 04 5E 00 EE
```

Then type: M 0600 0648 and you will see something similar to (the numbers will be different, but we are going to type over them which, after hitting RETURN on each line, will enter the new numbers into the computer's memory.):

Figure 2.

```
.M 0600 0648
.:  0600 28 58 FF FF 00 0B 06 AD
.:  0608 FF FC 00 21 06 03 AD A9
.:  0610 CB 85 1F A9 0C 85 20 A5
.:  0618 34 85 21 A5 35 85 22 A0
.:  0620 00 93 06 06 D0 16 20 38
.:  0628 06 F0 11 85 23 20 38 06
.:  0630 18 65 34 AA A5 23 65 35
.:  0638 20 43 06 8A 20 43 06 20
.:  0640 50 06 90 DB 60 EA EA A5
.:  0648 1F D0 02 C6 20 C6 1F B1
.
```

We have divided Supermon into 21 blocks with 80 hexadecimal numbers per block to make typing easier. There is a final, shorter block with 64 numbers. Type right over the numbers on the screen so that line 0600 looks like it does in Program 2. Then hit RETURN and cursor over to the A5 on line 0608. (Set a TAB to this position if your keyboard has a TAB key.) Then type over the numbers in this line and so on. When you have finished typing your RETURN on line 0648, type in: M 0650 0698 and the next block will appear for you to type over. Continue this way until you finish entering the new version of line 0CC8 at the end. (Hope that no lightning or fuses blow.)

3. If you have Upgrade ROMs, you will need to correct the lines listed in Program 3 at this point. To change line 06D0, simply type M 06D0 06D0 and it will appear so that you can type over it and RETURN as in step 2.

4. Now Supermon is in your memory and you must SAVE it. Hit RETURN so that you are on a new line and type: S "SUPERMON", 01,0600, 0CCC (to SAVE to tape) or type: S "0:SUPERMON",08,0600,0CCC (to SAVE to disk drive 0).

5. Finally, you will want to use the Checksum program to see if you made any errors during the marathon. You probably did, but to make it as painless as possible, the Checksum program will flash through your Supermon and let you know which blocks need to be corrected. So, type in Program 4 (or if you have Upgrade ROMs, use the first three lines from Program 5). SAVE Checksum just in case. Then LOAD "SUPERMON" (an ordinary LOAD as with a BASIC program will slide it in starting at

WATCH

THIS

SPACE

WATCH

THIS

SPACE

address 1536, above the end of Checksum). Then RUN. Incorrect blocks will be announced. When you know where the errors are, type SYS 1024 and then M XXXX XXXX for the starting and ending addresses of the bad block. Check the numbers against Program 2 (or Program 3) and in all corrections. If, despite everything, you cannot find an error within a block, make sure that the corresponding number within the DATA statement of the Checksum program is correct. Then SAVE the good version "SUPERMON1" as in step 4. "SUPERMON1" as in step 4.

6. Your reward is near. LOAD "CONTROL" and then LOAD SUPERMON1. Then type RUN and hold your breath. If all goes well, you should see:

Figure 3.

```
 SUPERMON4!

     DISSASSEMBLER  BY WOZNIAK/BAUM
       SINGLE STEP
 BY JIM RUSSO
 MOST OTHER STUFF ,BY BILL SEILER

 TIDIED & WRAPPED BY JIM BUTTERFIELD


 LINK TO MONITOR -- SYS 31283

 SAVE WITH MLM:
.S "SUPERMON",Ø1,7A33,8ØØØ
 READY.
```

And you should be able to use all the commands listed in the Supermon Summary. If some, or all, of the commands fail to function, check the last, short block of code to see if there are any errors.

After Supermon is relocated to the top of your memory, use a ML SAVE to save it in its final form. Instructions are on screen after RUN.

SUPERMON SUMMARY

```
COMMODORE MONITOR INSTRUCTIONS:
G GO RUN
L LOAD FROM TAPE OR DISK
M MEMORY DISPLAY
R REGISTER DISPLAY
S SAVE TO TAPE OR DISK
X EXIT TO BASIC
SUPERMON ADDITIONAL INSTRUCTIONS:
A SIMPLE ASSEMBLER
D DISASSEMBLER
F FILL MEMORY
H HUNT MEMORY
I SINGLE INSTRUCTION
P PRINTING DISASSEMBLER
T TRANSFER MEMORY
SUPERMON WILL LOAD ITSELF INTO THE
TOP OF MEMORY .. WHEREVER THAT HAPPENS
TO BE ON YOUR MACHINE.
   YOU MAY THEN SAVE THE MACHINE CODE
FOR FASTER LOADING IN THE FUTURE.
BE SURE TO NOTE THE SYS COMMAND WHICH
LINKS SUPERMON TO THE COMMODORE
MONITOR.


 SIMPLE ASSEMBLER
.A 2ØØØ LDA #$12
.A 2ØØ2 STA $8ØØØ,X
.A 2ØØ5 (RETURN)
.
     IN THE ABOVE EXAMPLE THE USER
STARTED ASSEMBLY AT 1ØØØ HEX.  THE
FIRST INSTRUCTION WAS LOAD A REGISTER
WITH IMMEDIATE 12 HEX.  IN THE SECOND
LINE THE USER DID NOT NEED TO TYPE THE
A AND ADDRESS.  THE SIMPLE ASSEMBLER
PROMPTS WITH THE NEXT ADDRESS.  TO EXIT
THE ASSEMBLER TYPE A RETURN AFTER THE
THE ADDRESS PROMPT.  SYNTAX IS THE SAME
AS THE DISASSEMBLER OUTPUT.

  DISASSEMBLER
.D 2ØØØ
(SCREEN CLEARS)
.,   2ØØØ A9 12        LDA #$12
.,   2ØØ2 9D ØØ 8Ø     STA $8ØØØ,X
.,   2ØØ5 AA           TAX
.,   2ØØ6 AA           TAX
(FULL PAGE OF INSTRUCTIONS)
     DISASSEMBLES 22 INSTRUCTIONS
STARTING AT 1ØØØ HEX.  THE THREE BYTES
FOLLOWING THE ADDRESS MAY BE MODIFIED.
USE THE CRSR KEYS TO MOVE TO AND MODIFY
THE BYTES.  HIT RETURN AND THE BYTES
IN MEMORY WILL BE CHANGED.  SUPERMON
WILL THEN DISASSEMBLE THAT PAGE AGAIN.

 PRINTING DISASSEMBLER
.P 2ØØØ,2Ø4Ø
2ØØØ A9 12        LDA #$12
2ØØ2 9D ØØ 8Ø     STA $8ØØØ,XY.
2ØØ5 AA           TAX
   ....
2Ø3F A2 ØØ        LDX #$ØØ
TO ENGAGE PRINTER, SET UP BEFOREHAND:
     OPEN 4,4:CMD4
ON 4.Ø, ACCESS THE MONITOR VIA A CALL
SYS 54386 (*NOT* A BREAK) COMMAND

 SINGLE STEP
.I
     ALLOWS A MACHINE LANGUAGE PROGRAM
TO BE RUN STEP BY STEP.
CALL REGISTER DISPLAY WITH .R AND SET
```

THE PC ADDRESS TO THE DESIRED FIRST INSTRUCTION FOR SINGLE STEPPING.
THE .I WILL CAUSE A SINGLE STEP TO EXECUTE AND WILL DISASSEMBLE THE NEXT.
CONTROLS:

< FOR SINGLE STEP;
RVS FOR SLOW STEP;
SPACE FOR FAST STEPPING;
STOP TO RETURN TO MONITOR.

[ON BUSINESS KEYBOARDS-- USE 8,←,6 AND STOP].

FILL MEMORY

.F 1000 1100 FF

FILLS THE MEMORY FROM 1000 HEX TO 1100 HEX WITH THE BYTE FF HEX.

GO RUN

.G

GO TO THE ADDRESS IN THE PC REGISTER DISPLAY AND BEGIN RUN CODE. ALL THE REGISTERS WILL BE REPLACED WITH THE DISPLAYED VALUES.

.G 1000

GO TO ADDRESS 1000 HEX AND BEGIN RUNNING CODE.

HUNT MEMORY

.H C000 D000 'READ

HUNT THRU MEMORY FROM C000 HEX TO D000 HEX FOR THE ASCII STRING READ AND PRINT THE ADDRESS WHERE IT IS FOUND. A MAXIMUM OF 32 CHARACTERS MAY BE USED.

.H C000 D000 20 D2 FF

HUNT MEMORY FROM C000 HEX TO D000 HEX FOR THE SEQUENCE OF BYTES 20 D2 FF AND PRINT THE ADDRESS. A MAXIMUM OF 32 BYTES MAY BE USED.

LOAD

.L

LOAD ANY PROGRAM FROM CASSETTE #1.

.L "RAM TEST"

LOAD FROM CASSETTE #1 THE PROGRAM NAMED RAM TEST.

.L "RAM TEST",08

LOAD FROM DISK (DEVICE 8) THE PROGRAM NAMED RAM TEST.
THIS COMMAND LEAVES BASIC POINTERS UNCHANGED.

MEMORY DISPLAY

```
.M 0000 0080
.:  0000 00 01 02 03 04 05 06 07
.:  0008 08 09 0A 0B 0C 0D 0E 0F
```

DISPLAY MEMORY FROM 0000 HEX TO 0080 HEX. THE BYTES FOLLOWING THE .: CAN BE ALTERED BY TYPING OVER THEM THEN TYPING A RETURN.

REGISTER DISPLAY

```
.R
     PC   IRQ  SR AC XR YR SP
.;  0000 E62E 01 02 03 04 05
```

DISPLAYS THE REGISTER VALUES SAVED WHEN SUPERMON WAS ENTERED. THE VALUES MAY BE CHANGED WITH THE EDIT FOLLOWED BY A RETURN.
USE THIS INSTRUCTION TO SET UP THE PC VALUE BEFORE SINGLE STEPPING WITH .I

SAVE

.S "PROGRAM NAME",01,0800,0C80

SAVE TO CASSETTE #1 MEMORY FROM 0800 HEX UP TO BUT NOT INCLUDING 0C80 HEX AND NAME IT PROGRAM NAME.

.S "0:PROGRAM NAME",08,1200,1F50

SAVE TO DISK DRIVE #0 MEMORY FROM 1200 HEX UP TO BUT NOT INCLUDING 1F50 HEX AND NAME IT PROGRAM NAME.

TRANSFER MEMORY

.T 1000 1100 5000

TRANSFER MEMORY IN THE RANGE 1000 HEX TO 1100 HEX AND START STORING IT AT ADDRESS 5000 HEX.

EXIT TO BASIC

.X

RETURN TO BASIC READY MODE. THE STACK VALUE SAVED WHEN ENTERED WILL BE RESTORED. CARE SHOULD BE TAKEN THAT THIS VALUE IS THE SAME AS WHEN THE MONITOR WAS ENTERED. A CLR IN BASIC WILL FIX ANY STACK PROBLEMS.

Program 1.

```
100 PRINT"{CLEAR}{02 DOWN}{REV} SUP
    ERMON!!"
110 PRINT"{DOWN}      DISSASSEMBLER ~
    {REV}D{OFF} BY WOZNIAK/BAU
    M
120 PRINT"          SINGLE STEP {REV}I
    {OFF} BY JIM RUSSO
130 PRINT"MOST OTHER STUFF {REV},HA
    LT{OFF} BY BILL SEILER
150 PRINT"{DOWN}TIDIED & WRAPPED BY
     JIM BUTTERFIELD"
170 L=PEEK(52)+PEEK(53)*256:SYS1536
    :M=PEEK(33):N=PEEK(34)
180 POKE52,M:POKE53,N:POKE48,M:POKE
    49,N:N=M+N*256
210 PRINT"{02 DOWN}LINK TO MONITOR ~
    -- SYS";N
220 PRINT:PRINT"SAVE WITH MLM:"
230 PRINT".S ";CHR$(34);"SUPERMON";
    CHR$(34);",01";:X=N/4096:G
    OSUB250
```

```
240 X=L/4Ø96:GOSUB25Ø:END
25Ø PRINT",";:FORJ=1TO4:X%=X:X=(X-X
    %)*16:IFX%>9THENX%=X%+7
26Ø PRINTCHR$(X%+48);:NEXTJ:RETURN
```

SUPERMON 4.0 Program 2.

```
.
.:  Ø6ØØ A9 CB 85 1F A9 ØC 85 2Ø
.:  Ø6Ø8 A5 34 85 21 A5 35 85 22
.:  Ø61Ø AØ ØØ 2Ø 38 Ø6 DØ 16 2Ø
.:  Ø618 38 Ø6 FØ 11 85 23 2Ø 38
.:  Ø62Ø Ø6 18 65 34 AA A5 23 65
.:  Ø628 35 2Ø 43 Ø6 8A 2Ø 43 Ø6
.:  Ø63Ø 2Ø 5Ø Ø6 9Ø DB 6Ø EA EA
.:  Ø638 A5 1F DØ Ø2 C6 2Ø C6 1F
.:  Ø64Ø B1 1F 6Ø 48 A5 21 DØ Ø2
.:  Ø648 C6 22 C6 21 68 91 21 6Ø
.
.:  Ø65Ø A9 8Ø C5 1F A9 Ø6 E5 2Ø
.:  Ø658 6Ø AA AA AA AA AA AA AA
.:  Ø66Ø AA AA AA AA AA AA AA AA
.:  Ø668 AA AA AA AA AA AA AA AA
.:  Ø67Ø AA AA AA AA AA AA AA AA
.:  Ø678 AA AA AA AA AA AA AA AA
.:  Ø68Ø AD FE FF ØØ 85 34 AD FF
.:  Ø688 FF ØØ 85 35 AD FC FF ØØ
.:  Ø69Ø 8D FA Ø3 AD FD FF ØØ 8D
.:  Ø698 FB Ø3 ØØ ØØ A2 Ø8 DD DE
.
.:  Ø6AØ FF ØØ DØ ØE 86 B4 8A ØA
.:  Ø6A8 AA BD E9 FF ØØ 48 BD E8
.:  Ø6BØ FF ØØ 48 6Ø CA 1Ø EA 4C
.:  Ø6B8 9A FA ØØ A2 Ø2 2C A2 ØØ
.:  Ø6CØ ØØ B4 FB DØ Ø8 B4 FC DØ
.:  Ø6C8 Ø2 E6 DE D6 FC D6 FB 6Ø
.:  Ø6DØ 2Ø 98 D7 C9 2Ø FØ F9 6Ø
.:  Ø6D8 A9 ØØ ØØ 8D ØØ ØØ Ø1 2Ø
.:  Ø6EØ 79 FA ØØ 2Ø 6B D7 2Ø 57
.:  Ø6E8 D7 9Ø Ø9 6Ø 2Ø 98 D7 2Ø
.
.:  Ø6FØ 54 D7 BØ DE AE Ø6 Ø2 9A
.:  Ø6F8 4C A4 D7 2Ø 31 D5 CA DØ
.:  Ø7ØØ FA 6Ø E6 FD DØ Ø2 E6 FE
.:  Ø7Ø8 6Ø A2 Ø2 B5 FA 48 BD ØA
.:  Ø71Ø Ø2 95 FA 68 9D ØA Ø2 CA
.:  Ø718 DØ F1 6Ø AD ØB Ø2 AC ØC
.:  Ø72Ø Ø2 4C CE FA ØØ A5 FD A4
.:  Ø728 FE 38 E5 FB 8D 1B Ø2 98
.:  Ø73Ø E5 FC A8 ØD 1B Ø2 6Ø 2Ø
.:  Ø738 81 FA ØØ 2Ø 44 D7 2Ø 92
.
.:  Ø74Ø FA ØØ 2Ø AF FA ØØ 2Ø 92
.:  Ø748 FA ØØ 2Ø CA FA ØØ 2Ø 44
.:  Ø75Ø D7 9Ø 15 A6 DE DØ 65 2Ø
.:  Ø758 C1 FA ØØ 9Ø 6Ø A1 FB 81
.:  Ø76Ø FD 2Ø A8 FA ØØ 2Ø 39 D5
.:  Ø768 DØ EB 2Ø C1 FA ØØ 18 AD
.:  Ø77Ø 1B Ø2 65 FD 85 FD 98 65
.:  Ø778 FE 85 FE 2Ø AF FA ØØ A6
.:  Ø78Ø DE DØ 3D A1 FB 81 FD 2Ø
.:  Ø788 C1 FA ØØ BØ 34 2Ø 65 FA
.
.:  Ø79Ø ØØ 2Ø 68 FA ØØ 4C 1B FB
.:  Ø798 ØØ 2Ø 31 FA ØØ 2Ø 44 D7
.:  Ø7AØ 2Ø 92 FA ØØ 2Ø 44 D7 2Ø
.:  Ø7A8 98 D7 2Ø 63 D7 9Ø 14 85
.:  Ø7BØ B5 A6 DE DØ 11 2Ø CA FA
.:  Ø7B8 ØØ 9Ø ØC A5 B5 81 FB 2Ø
.:  Ø7CØ 39 D5 DØ EE 4C 9A FA ØØ
.:  Ø7C8 4C BA D4 2Ø 81 FA ØØ 2Ø
.:  Ø7DØ 44 D7 2Ø 92 FA ØØ 2Ø 44
.:  Ø7D8 D7 2Ø 98 D7 A2 ØØ ØØ 2Ø
.
.:  Ø7EØ 98 D7 C9 27 DØ 14 2Ø 98
.:  Ø7E8 D7 9D 1Ø Ø2 E8 2Ø CF FF
.:  Ø7FØ C9 ØD FØ 22 EØ 2Ø DØ F1
.:  Ø7F8 FØ 1C 8E ØØ ØØ Ø1 2Ø 6B
.:  Ø8ØØ D7 9Ø C6 9D 1Ø Ø2 E8 2Ø
.:  Ø8Ø8 CF FF C9 ØD FØ Ø9 2Ø 63
.:  Ø81Ø D7 9Ø B6 EØ 2Ø DØ EC 86
.:  Ø818 B4 2Ø 34 D5 A2 ØØ ØØ AØ
.:  Ø82Ø ØØ ØØ B1 FB DD 1Ø Ø2 DØ
.:  Ø828 ØC C8 E8 E4 B4 DØ F3 2Ø
.
.:  Ø83Ø 17 D7 2Ø 31 D5 2Ø 39 D5
.:  Ø838 A6 DE DØ 92 2Ø CA FA ØØ
.:  Ø84Ø BØ DD 4C BA D4 2Ø 81 FA
.:  Ø848 ØØ 8D ØD Ø2 A5 FC 8D ØE
.:  Ø85Ø Ø2 A9 Ø4 A2 ØØ ØØ 8D Ø9
.:  Ø858 Ø2 8E ØA Ø2 A9 93 2Ø D2
.:  Ø86Ø FF A9 16 85 B5 2Ø Ø6 FC
.:  Ø868 ØØ 2Ø 64 FC ØØ 85 FB 84
.:  Ø87Ø FC C6 B5 DØ F2 A9 91 2Ø
.:  Ø878 D2 FF 4C BA D4 AØ 2C 2Ø
.
.:  Ø88Ø 79 D5 2Ø 17 D7 2Ø 31 D5
.:  Ø888 A2 ØØ ØØ A1 FB 2Ø 74 FC
.:  Ø89Ø ØØ 48 2Ø BB FC ØØ 68 2Ø
.:  Ø898 D3 FC ØØ A2 Ø6 EØ Ø3 DØ
.:  Ø8AØ 13 AC 1C Ø2 FØ ØE A5 FF
.:  Ø8A8 C9 E8 B1 FB BØ 1C 2Ø 5C
.:  Ø8BØ FC ØØ 88 DØ F2 Ø6 FF 9Ø
.:  Ø8B8 ØE BD 51 FF ØØ 2Ø 45 FD
.:  Ø8CØ ØØ BD 57 FF ØØ FØ Ø3 2Ø
.:  Ø8C8 45 FD ØØ CA DØ D4 6Ø 2Ø
.
.:  Ø8DØ 68 FC ØØ AA E8 DØ Ø1 C8
.:  Ø8D8 98 2Ø 5C FC ØØ 8A 86 B4
.:  Ø8EØ 2Ø 22 D7 A6 B4 6Ø AD 1C
.:  Ø8E8 Ø2 38 A4 FC AA 1Ø Ø1 88
.:  Ø8FØ 65 FB 9Ø Ø1 C8 6Ø A8 4A
.:  Ø8F8 9Ø ØB 4A BØ 17 C9 22 FØ
```

```
.: Ø9ØØ 13 29 Ø7 Ø9 8Ø 4A AA BD
.: Ø9Ø8 ØØ FF ØØ BØ Ø4 4A 4A 4A
.: Ø91Ø 4A 29 ØF DØ Ø4 AØ 8Ø A9
.: Ø918 ØØ ØØ AA BD 44 FF ØØ 85
.
.: Ø92Ø FF 29 Ø3 8D 1C Ø2 98 29
.: Ø928 8F AA 98 AØ Ø3 EØ 8A FØ
.: Ø93Ø ØB 4A 9Ø Ø8 4A 4A Ø9 2Ø
.: Ø938 88 DØ FA C8 88 DØ F2 6Ø
.: Ø94Ø B1 FB 2Ø 5C FC ØØ A2 Ø1
.: Ø948 2Ø A1 FA ØØ CC 1C Ø2 C8
.: Ø95Ø 9Ø FØ A2 Ø3 CC Ø9 Ø2 9Ø
.: Ø958 FØ 6Ø A8 B9 5E FF ØØ 8D
.: Ø96Ø ØB Ø2 B9 9E FF ØØ 8D ØC
.: Ø968 Ø2 A9 ØØ ØØ AØ Ø5 ØE ØC
.
.: Ø97Ø Ø2 2E ØB Ø2 2A 88 DØ F6
.: Ø978 69 3F 2Ø D2 FF CA DØ EA
.: Ø98Ø 4C 31 D5 2Ø 81 FA ØØ 2Ø
.: Ø988 44 D7 2Ø 92 FA ØØ 2Ø 44
.: Ø99Ø D7 A9 Ø4 A2 ØØ ØØ 8D Ø9
.: Ø998 Ø2 8E ØA Ø2 2Ø 34 D5 2Ø
.: Ø9AØ ØB FC ØØ 2Ø 64 FC ØØ 85
.: Ø9A8 FB 84 FC 2Ø 35 F3 FØ Ø5
.: Ø9BØ 2Ø CA FA ØØ BØ E9 4C BA
.: Ø9B8 D4 2Ø 81 FA ØØ A9 Ø3 85
.
.: Ø9CØ B5 2Ø 98 D7 2Ø ØB D5 DØ
.: Ø9C8 F8 AD ØD Ø2 85 FB AD ØE
.: Ø9DØ Ø2 85 FC 4C E7 FB ØØ CD
.: Ø9D8 ØA Ø2 FØ Ø3 2Ø D2 FF 6Ø
.: Ø9EØ A9 Ø3 A2 24 8D Ø9 Ø2 8E
.: Ø9E8 ØA Ø2 2Ø 34 D5 78 AD FA
.: Ø9FØ FF ØØ 85 9Ø AD FB FF ØØ
.: Ø9F8 85 91 A9 AØ 8D 4E E8 CE
.: ØAØØ 13 E8 A9 2E 8D 48 E8 A9
.: ØAØ8 ØØ ØØ 8D 49 E8 AE Ø6 Ø2
.
.: ØA1Ø 9A 4C 55 D6 2Ø CØ FC 68
.: ØA18 8D Ø5 Ø2 68 8D Ø4 Ø2 68
.: ØA2Ø 8D Ø3 Ø2 68 8D Ø2 Ø2 68
.: ØA28 8D Ø1 Ø2 68 8D ØØ ØØ Ø2
.: ØA3Ø BA 8E Ø6 Ø2 58 2Ø 34 D5
.: ØA38 2Ø 23 D5 85 B5 AØ ØØ ØØ
.: ØA4Ø 2Ø FE D4 2Ø 31 D5 AD ØØ
.: ØA48 ØØ Ø2 85 FC AD Ø1 Ø2 85
.: ØA5Ø FB 2Ø 17 D7 2Ø ØE FC ØØ
.: ØA58 2Ø 35 F3 C9 F7 FØ F9 2Ø
.
.: ØA6Ø 35 F3 DØ Ø3 4C BA D4 C9
.: ØA68 FF FØ F4 4C 5B FD ØØ 2Ø
.: ØA7Ø 81 FA ØØ 2Ø 44 D7 8E 11
.: ØA78 Ø2 A2 Ø3 2Ø 79 FA ØØ 48
.: ØA8Ø CA DØ F9 A2 Ø3 68 38 E9
.: ØA88 3F AØ Ø5 4A 6E 11 Ø2 6E
.: ØA9Ø 1Ø Ø2 88 DØ F6 CA DØ ED
.: ØA98 A2 Ø2 2Ø CF FF C9 ØD FØ
.: ØAAØ 1E C9 2Ø FØ F5 2Ø F7 FE
.: ØAA8 ØØ BØ ØF 2Ø 78 D7 A4 FB
.
.: ØABØ 84 FC 85 FB A9 3Ø 9D 1Ø
.: ØAB8 Ø2 E8 9D 1Ø Ø2 E8 DØ DB
.: ØACØ 8E ØB Ø2 A2 ØØ ØØ 86 DE
.: ØAC8 FØ Ø4 E6 DE FØ 7B A2 ØØ
.: ØADØ ØØ 86 B5 A5 DE 2Ø 74 FC
.: ØAD8 ØØ A6 FF 8E ØC Ø2 AA BC
.: ØAEØ 5E FF ØØ BD 9E FF ØØ 2Ø
.: ØAE8 EØ FE ØØ DØ E2 A2 Ø6 EØ
.: ØAFØ Ø3 DØ 1A AC 1C Ø2 FØ 15
.: ØAF8 A5 FF C9 E8 A9 3Ø BØ 21
.
.: ØBØØ 2Ø E6 FE ØØ DØ CA 2Ø E8
.: ØBØ8 FE ØØ DØ C5 88 DØ EB Ø6
.: ØB1Ø FF 9Ø ØB BC 57 FF ØØ BD
.: ØB18 51 FF ØØ 2Ø EØ FE ØØ DØ
.: ØB2Ø B3 CA DØ DØ FØ ØA 2Ø DF
.: ØB28 FE ØØ DØ A9 2Ø DF FE ØØ
.: ØB3Ø DØ A4 AD ØB Ø2 C5 B5 DØ
.: ØB38 9D 2Ø 44 D7 AC 1C Ø2 FØ
.: ØB4Ø 2F AD ØC Ø2 C9 9D DØ 2Ø
.: ØB48 2Ø CA FA ØØ 9Ø ØB 98 DØ
.
.: ØB5Ø Ø5 AE 1B Ø2 1Ø ØB 4C 9A
.: ØB58 FA ØØ C8 DØ FA AE 1B Ø2
.: ØB6Ø 1Ø F5 CA CA 8A AC 1C Ø2
.: ØB68 DØ Ø3 B9 FC ØØ ØØ 91 FB
.: ØB7Ø 88 DØ F8 A5 DE 91 FB 2Ø
.: ØB78 64 FC ØØ 85 FB 84 FC AØ
.: ØB8Ø 41 2Ø 79 D5 2Ø 17 D7 2Ø
.: ØB88 31 D5 4C D8 FD ØØ A8 2Ø
.: ØB9Ø E6 FE ØØ DØ 11 98 FØ ØE
.: ØB98 86 B4 A6 B5 DD 1Ø Ø2 Ø8
.
.: ØBAØ E8 86 B5 A6 B4 28 6Ø C9
.: ØBA8 3Ø 9Ø Ø3 C9 47 6Ø 38 6Ø
.: ØBBØ 4Ø Ø2 45 Ø3 DØ Ø8 4Ø Ø9
.: ØBB8 3Ø 22 45 33 DØ Ø8 4Ø Ø9
.: ØBCØ 4Ø Ø2 45 33 DØ Ø8 4Ø Ø9
.: ØBC8 4Ø Ø2 45 B3 DØ Ø8 4Ø Ø9
.: ØBDØ ØØ ØØ 22 44 33 DØ 8C 44
.: ØBD8 ØØ ØØ 11 22 44 33 DØ 8C
.: ØBEØ 44 9A 1Ø 22 44 33 DØ Ø8
.: ØBE8 4Ø Ø9 1Ø 22 44 33 DØ Ø8
.
.: ØBFØ 4Ø Ø9 62 13 78 A9 ØØ ØØ
.: ØBF8 21 81 82 ØØ ØØ ØØ ØØ 59
.: ØCØØ 4D 91 92 86 4A 85 9D 2C
.: ØCØ8 29 2C 23 28 24 59 ØØ ØØ
.: ØC1Ø 58 24 24 ØØ ØØ 1C 8A 1C
.: ØC18 23 5D 8B 1B A1 9D 8A 1D
.: ØC2Ø 23 9D 8B 1D A1 ØØ ØØ 29
.: ØC28 19 AE 69 A8 19 23 24 53
```

```
.:  ØC3Ø 1B 23 24 53 19 A1 ØØ ØØ
.:  ØC38 1A 5B 5B A5 69 24 24 AE
.:  ØC4Ø AE A8 AD 29 ØØ ØØ 7C ØØ
.:  ØC48 ØØ 15 9C 6D 9C A5 69 29
.:  ØC5Ø 53 84 13 34 11 A5 69 23
.:  ØC58 AØ D8 62 5A 48 26 62 94
.:  ØC6Ø 88 54 44 C8 54 68 44 E8
.:  ØC68 94 ØØ ØØ B4 Ø8 84 74 B4
.:  ØC7Ø 28 6E 74 F4 CC 4A 72 F2
.:  ØC78 A4 8A ØØ ØØ AA A2 A2 74
.:  ØC8Ø 74 74 72 44 68 B2 32 B2
.:  ØC88 ØØ ØØ 22 ØØ ØØ 1A 1A 26
.:  ØC9Ø 26 72 72 88 C8 C4 CA 26
.:  ØC98 48 44 44 A2 C8 54 46 48
.:  ØCAØ 44 5Ø 2C 41 49 4E ØØ ØØ
.:  ØCA8 DB FA ØØ 3Ø FB ØØ 5E FB
.:  ØCBØ ØØ D1 FB ØØ F8 FC ØØ 28
.:  ØCB8 FD ØØ D4 FD ØØ 4D FD ØØ
.:  ØCCØ B9 D4 7F FD ØØ 4A FA ØØ
.:  ØCC8 33 FA ØØ AA AA AA AA AA
```

SUPERMON 3.0 Progrm 3.

```
.:  Ø6DØ 2Ø EB E7 C9 2Ø FØ F9 6Ø
.:  Ø6EØ 79 FA ØØ 2Ø BE E7 2Ø AA
.:  Ø6E8 E7 9Ø Ø9 6Ø 2Ø EB E7 2Ø
.:  Ø6FØ A7 E7 BØ DE AE Ø6 Ø2 9A
.:  Ø6F8 4C F7 E7 2Ø CD FD CA DØ
.:  Ø738 81 FA ØØ 2Ø 97 E7 2Ø 92
.:  Ø748 FA ØØ 2Ø CA FA ØØ 2Ø 97
.:  Ø75Ø E7 9Ø 15 A6 DE DØ 65 2Ø
.:  Ø76Ø FD 2Ø A8 FA ØØ 2Ø D5 FD
.:  Ø798 ØØ 2Ø 81 FA ØØ 2Ø 97 E7
.:  Ø7AØ 2Ø 92 FA ØØ 2Ø 97 E7 2Ø
.:  Ø7A8 EB E7 2Ø B6 E7 9Ø 14 85
.:  Ø7CØ D5 FD DØ EE 4C 9A FA ØØ
.:  Ø7C8 4C 56 FD 2Ø 81 FA ØØ 2Ø
.:  Ø7DØ 97 E7 2Ø 92 FA ØØ 2Ø 97
.:  Ø7D8 E7 2Ø EB E7 A2 ØØ ØØ 2Ø
.:  Ø7EØ EB E7 C9 27 DØ 14 2Ø EB
.:  Ø7E8 E7 9D 1Ø Ø2 E8 2Ø CF FF
.:  Ø7F8 FØ 1C 8E ØØ ØØ Ø1 2Ø BE
.:  Ø8ØØ E7 9Ø C6 9D 1Ø Ø2 E8 2Ø
.:  Ø8Ø8 CF FF C9 ØD FØ Ø9 2Ø B6
.:  Ø81Ø E7 9Ø B6 EØ 2Ø DØ EC 86
.:  Ø818 B4 2Ø DØ FD A2 ØØ ØØ AØ
.:  Ø83Ø 6A E7 2Ø CD FD 2Ø D5 FD
.:  Ø84Ø BØ DD 4C 56 FD 2Ø 81 FA
.:  Ø878 D2 FF 4C 56 FD AØ 2C 2Ø
.:  Ø88Ø 15 FE 2Ø 6A E7 2Ø CD FD
.:  Ø8EØ 2Ø 75 E7 A6 B4 6Ø AD 1C
.:  Ø98Ø 4C CD FD 2Ø 81 FA ØØ 2Ø
.:  Ø988 97 E7 2Ø 92 FA ØØ 2Ø 97
.:  Ø99Ø E7 A9 Ø4 A2 ØØ ØØ 8D Ø9
.:  Ø998 Ø2 8E ØA Ø2 2( DØ FD 2Ø
.:  Ø9A8 FB 84 FC 2Ø Ø1 F3 FØ Ø5
.:  Ø9BØ 2Ø CA FA ØØ BØ E9 4C 56
.:  Ø9B8 FD 2Ø 81 FA ØØ A9 Ø3 85
.:  Ø9CØ B5 2Ø EB E7 2Ø A7 FD DØ
.:  Ø9E8 ØA Ø2 2Ø DØ FD 78 AD FA
.:  ØA1Ø 9A 4C F1 FE 2Ø 7B FC 68
.:  ØA3Ø BA 8E Ø6 Ø2 58 2Ø DØ FD
.:  ØA38 2Ø BF FD 85 B5 AØ ØØ ØØ
.:  ØA4Ø 2Ø 9A FD 2Ø CD FD AD ØØ
.:  ØA5Ø FB 2Ø 6A E7 2Ø ØE FC ØØ
.:  ØA58 2Ø Ø1 F3 C9 F7 FØ F9 2Ø
.:  ØA6Ø Ø1 F3 DØ Ø3 4C 56 FD C9
.:  ØA7Ø 81 FA ØØ 2Ø 97 E7 8E 11
.:  ØAA8 ØØ BØ ØF 2Ø CB E7 A4 FB
.:  ØB38 9D 2Ø 97 E7 AC 1C Ø2 FØ
.:  ØB8Ø 41 2Ø 15 FE 2Ø 6A E7 2Ø
.:  ØB88 CD FD 4C D8 FD ØØ A8 2Ø
.:  ØCCØ 55 FD 7F FD ØØ 4A FA ØØ
```

SUPERMON Program 4.

```
100 REM SUPERMON 4 CHECKSUM
110 DATA7331,12186,10071,10387,1082
    9,9175,10314,9823,9715,871
    4,8852
120 DATA8850,9748,7754,10247,10423,
    10948,10075,6093,5492,7805
    :S=1536
130 FORB=1TO21:READX:FORI=STOS+79:N
    =PEEK(I):Y=Y+N
140 NEXTI:IFY<>XTHENPRINT"ERROR IN 
    BLOCK #"B:GOTO160
150 PRINT"BLOCK #"B" IS CORRECT"
160 S=I:Y=0:NEXTB:PRINT"CHECK THE F
    INAL, SHORT BLOCK BY HAND"
```

SUPERMON Program 5.

```
100 REM SUPERMON 3 CHECKSUM
110 DATA7331,12186,10467,10880,1112
    4,10005,10906,10196,9951,8
    813
120 DATA8852,9329,10239,8457,10334,
    10423,11047,10311,6093,549
    2,7805:S=1536
```

©

PET To PET Communication Over The User Port

John Winn
Department of Chemistry
University of California at Berkeley

If you (or you and a friend) have access to two PETs, you may have wanted to connect the two together and transfer data from one to the other. The built-in IEEE bus is not suitable, since each PET is a bus controller and the rules allow only one controller on the bus. You could buy any of a number of attachments for serial, parallel or modem input/output, but the simplest method is to interconnect the PET's through the built-in parallel user port. Here's how it's done, using fairly simple BASIC and twelve wires.

First, what hardware is required? The user port connections are on the *bottom* row of the PC output edge connector. Looking at the rear of the PET, these are labelled A through N with keying slots sawed between A and B and between L and M. A and N are ground connections. C through L are the eight parallel data lines. Each will correspond, in effect, to one of the eight bits in a memory byte. Connection B is called "CA1"; it will be used to signal the presence of data to be read by the receiving PET. Connection M, called "CB2," will control (signal) CA1 on the other PET. (How this is done will be clearer later on.) To connect the two ports together, use two edge connector plugs, wiring A to A, N to N, C through L to C through L, but wire B on one connector to M on the other and vice versa (i.e. CA1 on one to CB2 on the other). The total length of the cable should not be more than about 20 feet. (Longer distances would require external "line drivers" to keep the signal from degrading.)

To control these dozen wires, various PEEKs and POKEs are used. One PET will transmit, and the other will receive at any one time, although each can do both. To send one byte, the transmitter will first activate the eight data lines. Then it will signal the receiver that the byte is set to be read. The receiver will read the byte and signal back to the transmitter that it has done so and is ready for the next.

Suppose we want to send one character from one PET to the other. Program 1 gives the program for the transmitter and Program 2, for the receiver. Line 20 in each program shows how the direction of data transfer is controlled. Line 40 of the transmitter program shows how one byte (ASC(A$)) is placed on the data lines. Meanwhile, the receiver is stuck on its line 40, waiting for bit two of memory location 59469 to be a one instead of a zero. This transition will signal the receiver that it can read the data lines. The signal is sent (from CB2 of the transmitter to CA1 of the receiver) by lines 60 and 70 of the transmitter program. Line 60 forces the three most significant bits of memory location 59468 to be ones. (The other bits are unchanged.) Line 70 forces the third most significant bit back to zero, forcing the first two to be ones and leaving the low order five bits (which are used for other things) as they were. This sequence turns CB2 on, then off.

stuck on line 90 waiting for the receiver to signal back that it has read the data. The receiver signals with lines 70 and 80. It then prints the received character on its screen and goes after another byte. The transmitter will get the signal and ask for another character to send, and the process will repeat.

Most applications will involve the transfer of more than just one character. Transmitting a whole string of many characters or a floating point number requires more elaborate programs, but they will be based on these simple versions. To send a string, the length of the string must be sent first, and then the string can be sent character by character. To send a floating point number, the simplest technique seems to be to use one BASIC variable at a known location in memory as an intermediary buffer, as is done in the programs described below.

You *Could* WAIT

Two other concerns arise. The first is the initial synchronization of the data transfer. This is perhaps best taken care of by a one byte "preamble" sent at the beginning of the program just to clean out any unsuspected data or transfer signals. The second concern is the ability to interrupt the transmission gracefully should something go wrong. (Along this same line, it is worth pointing out that line 90 of Program 1 and line 40 of Program 2 could be written using WAIT statements. But, since WAITs are not interruptable, except by pulling the plug, this is a bit dangerous.) The easiest way to interrupt a program without stopping it directly is to use the SHIFT key in the way described below.

Programs 3 and 4 give more elaborate pro-

grams which send a string of arbitrary length and arbitrary number of random floating point numbers. They both use the SHIFT key to signal an interrupt. (With Original ROM's, location 516 is zero if the SHIFT key is up, and one if it is down. With Upgrade ROM's, it's location 152.) The transmitter sends a preamble — one "%" — to guarantee synchronization. The character is arbitrary, but it should be as unique (or obscure) as possible.

The floating point buffer variable, called QQ in each program, *must* be the first defined variable of the program. This is so its location in memory can be found easily. At the beginning of variable data storage, one finds two bytes for the two character name of the first variable followed by five bytes representing the floating point number itself. Variables start at memory location 256*PEEK(43) + PEEK(42) in Upgrade ROM's (256*PEEK(125) + PEEK(124) in Original ROM's); hence, variable SQ in each program gives the location, two bytes along from the start, for QQ's five data bytes.

Data are transmitted (or received) in subroutines 1000 and 2000. Starting at 1000 is the subroutine for transmitting or receiving the five bytes of QQ. Transmitting or receiving only one byte (variable D in the program) is done by the subroutine starting at line 2000. Note that this subroutine is called by the first one.

Interruption requires that you hold down the SHIFT key until the program can branch to line 3000. Both the transmitter and the receiver have to be interrupted separately, but either can be interrupted first.

These programs illustrate the main techniques needed for more useful and interesting applications. For many games ("Battleship" comes to mind), the transfer rate of the BASIC code is fast enough, around 10 bytes per second or so.

ML For Fast Transfer

For much greater speed, machine language code is needed. Program 5 is a machine language version of the BASIC code in Programs 3 and 4, implemented in a slightly different way. Line 10 sets up a variable, D%, for receiving single bytes. It must be the first variable defined in the program, and the PEEKs must be changed to 125 and 124 for Original ROM's. The POKE 2,3 statement sets part of the linkage for the USR function. Line 20 POKEs the machine language code into the second cassette buffer. Line 30 puts the address of the low-order byte of D% into this code and sets D% back to zero. (Note: POKEX, PEEK(Y) does not work on Original ROM's. That's why line 30 is written the way it is.) The DATA statements contain the machine language for Upgrade ROM's. For Original ROM's, change the two occurrences (lines 1035 and 1057) of 94 both to 176. They locate the floating point accumulator used by USR.

To set the program into the transmit mode (line 100), POKE 1,91 first to complete the USR linkage for transmission. Next, send a one byte preamble ("%" is used here again) to insure synchronization. To send individual bytes (line 200), POKE them into location 832 and call SYS826. To transmit a floating point number (line 300), pass the number (or variable) as the argument of USR. Since USR has to be set equal to something, it can safely be set equal to the variable being passed or to any other variable which you want to equate to the variable being passed.

Of course, when one program is set up to transmit, the other must be set up to receive. First, (line 400), POKE 1,139 to complete the USR linkage for reception. Next, look for the preamble and warn yourself (line 440) if it was not received as expected. The FOR-NEXT loop in 420–430 should never go past I = 2. To receive individual bytes (line 500), call SYS873, and find the byte in the variable D%. To receive a floating point number (line 600), equate the variable you wish to input to USR. The argument to USR is not important here, nor is it disturbed if a variable is used.

In most programs, lines 100–120 and 400–440 would best be made subroutines which could be called to switch the program from one mode to the other at will. The main disadvantage of this program is that it cannot be easily interrupted. Data synchronization between the two PETs must be exact or one will finish first, leaving the other hung up. One or more direct SYS826 or SYS873 commands from the un-hung PET will, eventually, clear the other. (Which SYS you use will depend on the state– transmitter or receiver– of the hungup PET.

Transmission Rate

The data rate is quite good. Sending 2000 numbers in a command FOR I = 1 TO 2000: X = USR(I): NEXT takes about 8.6 seconds. That works out to (2000x6)/8.6 = 1400 bytes per second. In this test, the receiver just read the numbers, but did nothing with them. When the receiver stuck the numbers into an array, the time went up to 12.5 seconds.

Finally, if you want to locate the machine language somewhere other than 826 to 917 (or $033A to $0395), the only six numbers in DATA which change are the thirty-ninth (64), fortieth (3), forty-second (58), forty-third (3), eighty-fifth (69), and eighty-sixth (3). These, in pairs, are low and high order absolute address bytes (i.e. 64 + 3*256 = 832). They will have to be changed along with the various POKE locations in BASIC (and the numbers POKEd into locations 1 and 2) if the program is relocated. [*It is suggested that 4.0 users move the routine to avoid DOS usage of the bottom of this buffer. — Ed.*]

A Wooden Computer?

Not from Commodore!

So why should the desk look like wood? A pleasant cream and charcoal trimmed desk looks so much better with Commodore systems. One look and you'll see. Interlink desks are right. By design.

The specifications only confirm the obvious:

- Cream and charcoal color beautifully matches the Commodore hardware and blends with your decor.
- An ideal 710 mm (28") keyboard height yet no bumping knees because a clever cutout recesses the computer into the desktop.
- High pressure laminate on both sides of a solid core for lasting beauty and strength.
- Electrostatically applied baked enamel finish on welded steel legs—no cheap lacquer job here.
- T-molding and rounded corners make a handsome finish on a durable edge that won't chip.
- Knocked down for safe, inexpensive shipment.
- Patented slip joints for quick easy assembly.
- Leveling glides for uneven floors.
- Room enough for a Commodore printer on the desk, yet fits into nearly any den or office niche—H: 660 mm (26") W: 1170 mm (46") D: 660 mm (26").
- Matching printer stand available with slot for bottom feeding.

Price: $299

In short, as Commodore dealers, we won't settle for anything that looks good only in the catalog! Our customers won't let us. They don't buy pictures. And neither should you. This is why **we will let you use one of our desks for a week** and then decide. If for any reason you don't like it, just return it in good condition for a cheerful refund.

If your Commodore dealer doesn't carry our desks yet, send a check for $299 and we will ship your desk freight paid!

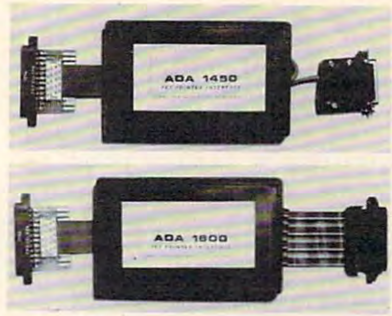
Name____________________

Address____________________

City__________ **St**_____ **Zip**______

Interlink, Inc., Box 134, Berrien Springs, MI 49103
Master Charge and Visa welcome. Call our order line:
616-473-3103

CBM/PET INTERFACES



RS-232 SERIAL PRINTER INTERFACE – addressable – baud rates to 9600 – switch selectable upper/lower, lower/upper case – works with WORDPRO, BASIC and other software – includes case and power supply.
MODEL – ADA1450 149.00

CENTRONICS/NEC PARALLEL INTERFACE - addressable – high speed – switch selectable upper/lower, lower/upper case – works with WORDPRO, BASIC and other software – has Centronics 36 pin ribbon connector at end of cable.
MODEL – ADA1600 129.00

CENTRONICS 730/737 PARALLEL INTERFACE – as above but with Centronics card edge connector at end of cable.
MODEL – ADA730 129.00

COMMUNICATIONS INTERFACE WITH SERIAL AND PARALLEL PORTS – addressable – software driven – true ASCII conversion – selectable reversal of upper-lower case – baud rates to 9600 – half or full duplex – X-ON, X-OFF – selectable carriage return delay – 32 character buffer – centronics compatible — much more.
MODEL – SADI 295.00

ANALOG TO DIGITAL CONVERTER – 16 channels – 0 to 5.12 volt input voltage range – resolution is 20 millivolts per count – conversion time is less than 100 microseconds per channel.
MODEL – PETSET1 295.00

REMOTE CONTROLLER WITH CLOCK/CALENDAR – controls up to 256 devices using the BSR X10 remote control receivers – 8 digital inputs, TTL levels or switch closure — 8 digital outputs, TTL levels.
MODEL – PETSET2 295.00

All prices are in US dollars for 120VAC.
Prices on 220 VAC slightly higher.
Allow $5.00 shipping & handling, foreign orders add 10% for AIR postage.
Connecticut residents add 7½% sales tax.
All prices and specifications subject to change without notice.
Our 30 day money back trial period applies.
MASTER CHARGE/VISA accepted.

MENTION THIS MAGAZINE WITH YOUR ORDER AND DEDUCT 5% FROM TOTAL.

IN CANADA order from: Batteries Included, Ltd., 71 McCaul Street, F6 Toronto, Canada M5T2X1, (416)596-1405.
IN THE USA order from your local dealer or direct: Connecticut microComputer, Inc., 34 Del Mar Drive, Brookfield, CT 06804, (203)775-4595.

Dealer inquiries invited.

CmC

Connecticut microComputer, Inc.
34 Del Mar Drive, Brookfield, CT 06804
203 775-4595 TWX: 710 456-0052

Program 1.

```
 10 REM **** SIMPLE TRANSMITTER
 20 POKE59459,255 :REM SET DATA
               LINES FOR OUTPUT
 30 INPUT"ENTER A CHARACTER";A$
 40 POKE59457,ASC(A$) :REM OUTPUT
                     CHARACTER
 50 REM NEXT 2 LINES SIGNAL THE RECEIVER
                        TO READ DATA
 60 POKE59468,PEEK(59468) OR 224
 70 POKE59468,PEEK(59468) AND 31 OR 192
 80 REM WAIT FOR RECEIVER TO SIGNAL BACK
 90 IF(PEEK(59469) AND 2)<>2 THEN 90
 100 GOTO30
READY.
```


Program 3.

```
 10 REM **** ELABORATE TRANSMITTER (UPGRADE ROM VERSION)
 20 QQ=0 :REM QQ MUST BE FIRST VARIABLE
 30 SQ=PEEK(42)+256*PEEK(43)+2 :REM ADDRESS OF FIRST QQ DATA BYTE
 40 SH=152 :REM ADDRESS OF 'SHIFT' KEY FLAG
 50 POKE59459,255 :REM SET DATA LINES FOR OUTPUT
 60 REM SEND SYNCHRONIZATION PREAMBLE
 70 D=ASC("%") : GOSUB2000
 80 PRINT"READY TO TRANSMIT":PRINT"USE 'SHIFT' KEY TO INTERRUPT"
 90 INPUT"ENTER A STRING";A$
 100 QQ=LEN(A$) : GOSUB1000 :REM TRANSMIT LEN(A$)
 110 FORI=1TOQQ
 120 D=ASC(MID$(A$,I,1))
 130 GOSUB2000 :REM SEND STRING 1 CHARACTER AT A TIME
 140 NEXT
 150 INPUT"HOW MANY RANDOM NUMBERS";N
 160 QQ=N : GOSUB1000 :REM TRANSMIT N
 170 FORI=1TON
 180 QQ=RND(1)
 190 GOSUB1000 :REM TRANSMIT EACH RANDOM NUMBER
 200 NEXT
 999 END
 1000 REM SUBROUTINE FOR FLOATING POINT TRANSMISSION
 1010 FORIJ=0TO4
 1020 D=PEEK(SQ+IJ)
 1030 GOSUB2000 :REM SEND QQ BYTE BY BYTE
 1040 NEXT
 1050 RETURN
 2000 REM SUBROUTINE FOR BYTE TRANSMISSION
 2010 POKE59459,D :REM OUTPUT BYTE
 2020 REM SIGNAL 'DATA READY'
 2030 POKE59468,PEEK(59468) OR 224
 2040 POKE59468,PEEK(59468) AND 31 OR 192
 2050 REM WAIT FOR RECEPTION AND ALLOW INTERRUPT
 2060 IF((PEEK(59469) AND 2)<>2) AND (PEEK(SH)<>1) THEN2060
 2070 IFPEEK(SH)=1THEN3000 :REM INTERRUPT
 2080 RETURN
 3000 PRINT"INTERRUPTED"
 3010 GOTO999 :REM END IF INTERRUPTED
READY.
```

SM-KIT

for Commodore Computers

A Programming Productivity Tool

ONLY $40

The SM-KIT is a collection of machine language firmware programming and test aids for BASIC programmers. SM-KIT is a 4K ROM (twice the normal capacity) which you simply insert in a single ROM socket on any BASIC 4 CBM/PET—either 80 column or 40 column. Includes both programming aids and disk handling commands.

ERROR DETECTION: the SM-KIT automatically indicates the erroneous line and statement for any BASIC program error.
LINE NUMBERING: the SM-KIT automatically numbers BASIC statements until you turn the function off.
SCREEN OUTPUT: the commands FIND, DUMP, TRACE and DIRECTORY display on the CRT while you hold the RETURN key (display pauses when the key is released). Continuous output is selected with shift-lock.
OUTPUT CONTROL to DISK or PRINTER: in addition to displaying on the CRT, you can direct output to either disk or printer.
HARDCOPY: allows screen displays to be either printed or stored on disk.
FIND: searches all or any part of a program for text or command strings or variable names. Either exact search or wild card search supported.
RENUMBER: the SM-KIT can renumber all or any part of a program. The selective renumbering allows you to move blocks of code within your program.
VARIABLE DUMP: displays the contents of floating point, integer, and string variables (both simple and array). Can display all variables or any selected variables.
TRACE: SM-KIT can trace program execution either continuously or step by step starting with any line number. Selected program variables can be displayed while tracing.
DISK COMMANDS: as in DOS Support (Universal Wedge), the "shorthand" versions of disk commands may be used for displaying disk directory, initializing, copying, scratching files, load and run, etc.
LOAD: SM-KIT can load all or part of BASIC or machine language programs. It can append to a program in memory, overwrite any part of a program, load starting with any absolute memory location, and load without changing variable pointers.
MERGE: allows merging all or any part of a program on disk with a program in memory.
SAVE and VERIFY: SM-KIT provides one step program save and verification. It also allows you to save any part of a program, or any address range.

A 4K ROM with both programming and disk handling aids.

Developed by (and available in Europe from) SM Softwareverbund-Microcomputer GmbH, Scherbaumstrasse 29, 8000 Munchen 83, Germany

252 Bethlehem Pike
Colmar, PA 18915

215-822-7727 **A B Computers**

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

PIE-C

PET/CBM* IEEE-488 TO PARALLEL PRINTERS By LemData Products

P.I.E.-C MEANS—Professional design, Indispensible features, Excellent quality and Cost effectiveness. You can't buy a better parallel interface for your PET/CBM.

Our P.I.E.-C will interface your PET/CBM through the IEEE-488 bus to . . . . . .

the NEC Spinwriter, the C. Itoh Starwriter, printers by Centronics, Epson, Anadex, Escon Products, the Paper Tigers by IDS, the MIPLOT by Watanabe, the DIP printers, the AJ-841, the OKIDATA printers, plus ALL OTHER parallel ASCII printers.

Assembled with custom case, CBM-TO-ASCII code converter and appropriate cable, the **P.I.E.-C** is only $129.95 (+ $5 S&H). Md. Res. +5% tax. Specify printer and CBM models.

LemData Products, P.O. Box 1080, Columbia, Md. 21044 Phone (301) 730-3257
*PET/CBM are trademarks of Commodore Business Machines

(Dealer Inquiries Invited)

Program 2.

```
 10 REM **** SIMPLE RECEIVER
 20 POKE59459,0 :REM SET DATA
                 LINES FOR INPUT
 30 REM WAIT FOR DATA TO BE SENT
 40 IF(PEEK(59469) AND 2)<>2 THEN40
 50 D=PEEK(59457) :REM READ DATA
 60 REM NEXT 2 LINES SIGNAL
       THE TRANSMITTER "DATA READ"
 70 POKE59468,PEEK(59468) OR 224
 80 POKE59468,PEEK(59468) AND 31 OR 192
 90 PRINT CHR$(D) :REM PRINT THE
                   RECEIVED CHARACTER
 100 GOTO30
READY.
```


Program 4.

```
 10 REM **** ELABORATE RECEIVER (UPGRADE ROM VERSION)
 20 QQ=0 :REM QQ MUST BE FIRST VARIABLE
 30 SQ=PEEK(42)+256*PEEK(43)+2 :REM ADDRESS OF FIRST QQ DATA BYTE
 40 SH=152 :REM ADDRESS OF 'SHIFT' KEY FLAG
 50 POKE59459,0 :REM SET DATA LINES FOR INPUT
 60 REM LOOK FOR SYNCHRONIZATION PREAMBLE
 70 FORI=1TO3 : GOSUB2000 : IFD=ASC("%")THEN80
 72 NEXT
 74 PRINT"EXPECTED PREAMBLE NOT RECEIVED" : END
 80 PRINT"READY TO RECEIVE":PRINT"USE 'SHIFT' KEY TO INTERRUPT"
 90 GOSUB1000 :REM READ LENGTH OF TRANSMITTED STRING
 100 A$="" : FORI=1TOQQ : GOSUB2000 :REM INPUT ONE BYTE
 110 A$=A$+CHR$(D) :REM BUILD UP STRING
 120 NEXT
 130 PRINT"RECEIVED ";A$
 140 REM READ NUMBER OF RANDOM INPUTS TO EXPECT
 150 GOSUB1000 : N=QQ
 160 FORI=1TON
 170 GOSUB1000 : PRINT QQ :REM READ RANDOM NUMBERS
 180 NEXT
 999 END
 1000 REM SUBROUTINE FOR FLOATING POINT RECEPTION
 1010 FORIJ=0TO4
 1020 GOSUB2000 :REM READ QQ BYTE BY BYTE
 1030 POKESQ+IJ,D :REM BUILD NEW QQ
 1040 NEXT
 1050 RETURN
 2000 REM SUBROUTINE FOR BYTE RECEPTION
 2010 REM WAIT FOR DATA TO BE SENT AND ALLOW INTERRUPTION
 2020 IF((PEEK(59469) AND 2)<>2) AND (PEEK(SH)<>1) THEN2020
 2030 IFPEEK(SH)=1THEN3000 :REM INTERRUPT
 2040 D=PEEK(59457) :REM READ DATA BYTE
 2050 REM SIGNAL 'DATA RECEIVED'
 2060 POKE59468,PEEK(59468) OR 224
 2070 POKE59468,PEEK(59468) AND 31 OR 192
 2080 RETURN
 3000 PRINT"INTERRUPTED"
 3010 GOTO999 :REM END IF INTERRUPTED
READY.
```

Program 5.

```
 10 D%=256*PEEK(43)+PEEK(42)+3:POKE2,3
 20 FORI=826TO917:READJ:POKEI,J:NEXT
 30 I=PEEK(D%):POKE889,I:I=PEEK(D%-1):
    POKE890,I:D%=0
 100 REM ****** TRANSMIT
 110 POKE1,91:REM SET USR FOR TRANSMISSION
 120 POKE832,ASC("%"):SYS826:REM SEND
     PREAMBLE
 200 REM TRANSMIT A BYTE ("A")
 210 POKE832,ASC("A"):SYS826
 300 REM TRANSMIT THE NUMBER 1.23
 310 X=1.23:X=USR(X)
 400 REM ****** RECEIVE
 410 POKE1,139:REM SET USR FOR RECEPTION
 420 FORI=1TO3:SYS873:IFD%=ASC("%")THEN500
 430 NEXT:REM LOOK FOR PREAMBLE
 440 PRINT"PREAMBLE NOT RECEIVED":STOP
 500 REM RECEIVE A BYTE
 510 SYS873:A$=CHR$(D%):PRINTA$
 600 REM RECEIVE A NUMBER
 610 X=USR(0):PRINTX
 1000 REM DATA & CORRESPONDING MNEMONICS
 1010 REM
 1020 DATA 169,255    : TBYTE  LDA #$FF
 1021 DATA 141,67,232:         STA $E843
 1022 DATA 169,0      :        LDA #***
 1023 DATA 141,65,232:         STA $E841
 1024 DATA 173,76,232:         LDA $E84C
 1025 DATA 9,224      :        ORA #$E0
 1026 DATA 141,76,232:         STA $E84C
 1027 DATA 41,31      :        AND #$1F
 1028 DATA 9,192      :        ORA #$C0
 1029 DATA 141,76,232:         STA $E84C
 1030 DATA 173,77,232: TWAIT   LDA $E84D
 1031 DATA 41,2       :        AND #$02
 1032 DATA 240,249    :        BEQ TWAIT
 1033 DATA 96         :        RTS
 1034 DATA 162,5      :        LDX #$05
 1035 DATA 181,94:<--: TFLPT   LDA $5E,X
 1036 DATA 141,64,3   :        STA $0340
 1037 DATA 32,58,3    :        JSR TBYTE
 1038 DATA 202        :        DEX
 1039 DATA 16,245     :        BPL TFLPT
 1040 DATA 96         :        RTS
 1041 DATA 169,0      : RBYTE  LDA #$00
 1042 DATA 141,67,232:         STA $E843
 1043 DATA 173,77,232: RWAIT   LDA $E84D
 1044 DATA 41,2       :        AND #$02
 1045 DATA 240,249    :        BEQ RWAIT
 1046 DATA 174,65,232:         LDY $E84C
 1047 DATA 142,0,0    :        STY $****
 1048 DATA 173,76,232:         LDA $E84C
 1049 DATA 9,224      :        ORA #$E0
 1050 DATA 141,76,232:         STA $E84C
 1051 DATA 41,31      :        AND #$1F
 1052 DATA 9,192      :        ORA #$C0
 1053 DATA 141,76,232:         STA $E84C
 1054 DATA 96         :        RTS
 1055 DATA 162,5      :        LDX #$05
 1056 DATA 32,69,3    : RFLPT  JSR RBYTE
 1057 DATA 148,94:<--:         STY $5E,X
 1058 DATA 202        :        DEX
 1059 DATA 16,248     :        BPL RFLPT
 1060 DATA 96         :        RTS
READY.
```

©

Replacing The INPUT# Command

Jerry E. Dunmire
San Jose, CA

At last you have your PET and now you can keep track of all those magazine articles, recipes, addresses or whatever else you promised your spouse! At least that's how I felt, and I immediately sat down to write the programs.

If you have tried to write a program that uses the INPUT command, then you know the problems I encountered. The INPUT command will not accept commas, quotes, or colons and using the GET command to construct a string is very slow. Since a proper bibliography of magazine articles must contain quote marks, I was stuck with the GET command. There had to be a better way.

There is! Nothing says that all programs must be written in BASIC. I could write a machine language routine to replace the INPUT# command. The new routine would accept all characters. Replacing the INPUT# command would also solve the same problems I encountered when reading from the tape or disk.

There are three items that we need to know in order to write a new version of the INPUT# command: how strings are stored, where the string is located, and how to input characters. The *PET/CBM Personal Computer Guide* by Adam Osborne and Carroll S. Donahue provided the information on string storage. Raymond Diedrichs explained how to input from a file in his article "Pet File I/O in Machine Language" **COMPUTE!** #11.

Strings are stored at the top of the available memory. As each string is entered, it is added to the bottom of the list. In order to identify a particular string we must know where it begins and how long it is. The PET uses one byte to represent the length of the string, and two bytes to identify the address where the string begins. The particular format that identifies a string depends on whether the string is an element of an array or a simple variable.

A simple variable has the form shown in Figure 1. If the string is an element of an array, it would be identified as shown in Figure 2. We can disregard the information in the header of an array.

This is only part of the information we need to locate a string in memory. The location of the pointer to the string is still unknown. Must our routine search for the name of the particular string we wish to input? Well, it could, but there is an easier way. Locations $44 and $45 point to the last variable referenced. If that last variable were the string we wish to input, then these locations will point to the length of the string, and the next two locations will be the address where the string is stored. Figure 3 shows the relationship between locations $44, $45, variables, and strings.

Reading characters from a file is even easier than dealing with strings. If a file has been opened by a BASIC statement, the subroutine at $FFC6 will set the file up so we can read from it. Then the subroutine at $FFCF will input a character from that file. When we have all the characters we want, the default I/O devices should be restored.

Armed with this knowledge, I wrote two routines. The two routines are named READString and INPUTLine. They are located in the second cassette buffer. Both use locations $44 and $45 to locate the variable, so the last variable you reference before calling these routines must be a string.

READS inputs a fixed number of characters from file #1. The number of characters is determined by the length of the string referenced by locations $44, 45. As the characters are read in, they replace the characters that are already in the string. This routine will cause strange problems if locations $44, $45 point to a string with zero length. To prevent this occurrence, I use the following commands to call READS:

```
10 IF LEN(A$) THEN SYS(826)
```

If A$ has a zero length, READS will never be called. As you can see, the starting address of READS is 826 ($033A).

The version of READS shown in Program 1 reads one additional character after it has filled the referenced string. The file has a carriage return at the end of each string. To remove this extra character input, place NOP's ($EA) in locations $0361 through $0363.

INPUTL also uses file #1. A carriage return must mark the end of a string just like the INPUT# command. INPUTL will accept any character other than a carriage return. Up to 80 characters can be input. If more than 80 characters are input, the ST variable will be set to a value of –1.

INPUTL works more like INPUT# than READS does. As the individual characters of a string are input, they are placed in an input buffer. Only after the string has been terminated with a carriage return is it transferred to the string storage area and assigned to the variable pointed to by $44, $45. The string is copied from the input buffer to

just below the string storage area. Then the pointer to the beginning of the string storage area is adjusted to account for the new string.

I use the following line to call INPUTL, but you can use any function that leaves locations $44, $45 pointing to the variable you wish to input.

10 A$ = " " : SYS(872)

As you can see, the starting address for INPUTL is 872. As with READS, if the last variable you referenced were not a string then the results are almost unpredictable and certainly bad.

You can change the file number used by these programs to suit your needs. Simply POKE the number of a file you have opened into location 827 for READLINE and 873 for INPUTSTRING.

INPUTL and READS will work with BASIC 3.0 or BASIC 4.0. If you need to use them with BASIC 1.0 then you will have to adjust all of the references to memory locations less than $0400 (1024 decimal).

INPUT# is still the fastest way to input a string. However, both INPUTL and READS are at least three to four times faster than using GET# commands. If you are short on memory, using the GET# command will be exceedingly slow since it will cause the garbage collection routine to execute more often than any of the other methods.

Program 1.

```
800 FOR ADRES=826TO949:READ DATTA:P
    OKEADRES,DATTA
805 NEXT
826 DATA 162, 1, 32, 198, 255, 160
832 DATA 0, 177, 68, 133, 96, 200
838 DATA 177, 68, 133, 94, 200, 177
844 DATA 68, 133, 95, 169, 0, 133
850 DATA 97, 32, 207, 255, 164, 97
856 DATA 145, 94, 200, 132, 97, 198
862 DATA 96, 208, 242, 32, 207, 255
868 DATA 32, 204, 255, 96, 162, 1
874 DATA 32, 198, 255, 169, 0, 133
880 DATA 5, 32, 207, 255, 201, 13
886 DATA 240, 15, 166, 5, 232, 224
892 DATA 81, 240, 47, 157, 0, 2
898 DATA 134, 5, 76, 113, 3, 166
904 DATA 5, 160, 0, 198, 48, 165
910 DATA 48, 201, 255, 208, 2, 198
916 DATA 49, 189, 0, 2, 145, 48
922 DATA 202, 208, 238, 165, 5, 145
928 DATA 68, 165, 48, 200, 145, 68
934 DATA 165, 49, 200, 145, 68, 76
940 DATA 178, 3, 169, 255, 133, 150
946 DATA 32, 204, 255, 96
```

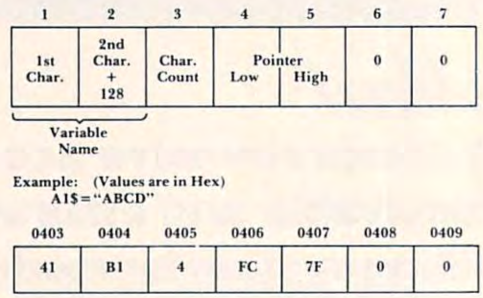


Figure 1. Simple String Variable Storage

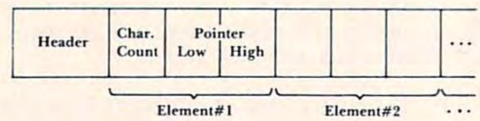


Figure 2. Array String Storage

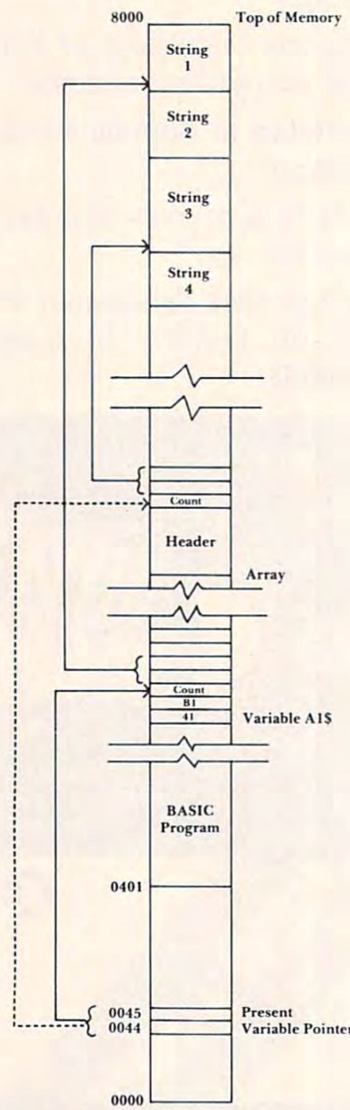


Figure 3. (Upgrade or 4.0 BASIC) Memory Map

Typing Foreign Language Text With The Commodore Printer

Zoltan Szepesi
Pittsburgh, PA

Most languages, unlike English, use different kinds of marks or accents above some of the vowels or even above or below some consonants. The French has the "accent aigu" (´), "accent grave" (`) and the "accent circonflexe" (^) placed in many words above vowels e,a,o,u and they have also the "cédille" placed below C (as ç) in some words. English typewriters and printers generally do not have the facility for printing these orthographic signs. However, with the CBM series 2022 and 2023 printers one can create special characters, thereby printing any of the wanted letters.

We could create the complete special character (letter + accent) for each vowel. However, for 4 vowels and 3 accents we would need 4x3 = 12 special characters. It is simpler to program only the 3 accents and, any time one needs the accent on the vowel, one goes to a subroutine to print the accent in the proper position. After the accent is printed, one has a carriage return without line feed, and the standard characters are printed after a line was typed.

At first, I made a program according to this plan. However, as each accent needs a full printer head scan, the printing time was slowed down very much if the number of accents in a line were great. Therefore, I modified the program so that the accents are printed after the full line has been printed, and any number of accents of one kind is printed in one printer head movement. This improved the speed to a practically acceptable level.

This paper and program will not handle the printing of special symbols below the letters as the cédille in French. This problem is the same as printing descenders on letters g,j,q,p,y. According to the same principles as described above for the accents, one can create these special characters. However, the printing can be done only with the tractor feed printer (2022 series). A paper and program on this problem will be published elsewhere.

In writing this program, I started with the "TYPEWRITER 1.5" program of Warren D. Swan, published in THE PAPER (pages 11-15, Vol. II, Issue 10, January 1980), modified for the new CBM/PET (ROM 3) and the new printer ROM (4). It is a very simple, but powerful, mini-word-processor.

Listing 1 is the accent printing program. There, first we have to design the special character strings A$(I), where I = 0 TO 2. They are defined in statement 240, using the DATA in statements 150 to 170 for the three above mentioned French accents. (See instructions in the Printer User manual or in Swan's article). Second, one has to decide which keys to sacrifice for calling the subroutine for the special accents. We used the "and" key (&) for the accent aigu (´) as specified in statement 410, the "shift and" for accent grave (`) — statement 420 — and the "shift apostrophe" key for accent circonflexe (^) — statement 430. For the printing of the accents in the proper place, the strings S$(I) are created, one for each accent type. For tape recording and reprinting the text from tape the string T$ is created.

Swan gives the instructions for how to use the original program. I will tell you shortly what to do and how to do it in the modified program.

For Input one can choose:

1. The keyboard (device #0)
2. The tape recorder (device #1 or #2)

For Output:

1. Tapes 1 or 2.
2. Screen (device #3)
3. Printer (device #4)

For the tape files you can give a file title. If you do not need a title just press two apostrophes ("").

When using the DEL command in or after an enhanced text, the following rule has to be applied:

Within the enhanced text the correction can be made the normal way, if the SHIFT- (key was not yet pressed.

If you want to go back to the enhanced characters after an exit from them, use the DEL until you delete the exit character. Here press the enhanced command (SHIFT-BACKSLASH) and continue with DEL. Do not again use the SHIFT-BACKSLASH when you type the corrected text, but use the SHIFT-(key when you want to continue with standard characters.

Boldface characters can be printed also on the full line by printing the same text on the same line

SYS8010

for
8010 or
TNW2000

Software
by
MICROFEX
fits in 3.5K

VIC-20

VIC-MAN
eats dots and monsters
in maze **$14.95**

VIC-ROIDS
shoot at asteroids as they
float in space... **$14.95**

ROM based terminal
package for CBM 8000
with 8010 or TNW 2000
SYS8010 ROM $49.95

OR

FREE with purchase
of 8010 modem!
8010 modem........... $279.00

PRIVATE I

Eliminates data errors
when using a modem with
an extension line
$11.95

Pet Bulletin Board System

Start a bulletin board with your 16K Pet or CBM. Use 4040, 8050 or 2031 drives. See it in operation at (414) 282-8118.

See Message Number 3.

PBBS software
$49.95

We also carry and demo the entire Commodore and Jim-Pack lines of equipment.

Send check or money order with order. Allow $5.00 for shipping and handling.
PETTED microsystem

4265 W. Loomis
Milwaukee, WI 53221
under the "Shell" sign
at I-894 & Loomis Rd.
(414) 282-4181

PETTED

VISA & MASTERCHARGE
ACCEPTED

HOME FINANCES
Commodore PET™

These programs were developed on a Commodore 2001 series, thirty-two kilobyte (32K required) personal electronic transactor with a single tape cassette.

Will categorize entries, give totals & tallies for all past-entered data (not a ledger; no dates or specific item names).

Great for use every 2-6 months in conjunction with check book records, stubs, bills, or your cancelled checks.

(a) Personal cash flow & tax deduction-related; 24 specific to general categories:
"tax accounts" $30
or "TAX ACCOUNTS"
(No shift)

(b) Forty categories mostly specific for common home expenditures:
"house accounts" $50
or "HOUSE ACCOUNTS"
(No shift)

(c) Both above programs; very slight overlap: $70

Choice desired . . . ________

Mail, with check or other payment and your address, to:

PractEd Tapes, Inc.
12162 S.E. 14th St.
Bellevue, Washington 98005

Programs not guaranteed against loss, misuse or system difficulties.

PET/CBM COMPUTERS
Cross Reference Program

FORMATTED LISTINGS
- .Easy to read
- .Multiple statement lines can be listed on separate lines or on one line, as on the screen

LINE NUMBER CROSS REFERENCE
- .Shows all GOTO's, GOSUB's and ON GOTO/GOSUB's
- .Flags unresolved branches

VARIABLE CROSS REFERENCE
- .Shows allocation of all variables and user defined functions
- .Variables sorted into alpha-numeric order

CODE OPTIMIZATION
- .Shows the amount of memory wasted on remarks, long variables and unneeded spaces

NEW!
SPECIAL FEATURES FOR SUPERKRAM USERS

Will work on all PET 2001, CBM 4032 and CBM 8032, and 4040/8050 disk drives.
Specify your hardware configuration.

Send $29.95 (includes postage and handling) to:
OPPENHEIMER SOFTWARE
79 ST. BOAT BASIN #39
NEW YORK, N.Y. 10024
(212) 787-2416

PET & CBM are trademarks of COMMODORE BUSINESS MACHINES, INC.
SUPERKRAM is a trademark of UNITED SOFTWARE OF AMERICA, INC.

Dealer Inquiries Invited

KEY	ASCII	Statement No.	Function
CLR	147	320	Sets the paging mode to the printer.
HOME	19	330	Sends a cursor-home character to the output device only.
DEL	20	340	Deletes the last character.
SHIFT BACKSLASH	220	350	Enhanced print
SHIFT (	168	360	Unenhanced print for stopping enhanced
INST	148	370	Deletes entire line
BACKSLASH	92	380	TAB for next 8 spaces
SHIFT #	163	400	Prints programmable character defined previously.
RETURN	13	310	Brings the printer to the print subroutine.
SHIFT RETURN	141		
&	38	410	Accent aigu
SHIFT &	166	420	Accent grave
SHIFT APOSTROPHE	167	430	Accent circonflexe
CURSOR LEFT	157	390	Program goes to the special command mode, where one can ask: 1. A programmable character 2. Change the # of lines/inch 3. End the program.

Table 1. List of operations for the TYPEWRITER ACCENT program.

two times. For achieving this, do the following:

After the line you want to print bold face, do not press RETURN, but use SHIFT-RETURN. For the next line press SHIFT-BACKSLASH and RETURN.

If the language you want to print has other accents than the ones given in this program, just construct their forms according to the instructions of the printer and substitute the resulting six numbers into the data statements 150-170. E.g.: the German text needs only the "Umlaut," which could be printed by the following data:

0,0,64,0,64,0

Since more accents are not used in the German, the other 2 accents can be deleted in the program. Since there will be just a single A$ and S$, statements 240,270,280,500,510,2090,2510 could be modified accordingly and statements 160,170,420,430,1110,1120,1220,1230 could be deleted.

In the Hungarian, beside the accent grave and the Umlaut, one needs an accent similar to the quotation mark. The following data would define this:

0,32,64,32,64,0

In several other languages one uses a wave-shaped accent. The previous Hungarian accent could be acceptable for this accent too.

Copyright registration of this program is being requested. You can use this program for your personal use, or you can have it on tape by sending $3.00 to my address: 2611 Saybrook Drive, Pittsburgh, PA 15235.

```
10 REM TYPE ACCENT  PROGRAM BY Z.SZEPESI
     ¬(COPYRIGHT REGISTRATION APPLIED)
20 REM MODIFIED FROM TYPEWRITER1.5 BY ¬
     ¬W.D.SWAN(THE PAPER VOL.II.ISSUE ¬
     ¬10)
30 REM             INITIALIZATION
35 REM            "----------------"
40 POKE 59468,14:OU=4:IN=1:Q=205:
     ¬CO=59467:T0=59466:R=59464
50 K=0:H=2:PRINT"WHAT IS THE INPUT ¬
     ¬DEVICE # (0 TO 2)?";:GOSUB3000
60 K=1:H=4:C=D:IFC=1ORC=2THENINPUT"FILE ¬
     ¬TITLE:";TL$:GOTO80
70 OPENIN,C
80 PRINT"WHAT IS THE OUTPUT DEVICE # (1 ¬
     ¬TO 4)?";:GOSUB3000
90 IFC=DGOTO80
100 IFD=1ORD=2THENINPUT"FILE TITLE";TL$
105 IFC$=""GOTO100
110 S=-(D<4):IFC=1ORC=2THENOPENIN,C,0,
     ¬TL$
120 IFD=1ORD=2THENOPENOU,D,S,TL$:GOTO140
130 OPENOU,D,S
140 PRINT"ĥ":OPEN5,OU,5:DIM A$(2),S$(2)
150 DATA 0,0,0,32,64,0:REM ACCENT AIGU
160 DATA 0,64,32,0,0,0:REM ACC. GRAVE
170 DATA 0,32,64,32,0,0:REM ACC. CIRC.
240 FORI=0TO2:A$(I)="":FORJ=1TO6:READA:
     ¬A$(I)=A$(I)+CHR$(A):NEXTJ:NEXTI
250 REM            MAIN PROGRAM LOOP
255 REM           "-----------------"
```

```
260 OPEN7,4,7:PRINT#7:CLOSE7:OPEN6,4,6:
    ¬M=6
270 POKEQ,Ø:PRINT"h̲↓↓↓↓↓S̲<";:P=1:L$="":
    ¬G$="":T$="":FORI=ØTO2:S$(I)="":
    ¬NEXT
280 AA=Ø:AØ=Ø:A1=Ø:A2=Ø:E=Ø
290 GET#IN,C$:IF64ANDSTGOTO7ØØØ
300 IFC$=""GOTO29Ø
310 IFC$=CHR$(13)ORC$=CHR$(141)GOTO1ØØØ:
    ¬REM TO PRINT LINE
320 IFC$="ĥ"THENPRINT#OU,"ĥ":GOTO29Ø
330 IFC$="h̲"THENPRINT#OU,"h̲":GOTO29Ø
340 IFC$=CHR$(2Ø)THENGOSUB2ØØØ:GOTO29Ø
350 IFC$="\̲"THENC$=CHR$(1):E=1:IFD=1ORD=
    ¬2THENT$=T$+"\̲":GOTO29Ø
360 IFC$="⌊"THENC$=CHR$(129):E=Ø:
    ¬IFD=1ORD=2THENT$=T$+"⌊":GOTO29Ø
370 IFC$=CHR$(148)THENFORK=1TOLEN(L$):
    ¬GOSUB2ØØØ:NEXT:GOTO27Ø
380 IFC$="\"GOTO25ØØ
390 IFC$="<"GOTO4ØØØ
400 IF C$="#̲"THENC$=CHR$(254)
410 IFC$="&"THENT$=T$+C$:GOSUB11ØØ:AA=1:
    ¬AØ=1:GOTO29Ø
420 IFC$="&̲"THENT$=T$+C$:GOSUB111Ø:AA=1:
    ¬A1=1:GOTO29Ø
430 IFC$="!̲"THENT$=T$+C$:GOSUB112Ø:AA=1:
    ¬A2=1:GOTO29Ø
500 FORI=ØTO2:S$(I)=S$(I)+" ":NEXTI:
    ¬IF E=ØGOTO52Ø
510 FORJ=ØTO2:S$(J)=S$(J)+" ":NEXTJ
520 C=ASC(C$)AND127
530 L$=L$+C$:T$=T$+C$
540 IFC>31ORC$=">"THENP=P+1:IFE=1THENP=P
    ¬+1
550 IFP=72THENGOSUB26ØØ
560 POKEQ,1:G$=G$+C$:IFE=1THENG$=G$+" "
570 PRINTC$;:IFE=1THENPRINT" ";
580 POKEQ,Ø:PRINT"S̲<";:GOTO29Ø
998 REM           PRINT THE LINE
999 REM        "----------------"
1000 IFL$="r̲A"THENL$=M$:G$=H$
1010 PRINT:IFD=4THENPRINT#OU,L$;CHR$(141
    ¬);
1020 IFD=1ORD=2THENPRINT#OU,T$;CHR$(141)
    ¬;
1030 PRINT"ĥ";:FORK=1TOLEN(G$):POKEQ,1:
    ¬PRINTMID$(G$,K,1);:NEXT:M$=L$:
    ¬H$=G$
1040 POKEQ,Ø:IF AA=1 THENGOSUB12ØØ
1060 PRINT#OU,CHR$(13);:GOTO27Ø
1100 PRINT#5,A$(Ø):S$(Ø)=LEFT$(S$(Ø),
    ¬P-2)+CHR$(254)
1105 RETURN
1110 PRINT#5,A$(1):S$(1)=LEFT$(S$(1),
    ¬P-2)+CHR$(254)
1115 RETURN
1120 PRINT#5,A$(2):S$(2)=LEFT$(S$(2),
    ¬P-2)+CHR$(254)
1125 RETURN
1200 IFD<>4THENRETURN
1210 IFAØ=1THENPRINT#5,A$(Ø):PRINT#OU,
    ¬S$(Ø);CHR$(141);
1220 IFA1=1THENPRINT#5,A$(1):PRINT#OU,
    ¬S$(1);CHR$(141);
1230 IFA2=1THENPRINT#5,A$(2):PRINT#OU,
    ¬S$(2);CHR$(141);
1240 RETURN
1998 REM           DELETE A CHAR.
1999 REM        "----------------"
2000 IFLEN(L$)=ØORLEN(T$)=ØTHENRETURN
2010 PRINTCHR$(2Ø);:IFE=1THENPRINTCHR$(2
    ¬Ø);
2020 F$=RIGHT$(G$,1):G$=MID$(G$,1,
    ¬LEN(G$)-1):IFE=1THENG$=LEFT$(G$,
    ¬LEN(G$)-1)
2030 F1$=RIGHT$(T$,1):T$=LEFT$(T$,
    ¬LEN(T$)-1)
2040 IF F1$="&"ORF1$="&̲"ORF1$="!̲"THENT$=
    ¬LEFT$(T$,LEN(T$)-1)
2050 IFF$<>"<"GOTO2Ø8Ø
2060 O$=RIGHT$(L$,1):L$=MID$(L$,1,
    ¬LEN(L$)-1):IFO$<>"<"ORF$<>"<"GOTO2
    ¬Ø5Ø
2070 RETURN
2080 L$=MID$(L$,1,LEN(L$)-1):P=P+((ASC(F
    ¬$)AND127)>31)+(F$=">"):IFE=1THENP=
    ¬P-1
2090 FORI=ØTO2:S$(I)=LEFT$(S$(I),P-1):
    ¬NEXT
2100 IFD=1ORD=2GOTO212Ø
2110 RETURN
2120 IF F1$="&"ORF1$="&̲"ORF1$="!̲"THENT$=
    ¬LEFT$(T$,P-1)
2130 IFE=1THEN T$=LEFT$(T$,LEN(T$)+1)
2140 RETURN
2498 REM           TAB TO NEXT STOP
2499 REM        "------------------"
2500 T=8-(PAND7):P=P+T:FORK=1TOT:
    ¬L$=L$+" ":G$=G$+" ":T$=T$+" ":
    ¬PRINT" ";
2510 FORI=ØTO2:S$(I)=S$(I)+" ":NEXTI
2520 NEXTK:PRINT"S̲<";
2530 IFP>=72THENGOSUB26ØØ
2540 GOTO29Ø
2598 REM           END OF LINE BEEP
2599 REM        "------------------"
2600 POKER,Ø:POKECO,16:POKETØ,15:
    ¬POKER,15Ø:FORK=1TO2E2:NEXT
2610 POKER,Ø:POKETØ,Ø:POKECO,Ø::RETURN
2998 REM           GET A DEVICE
2999 REM        "--------------"
3000 GETC$:IFC$=""GOTO3ØØØ
3010 D=ASC(C$)-48:IFD<KORD>HGOTO3ØØØ
3020 PRINTD:RETURN
3998 REM           EXTRA COMMANDS
3999 REM        "----------------"
4000 PRINT"ENTER COMMAND:"
4010 PRINT"↓1. DEFINE A PROGRAMMABLE ¬
    ¬CHARACTER."
4020 PRINT"↓2. SET LINES/INCH.":
    ¬PRINT"↓3. END PROGRAM"
4030 GET#IN,F$:IF64ANDSTGOTO7ØØØ
4040 IFF$<"1"ORF$>"3"GOTO4Ø3Ø
4050 IFD=1ORD=2THENL$=L$+"<"+F$:
    ¬G$=G$+"<"
4060 IF F$="3"GOTO7ØØØ
4070 IF F$="2"GOTO6ØØØ
4998 REM           DEFINE CHARACTER
4999 REM        "------------------"
```

```
5000 PG$="":PRINT"ĥENTER 6 NUMBERS TO ¬
     ¬DEFINE THE CHARACTER:
5010 PRINT"(ONE AT A TIME FOLLOWED BY ¬
     ¬RETURN)
5020 IFD=4THENCLOSE5:OPEN5,OU,5
5030 FORK=1TO6:PRINTK;:INPUT#IN,F$:
     ¬IF64ANDSTGOTO7000
5040 C=VAL(F$):PRINTC:IFD=1ORD=2THENL$=L
     ¬$+F$+CHR$(13)
5050 PG$=PG$+CHR$(C):NEXT:IFD=4THENPRINT
     ¬#5,PG$
5060 PRINT"ĥ";:FORK=1TOLEN(H$):POKEQ,1:
     ¬PRINTMID$(H$,K,1);:NEXT
5070 POKEQ,0:PRINT"ḫ↓↓↓↓":FORK=1TOLEN(G$
     ¬):POKEQ,1:PRINTMID$(G$,K,1);:NEXT
5080 POKEQ,0:PRINT"$<";:GOTO290
5998 REM          SET LINES/INCH
5999 REM        "----------------"
6000 PRINT"ĥNUMBER OF LINES PER INCH? ";
6010 IFD=4THENCLOSE6:OPEN6,OU,6
6020 INPUT#IN,F$:IF64ANDSTGOTO7000
6030 M=VAL(F$):PRINTM:IFD=1ORD=2THENL$=L
     ¬$+F$+CHR$(13)
6040 IFD=4THENPRINT#6,CHR$(144/M)
6050 GOTO5060
6998 REM          END OF PROGRAM
6999 REM        "----------------"
7000 PRINT:IFD=1ORD=2THENCLOSEOU
7010 END:IFD>2GOTO5060
7020 PRINT"?CAN'T CONTINUE ERROR":END:
     ¬RUN
```

©

Three Reviews:

Superchip, Spacemaker, Sort

Harvey B. Herman
Associate Editor

The Petmaster Superchip

Some of us may have envied the tricks one can play with the new 80 column PETs using BASIC 4.0. For example, one can define a window which is seemingly immune from scrolling. SUPERCHIP, firmware from our English cousins, is intended to provide some of these screen handling functions and additional goodies also. It is available for all the current PET ROMs and does not conflict with the TOOLKIT.

The first feature I made use of (and liked) is called single key BASIC. That is, G stands for GO, N stands for NEXT,R stands for RETURN, etc. The full word appears, as if by magic, when a control key is pressed simultaneously with a letter. Another function that caught my fancy is called escape. This allows you to toggle back and forth between quote and direct modes of cursor control. If you ever get stuck in the wrong mode you know how useful that could be. I also made frequent use of the hold function which suspends execution until RETURN is pressed.

SUPERCHIP has a total of 18 functions:

erase begin	**erase end**
scroll up	**scroll down**
escape	**retrace**
message	**functions**
movit	**single key**
delete line	**insert line**
scroll window	**graphics toggle**
hold	**stop**
shrink	**reverse**

Most functions can be accessed either in immediate mode or from a BASIC program. A concise reference chart on the rear cover of the 26 page user manual summarizes the functions and states any exceptions or limitations. The manual is, for the most part, easy to understand by a first time user. However, I did have trouble with the scroll window section and I was confused by the use of the word "bracket" for "open parenthesis."

SUPERCHIP will appeal, I think, to many people and, if the reader is in this group, by all means buy it. You will have added quite a few useful functions to your repertoire which are not available elsewhere. Programs which use these features will be able to generate displays which ordinary PETs cannot do without great difficulty. As for me, I am not convinced that it is a good buy. Even with the recent drop in the pound exchange rate, SUPERCHIP still costs more than comparable firmware such as the TOOLKIT. In its favor, however, is that it offers some desirable features of BASIC 4.0 without losing the use of previously developed machine language software, which may be ROM-dependent.

Supersoft
28 Burwood Ave.
Eastcote, Pinner, Middx., England
£45

Spacemaker II

New Commodore CBM/PETs have empty ROM sockets on the main logic board which allow users to install special software packages. These include the CBM word processors, VISICALC, and the TOOLKIT. Recently I received for review two firmware (EPROM) programs which, alas, required installation in the same empty ROM socket. This meant that I could not switch back and forth between the two programs without risk of permanent damage to the IC pins (or to my psyche). My problem was solved when I received the SPACEMAKER II for review. This nicely crafted piece of hardware is capable of switching between as many as four different ROMs when plugged into a single socket on the PET logic board.

I had no trouble working with SPACEMAKER II. The hardest part is insertion of the ROMs, but this time everything went smoothly. Jumpers which depend on ROM type, are placed on posts and no soldering is required. SPACEMAKER II is particularly easy to plug into a socket on the PET as you can get a grip on it more easily than a much smaller ROM. The four page instruction leaflet had quite explicit directions and I noticed only one typo (figure 2 instead of figure 3). The version I received employed manual switching with a switch mounted on the side of the PET (no drilling necessary). It is also possible to switch using software, with control by the User Port or with optional hardware (ROMDRIVER).

I have no hesitation about recommending this hardware to PET users who require software on ROM, but have addressing conflicts. SPACEMAKER II is professionally done and is reasonably priced. My only gripe is that they did not include a circuit diagram in the unlikely event that service is needed. In a way I'm glad they didn't as I was hard pressed to find any negative comments.

CGRS Microtech
P.O. Box 102
Langhorne, PA 19047
$39.00

SORT

(3.0 or 4.0 ROMs)
(40 or 80 column screen)

If you do much computing you will eventually need a good sort routine. I started to write a program recently which sorted and printed the names of up to 256 programs on PEDISK I diskettes. To my horror, I realized that I did not have, in my "junk box" of programs, a fast sort routine. **COMPUTE!** came to my rescue. An early issue compared sort routines and I was able to adapt one of the BASIC listings in the article. However, not everyone has the ability or inclination to fit published programs to their own use. Matrix software offers a SORT program (on EPROM) for people who need a fast machine language sort that can be used with a minimum of effort even by novice programmers.

I had little trouble writing my first simple sort program. Their seven pages of instructions were quite helpful. I was able to do a four character sort on 1000 items in under seven seconds (average). Try doing that in BASIC sometime and you will be as impressed as I was. The program is executed with a SYS call after a few required POKEs. For example,

```
POKE 905,a — which dimensioned array
POKE 906,b — number of keys
POKE 907,c — dimension of array
POKE 927,d — number of characters to evaluate
POKE 947,e — what character to begin sort at
SYS 36864 — for EPROM at $9000 (specify when
ordering)
```

For review purposes only, the company included a demonstration program. I believe they should include a listing of this program with each order. Otherwise, I have no complaints about this package. The sort is fast. It works with integers, real numbers, or strings. And, as a bonus, they include a printer screen dump in the unused space on the ROM. Check this program out if you do lots of sorts and you need a fast routine resident at all times. You should find it very useful.

Matrix Software
315 Marion Ave.
Big Rapid, MI 49307
$55

©

Machine Language: Jumbo Numbers

Jim Butterfield
Toronto, Canada

A single byte will hold an unsigned number whose value may be from 0 to 255. Most of us, sooner or later, want to handle larger numbers. The techniques are fairly straightforward.

A number may occupy several bytes of storage. The usual convention is for the higher order bytes to contain powers of 256. In simple terms, this means that one byte counts in "ones"; another byte counts in "256-s"; the next byte, if used, counts in "4096-s" and so on. It's easier than it sounds if you convert the number to hexadecimal. One million, which in hexadecimal is 0F4240, fits nicely into three bytes: from high order to low order these bytes contain 0F, 42, and 40 hexadecimal.

It is possible to hold numbers in a decimal type of format. This makes input and output easy, since no conversion is needed to convert the decimal digits, and addition and subtraction can be quite easily accomplished. More complex arithmetic is difficult — even multiplication and division requires an effort — so that we choose binary if any real math crunching is needed. Decimal numbers can be held two ways: packed, with two digits to a byte; and unpacked, with one digit to a byte.

Sizing

We must make room for the largest possible numbers we expect to handle. The following table may be halpful:

	Unsigned	Signed	Packed Decimal
1 Byte:	0 to 255	−128 to +127	0 to 99
2 Bytes:	0 to 65535	−32768 to +32767	0 to 9999
3 Bytes:	0 to 1677215	−8388608 to +8388607	0 to 999999

The table grows proportionately; if a count of over sixteen million in three bytes won't do, four bytes reaches to over four billion (after taxes, that's four thousand million in Great Britain). Enough for most applications, but you can continue to add bytes as you wish.

What about fractions? The most common method is to use an assumed decimal point. In other words, count in pennies instead of in dollars and you won't need fractions. There are more exacting methods, but most of us sidestep them if we can.

Memory Arrangement

There's really no special law regarding how you arrange these bytes in memory. You can have high order values at the higher addresses, or turn it around and have high order values at the low end. I like to have low order at the low address end, etc.: it's easier to remember and is more consistent with address modes. On the other hand, storing the bytes the other way around (high order at the low address) makes it a little easier to handle a number with indexing. Why? Well, if we have to test an index register for the end of its range with CPX or CPY, we'll affect the Carry flag ... and we often need that flag to link information between the various bytes. A fine point; the choice is really up to you.

You can even scatter the values through memory rather than having them consecutive. Often it's better to keep them together so that you can "walk through" a number using indexing. But there are exceptions to every rule.

Some Simple Operations

We can manipulate multi-byte numbers just as readily as single bytes. All we need is some new rules.

For the following sample code, let's assume a two-byte value stored in locations 0300 low-order and 0301 high-order.

Moving: move both bytes instead of one. To move 0300/0301 to 0320/0321 we might code: LDX #$01; LOOP: LDA $0300,X; STA $0320,X; DEX; BPL LOOP. We have moved the high order byte first, but this makes no difference.

Addition and subtraction: start at the low end; fix up the Carry flag before you start, and then let the Carry link the bytes together. To add the contents of $0300/0301 to $0320/0321 and place the result at $0320/0321, we might code: CLC; LDA $0300; ADC $0320; STA $0320; LDA $0301; ADC $0321; STA $0321. Note that it's vital that we start at the low end of the numbers, in this case the low addresses. We might wish to check to insure that the result hasn't overflowed (overflew?) the space available. For unsigned numbers, we do this by checking that the Carry flag is clear.

Subtraction goes the same way, except we give SEC and use the SBC command. A valid subtraction will complete with the Carry flag set; otherwise there's an unsigned number overflow.

Comparisons

Comparison is a little different from the single-byte compare. We need to decide in advance if we're testing for equality or for greater-than; it's hard to check for both in a single sequence.

80 x 25

PET/CBM™

2000/3000/4000 Series

not using a CRT, or display controller chip

$275.00*

Select either **80 x 25** or **40 x 25** On The Built-in Display

From the keyboard or program

Displays the full, original character set

Available from your local dealer or:

EXECOM CORP.
1901 Polaris Ave.
Racine, WI 53404
Ph. 414-632-1004

*Plus installation charge of $75.00

Available only for Basic 3.0 & Basic 4.0

PET& CBM™a

trademark of Commodore Business Machines

Commodore 3.0/4.0/2.1 Dos Source Code!

Complete annotated source code for 3.0 ROMs, 4.0 ROMs and 2.1 DOS (4040-,8050-disk) now available. All entry points, routine/variable names, info on routine operation, register contents, etc. included. 4.0 source code comes with 3.0 correspondence codes so you can change 3.0 machine-language programs to 4.0 and vice versa. Hardcopy only. $129. each. All 3 (3.0, 4.0, 2.1 DOS) for $310.00. Source files on diskette on special order.

Software/Firmware

- EXTRAMON extended monitor in 2K EPROM for any slot, 3.0/4.0 ROMs, plus hardcopy documentation. $19.95
- DISK MONITOR© - reads T/S from disk to screen at 0.21 seconds/blk. Allows editing like the resident monitor, updates disk, traces links, etc. FAST! 31.1 Kbytes/sec. $49.95
- BULLETPROOF!© - full-screen input routine. Uses machine-language to get user input, places it in a user-defined variable. Text on screen outside fields impossible to disturb. Documented source code incl. with disk. $29.95

THE GREAT WESTERN SOFTWARE CO.

Authorized dealer for RPL© compilers

1624 N.W. 9th Ave., Ft. Lauderdale, FL 33311 (305) 523-1351

PETTERM

INTELLIGENT TERMINAL

Turn your PET into an intelligent terminal with one of our terminal packages. These are complete assembled hardware and software packages. All include line editing/resend, repeat key, shift lock, output to CBM printer, and more . . . Delivered on PET cassette with manuals.

PETTERM I	All features above $75.00
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

for PET/CBM

FORTH is a new concept in programming, with the speed of compilers and interactive ease of BASIC. Programs become a part of FORTH extending the power of FORTH and your PET.

8050,4040 disk, cassette all PET-CBMs 16k+

Starter	fig-FORTH w editor assembler	$35
Personal	floating point; strings; source	$75
Professional	turnkey development/data base	$259

F S S

software for small computers

1903 Rio Grande
Austin, Texas
78705

1-512-477-2207

P.O. Box 8403
Austin, Texas
78712

master charge

DEALERS INQUIRE

VISA

Equality tests are quite straightforward: test each of the pairs of bytes, and if any are not the same, the two values are unequal. We might code: LDX #$01; LOOP: LDA $0300,X; CMP $0320,X; BNE UNEQUAL; DEX; BPL LOOP; EQUAL: ... The code is fairly self-evident.

To compare for greater-than, we might do a full subtraction. We won't need to keep the result; the flags will tell us the answer. We might code: SEC; LDA $0300; SBC $0320; LDA $0301; SBC $0321. At this point, the Carry flag will be set if the value in $0300/0301 is greater than or equal to that in $0320/0321.

It's possible to compare from the high-order end down, on the theory that if the first byte is different, you don't need to look at the rest. Additionally, such a comparison can more easily test both equal and greater-than conditions. There's often not much difference; speed is likely to depend on whether or not the numbers are likely to be close or far apart.

Shifts And Rotates

Shifts and Rotates propogate readily through the Carry bit. The first operation must start at the proper end of the number: Right shifts start from the high end, Left shifts from the low. The remaining operations, which work their way through the number, must always be Rotates, regardless of whether the overall operation is Shift or Rotate.

To shift the two-byte number at $0300/0301 left, we might code: ASL $0300; ROL $0301. To rotate the same number, we would give ROL $0300; ROL $0301.

To shift the same number right, we would code; LSR $0301; ROR $0300. Finally we would rotate the number right with ROR $0301; ROR $0300.

Big numbers are not much harder to work with than small ones. All the usual operations are still available to you. There are more items to keep track of, but that's a natural result of expansion.

Make provision for future big numbers now. You wouldn't want to tell your boss that he can't give you a raise because there isn't room enough in the computer to hold what he wants to pay you...

©

File Recovery

M. R. D'Amato
Plainfield, NJ

If you have inadvertently scratched a file (and who hasn't?) on the 2040 (DOS 1.0) or the 8050 disk drive you can easily recover the information if you avoid saving additional files on the disk. (For really bad (slipped?) disk problems, see Cones' more sophisticated file recovery program; **COMPUTE!** #10.) The task is easier on the 8050, so let's start there.

On The 8050

When a program file is scratched, the file identifier, located in byte #0 of the 30-byte file entry in the directory, is changed from 130 to 0. Also, the blocks in which the file was written are recovered by DOS for subsequent reuse. Program 1 does the following. It searches the directory for scratched files and presents them on the screen one at a time. Press key "N" if the scratched file is not the one you want. (Also press "N" if all zeros appear as the file name, but don't respond to the directory track and sector numbers, which are provided for your information.) When the desired file appears on the screen, press "Y." This results in changing the file label from 0 to 130 and depositing this value on the disk. If the directory is then accessed, the name of the scratched file will appear in the displayed directory, and the file can be loaded into memory.

It is essential that the file be reSAVEd or it will be again lost when DOS assigns one or more of its blocks to a subsequently saved file. Therefore, after recovering the file and loading it into memory you should (a) SCRATCH the file (to remove it from the directory) and (b) SAVE the file under its original or a new name. That's all there is to it.

And the 2040

The task is a bit more difficult on the 2040 with DOS 1.0 because the track number of the first data block is, like the file identifier, also set to 0. However, it is usually possible to infer the number of the initial track by examining the starting track numbers of the neighboring files in the directory. Program 2, which lists a block of data on the screen, is meant to help in this task. When prompted for track and sector, enter 18,1, the first directory block. If your scratched file is in this directory block, it can be identified by its name, which appears in ASCII format. The first two bytes of the block give the track and sector of the next directory block (18,4). The 30 bytes of the first file entry follow.

As already noted, byte 0 holds the file type. The track and sector of the first data block are located in bytes one and two, followed by a file name in bytes 3-18, padded with shifted spaces (160's). A total of eight file entries (separated by two zero bytes) can be contained within a directory block. If your scratched file is not in the first directory block, access the next directory block by running Program 2 and entering 18, 4, continuing the process as necessary.

After locating the lost file, compare the track and sector number of the first data block in the preceding and the following file entries. These values will often immediately suggest the number of the first track of the scratched file. Having inferred the initial track of the scratched file, add lines 145, 414, and 416 to Program 1, change T in line 170 to 18 and run the program. It's a good idea to work with a duplicate disk, just in case you have the wrong track number and cause DOS some confusion.

Sequential files can also be recovered by changing the CHR$ (130) in line 410 of Program 1 to CHR$ (129). As with program files, once the sequential file is recovered, the name of the original file should be scratched from the directory and the recovered file saved.

The omission of a disk-error handling routine in the program is a concession to simplicity. It seemed just as easy to rerun the program if anything went wrong, but it's simple enough to include an error routine if needed.

The 8050 recovery program also works for files generated by the Wordpro 4 word processor, which stores text as program files. In fact, it was the humiliation of having mindlessly scratched a couple of such files that led to the development of the present program.

Murky BAM

For those of you who might want to poke around in murky BAM (block availability map), Programs 3 and 4 will illuminate things a bit by highlighting, in reverse field, the BAM bytes associated with a particular track (four on the 2040 disk and five on the 8050). In both cases, the first byte reveals the number of free blocks in the specified track and the subsequent bytes indicate their identity. Block zero is represented by byte zero of the second byte, block eight is represented by byte zero of the third byte, and so on. Not a very intuitive layout, but computers have little concern for such matters.

Because the DOS support program ("wedge") jumps into action when it sees the ASCII of ›, /, ↑ or @, it's best not to have the wedge concurrently in memory when using Programs 1-3 on the 2040 with DOS 1.0. This is not a problem on the 8050 and may be on the 2040 with versions of DOS ›

1.0. As if in compensation, the 8050 (but not the 2040) may give a "70, no channels" error if the disk holding the scratched program is not accessed with a load or a directory command after initialization. If this occurs, simply display the directory and rerun the program.

Program 1.

```
100 REM ***RESTORE SCRATCHED PROGRA
    M FILE***********
110 REM ***ON 8050 DISK DRIVE*** M.
     R. D'AMATO *****
120 OPEN15,8,15
130 OPEN1,8,3,"#"
140 PRINT"DRIVE NUMBER Ø OR 1?":INP
    UT D
150 PRINT"PRESS Y IF FILE IS FOUND,
     PRESS N IF NOT"
160 REM ***FIND AND PRINT SCRATCHED
     FILES***********
170 T=39:S=1
180 PRINT#15,"B-R:"3;D;T;S
190 PRINT#15,"B-P:"3;2+32*R
200 GET#1,A$:IFA$=""THENA$=CHR$(Ø)
210 IFASC(A$)>128THEN300
220 FOR K=5+32*R TO 20+32*R
230 PRINT#15,"B-P:"3;K
240 GET#1,A$:IFA$=""THENPRINT"Ø";:G
    OTO260
250 PRINTA$;
260 NEXT
270 PRINT
280 GETA$:IFA$=""THEN280
290 IFA$="Y"THEN400
300 R=R+1:IFR<8THEN180
310 REM ***GET NEXT DIRECTORY TRACK
     & SECTOR********
320 PRINT#15,"B-P:"3;Ø
330 GET#1,A$:IFA$=""THENA$=CHR$(Ø)
340 T=ASC(A$):IFT=ØTHENPRINT"FILE N
    OT FOUND":GOTO430
350 PRINT#15,"B-P:"3;1
360 GET#1,A$:IFA$=""THENA$=CHR$(Ø)
370 S=ASC(A$):PRINT T,S
380 R=Ø:GOTO180
390 REM ***RESTORE PROGRAM FILE LAB
    EL (13Ø)*********
400 PRINT#15,"B-P:"3;5+32*R-3
410 PRINT#1,CHR$(13Ø);
420 PRINT#15,"U2:"3;D;T;S
430 CLOSE1:CLOSE15
```

Program 2.

```
100 REM **READ A BLOCK ON 2Ø5Ø OR 8
    Ø5Ø**
110 OPEN15,8,15
120 OPEN1,8,3,"#"
130 PRINT"ENTER TRACK AND SECTOR (B
    LOCK)"
135 INPUT T,S:D=1:REM D=DRIVE NUMBE
    R
140 PRINT:PRINTT"-"S":";
150 PRINT#15,"B-R:"3;D;T;S
160 FORK=ØTO255
170 PRINT#15,"B-P:"3;K
180 GET#1,A$
190 IFA$=""THENPRINT"Ø";:GOTO21Ø
200 PRINTASC(A$);
210 NEXTK
220 CLOSE1:CLOSE15
```

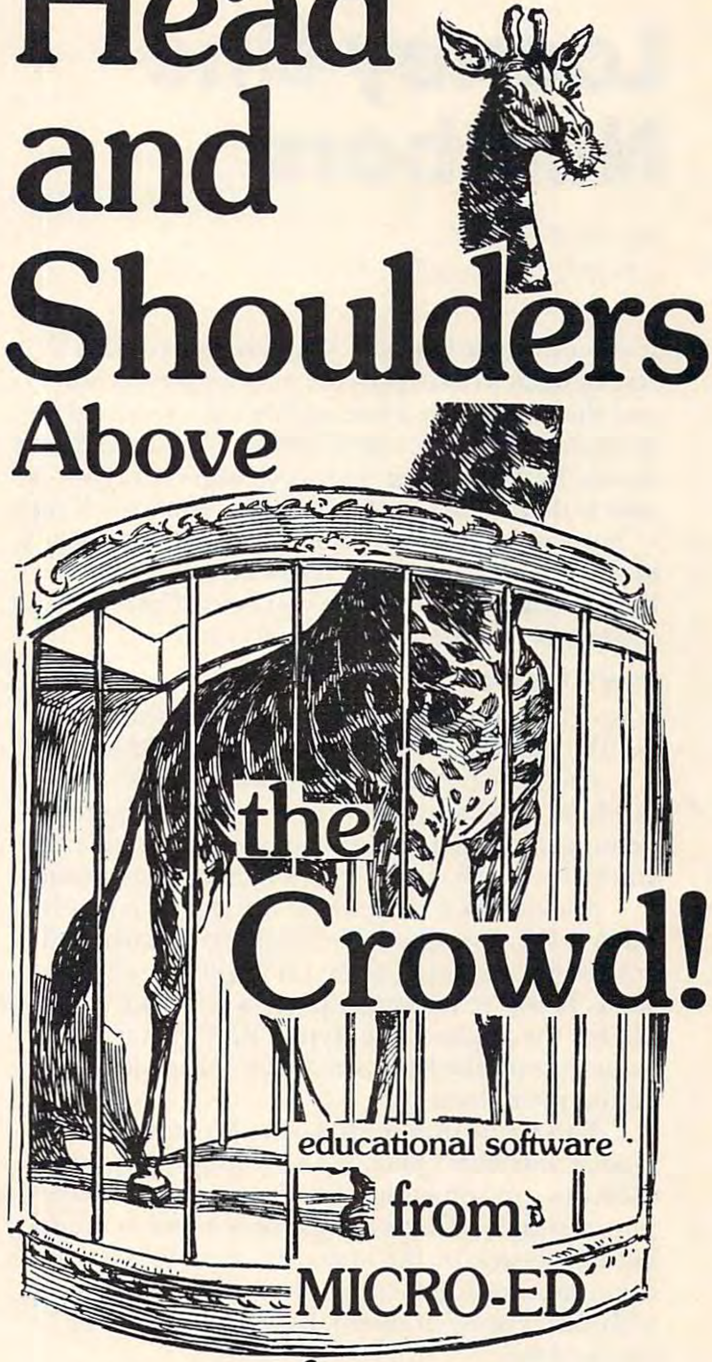
Program 3.

```
100 REM **BAM HIGHLIGHT PROGRAM 2Ø4
    Ø**
105 REM ******** M. R. D'AMATO ****
    ***
110 OPEN15,8,15
120 OPEN1,8,3,"#"
130 PRINT"ENTER TRACK FOR WANTED BA
    M"
135 INPUT T:D=1:REM D=DRIVE NUMBER
140 PRINT:PRINT18"-"Ø":";
150 PRINT#15,"B-R:"3;D;18;Ø
160 FORK=ØTO255
170 PRINT#15,"B-P:"3;K
180 GET#1,A$
182 IFK=>(4*T)ANDK<4*(T+1)THENPRINT
    "{REV}";
190 IFA$=""THENPRINT"Ø";:GOTO21Ø
200 PRINTASC(A$);
210 PRINT"{OFF}";:NEXTK
220 CLOSE1:CLOSE15
```

Program 4.

```
100 REM **BAM HIGHLIGHT PROGRAM 8Ø5
    Ø**
105 REM ******** M. R. D'AMATO ****
    ***
110 OPEN15,8,15
120 OPEN1,8,3,"#"
130 PRINT"ENTER TRACK FOR WANTED BA
    M"
135 INPUT T:D=1:REM D=DRIVE NUMBER
137 IF T>5ØTHENS=3:T=T-5Ø:GOTO14Ø
138 S=Ø
140 PRINT:PRINT38"-"S":";
150 PRINT#15,"B-R:"3;D;38;S
160 FORK=ØTO255
170 PRINT#15,"B-P:"3;K
180 GET#1,A$
182 IFK=>5*T+1ANDK<5*T+6THENPRINT"{
    REV}";
190 IFA$=""THENPRINT"Ø";:GOTO21Ø
200 PRINTASC(A$);
210 PRINT"{OFF}";:NEXTK
220 CLOSE1:CLOSE15
```

©



Looney Line Numbers

Jim Butterfield
Toronto, Canada

It should never happen. You have a program that you've been working on for hours (days? weeks?) and then suddenly a line number goes wrong. In between lines 6340 and 6360 the line number that should be 6350 has suddenly changed to 2254. Not only is that wrong — the GOTO's won't work right — but you can't get rid of it! The line seems stuck in your program forever. How does it happen? More to the point, how do you get rid of it without completely reentering the program?

How It Happens

It won't happen under normal circumstances. BASIC guards carefully against this kind of error.

An unwise POKE instruction or a SYS to a machine language program that's not completely debugged can get you into all sorts of trouble. If you're lucky, all you'll get is a looney line number.

Sometimes a bad LOAD will do the trick. In theory, the computer should guard against load errors; but it doesn't always tell you the whole story. If you're loading tape on a CBM/PET, always ask for the Status value (type PRINT ST): if the value is zero, the load is reliable; otherwise, you're taking your chances.

Bad RAM (Random Access Memory) can plague you with faults. It's not always obvious. Memory can sometimes fail erratically: perhaps the power supply voltage drops for a moment, and a bit disappears; or the malfunction only starts after the computer's innards get hot. If you're plagued with this type of problem, have your machine checked out.

All of the above may cause goofy line numbers; but they also may randomly cause other errors. Some are fatal, and some cause your program to look weird. Try to pin down the cause; it's worth the effort.

Fixing Numbers That Are Too High

There are two cases: high line numbers (out of proper order) and very high line numbers.

If an out-of-sequence line number is too high, but less than 64000, the trick is easy: delete the bad line and reenter it with the proper line number.

If the line number is 64000 or more, we must go to the next section and run the program there. You're not allowed to enter a line number of 64000 or more, even to delete the line concerned. Try typing 64000 followed by RETURN; you'll get a ?SYNTAX ERROR.

Fixing Low And Super-high Numbers

Type in the following lines at the front of your program. If your program happens to have lines numbered in the range from 0 to 8, take them out and put them back later.

```
1 A=1025: V=256: X=-1
2 B=A:A=PEEK(B)+PEEK(B+1)*V
3 PRINT : IF A=0 THEN END
4 Z=PEEK(B+2):Y=Z+PEEK(B+3)*V
5 PRINT CHR$(145);Y;:IF Y>X AND Z<250 GOTO 8
6 Y=X+1 : Y%=Y/V:PRINT"TO";Y
7 POKE B+2,Y-Y%*V:POKE B+3,Y%
8 X=Y:GOTO 2
```

The above coding is for the PET/CBM; you can adapt it to other computers by changing the value 1025 in line 1 to the Start-of-BASIC address in your machine. The CHR$(145) can be changed to match your machine's Cursor-Up character.

Meaning of the variables: B is the address of the current line of BASIC being examined; A is the address of the next line. X is the previous line number and Y is the new line number. Z is the "high byte" of the new line number; it's used to test for a super-high number. V is a constant of 256.

The program goes through each line of BASIC including itself and checks that each line number is higher than the previous one and not over 63999. If the line number fails the test, it is set to one higher than the previous line number.

Note the logic: can you see why the program must not be used on a normal "too-high" looney line number? It would "pass" the bad line number, and then bump up the numbers on all following lines.

What do you do if you have both too-low and too-high? Fix the too-high first before you run this program. If you do have multiple faults, chances are your program is in really bad shape anyway; get your computer fixed and redo the whole program.

Looney line numbers should never happen. Look for the cause if it happens to you.

You can fix them, however. And the mechanics of fixing bad line numbers has a tiny bonus: look at the coding and see if you can gain an insight into how BASIC is put together.

Super-coders can go after the same problems by attacking the program directly as it lies on disk, copying the program over and correcting it on the way. Users with BASIC enhancement packages (Toolkit, Command-O, Power, etc.) can fix everything in a trice with program renumber.

There are many ways of fixing it ... once you know how. ©

Editor's Note: Stephen suggests that this game can be a source of ideas for a variety of other games. If you come up with an exciting variation, send it in and share it with other **COMPUTE!** *readers. — RTM*

Mine Maze

Stephen Vermeulen
Calgary, Canada

This two-player game for 40 column PETs illustrates how the character oriented graphics of the PET, coupled with keyboard input from only three keys, can produce a fast, frustrating, and addictive game. Also, by keeping the graphics as simple and as clear as possible, implementation in BASIC is practical.

Rules Of The Game

A random maze with a clear border around it is drawn on the screen. The two players are then placed within the clear border and play starts. The object, for both players, is to be the survivor of what might appear at first to be a one-sided conflict. The aggressor in the battle is the left-hand player who is represented by the solid ball (shifted Q) graphic character. The ball can only win by running into his opponent, the circle (shifted W).

The circle is usually the defensive player, and is able to drop mines on the playing field. When the ball hits one of these mines it looses one life. To even things up a bit, the ball is given one free life for every ten mines the circle deposits on the playing field and, also, the number of mines that must be dropped before the ball gets its first free life is set randomly.

Lines 100 to 280 print the instructions and get the players' names and the difficulty factor. Default values for these inputs are provided so that eager (or lazy) players can get into play by pressing return 3 times. The next portion (lines 290–380) sets up the playing field and starts the play. The graphic characters used are Q [81] for the ball, W [87] for the circle, [102] for the maze, and * [42] for the mines. The values in brackets are the screen POKE values. The last section (lines 600–680) of the program displays the score and prompts the players for another game.

Heart Of The Program

Now that the sundries have been discussed, the heart of the program can be dissected. The keypress interpreter is the code found in lines 390–410. The branches on "@" and "=" to lines 470 and 480 serve to rotate the player's direction of movement 90 degrees clockwise. The branch on "*" proceeds to lines 490–510 which drop a mine and increment the mine counter and the ball's life counter. After the present key press has been processed the program then moves both players. Before a player can be moved, the program must check for obstacles and hazards by PEEKing the next position and, if necessary, going to one of the two obstruction test routines. The first of these (lines 520–560) is for use when moving the ball. Here the next square is checked to see if it is a maze wall (the ball bounces), or the circle (the ball wins the game), or a mine (the ball looses a life and possibly the game). The second routine (lines 570–590) is the obstacle routine for the circle, here the next square is checked to see if it is a maze wall (making the circle bounce back) or the ball (the ball wins). If the square happened to contain a mine the circle would just erase it (which can be very frustrating if you are controlling the circle).

Official Decrees

And now for some final rules for situations which do arise (these rules were adopted for play in the most recent World Mine Maze 1.9 Championships consisting of a round-robin three player tournament):

1. It is decreed that, the Mine Layer shall *not* lock himself in between the outer boundary and a wall of the maze (see Figure 1).

2. It is decreed that, if the Mine Layer has successfully sealed himself off in the inner part of the maze, the Ball must commit suicide by running into as many mines as it takes to die.

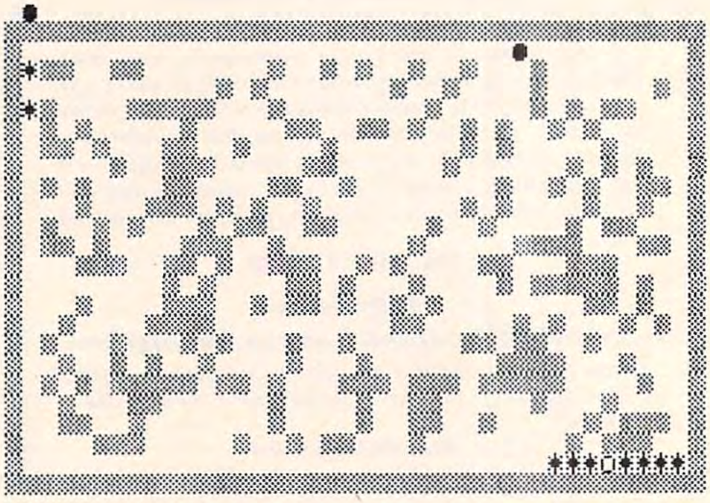


Figure 1: The circle is not allowed to win by blocking off an edge position such as this because it is so easy to do.

```
100 POKE59468,14
110 REM MINES VS DESTROYER
120 PRINT"{CLEAR}{04 DOWN}                    MINE
    S OR BUST"
130 PRINT"{02 DOWN}INSTRUCTIONS:
140 PRINT"{DOWN}THE PLAYER ON THE LEFT US
    ES THE @ KEY
150 PRINT"TO STEER THE BALL CLOCKWISE."
160 PRINT"{DOWN}THE PLAYER ON THE RIGHT U
    SES THE = KEY
170 PRINT"TO TURN THE CIRCLE CLOCKWISE, A
    ND THE *
180 PRINT"TO PLACE A MINE ON THE FIELD.
190 PRINT"{DOWN}IF THE BALL [@] HITS A MI
    NE THE CIRCLE
200 PRINT"WINS, IF THE CIRCLE HITS A MINE
     THE MINE";
210 PRINT"IS DESTROYED.
220 PRINT"{REV}THE{OFF} {REV}BALL{OFF} {R
    REV}GETS{OFF} {REV}AN{OFF} {REV}
    EXTRA{OFF} {REV}LIFE{OFF} {REV}F
    OR"
230 PRINT"{REV}10{OFF} {REV}MINES{OFF} {R
    REV}PLACED{OFF} {REV}ON{OFF} {RE
    REV}THE{OFF} {REV}FIELD"
240 PRINT"{DOWN}IF THE BALL HITS THE CIRC
    LE THEN THE
250 PRINT"BALL WINS.
260 INPUT"{DOWN}RIGHTIST____RIGHT{08 LEFT}
    ";C$
270 INPUT"{DOWN}LEFTIST_____LEFT{07 LEFT}"
    ;B$
280 INPUT"{02 DOWN}DIFFICULTY 1...20  9{0
    3 LEFT}";DD
290 POKE59468,12
300 MC=INT(9*RND(0)+1)
310 REM SET UP PLAYING FIELD
320 PRINT"{CLEAR}":S=32768
330 FORI=0TO39:POKES+40+I,102:POKES+1000-
    40+I,102:NEXTI
340 FORI=2TO23:POKES+40*I,102:POKES+40*I+
    39,102:NEXTI
350 FORI=1TO25*DD:POKES+INT(36*RND(0)+2)+
    40*INT(20*RND(0)+3),102:NEXTI
360 U=-40:D=40:L=-1:R=1
370 B=S+81+160:BD=D:DB=D:C=S+78+40:DC=D:C
    D=D
380 POKEB,81:POKEC,87
390 GETP$:IFP$="@"THEN470
400 IFP$="="THEN480
410 IFP$="*"THEN490
420 X=PEEK(B+BD):IFX<>32THEN520
430 POKEB,32:B=B+BD:POKEB,81
440 X=PEEK(C+CD):IFX<>32THEN570
450 POKEC,32:C=C+CD:POKEC,87
460 GOTO390
470 DB=-1/DB:BD=BD*DB:GOTO390
480 DC=-1/DC:CD=CD*DC:GOTO390
490 MC=MC+1:IFMC=10THENMC=0:BL=BL+1:POKES
    +BL,81
500 IFPEEK(C-CD)<>102THENPOKEC-CD,42:GOTO
    390
510 GOTO390
520 IFX=102THENBD=-BD:GOTO440
530 IFX=87THENBS=BS+1:GOTO600
540 IFX<>42THEN430
550 BL=BL-1:POKES+BL+1,32:IFBL>=0THENGOTO
    430
560 CS=CS+1:GOTO600
570 IFX=102THENCD=-CD:GOTO390
580 IFX<>81THEN450
590 BS=BS+1
600 FORI=1TO100:GETP$:NEXTI
610 GETP$:IFP$=""THEN610
620 POKE 59468,14:PRINT"{CLEAR}{05 DOWN}
                   SCORE
630 PRINT"{03 DOWN}"B$" ="BS;TAB(37-LEN(C
    $+STR$(CS)));CS"= "C$
640 PRINT"{04 DOWN}NEW DIFFICULTY (YES, N
    O, END)
650 GETP$:IFP$="E"THENEND
655 MC=0:BL=0
660 IFP$="Y"THENPRINT"{CLEAR}":POKE59468,
    14:MC=0:BL=0:GOTO280
670 IFP$=""THEN650
680 GOTO290
```

©

COMAL: Another Language?

Jim Butterfield
Toronto, Canada

There are a lot of "languages" around for small computers. Only two are in common use: BASIC (most often by Microsoft) and Machine Language.

Most of the others classify as in-between languages: not as friendly as BASIC, not as fast as machine language. Each have their advocates (fanatics?) who extol the advantages which a specific language can bring to a specific application. But BASIC and machine language look like they will reign supreme for quite a while yet. Until local computer shops bristle with book titles such as "101 FORTH Games," "Some Common SMALLTALK Programs," and "Hands-On SNOBOL For The PET/CBM," many users will prefer to stay in the mainstream of the home computing community and learn one or two simple languages well.

Now we have COMAL to add to the computer Tower of Babel. Another language, another chance to sidetrack? COMAL does have its interesting features, especially to those raised on BASIC. Let's take a look.

Free!

COMAL is public domain. In other words, it's free; and you can help yourself to a friend's copy with a clear conscience. In Canada, Commodore has distributed copies to all dealers; you can get one by asking your dealer to make you a copy of the disk and documentation. In the USA, the COMAL Users Group, 5501 Groveland Terrace, Madison WI 53716, will send you the disk for a charge of $12.95; or for $47.50 you get a kit including disks, documentation, binder, and a newsletter subscription; in either case, add $2.00 for shipping/handling.

You get what you pay for, right? Not in this case: COMAL is a massive system (it will fit only into a 32K system with disk) and has features that make it well worth considering, particularly for educational use.

COMAL came into existence in Denmark. It was first defined by Borge R. Christensen and Benedict Leofstedt in 1974. At that time it was a form of extended BASIC. It has been expanded and refined into the current version, COMAL-80, by Mogens Kjaer.

Super-Basic?

COMAL still retains much of the flavor of BASIC, and for that reason it's a very easy language for BASIC users to pick up. The first impression that a user gets is rather intriguing: it seems as if you may type in your program in BASIC — and when you say LIST, it comes back in a PASCAL-like language!

How can this happen? Most BASIC users learn that their program is "tokenized" as it is input. The individual letters of PRINT get scrunched together and stored as a single byte called a token; that's why you can type in a line such as 100 ? A and list it back as 100 PRINT A. COMAL tokenizes your input to a remarkable extent, so that a line input as 5 FOR J = 1 TO 8 will list back as 5 FOR J: = 1 TO 8 DO. Note that a colon has crept in after the J, and that the word DO has been added.

So: with a few new rules, you may just type in a program in its BASIC form, and COMAL will adapt it into its own internal format. Some of the new rules are easy: for example, be sure to put a space after each keyword (don't say FORK, say FOR K). Others take a little more practice: subroutines are now called Procedures and, instead of GOSUB-ing to them, you EXEC-ute them. And you don't use line numbers for GOTO and GOSUB, you use names (or "Labels"). But these are not difficult. It's easy for a BASIC person to jump into COMAL.

...And More

But it's more than just a reworded BASIC system. There are a whole new series of capabilities.

Some are easy to understand and on most people's wish-lists. You may now use variables HORSESHOE and HOUSEFLY without confusion (in BASIC, only the first two letters are meaningful). IF has been beefed up to include ELSE and other features, allowing you to code IF M = 12 THEN M = 1; Y = Y + 1 ELSE M = M + 1 ENDIF. Note that we are using a semicolon instead of a colon to separate statements, and we terminate the IF sequence with an ENDIF. This isn't just needless bookkeeping: ENDIF allows us to set the range of the IF statement to part of a line or multiple lines.

Other COMAL features are recognizable as structured language extensions. The user will find CASE (replacing ON A GOTO...), WHILE, and REPEAT .. UNTIL. Procedures can be used as subroutines or as function definitions; and you may pass parameters to or from procedures. Strings require a little more care than in BASIC, but string handling is more powerful after you get used to it.

COMAL is fast. You'll see no loss of speed from BASIC.

However...

The language is nice, but it's new. You may have to wait a while before you find a community of other COMAL users around you. The BASIC language feature I miss most is the SYS command — at least I haven't found it yet. I like to be able to extend some programs with machine language inserts if necessary. Some 4.0 disk commands don't appear to be supported by COMAL; I haven't found a way to initiate a SCRATCH or COLLECT from the language. There doesn't seem to be a built-in exit to BASIC cold start, which would be a way around the previous problem.

COMAL for the PET/CBM comes in two forms; the smallest takes up 16K of memory. This may cramp the size of programs. COMAL does pack programs in memory more efficiently, but you have less space to work with from the start.

COMAL seems ideal for educational environments, particularly for those who like to teach structured programming techniques.

It's a helpful language in many ways: as easy as BASIC and very like it. It has goodies that BASIC can't match. The structure and balance of COMAL lead me to suspect that there will be a compiler along one of these days.

And the price is right. ©

CAPUTE!:

Corrections And Amplifications

—**COMPUTE!** #14, pg. 106, "The Apple Hi-Res Shape Writer." The following routine was missing (Program 2):

Program 2.

```
10 D$ =  CHR$ (4):INC = 10:S = 16
     384
20  PRINT D$"OPEN POKE ROUTINE"
30  PRINT D$"WRITE POKE ROUTINE"
90 LINE = 5000:B = 16500
100  PRINT LINE"FORI=16384TO"B":R
     EADA:POKEI,A:NEXTI"
120  FOR I = S TO B: IF (I - S) /
     10 =  INT ((I - S) / 10) THEN
      PRINT :LINE = LINE + INC: PRINT
     LINE"DATA" PEEK (I);: GOTO 1
     40
130  PRINT "," PEEK (I);
140  NEXT : PRINT : PRINT D$"CLOS
     E"
]LIST
```

—**COMPUTE!** #16, pg. 66: Line 62005 should read:

FOR I=LO TO HI

— **COMPUTE!** #16, pg. 118: "The = sign does concatenation...". No it doesn't! The + sign does concatenation. Who wrote that? Who is this guy Butterfield anyway? He deserves thirty lashes with a wet noodle.

Unless, of course, he wrote it correctly and somebody goofed it up down there. In which case, transfer the lashes (and the noodle) to the appropriate culprit. Heck, I have enough trouble spelling concatenation without worrying about how to do it...

The whole thing is bristling with = signs that shouldn't be there. M$=A$=B$=C$ should be M$=A$+B$+C$; PRINT J$=" "=M$ should be PRINT J$=" "+M$. However, to make up for it, you've changed a character the opposite way in your last paragraph: Z$=Z$+"+" should read Z$=Z$+"=".

Try typing in this line:

FOR J=1 TO 10:PRINT"TSK.":NEXT J

[Our thanks to Jim for his corrections. The typos (and the noodle) belong here.]

—**COMPUTE!** #16, pg. 134: line 9010 should read FOR I=4 TO 35*PV STEP 5*PV. The value POKEd in line 9520 should be 43.

—**COMPUTE!** #16, pg. 124: To permit the program to also work on the 8050 drive, change line 290 to PRINT#15, "M–E"CHR$(180)CHR$(255)

—**COMPUTE!** #16, pg. 10: Many readers suggested modifications to David Thornburg's excellent 20-questions program to permit more random responses and to prevent the same response if the questions always began: "is it animal," "is it vegetable," "is it mineral." One of Mr. Thornburg's points was the brevity of the program in contrast to the "intelligence" it seemed to evidence. There are indeed a variety of ways to make the program even more remarkable for Turing tests on the unsuspecting.

—**COMPUTE!** #10, pg. 112: To allow the disk program to work with the 4044 CBM disk drives (or 2040s with upgraded ROM), change line 5012 to: IF P(0) < 25 THEN SM =18 ©

ECLECTIC SYSTEMS CORPORATION

Authorized Commodore service center
Repair of the complete line of Commodore products
In a hurry? Check our modular exchange program

HARDWARE:		SOFTWARE:	
CBM 8032 Computer, 80 Column	$1195	OZZ	$299
CBM 8050 Disk Drive	1395	Wordcraft 80	299
CBM 4032 Computer, 40 Column	995	Tax Preparation System	380
CBM 4040 Disk Drive	995	IRMA	380
CBM 4022 Printer	649	Dow Jones Portfolio Management System	115
CBM VIC 20 Computer	263	Personal Tax	55
CBM VS100 Cassette	68	Pascal	229
PET to IEEE Cable	33	Assembler Development Package	77
IEEE to IEEE Cable	39	Wordpro 4+	329
BASF Diskette, Box of 10	30		

Order TOLL FREE 1+800-527-3135

10 AM to 4 PM CDT Monday through Friday

Texas residents call 1+214-661-1370

VISA, MASTER CHARGE, MONEY ORDERS, AND C.O.D. "Certified Check" accepted.
Units in stock shipped within 24 hours, F.O.B. Dallas, Texas.
All equipment shipped with manufacturer's warranty.

Residents of Texas, Louisiana, Oklahoma City and Tulsa, Oklahoma must add applicable taxes.

Eclectic shortly will be announcing products that are designed to work with CBM systems:

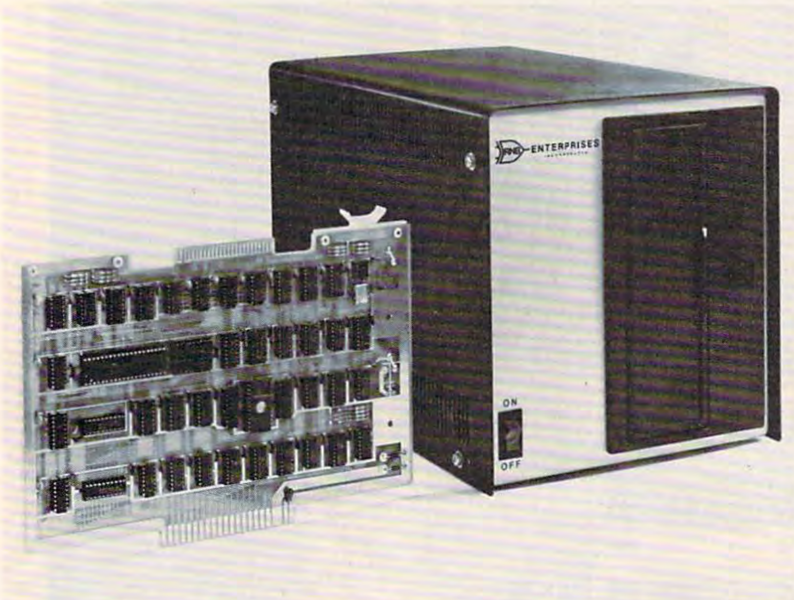
1. ROMIO: two RS232 ports—three parallel ports—26K EPROM memory-managed alternate character set, software controlled—EDOS (extended DOS).
2. Terminal program (options with ROMIO)
3. EPROM programmer
4. Front-end processor
5. Additional firmware to be announced

Be sure to write the address below for more information; dealer inquiries welcome.

P.O. Box 1166 • 16260 Midway Road • Addison, Texas 75001 • (214) 661-1370

VAK-7 8″ FLOPPY DISK SYSTEM

FOR AIM-65, SYM-1



The VAK-7 Disk System was specifically designed for use with AIM-65 and SYM-1 Microcomputer Systems. The VAK-7 is a complete full size (8″) Floppy Disk System. This system will Read, Write and Format IBM Single and Dual Density diskettes. The VAK-7 is available with both Single and Dual Sided Disk Drives. Each Disk Drive comes with its own cabinet and Power Supply. The VAK-7 can handle up to 4 disk drives, totaling more than 4.98 Megabytes of storage.

The VAK-7 Disk System incorporates both advanced hardware and innovative software designs. The addition of the VAK-7 produces a very powerful and useful computer system. Unlike most other disk systems, there is no requirement for the user to provide RAM to hold the Disk Operating System software. No valuable time is wasted loading in the DOS from cassette tape, because the VAK-7 DOS software is in onboard ROM. The VAK-7 is located above the 32K boundary (8000 HEX), leaving the user with a full 32K bytes of contiguous address space for his own use.

AIM-65—Allows the user to save and load object code thru the AIM Monitor; to load, save, and append Text thru the AIM Editor; to load, save, and append Basic Programs thru the BASIC INTERPRETER; to assemble directly from disk single or multiple file programs.

SYM-1—Allows the user to save and load Files for use with the SYM Monitor, SYM Basic, and RAE-1.

ADDITIONAL COMMANDS:

ACTIVATE A DELETED FILE
COMPRESS A DISK
RENAME A DISK FILE
COPY A DISK
FORMAT A DISK
DELETE A DISK FILE
INITIALIZE A DISK
LIST CATALOG

SPECIFICATIONS:

- Completely assembled, tested, and burned in.
- Occupies address 8000-8FFF for AIM-65, $9000-9FFF for SYM-1, or $E000-EFFF for KIM-1.
- IBM Format; Single Density (128 bytes/sector); Dual Density (256,512, or 1024 bytes/sector).
- All ICs are in sockets.
- Fully buffered address and data bus.
- Standard KIM-4* BUS (both electrical pin-out and card size).
- Designed for use with a regulated power supply, but has provisions for adding regulators for use with an unregulated power supply.
- Dimensions: Board—10″ wide x 7″ high (including card-edge). Cabinet—9.25″ wide x 10″ high x 16″ deep.
- Power Requirements: +5v DC @ 2 Amps.
  117 AC 60Hz @ 2 Amps.
- Shipping Weight: 25 lbs.

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

PRICING:

VAK-7 **$1,299.00**
Controller and One Single-Sided Drive

VAK-7A **$599.00**
Additional Single-Sided Drive with Cabinet and Power Supply

VAK-7B **$1,599.00**
Controller and One Dual-Sided Drive

VAK-7C **$899.00**
Additional Dual-Sided Drive with Cabinet and Power Supply

CALL OR WRITE FOR FREIGHT CHARGES

RNB ENTERPRISES INCORPORATED

4030 N. 27th Avenue, Suite D
Phoenix, AZ 85017
(602) 265-7564

Advanced Operating Systems Publishes 10 Program Household Package

Michigan City, IN, October 1, 1981 — Advanced Operating Systems has published a package of 10 programs useful in performing household duties. The programs are written in BASIC.

The "Mostly BASIC Household Program" package offers two programs which give a synopsis of energy consumption. The "Electric Energy Usage" selection compares energy units used during two different years. The "Gas Mileage Calculator" uses basic data to figure the gas mileage of a vehicle.

Two programs focusing on diet and eating habits of the user are the "Recipe Amount Calculator" and "The Basic Diet."

"The House Buying Guide" and "The Amortization Schedule" focus on budgeting and investing of money. The schedule calculates the balance, principal, interest, and cumulative interest for each month of a loan. The buying guide weighs answers from questions it gives the user and advises on the possibilities of making a profit from buying a structure. The program can be used whether the operator will be renting out the structure or selling it at a later date.

Two programs which offer no frills, but are unlimited in their usefulness, are the "Digital Stopwatch" and "The Message Taker." The stopwatch counts off minutes, hours, and seconds until told to stop. The message program records up to six messages and enables members of the household to leave "notes" for one another.

Medical expenses can also be cataloged through use of this package. The "Medical Expense Record files away the type of expense, cost, and cumulative total for each year.

"The Tarot Card Reader" is an entertaining program based on a deck of 78 cards used in fortune telling. The program answers questions from the user by picking 10 of the cards. The position they hold after being layed out has a meaning, as well as whether the figure on the card is right side up or upside down. The translation of the 10 cards is listed briefly by the computer. This program requires 16K to run.

The package is available through computer retail outlets. Advanced Operating Systems is the microcomputer software division of Howard W. Sams & Co., Inc., a subsidiary of International Telephone and Telegraph Corporation.

Smart Terminal Program For Atari® Features Autodialing

Redmond, WA — The MicroPeripheral Corporation has announced TSMART, the first smart terminal program written for the ATARI 800®, with provision for autodialing other computers. The program is availble on cassette with instructions for transferring to disk. TSMART permits transfer of BASIC programs between a remote host computer and an ATARI cassette or disk storage device. The autodial feature works in conjunction with the AUTO-MICROCONNECTION, a direct connection modem ($199.50), manufactured by the MicroPeripheral Corporation.

The program permits off-line text preparation (messages, manuscripts, letters, etc.) with a text editing or word processing program for on-line transmission. A built-in feature permits creation and storage of text, then transmission by TSMART for those who do not have a text editor.

TSMART also permits transfer of source code assembler files. The recipient can create the object code using an editor/assembler program. A separate command is available for transferring object (hexadecimal) code files, such as ATARI Music Composer Files.

An AUTOBUF feature will open and close the memory storage buffer automatically when uploading or downloading. TSMART recognizes the automatic buffer open/close (X-on/X-off) codes transmitted by TSMART or other compatible programs. Downloading from FORUM 80 bulletin boards is also accomplished automatically. The buffer can be "toggled" on and off as many times as desired while data is being downloaded. Another feature is software selectable half or full duplex operation.

The program will also automatically send messages to bulletin boards using the standardized block mode or 16 line prompt recognition message entry.

The program was written for

the ATARI 800® by Dr. James W. Clark. It can be used with any RS-232 compatible modem, although the dialer feature cannot be used with obsolete acoustic modems. TSMART is supplied in a protective binder with extensive easy-to-use operating instructions and is priced at $79.95. For additional information contact the Micro-Peripheral Corporation, 2643 151st Place N.E., Redmond, WA 98052, Telephone (206)881-7544.

Service Of Commodore Computers Begins At 38 TRW Service Centers

Valley Forge, PA, October 9, 1981 — Service of Commodore Business Machines, Inc., microcomputers has begun at 38 service centers operated by the Customer Service Division of TRW Inc.

As per a five-year agreement signed between Commodore and TRW in August, TRW will service and maintain Commodore microcomputers throughout the United States both on-site and at walk-in depots.

The first 38 service centers to complete comprehensive training on Commodore equipment and go on-line as part of the agreement cover some 75 percent of the nation's microcomputer users. Additional TRW centers will be brought on-line over the next year.

Commodore products covered by the TRW service agreement include the CBM™ 8032 central processing unit, the 4032N and 4032B central processors with 12-inch monitors and 8040 universal logic boards, 8050 disk drive, 4022 matrix printer, and 8010 communications modem.

The first group of TRW service depots handling Commodore equipment are in Atlanta, Birmingham (AL), Boston, Charlotte (NC), Chicago, Cincinnati, Columbus (OH), Dallas, Denver, Detroit, Fresno, Grand Rapids (MI), Hartford (CT), Houston, Indianapolis, Jacksonville, Little Rock, Los Angeles, Memphis, Miami, Milwaukee, New Orleans, Oklahoma City, Philadelphia, Phoenix, Pittsburgh, Richmond, Rochester, Sacramento, St. Louis, Salt Lake City, San Diego, San Francisco, Seattle, Springfield (NJ), Tampa, and Washington, DC.

"The opening of these TRW service centers for Commodore equipment effectively triples our service capability overnight," said James Finke, president and chief operating officer of Commodore.

consumer computers Mail Order DISCOUNTS

ORDER TOLL-FREE

NEC Microcomputer

PC-8001A 32K Computer	CALL
PC-8012A I/O Unit w/32K RAM	CALL
PC-8013A Dual Mini-Disk Drive Unit	CALL
PC-8001 Multi Cardware (FDI/O & 32K)	CALL
CP/M 2.2 Operating System for NEC	129
WordStar configured for NEC	299
SuperCalc configured for NEC	279
NEC Wordprocessor & Accounting Software	CALL

Many more software packages and languages; (Pascal, Fortran, Cobol, etc) are available configured for the NEC 8001A Computer.
Please call or write for a product price list.

ATARI 800 16K $749

Atari 400 w/16K	349
410 Program Recorder	65
810 Disk Drive	449
825 80 col. 7x8 Dot matrix impact printer	699
822 40 col. Quiet Thermal Printer	349
850 Interface Module	159
Atari 16K Ram Module	69
Axlon Ramcram 32K Module	189

Video Monitors

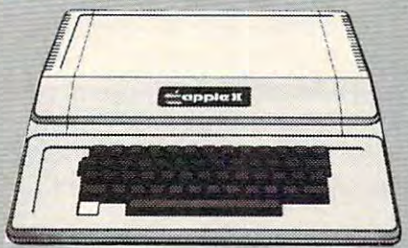
Amdek/Leedex Video 100 12" B&W	155
Amdek/Leedex Video 100G 12" Green Phospher	179
Amdek (Hitachi) 13" Color w/audio output	389
NEC 12" Green Phospher Display JB-1201M	CALL
NEC 12" Lo-Res Color Display	CALL
NEC 12" Hi-Res RGB Color Display	CALL
Sanyo 9" B&W Display	185
Sanyo 9" Green Phospher Display	CALL
Sanyo 12" B&W Display	269
Sanyo 12" Green Phospher Display	285
Sanyo 13" Color Display	449
Zenith 12" Green Phospher Display ZVM-121	149

ZENITH 12" GREEN $149

VIC20™ $259

Personal Computer
Color · Sound · Graphics
Call or write for more info.
Disk drives available soon!

apple computer
Authorized Dealer



APPLE II PLUS

16K	NOW	$1025
48K	NOW	$1089
64K*	NOW	$1199

*48K Apple with 16K RamBoard

APPLE DISK DRIVES $439

w/controller and DOS 3.3 **$499**

Apple Cards and Hardware

Language System w/Pascal & BASICS	379
Silentype Printer w/Interface card	349
Hayes Micromodem II	299
Novation Apple-Cat	339
Videx Videoterm 80 column card	269
Videx Keyboard Enhancer	115
Z-80 Softcard by Microsoft	299
16 K RamCard by Microsoft	169
CPS Multi-function card	189

Software for the Apple

VisiCalc version 3.3	159
VisiFile (NEW data base manager)	199
VisiTrend/VisiPlot	219
DB Master	169
WordStar (Apple 80 col. version)	249
Dow Jones Portfolio Evaluator	45
Apple Post	45
Apple Writer	65
Dow Jones News & Quotes Reporter	85
Apple Plot	60
Tax Preparer	99
Real Estate Analyzer	129

16K RAMBOARD by ConComp
for Apple II Computers

FOR ONLY $129.95

AVAILABLE NOW

S-100
California Computer Systems

Floppy Disk Controller	**$369**
64K Dynamic Ram Board, 200ns	**$499**
Z-80 CPU board w/monitor ROM	**$269**
16K Static memory board, 200ns	369
32K Static memory board, 200ns	599
S-100 12 Slot Mainframe	475
4-Port Serial Interface	299
2-Port Serial/2-Port Parallel Interface	299
4-Port Parallel Interface	229

Printers

Silentype w/Apple II interface **$349**

Epson MX-80 or MX-80 FT CALL

Anadex 9501 w/2K Buffer	1349
C. Itoh Starwriter 25 CPS daisywheel	1449
C. Itoh Starwriter 45 CPS daisywheel	1649
Epson MX-70	CALL
Epson MX-80 & MX-80 F/T	CALL
Epson MX-100	CALL
NEC 8023 Impact Dot Matrix	695
NEC Spinwriters (Latest models)	CALL
Paper Tiger IDS-445G w/graphics	699
Paper Tiger IDS-460G w/graphics	949
Paper Tiger IDS-560G w/graphics	1249
Silentype Printer w/Apple interface	349
Qume Sprint Daisywheels (Latest models)	CALL

ORDER TOLL FREE 800-854-6654

In California and outside continental U.S.
(714) 698-8088
Telex 695-000 Beta CCMO

Send Orders To:

Ordering information: Phone orders using VISA, MASTERCARD, AMERICAN EXPRESS, DINER'S CLUB, CARTE BLANCHE, bank wire transfer, cashier's or certified check, money order, or personal check (allow ten days to clear). Unless prepaid with cash, please add 5% for shipping, handling and insurance. (minimum 5.00). California residents add 6% sales tax. We accept CODs. OEM's, Institutions and corporations please send for a written quotation. All equipment is subject to price change and availability without notice. All equipment is new and complete with manufacturer's warranty (usually 90 days). Showroom prices may differ from mail order prices.

consumer computers Mail Order
8314 Parkway Drive
La Mesa, Calif. 92041

Okidata Adds Interface Options To Microline Printers

Mount Laurel, NJ, October 14 — Okidata Corporation, a supplier of quality dot-matrix printers announced the availability of significant new interface options for its Microline family of printers. These options are an IEEE 488 bus adapter and a current loop interface.

The bus adapter option will make all new and existing Microline printers compatible with the IEEE 488 bus. Users of Commodore Pet personal computers will find this option particularly helpful in integrating Microline printers into systems. The IEEE 488 bus adapter option comes in the form of a plug-in board which is easily installed by the user. The connecting cable converts Centronics-compatible parallel data into data compatible with the IEEE 488 bus.

The IEEE 488 bus adapter option will be available later this month. This feature is priced competitively, and quantity pricing is availalble. The board can be used with the Microline 82A, 83A, and 84.

For those users who need a current loop interface, the optional Microline RS-232C high speed serial interface now offered will add a current loop interface capability as a standard feature. The interface board can also be used with the Microline 82A, 83A, and 84.

The high speed RS-232C interface board comes with two different buffer sizes: a two-kilobyte memory or a 256-character storage buffer. The interface has switch-selectable baud rates of up to 9,600 bits per second. The built-in current loop feature will be available soon. The single quantity price for the RS-232C board with two kilobytes of buffer memory and the current loop interface is $160; the price for the one with 256-characters of memory is $130.

Okidata Corporation is headquartered at 111 Gaither Dr., Mt. laurel, NJ 08054.

Software For The Very Young

In October Edu-Ware Services, Inc. releases a new addition to the EDU-WARE line, COUNTING BEE. This friendly system introduces young learners (ages 3-6) to counting, addition, subtraction, shape discrimination, weight, and measurement.

COUNTING BEE features a learning management mode, which allows parents and teachers to preset the system, with emphasis and duration tailored to an indi-

ATARI

Computers for people.™

800™ $749

410 Recorder	$59.00
810 Disc Drive	$444.00
822 Printer	$359.00
825 Printer	$629.00
830 Modem	$159.00
820 Printer	$269.00
850 Interface	$159.00
New DOS 2 System	$21.00
CX70 Light Pen	$64.00
CX30 Paddle	$18.00
CX40 Joy Stick	$18.00
CX853 16K RAM	$89.00
Microtek 16K RAM	$75.00
Microtek 32K RAM	$169.00

400™ $329

ATARI SOFTWARE

CX404 Word Processor	$119.00
CX404 PILOT	$68.00
CX413 Microsoft Basic	$68.00
CX4101 Invitation To Programing I	$17.00
CX4102 Kingdom	$13.00
CX4103 Statistics	$17.00
CX4104 Mialing List	$17.00
CX4105 Blackjack	$13.00
CX4106 Invitation to Programing 2	$20.00
CX4107 Biorythm	$13.00
CX4108 Hangman	$13.00
CX4109 Graph It	$17.00
CX4110 Touch Typing	$20.00
CX4111 SPACE INVADERS	$17.00
CX4112 States & Capitals	$13.00
CX4114 European Countries & Capitals	$13.00
CX4115 Mortgage & Loan Analysis	$13.00
CX4116 Personal Fitness Program	$59.00
CX4117 Invitation To Programing 3	$20.00
CX4118-20 Conversational Languages (ea.)	$45.00
CX4121 Energy Czar	$13.00
CXL4001 Educational Master	$21.00
CX6001-17 Talk & Teach Series (ea.)	$23.00
CX8106 Bond Analysis	$20.00
CX8107 Stock Analysis	$20.00
CX8101 Stock Charting	$20.00
CXL4002 Basic Computing Language	$46.00
CXL4003 Assembler Editor	$46.00
CXL4004 Basketball	$24.00
CXL4005 Video Easel	$24.00
CXL4006 Super Breakout	$30.00
CXL4007 Music Composer	$45.00
CXL4009 Chess	$30.00
CXL4010 3-D Tic-Tac-Toe	$24.00
CLS4011 STAR RAIDERS	$39.00
CXL4012 MISSLE COMMAND	$32.00
CXL4013 ASTEROIDS	$32.00
CXL4015 TeleLink	$20.00
Visicalc	$149.00
Letter Perfect (Word Processor)	$109.00
Source	$89.00

commodore

CBM 8032 $1149

4016	$799.00
4032	$999.99
8096	$1795.00
CBM4022 Printer	$629.00
Tally 8024	$1699.00
CBM C2N Cassette Drive	$69.00
CBM4040 Dual Disk Drive	$1039.00
CBM8050 Dual Disk Drive	$1349.00

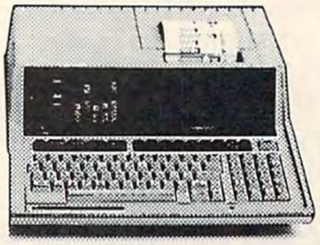
SOFTWARE

WordPro3 Plus	$229.00
WordPro4 Plus	$329.00
Commodore Tax Package	$399.00
Visicalc	$149.00
BPI General Ledger	$329.00
OZZ Information System	$329.00
Dow Jones Portfolio	$129.00
Pascal	$239.00
Legal Time Accounting	$449.00
Word Craft 80	$289.00
Create-A-Base	$249.00
Power	$89.00
Socket-2-Me	$20.00
Jinsam	$Call
MAGIC	$ Call

VIC 20 $259

Vic-TV Modual	$19.00
Vic Cassette	$69.00
Vic 6 Pack Program	$44.00
VIC1530 Commodore Datassette	$69.00
VIC1540 Disk Drive	$499.00
VIC1515 VIC Graphic Printer	$399.00
VIC1210 3K Memory Expander	$32.00
VIC1110 8K Memory Expander	$53.00
VIC1011 RS232C Terminal Interface	$43.00
VIC1112 VIC IEEE-488 Interface	$86.00
VIC1211 VIC 20 Super Expander	$53.00
VIC1212 Programmers Aid Cartridge	$45.00
VIC1213 VICMON Machine Language Monitor	$45.00
VIC1901 VIC AVENGERS	$23.00
VIC1904 SUPERSLOT	$23.00
VIC1906 SUPER ALIEN	$19.00
VIC1907 SUPER LANDER	$23.00
VIC1908 DRAW POKER	$23.00
VIC1909 MIDNIGHT DRIVE	$23.00
VT106A Recreation Pack A	$44.00
VT107A Home Calculation Pack A	$44.00
VT164 Programmable Character/Gramegraphics	$12.00
VT232 VICTerm I Terminal Emulator	$9.00

HEWLETT PACKARD



HP•85 $2595

NEW! HP•125	$3295.00
HP•83	$1795.00
HP•85 16K Memory Module	$249.00
5¼" Dual Master Disc Drive	$2129.00
Graphics Plotter (7225B)	$2079.00

Call for HP Software Prices & Information.
Call for Calculator prices.

Texas Instruments

TI-99/4 $379

PHC 004 TI-99/4 Home Computer	$379.00
PHP 1600 Telephone Coupler	$169.00
PHP 1700 RS-232 Accessories Interface	$169.00
PHP 1800 Disk Drive Controller	$239.00
PHP 1850 Disk Memory Drive	$389.00
PHP 2200 Memory Expansion (32K RAM)	$239.00
PHA 2100 R.F. Modulator	$43.00
PHP 1100 Wired Remote Controllers(Pair)	$31.00
PHM 3006 Home Financial Decisions	$26.00
PHM 3013 Personal Record Keeping	$43.00
PHD 5001 Mailing List	$60.00
PHD 5021 Checkbook Manager	$18.00
PHM 3008 Video Chess	$60.00
PHM 3010 Physical Fitness	$26.00
PHM 3009 Football	$26.00
PHM 3018 Video Games I	$26.00
PHM 3024 Indoor Soccer	$26.00
PHM 3025 Mind Challengers	$22.00
PHM 3031 The Attack	$35.00
PHM 3032 Blasto	$22.00
PHM 3033 Blackjack and Poker	$22.00
PHM 3034 Hustle	$22.00
PHM.3036 Zero Zap	$18.00
PHM 3037 Hangman	$18.00
PHM 3038 Connect Four	$18.00
PHM 3039 Yahtzee	$22.00
PHM 3017 Terminal Emulator I	$39.00
PHM 3026 Extended Basic	$88.00
PHM 3035 Terminal Emulator II	$45.00

DISCS

Sycom Blank Disk (10)	$29.00
Maxell MD I	$36.00
Maxell MD II (10)	$46.00

PRINTERS

Epson MX-70	Call for Prices
Epson MX-80	
Epson MX-80 FT	
Diablo 630	
TEC 1500 Starwriter 25cps	$1495.00
TEC 1500 Starwriter 45cps	$1795.00

No Risk, No Deposit On Phone Orders, COD or Credit Card, Shipped Same Day You Call *

* on all in stock units

IN PA, CALL (717) 327-9575

(800) 233-8950

COMPUTER MAIL ORDER

501 E. THIRD ST., WILLIAMSPORT, PA 17701

OVER 40 YEARS EXPERIENCE IN SOPHISITICATED ELECTRONICS

VISA

To Order:

Phone orders invited (800 number is for order desk only). Or send check or money order and receive free shipping. Pennsylvania residents add 6% sales tax. Add 3% for Visa or M.C. Equipment is subject to price change and availability without notice. Please call between 11 AM & 6 PM.

vidual child's needs. COUNTING BEE joins other EDU-WARE systems designed for young learners, SPELLING BEE and ARITHMETIC SKILLS. The sys uses Applesoft, 48K, DOS 3.2 or 3.3, and retails for $29.95.

Special Learning Ed Software Presents Educational Software For The PET®

The program consists of individualized computer-based instructional activities in spelling for students with specific language learning disabilities. However, this program will improve the skills of any persons with spelling deficits. The ramifications of using the tapes are many which include teaching average students, slow learners, and those who are studying English as a second language. The program can be used by anyone who can read.

The author is a Language Therapist who has for many years tutored students in grades kindergarten through twelve. These students with specific learning differences are in need of specialized tutoring in the language areas.

The spelling program includes spelling rules, exceptions, and generalizations which provide repetitive exercises and reinforcement as well as motivation to the learner. The drill and practice which persons with specific language learning disabilities require to learn to spell can be provided through tapes to be used on the PET Commodore. All tapes work with any 40-column PET, old or new.

The program consists of 21 tapes covering five basic spelling rules essential to encode words. The complete program is also available on disk.

The copyrighted program includes:

F-L-S-or Z rule	4 Tapes / $39.80
Doubling Rule or One, One, One	3 Tapes / $29.85
Silent E Rule	4 Tapes / $39.80
The Y Rule	4 Tapes / $39.80
Extension of the Doubling Rule	6 Tapes / $59.70

You can SAVE by purchasing the entire series, tapes or disk, for $199.00. Individual spelling rules can be purchased.

For further information on this program write to:

SLED SOFTWARE
P.O. Box 16322
Minneapolis, MN 55416
Telephone (612)926-5820

Lyco Computer Marketing & Consultants

We Specialize in Quality, Knowledge, Service, and Microcomputers

717-435-5197

SANTA'S X-MAS SPECIALS

ATARI®

800	16K	$744.00
400	16K	$329.00

ATARI HARDWARE

410 CASSETTE RECORDER	$ 57.00
810 DISK DRIVE	$444.00
850 INTERFACE	$159.00
825 PRINTER	$525.00
830 PHONE MODEM	$135.00

PROGRAMMING SOFTWARE

CX4101 INVITATION TO PROG	$ 17.00
CX4106 INVITATION TO PROG. 2	$ 19.00
CX4117 INVITATION TO PROG. 3	$ 19.00
CXL4003 ASSEMBLER EDITOR	$ 42.00
CXL4002 ATARI BASIC CART	$ 42.00
CXL4015 TELELINK	$ 19.00

EDUCATIONAL SOFTWARE

CXL4001 EDUCATIONAL SYSTEM	$ 19.00
CX6001 U.S. HISTORY	$ 22.00
CX6002 U.S. GOVERNMENT	$ 22.00
CX6003 SUPERVISORY SKILLS	$ 22.00
CX6004 WORLD HISTORY	$ 22.00
CX6005 BASIC SOCIOLOGY	$ 22.00
CX6006 COUNSELING PROCED	$ 22.00
CX6007 PRINCIPLES OF ACCT.	$ 22.00
CX6008 PHYSICS	$ 22.00
CX6009 GREAT CLASSICS	$ 22.00
CX6010 BUS. COMMUNCIATION	$ 22.00
CX6011 BASIC PSYCHOLOGY	$ 22.00
CX6012 EFFECTIVE WRITING	$ 22.00
CX6014 PRINCIPLES OF ECON	$ 22.00
CX6015 SPELLING	$ 22.00
CX6016 BASIC ELECTRICITY	$ 22.00
CX6017 BASIC ALGEBRA	$ 22.00
CX4108 HANGMAN	$ 12.00
CX4112 STATES & CAPITALS	$ 12.00
CX4114 EUROPE COUNT. & CAP.	$ 12.00
CX4121 ENERGY CZAR	$ 12.00
CX4123 SCRAM	$ 12.00
CX4102 KINGDOM	$ 12.00

ENTERTAINMENT

CXL4004 BASKETBALL	$ 29.00
CXL4005 VIDEO EASEL	$ 29.00
CXL4006 SUPER BREAKOUT	$ 29.00
CXL4009 CHESS	$ 29.00
CXL4010 3-D TIC-TAC-TOE	$ 29.00
CXL4011 STAR RAIDERS	$ 30.00
CXL4012 MISSILE COMMAND	$ 30.00
CXL4013 ASTEROIDS	$ 30.00
CX4015 BLACKJACK	$ 12.00
CX4111 SPACE INVADERS	$ 16.00

ACCESSORIES

CX30 PADDLES	$ 17.00
CX40 JOYSTICKS	$ 17.00
CX853 16K RAM	$ 85.00
******32K RAM by MICROTEK	$169.00

WE CARRY MANY OTHER LINES OF MICROCOMPUTERS YOU CAN CALL FOR PRICES ON:

CPU's	PRINTERS
COMMODORE	TEC
ALTOS	DIABLO
TEXAS INSTRUMENT	NEC
ZENITH	EPSON
INTERTEC DATA	ANADEX
AND OTHERS	AND OTHERS

At Lyco Computers we offer our expert services to help customers make their first computer purchase, schools establish a computer program, or evaluate multiterminal systems.

to help evaluate your needs, or if you wish to make a purchase.

CALL US AT 717-435-5197

LYCO COMPUTERS
P.O. BOX 10
COGAN STATION, PA 17728

ORDERS SHIPPED SAME DAY

NO COD CHARGES

WE PAY FREIGHT ON PRE-PAID ORDERS

PDI Establishes Division To Produce And Distribute Games

Program Design, Inc., the 4-year old Greenwich, CT educational software company is forming a division to produce and distribute entertainment software for the Atari 400 and 800 computer.

The division, called BEYOND SOFTWARE, is designed with independent software writers in mind. "We are interested in acquiring high-quality arcade, space, and simulation games," says PDI President, John Victor. "We are setting up a system that will cater to the needs of games designers. We will offer attractive terms, plus consulting services to top programmers."

BEYOND SOFTWARE is being established because the Program Design management feels the Atari computers offer exceptional entertainment possibilities.

BEYOND SOFTWARE will be managed by Craig Patchett, the author of CAPTIVITY, a 3-D maze game, and several other game programs.

For additional information, contact Patchett at Program Design, 11 Idar Court, Greenwich, CT 06830; (203)661-8799.

Six New SuperPET Books Now Available From Commodore

Valley Forge, PA, October 5, 1981 — Commodore Business Machines, Inc., has announced the availability of six new reference books to be used with its SuperPET "micro-mainframe-style" computer. The books are provided with the SuperPET system, but can also be purchased separately.

The SuperPET, which is Commodore's latest product addition, offers expanded capabilities by providing 96K RAM, an additional microprocessor, five languages, and a standard data communications interface.

The books include a System Overview of the SuperPET, and one book for each of the five languages available with the product — Waterloo microAPL, Waterloo microBASIC, Waterloo microFORTRAN, Waterloo microPASCAL, and Waterloo 6809 Assembler.

The System Overview book provides an introduction to the hardware of the SuperPET, an overview of the Waterloo software for the computer, and various descriptions that apply to the Waterloo software systems in general. The book retails for $5.95.

The Waterloo microAPL book, which retails for $9.95, is a tutorial introduction to the language, as well as a comprehensive reference manual.

The reference series also includes a Waterloo microBASIC book, which is divided into four parts: an introduction to the general characteristics of the system; a comprehensive reference guide describing the command language; an additional reference guide describing the programming language; and appendices containing summaries of both command and programming languages, as well as describing use of files with Waterloo microBASIC. It retails for $10.95.

The Waterloo microPascal book, also retailing for $10.95, features a tutorial introduction of Pascal language, and a reference manual defining the language.

The Waterloo microFORTRAN book is also divided into tutorial and reference sections, and retails for $10.95.

The final book in the series, pertaining to the Waterloo 6809 Assembler, describes the 6809

APPLE • ALTOS • ATARI • MAXELL • DYSAN • EPSON • CCS • SHARP • CASIO • HP • VERBATIM • MEMOREX • SOROC • CORVUS • ADDS

BMC • CENTRONIC • NEC • HAZELTINE • AMPEX • VERBATIM • HITACHI • HAYDEN • CANON • NOVATION • D.C. HAYS MICROCOMPUTERS • PAPER TIGER • PERSONAL SOFTWARE • GTE •

HP • D.C. HAYS MICROCOMPUTERS • PAPER TIGER • GTE • DIABLO • BMC • CENTRONIC • NEC • HAZELTINE • MEMOREX • AMPEX • PERSONAL SOFTWARE • ADDS • XEROX • DIABLO

MICRO BUSINESS WORLD INC.

Information Line (213) 996-2252
TOLL FREE MAIL ORDER LINES
(800) 423-5886 Outside Calif.

SPECIAL OF THE MONTH

APPLE II PLUS 48K W/16K EXPANSION BOARD = 64K

Our price $1199.00

List price $1780.00
Our price $1199.00
Save $581.00

COMPUTERS

apple computer

	List	Our Price	SAVE
Apple II-16K	$1330.00	$999.00	$331.00
Apple II-32K	1430.00	SPECIAL	$
Apple II-48K	1530.00	CALL	$
Apple II-64K	1725.00		$
Apple III 128-K	3915.00	2997.00	918.00
ALTOS 8CS 8000-15	5990.00	4450.00	1540.00
SHARP-64K Z80 FULL KBD YX-3200		CALL	
NEC PC 8001	1295.00	1050.00	245.00

ATARI PERSONAL COMPUTERS	List	Our Price	SAVE
400 16K Bytes of Ram	595.00	337.45	257.05
800 16K Bytes of Ram	$1080.00	739.00	332.00
410 Program Recorder	90.00	77.00	13.00
810 Disk Drive	600.00	457.00	143.00
825 Printer (80 col-Centronic 737)	999.95	769.00	230.00
820 Printer (40 col impact)	450.00	353.00	97.00
830 Acoustic Modem	200.00	155.00	45.00
850 Interface Module	220.00	192.00	48.00
Atari Visicalc	200.00	164.00	36.00
Atari 400-16K	595.00	327.00	304.00

HEWLETT PACKARD	List	Our Price	SAVE
HP-125 Microcomputer	3750.00	2990.00	760.00
HP-85 Microcomputer	3250.00	2475.00	775.00
HP-83 Microcomputer	2250.00	1777.00	473.00
16K Exp-Memory Module	295.00	255.00	40.00
Graphics Plotter 7225	2450.00	2075.00	375.00
Personality Mod. for 7225	750.00	665.00	85.00
2631B Impact/printer/hvy dty	3950.00	3250.00	700.00
Option 020 for 2631B	150.00	125.00	25.00
8 Drives to choose from 82902S	1300.00	1125.00	195.00
9895A 8" Dual Drive	6850.00	5500.00	1350.00
Graphics Tablet 9111A	2050.00	1678.00	374.00
HP-41 CV New 2.2 bytes mem	325.00	250.00	75.00
HP-41-C Calculator	250.00	185.00	65.00
Card Reader for 41CV/C	215.00	162.00	53.00
Printer for 41CV/C	385.00	289.00	101.00
Optical Wand for 41 CV/C	125.00	97.00	28.00
Quad Ram equals 4 Mem. Mods	95.00	81.00	14.00
Memory Modules for 41C		25.00	$
HP-97 Programable Printer	750.00	595.00	175.00
HP-67 Programable Calculator	375.00	295.00	80.00
HP-34C Programable Scientific	150.00	117.00	33.00
HP-38C Programable Bus. R/E	150.00	117.00	33.00
HP-32E Adv. Scientific	55.00	48.00	7.00
HP-37E Business Mgmt.	75.00	57.00	18.00

We carry a large inventory of Libraries, accessories and supplies.

PRINTERS

	List	Our Price	SAVE
EPSON			
MX 80 FT	745.00	549.00	196.00
MX 80 IMPACT	645.00	450.00	195.00
MX 70 IMPACT	500.00	390.00	110.00
MX 100	995.00	765.00	230.00
APPLE SILENTYPE	645.00	299.00	346.00
ANADEX 9501	1650.00	1299.00	351.00
NEC			
5510	3195.00	2445.00	750.00
5515	3295.00	2545.00	750.00
3510	2495.00	1795.00	700.00
3515	2545.00	1849.00	696.00
OKIDATA			
MICROLINE 80	545.00	395.00	150.00
MICROLINE 82	649.00	549.00	100.00
MICROLINE 83	1050.00	769.00	281.00
PAPER TIGER			
445G with Graphics	795.00	695.00	100.00
460G with Graphics	1,394.00	899.00	495.00
560G New full size	1,695.00	1,139.00	556.00
DIABLO (LETTER QUALITY)			
630 R102 bi-directional tractors	2,965.00	2,350.00	615.00
1640K109 keyboard tractors	4,000.00	2,899.00	1,100.05
630 RO Receive Only	2,710.00	2,250.00	460.00
1650K 136 keyboard tractors	4,000.00	3,100.00	900.00

APPLE II STUDENT SYSTEM

- Apple II Plus System-48K RAM
- Disk II Floppy Disk & Interface (DOS 3.3)
- 12" Grn. Phs. Video Monitor

SAVE $655

Our Price $1795

List $2450.00

MONITORS

	List	Our Price	SAVE
NEC Grn. Phs. 12"	$285.00	239.99	$46.00
BMC Green Phs. 12"	275.00	229.00	46.00
Zenith 12"	159.00	119.00	30.00

SANYO MONITORS
High Resolution, Number 1 seller

	List	Our Price	SAVE
13" Color (new) high quality	$550.00	388.00	$162.00
12" Green Phosphorous	360.00	240.00	120.00
12" Black & White	340.00	217.00	123.00
15" Black & White	370.00	235.00	135.00
9" Black & White The Best Seller!	235.00	145.00	90.00

DISKETTES

SOLD IN BOXES OF 10
(Min. Purchase)
BUY $100 of the Following Diskettes

	List	Our Price	SAVE
DYSAN		PRICE PER DISKETTE	
104/1 5" SOFT SECTOR	$ 6.00	3.99	$ 2.01
104/1D = " DBL. DEN.-SOFT SEC.	6.40	4.60	1.80
3740/1 8" SOFT SECTOR	7.25	4.75	2.50
3740/1D 8" DBL. DEN.-SOFT SECTOR	10.75	6.90	3.85
MAXELL			
MD-1 5" SOFT SECTOR/DBL. DEN.	5.00	4.00	1.00
MD-2 = " SOFT SECTOR/DBL. SIDE/DBL. DEN.	7.00	4.90	2.10
FD-1 8" SOFT SEC./DBL. DEN.	6.50	4.50	2.00
FD-2 8" SOFT SEC./DBL SIDE/DBL. DEN.	8.50	5.95	2.55

TERMINALS

TELEVIDEO	List	Our Price	SAVE
910	699.00	599.00	100.00
912C	950.00	699.00	251.00
920C	995.00	795.00	200.00
950C	1,195.00	949.00	246.00

DRIVES

CORVUS	List	Our Price	SAVE
5 MBYTES	3,750.00	3,050.00	700.00
10 MBYTES	5,350.00	4,449.00	901.00
20 MBYTES	6,450.00	5325.00	1,125.00

SOFTWARE

FOR APPLE II

	List	Our Price	SAVE
Apple Fortran	$ 200.00	$ 147.00	$ 53.00
DOS 3.3 (16 Sector)	60.00	45.00	15.00
Apple PILOT (16 Sector)	150.00	119.00	31.00
Apple FORTRAN (requires A2B0006) (16 Sector)	200.00	159.00	41.00
Language System with Apple Pascal	495.00	399.00	96.00
BPI General Ledger System	395.00	319.00	76.00
BPI Inventory Package	395.00	319.00	76.00
Visidex	200.00	159.00	41.00
Visicalc	200.00	159.00	41.00
Desktop Plan II	200.00	159.00	41.00
Microlab Database System	150.00	119.00	31.00
Stoneware DB Master	229.00	189.00	40.00
Muse Supertext II	150.00	119.00	31.00
Softape Magic Window	99.00	72.00	27.00

TEXAS INSTRUMENTS

	List	Our Pr	SAVE
TI 99/4A Console New Improved	950.00	385.95	564.05
10" Color Monitor High Resolution	374.95	317.95	57.00
32K Memory Module	399.95	312.95	87.00
Speech Synthesizer	149.95	127.95	22.00
Disk Memory Drive	499.95	390.95	109.00
RF Modulator	49.95	42.50	7.45
Telephone Coupler (Modem)	224.95	185.95	39.00
Printer (Solid State)	399.95	315.95	84.00

MODEMS

	List	Our Price	SAVE
NOVATION CAT MODEM	189.95	140.00	49.95
NOVATION D-CAT	199.00	150.00	49.00
NOVATION APPLE-CAT	349.00	310.00	39.00
HAYES MICROMODEM	379.00	295.00	84.00
HAYES SMARTMODEM	279.00	239.00	40.00

CALCULATORS

CASIO	List	Our Price	SAVE
HR-10	49.95	39.00	10.95
HR-12	54.95	42.00	12.95
FR-100	79.95	59.00	20.95
FR-1210	129.95	99.00	30.95
PQ-20	29.95	23.00	6.95
LC-785	12.95	10.00	2.95
LC-3165	12.95	10.00	2.95
FX-68 Scientific	29.95	23.00	6.95
FX-81 Scientific	19.95	17.00	2.95
FX-3600P Scientific	39.95	29.95	10.00
FX-602P "Computer Talk" 88 Memories Programming Upper & Lower Case Dot Matrix	129.95	99.00	30.95
FX-702P Solves Problems with Alpha-Numeric Clarity, uses Basic Language	199.95	159.00	40.95

TELE. ANSW DEVICES

PHONE MATE	List	Our Pr	SAVE
900	119.95	86.00	33.95
905 Remote	199.95	144.00	55.95
910	159.95	115.00	44.95
920	199.95	144.00	55.95
925 Remote	239.95	173.00	66.95
930 Remote	299.95	216.00	83.95
950 Remote	339.95	245.00	94.95
960 Remote	399.95	288.00	111.95

1 Year Extended WARRANTY
$99.00
INQUIRE

$AVINGS
ERVICE
ELECTION
ATISFACTION

MEMBER BBB

MICRO BUSINESS WORLD MAIL ORDER WAREHOUSE
18720 Oxnard, #108 Tarzana, CA 91356
Telex: 182852
Answer: MICRO TZNA

OUTSIDE CA CALL TOLL FREE 1 (800) 423-5886 IN CA (213) 996-2252

Name (Please Print) ______
Address ______
City ______ State ______ Zip ______

Qty	Make	Model	Description	Price	Total

TOTAL ORDER $ ______
TAX IF APPLICABLE* ______
SHIPPING & HANDLING** ______
TOTAL ENCLOSED $ ______

*California residents add 6% sales tax.
**Add 3% for Shipping & Handling — Add 3% surcharge for credit cards. Orders cannot be shipped unless accompanied by payment, including shipping handling and tax where applicable.

Certified Check or M.O. Bank Wire Transfer
Cash/Credit Card (Allow 2 weeks clearance for checks)

CREDIT CARD # ______
Exp. Date ______ Signature ______

We reserve the right to correct typographical errors. This ad supercedes all previous ads.

MAXELL • DYSAN • EPSON • CCS • SHARP • CASIO • HP • VERBATIM • MEMOREX • SOROC • CORVUS • PERSONAL SOFTWARE • CCS

Assembler, Linker, and Monitor systems. It contains all the details necessary to develop and debug programs written in the 6809 assembly language for the SuperPET. Retail price is $10.95.

An additional book for COBOL will be available in the first quarter of 1982. These books may be ordered through Commodore dealers nationwide.

Link Systems Introduces DataFax™

Santa Monica, CA, October 19, 1981 — DataFax™, a new approach to information management, is now being marketed by Link Systems. This new Pascal program is designed to allow the user to enter and access information according to individual needs without programming.

DataFax™ allows the user to enter data in virtually any form and to retrieve it in an individually meaningful way...thus eliminating the constrictions of set programs. Each screen of information you enter can be cross-referenced or categorized by the keywords associated with them such as names, dates, or any categories relevant to your use. Screens may be chained together if necessary and a hard copy obtained.

The DataFax™ user enters all data via a simple screen editor... anywhere on the screen. The screen editor features cursor movement, along with tabbing and word tabbing. Data can be printed exactly as it appears on the screen or in a personalized report form.

With the cross-referencing and indexing feature, a set of keywords can be entered for each screen. Keywords may be logically ANDed and ORed in any combination, thus providing a powerful data retrieval mechanism.

The program runs on the Apple II™ and will soon be available for the Apple III™. The suggested retail price is $250.

All Link software is hard disk compatible. For further information, or to order, contact Pat Merryman at (213)453-1851. Or write to Link Systems, 1655 26th St., Santa Monica, CA 90404.

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. ©

commodore spectacular

8032-32K 80 COL CRT	
REG. $1495	**$1095**
4032-32K 40 COL CRT	
REG. $1295	**$995**
4016-16K 40 COL CRT	
REG. $995	**$795**
64K ADD ON MEMORY	
REG. $500	**$395**
9000-134K SUPER PET	
REG. $1995	**$1795**
8050-DUAL DISK 950K	
REG. $1795	**$1345**
4040-DUAL DISK 343K	
REG. $1295	**$995**
2031-SINGLE DISK 170K	
REG. $695	**$555**
C2N-CASSETTE DRIVE	
REG. $95	**$69**
4022-80 COL PRINTER	
REG. $795	**$629**
8023P-136 COL PRINTER	
REG. $995	**$849**
8300P-40 CPS LETTER QUALITY	
REG. $2250	**$1995**
8024-MANNSMAN TALLEY	
REG. $1995	**$1595**
8024L - LETTER TALLEY	
REG. $2495	**$1995**
25 CPS-STARWRITER	
REG. $1895	**$1445**

SOFTWARE

	LIST	YOU PAY
OZZ — The Information Wizard	$395	$299
Wordcraft 80	$395	$299
IRMA — Info Retrieval & Mgmt. Aid	$495	$399
Dow Jones Portfolio Mgmt.	$149	$119
Pascal Development Pkg.	$295	$229
EBS — Receivables, Inventory	$750	$579
BPI — General Ledger	$395	$299
Word Pro 3 — 40 Column	$250	$179
Word Pro 4 — 80 Column	$375	$269
Word Pro 4 Plus	$450	$329

CBM-IEEE MODEM	
REG. $279	**$229**
VOICE SYNTHESIZER	
REG. $395	**$329**
VIC 20 ★★★★	
REG. $299	**$269**
VIC 1011 RS-232 INTER.	
REG. $49.95	**$39**
VIC 1112-IEEE INTER.	
REG. $99.95	**$79**
VIC 1540 - DISK 170K	
REG. $599	**$499**
VIC 1515 - 30 PS PRINTER	
REG. $395	**$349**
PET TO IEEE CABLE	
REG. $39.95	**$34**
IEEE TO IEEE CABLE	
REG. $49.95	**$39**

Philadelphia Computer Discount

155 E. LANCASTER, PA 19087

MAIL ORDER
800-345-1289
IN PA CALL (215) 687-8540

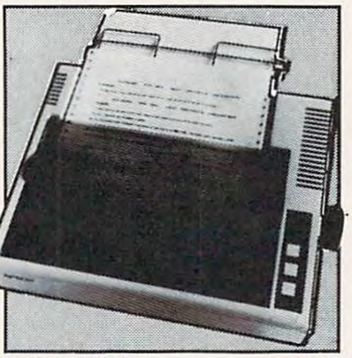
- PREPAID ORDERS SHIPPED FREE
- VISA & MASTERCARD ADD 3%
- COD - UPS
- PA RESIDENTS ADD 6% PA TAX

BRING IN THE NEW YEAR!

LOOK CLOSELY AT THESE SUPER SAVINGS!!

Ω OMEGA SALES CO.

• We Accept C.O.D.'s • Stock Shipments Same Day or Next • No Surcharge For Credit Cards • All Equipment Factory Fresh w/MFT Warranty • We Carry the Complete Line of Personal Software



EPSON MX-80 $ 469

ATARI 800 16K $ 749

EAST COAST
1-800-556-7586

Ω

WEST COAST
1-800-235-3581

PRICES ARE SUBJECT TO CHANGE W/O NOTICE.

SAVINGS!

AMAZING SPECIALS!

NEC Color Monitor
JC 1201 $319

INTERTEC SUPERBRAIN 64K RAM	**$2799**
QD SUPERBRAIN	$2999
NEC 5510 SPINWRITER (7710)	**$2345**
NEC 5520 SPINWRITER (7720)	$2695
NEC 5530 SPINWRITER (7730)	$2345
NEC 12" MONITOR	$ 189
NEC COLOR 12" MONITOR	$ 339
NEC PC 8023 Printer 100 CPS Tractor & Friction	$ 639
OKIDATA MICROLINE-80	**$ 379**
OKIDATA MICROLINE-82A	$ 529
OKIDATA MICROLINE-83A	$ 749
DIABLO 630	$1995
APPLE II PLUS 48K	**$1139**
APPLE DISK w/3.3 DOS Controller	$ 525
APPLE DISK w/o Controller	$ 449
EPSON MX-80	$ 469
Interfaces: IEEE $55, TRS-80 CABLE $35, APPLE INTERFACE & CABLE $90, RS-232 $70	
HAZELTINE 1420	$ 799
NORTHSTAR HORIZON II 32K QD	$2925
ANADEX DP-9500/9501	$1199
TELEVIDEO 912C	**$ 699**
TELEVIDEO 920C	$ 729
TELEVIDEO 950	$ 929

WE CARRY THE COMPLETE LINE OF ATARI SOFTWARE, PERIPHERALS AND ACCESSORIES.

Advertisers Index

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

For Fastest Service, Call Our Toll-Free US Order Line 800-345-8112 In Pennsylvania call 800-662-2444

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

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

Name, Address: ____________

Charge my: ☐ Visa ☐ MC
Number____________ Expires______/______/

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

For Fastest Service, Call Our Toll-Free US Order Line 800-345-8112 In Pennsylvania call 800-662-2444

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

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

Name, Address: ____________

Charge my: ☐ Visa ☐ MC
Number____________ Expires______/______/

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

Foreign Readers

New subscription rates for **COMPUTE!** readers outside of the US.
☐ $25 Anywhere/Surface Mail (2-4 months delivery)
☐ $38 Europe/Air Delivery (7-10 days)
☐ $48 Middle East, North Africa, Central America/Air Mail
☐ $88 South America, South Africa, Far East, Australia/Air Mail

Name ____________
Address ____________

☐ Check here if renewal subscription.

Payment must accompany this card.
Payment in US funds drawn on a US bank, International Money Order or charge card. VISA ☐ MasterCard ☐.
Account No. ____________. Expires ______/______/.

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

Place
Stamp
Here

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008
USA

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 236 BROOMALL, PA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008

Place
Stamp
Here

COMPUTE! Magazine
Post Office Box 5406
Greensboro, NC 27403

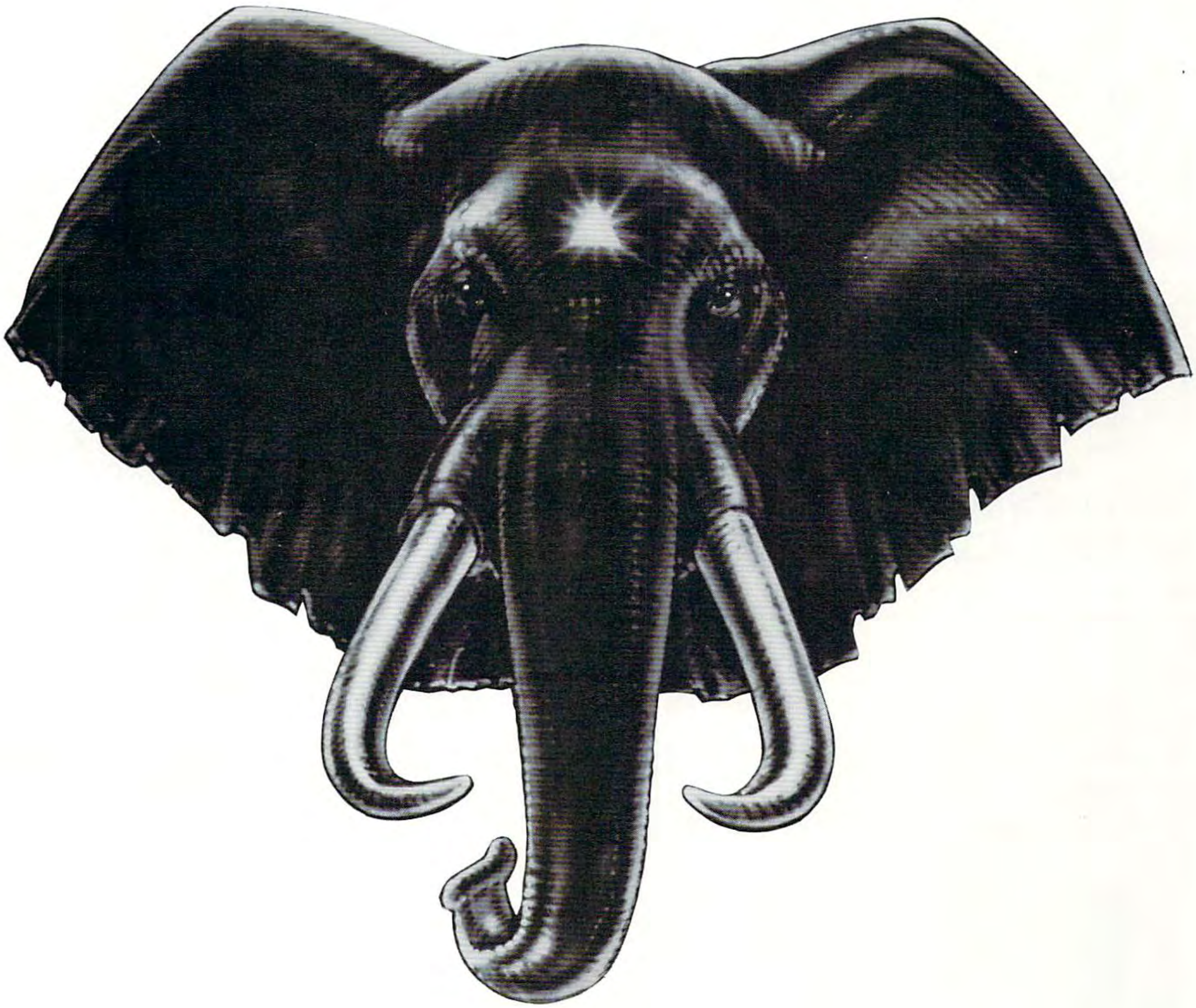
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 236 BROOMALL, PA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Magazine
515 Abbott Drive
Broomall, PA 19008

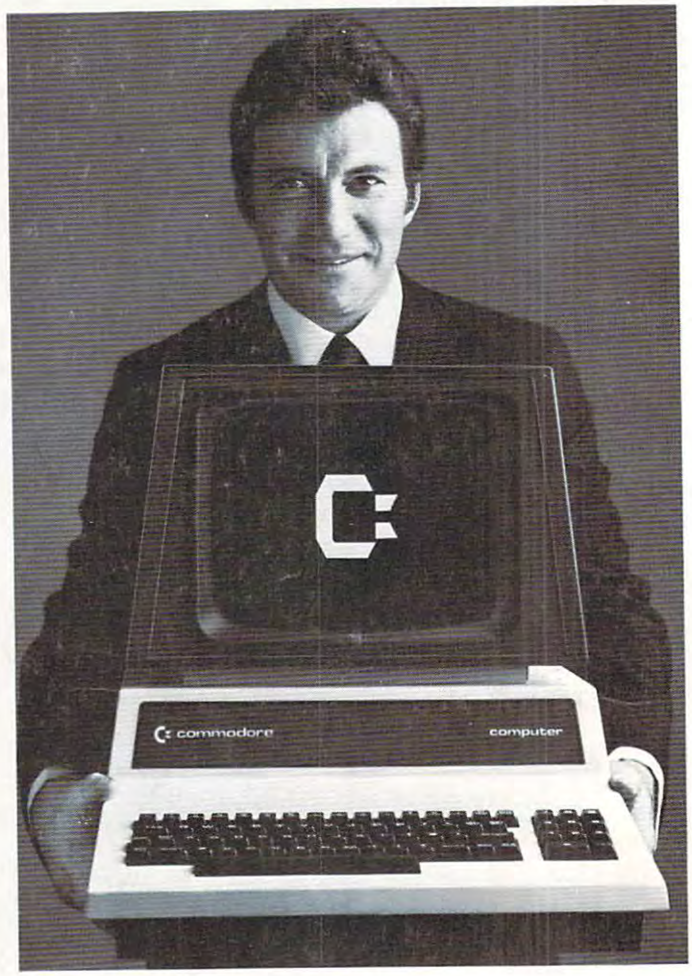
REMEMBER.



Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Toll-free: 1-800-343-6833; in Massachusetts (617) 828-8150. Telex 951-624.

THE COMMODORE COMPUTERS
"FROM $300 TO $1995, THEY COST LESS AND GIVE YOU MORE FOR YOUR MONEY. READ OUR CHART."
—William Shatner

The idea of a computer in every office and home used to be science fiction. Now it's becoming a reality. The question is, with so many to choose from, which computer should you buy? When you consider the facts, the clear choice is Commodore.



COMPARE OUR $995 COMPUTER

FEATURES	COMMODORE 4016	APPLE II	IBM
Base Price	$995	$1,330	$1,565
12" Green Screen	Standard	299	345
IEEE Interface	Standard	300	NO
TOTAL	$995	$1,929	$1,910
Upper & Lower Case Letters	Standard	NO	Standard
Separate Numeric Key Pad	Standard	NO	Standard
Intelligent Peripherals	Standard	NO	NO
Real Time Clock	Standard	NO	NO
Maximum 5½" Disk Capacity per Drive	500K	143K	160K

Prices are as of the most recent published price lists, September, 1981 and approximate the capabilities of the (16K) PET® 4016. Disk Drives and Printers are not included in prices. Models shown vary in their degree of expandability.

Many experts rate Commodore Computers as the best desk-top computers in their class. They provide more storage power—up to 1,000,000 characters on 5¼" dual disks—than any systems in their price range. Most come with a built-in green display screen. With comparable systems, the screen is an added expense. Our systems are more affordable. One reason: we make our own microprocessors. Many competitors use ours. And the compatibility of peripherals and basic programs lets you easily expand your system as your requirements grow. Which helps explain why Commodore is already the No. 1 desk-top computer in Europe with more than a quarter of a million computers sold worldwide.

WE WROTE THE BOOK ON SOFTWARE. The Commodore Software Encyclopedia is a comprehensive directory of over 500 programs for business, education, recreation and personal use. Pick up a copy at your local Commodore dealer.

FULL SERVICE, FULL SUPPORT. Commodore dealers throughout the country offer you prompt local service. In addition, our new national service contract with TRW provides nationwide support. Visit your Commodore dealer today for a hands-on demonstration.

Commodore Computer Systems
681 Moore Road
King of Prussia, PA 19406

Canadian Residents:
Commodore Computer Systems
3370 Pharmacy Avenue
Agincourt, Ontario, Canada, M1W 2K4

Please send me more information. CBM-CO

Name ____________________

Company ____________ Title ____________

Address ____________________

City ____________ State ________ Zip ________

Telephone ____________________

Interest Area ____________________

☐ Business ☐ Education ☐ Personal

commodore
COMPUTER