Pet Machine Language Graphics

David Malmberg, Fremont, CA

The PET has great graphics for almost any application, especially games. The only drawbacks I have found are that sometimes the graphics are not fast enough, or certain special effects (such as reversing only a section of the screen) cannot be done easily or quickly in a BASIC program without resorting to writing special subroutines in machine language. After many frustrating attempts to get the graphics to do exactly what I wanted in various machine language routines, I decided to write a general machine language subroutine that could be easily called from a BASIC program and would give the PET a wide-ranging repertoire of graphics "tricks."

Listing 1 is a BASIC program that POKEs this general machine language subroutine into the second cassette buffer and into the top three pages (a page is 256 bytes) of memory. This program then resets the memoy boundaries to protect the machine code from any BASIC programs. This is done automatically and is independent of the memory size. The program will also determine which of the various ROMs are in the PET and modify the machine code accordingly. It will work with "old", "new" of "4.0" ROMs. However, it will not work with the new 80-column PETs.

Once the subroutine has been loaded, it will give your BASIC programs significantly enhanced graphics capabilities. Specifically, you will be able to define a rectangular area on the screen and manipulate that area at machine language speed. The rectangle may be as small as a single space or as large as the entire screen. The area may be manipulated in the following ways:

Filled with any character

Reversed

Flashed on and off (i.e., fast multiple reversing)

Repositioned elsewhere on the screen

Moved (animation) in any direction at any speed with or without screen wraparound

Made to grow or shrink in size

Using The Subroutine

Your BASIC program would use the subroutine by POKEing various values into the subroutine's parameter list and then issuing a SYS(826) command. The parameter list and the corresponding POKE locations are given below:

LOCATION DESCRIPTION

700 Starting row (SR)
(0 to 24)
701 Last row (LR)
(0 to 24)
702 Starting column (SC)
(0 to 39)
703 Last column (LC)
(0 to 39)
704 Fill character (FC)
705 Row move direction (RD)
0 = Up
1 = Down or to side
706 # of rows to move (RM)
707 Column move direction (CD)
0 = Left
1 = Right or even
708 # of column to move (CM)
709 # of jiffies delay between iterations (JD)
710 Wraparound factor (WF)
0 = Wraparound is OK
1 = Disappear off screen edge
2 = Move to edge only
711 # of iterations before returning to BASIC (IT)

The letters inside the parentheses are short-hand variable names to which I have found it useful to assign the values of the POKE locations at the beginning of the BASIC program using the subroutine. In this way I don't have to remember that Jiffy Delay is location 709, rather I can just POKE JD,6 if I want a 6 jiffy (i.e., 1/10 second) delay between iterations. Using these parameter names also reduces the chance of errors, and is faster since BASIC handles variables faster than constants. I recommend you adopt the use of these parameter variables when using this subroutine.

Listing 2 is a BASIC program that demonstrates the full range of capabilities of the graphics subroutine. You are urged to key it in, run it and then study it to see just how each of the graphic effects was obtained. You will find it very informative.

At this point it is appropriate to describe in more detail just how the parameters can be used to generate various graphic effects. NOTE: In the discussions that follow all of the parameters are assumed to be zero unless specifically stated otherwise. In fact you will find it convenient to GOSUB to a routine to zero all of the parameters before beginning any new graphics, e.g., GOSUB 7000 in Listing 2.

Defining The Rectangle

The rectangular area is defined by the values of the parameters in locations 700 to 703. The area is the intersection of the defined rows and columns. The routine assumes that the "first" row or column on the screen is number zero, not number one. If the value of the starting row (starting column) is greater than the last row (last column) the routine will assume that the rectangle "wraparound" the edge of the screen. The rectangle may be the entire screen or a single space.

Filling The Area

If you wish to fill the rectangular area with a character, location 704 (short-hand FC) is POKEd with the ASCII value of the desired character. For example, the following lines of code will build a border around the screen "W" wide composed of character "C":

```
1 POKE FC,C : POKE SR,0 : POKE LR,24
2 POKE SC,40-W : POKE LC,W-1 : SYS(826)
3 POKE SC,0 : POKE LC,39
4 POKE SR,25-W : POKE LR, W-1 : SYS(826)
```

Lines 1 and 2 generate the sides of the border, and 3 and 4 the top and bottom. Notice that the routine uses the wraparound (start > last) feature to generate two sides of the border with the same subroutine call.

Reversing And Flashing

When you wish to reverse the area, the Fill Character, location 704, is POKEd with zero. A special case of reversing is to flash the rectangle on and off with fast multiple reversing. This effect is obtained by POKEing location 711 (IT) with the number of times the area is to be reversed, and location 709 (JD) with the number of jiffies to delay between each reverse cycle. For example, the following code will flash the entire screen on and off by reversing it "N" times at a speed of "D" jiffies:

```
1 POKE FC,0 : POKE JD,D : POKE IT,N
2 POKE SR,0 : POKE LR,24
3 POKE SC,0 : POKE LC,39 : SYS(826)
```

Repositioning The Area

The rectangle can be repositioned in a different location on the screen by setting the parameters in locations 705 to 708. Location 705 (RD) is POKEd with a zero if the relative displacement of the new position is up and with 1 if it is down or even. Location 707 (CD) is POKEd with 0 if the displacement is left and with 1 otherwise. Locations 706 (RM) and 708 (CM) are the number of rows and columns, respectively, the area is to be displaced. For example,

```
1 POKE RD,0 : POKE RM,10
2 POKE CD,1 : POKE CM,5
```

will reposition the area five columns to the right and ten rows up.

If the "old" area is to be blanked out after the repositioning, the Fill Character (FC = 704) should have been previously POKEd with 32, i.e., an ASCII blank. If FC is zero rather than 32, then both the "old" and "new" areas will be visible on the screen after returning from the graphics subroutine.

Since this repositioning is done by relative displacement rather than absolute positioning on the screen, there will be instances when the new position will be "off the edge." Just how the routine handles these situations is determined by the value of the Wraparound Factor (WF = 710). If this value has been POKEd with a zero, the routine will automatically wraparound to the other edge(s) of the screen. If WF is 1, the portion of the rectangle that goes over the edge will not be shown. If WF is 2, the routine will automatically recalculate the reposition parameters so that the rectangle stops just at the edge of the screen.

Motion Or Animation

Motion, or animation, is handled very much like repositioning, except that the increments of displacement are smaller (typically only one row and/or column) and the number of iterations (IT = 711) and jiffies delay (JD = 709) are used to control the distance and speed of the movement. For example to show the rectangle moving up and to the right at a 30 degree angle at a relatively fast pace these instructions could be used:

```
1 POKE RD,0 : POKE RM,1 : POKE CD,1 : POKE CM,2
2 POKE IT,10 : POKE JD,2 : POKE FC,32 : POKE WF,0
```

Setting WF to zero and FC to 32 assures the "old" area is erased and that wraparound is allowed if appropriate. Even though JD was set to 2, the actual "speed" of the movement will depend on the size of the rectangle — obviously larger areas take longer to move than smaller ones — even at machine language speed! You should experiment with various values of JD to get the speed you want for your specific areas to be moved.

After returning from the subroutine, the parameters defining the rectangle will be automatically updated to correspond to the new location, so it is unnecessary to keep track of these locations in your BASIC program or to rePOKE these locations before making another move. However if you are moving several different areas "simultaneously", you should save locations 700 to 703 after exiting the routine and then rePOKE these same values before moving again (if there are intervening moves of other areas).

Listing 2 has a number of examples of movement that should be helpful to you in understanding how to use this routine effectively. The code at lines 800 to 870 should be particularly useful because it shows an easy and fast way to control motion with the numeric key pad.

Shrink And Growing

Some very interesting graphic effects are possible if you use the routine for repositioning or motion but do not POKE the Fill Character with a ASCII blank, i.e., a 32. If FC is zero, the "old" area is not changed as the "new" area is created. This allows the total graphic area to give the appearance of growing in size. Once the area has grown, FC can be set to 32 and the direction of the movement switched by 180 degrees and the area will appear to shrink. If FC is POKEd with something other than zero or 32, movement can be handled against a non-blank background, or some other characters can be left behind as the "wake" of the movement.

Lines 880 to 980 in Listing 2 give a good example of using the routine to grow and shrink objects.

Conclusions

I hope you have as much fun using this routine as I did in writing it. If you develop any new or unusual uses for this routine drop me a note — or better yet, tape copy of the program.

If you don't want to spend the effort keying in the code in the Listings, send me $5.00 and I will send you a tape containing both the graphics loader program (Listing 1) and the demo program (Listing 2).

Listing 1

```
10 CLR:POKE59468,12
20 REM PROGRAM BY DAVID MALMBERG
30 REM 43064 VIA MORAGA
40 REM FREMONT,CALIF 94538
50 REM (415) 651-6921
60 IFPEEK(50000)=0THENPOKE134,0:
     ¬POKE135,PEEK(135)-3:GOTO80
70 POKE52,0:POKE53,PEEK(53)-3
80 CLR
90 ZZ=53:IFPEEK(50000)=0THENZZ=135
100 QQ=PEEK(ZZ)
110 PRINT"ħ"TAB(12)"LOADING 2ND ¬
     ¬CASSETTE"
120 PRINT"h↓↓MACHINE LANGUAGE GRAPHICS"
130 PRINT"h↓↓↓↓BY DAVID MALMBERG"
140 FOR I= 826 TO 1015 :READDC:POKEI,DC:
     ¬PRINT"h";I;DC:NEXTI
150 DATA32,150,3,56,165,62,201,25,176
160 DATA61,165,63,201,25,176,55,165,68
170 DATA201,25,176,49,165,64,201,40,176
180 DATA43,165,65,201,40,176,37,165,70
190 DATA201,40,176,31,165,68,208,10,165
200 DATA70,208,6,32,0,16,56,176,3,32
210 DATA119,17,32,133,3,165,73,240,7
220 DATA198,73,240,3,76,98,3,32,168,3
230 DATA96,165,71,240,12,169,0,141,143
240 DATA0,165,71,205,143,0,208,251,96
250 DATA162,0,181,48,188,174,2,157,174
260 DATA2,148,48,232,224,32,208,241,96
270 DATA162,0,189,174,2,180,48,149,48
280 DATA152,157,174,2,232,224,32,208
290 DATA240,96,169,0,133,51,162,8,10
300 DATA38,51,6,49,144,7,24,101,48,144
310 DATA2,230,51,202,208,239,133,50,96
320 DATA169,0,133,1,133,2,165,54,240
330 DATA17,133,48,169,40,133,49,32,187
340 DATA3,165,50,133,1,165,51,133,2,24
350 DATA165,2,105,128,133,2,96
360 PRINT"h"TAB(12)"LOADING HIGH MEMORY ¬
     ¬ "
370 AA=QQ*256
380 FORI=AATOAA+626:READDC:POKEI,DC:
     ¬PRINT"h";I;DC:NEXTI
390 DATA165,62,133,54,32,213,3,166,62
400 DATA164,64,165,66,208,4,177,1,73
410 DATA128,145,1,196,65,240,10,200,56
420 DATA192,40,144,236,160,0,240,232
430 DATA228,63,240,30,232,56,224,25,176
440 DATA14,24,165,1,105,40,133,1,144
450 DATA2,230,2,56,176,206,162,0,134
460 DATA1,169,128,133,2,208,196,96,169
470 DATA0,133,60,165,67,208,87,56,165
480 DATA54,229,68,16,36,72,165,72,201
490 DATA2,208,21,24,104,101,68,133,68
500 DATA169,0,133,73,133,1,133,54,169
510 DATA128,133,2,24,144,47,201,0,208
520 DATA34,24,104,105,25,230,59,197,59
530 DATA208,16,133,54,24,165,1,105,40
540 DATA133,1,144,22,230,2,24,144,17
550 DATA133,54,32,213,3,24,144,9,169
560 DATA15,133,60,104,169,0,133,54,165
570 DATA54,133,59,96,24,165,54,101,68
580 DATA56,201,25,144,45,233,25,72,165
590 DATA72,201,2,208,31,104,133,61,230
600 DATA61,169,0,133,73,56,165,68,229
610 DATA61,133,68,169,24,133,54,169,131
620 DATA133,2,169,192,133,1,24,144,44
630 DATA201,0,208,31,104,198,59,197,59
640 DATA208,16,133,54,56,165,1,233,40
650 DATA133,1,176,22,198,2,24,144,17
660 DATA133,54,32,213,3,24,144,9,169
670 DATA15,133,60,104,169,24,133,54,165
680 DATA54,133,59,96,169,0,133,74,165
690 DATA69,208,47,56,165,76,229,70,16
700 DATA93,72,165,72,201,2,208,12,24
710 DATA104,101,70,133,70,169,0,133,73
720 DATA240,74,201,0,208,7,24,104,105
730 DATA40,24,144,63,169,15,133,74,104
740 DATA169,0,24,144,53,24,165,76,101
750 DATA70,56,201,40,144,43,233,40,72
760 DATA165,72,201,2,208,19,104,133,61
770 DATA169,0,133,73,56,165,70,229,61
780 DATA133,70.169,39,24,144,15,201
790 DATA0,208,4,104,24,144,7,169,15
800 DATA133,74,104,169,39,133,75,96
810 DATA165,67,208,11,165,62,133,55
820 DATA165.63,133,56,24,144,8,165,63
830 DATA133,55,165,62,133,56,165,69
840 DATA208,11,165,64,133,57,165,65
850 DATA133,58,24,144,8,165,65,133,57
860 DATA165.64,133,58,165,55,133,54
870 DATA32.213,3,165.1,133,52,165,2
880 DATA133,53,169,175,133,59,166,55
890 DATA134,54,134,77,32,70,16,164,57
900 DATA132,76,32,8,17,166,77,165,60
910 DATA208,14,165,74,208,10,177,52
920 DATA132,61,164,75,145,1,164,61,165
930 DATA66,240,2,145,52,196,58,240,26
940 DATA165,69,240,10,192,0,208,2,160
950 DATA40,136,24,144,207,192,39,208
960 DATA4,160,0,240,199,200,24,144,195
970 DATA228,56,240.66,165,67,240,31
980 DATA224,0,208,12,162,24,169,131
990 DATA133,53,169,192,133,52,208,162
1000 DATA202,56,165,52,233,40,133,52
1010 DATA176,152,198,53,24,144,147,224
1020 DATA24,208,12,169,128,133,53,169
1030 DATA0,133,52,162,0,240,131,232,24
1040 DATA165.52,105,40,133,52,144,227
1050 DATA230,53,24,144,222,165,62,133
1060 DATA54,32,70,16,165,59,133,62,165
1070 DATA63,133,54,32,70,16,165,59,133
1080 DATA63,165,64,133,76,32,8,17,165
1090 DATA75,133,64,165,65,133,76,32,8
1100 DATA17,165,75,133,65,96
1110 REM MODIFICATIONS FOR RELOCATION
1120 FORI=1TO3:READA:POKEQQ*256+A,QQ:
     ¬NEXTI
1130 DATA448,588,599
1140 FORI=1TO3:READA:POKEQQ*256+A,QQ+1:
     ¬NEXTI
1150 DATA455,610,621
1160 POKE876,QQ
1170 POKE882.QQ+1
1180 REM MODIFICTIONS FOR OLD ROMS
```

```
1190 IFPEEK(50000)<>0THEN1220
1200 FORI=1TO4:READA:POKEA,2:NEXTI
1210 DATA908,909,913,914
1220 PRINT"ĥṟLOCATIONr̂     ṟDESCRIPTIONr̂"
1230 PRINT"   700      STARTING ROW"
1240 PRINT"   701      LAST ROW
1250 PRINT"   702      STARTING COLUMN"
1260 PRINT"   703      LAST COLUMN"
1270 PRINT"   704      FILL CHARACTER"
1280 PRINT"   705      ROW MOVE DIRECTION
     ¬"
1290 PRINT"              0=UP"
1300 PRINT"              1=DOWN OR TO ¬
     ¬SIDE"
1310 PRINT"   706      # OF ROWS TO MOVE"
1320 PRINT"   707      COLUMN MOVE ¬
     ¬DIRECTION"
1330 PRINT"              0=LEFT"
1340 PRINT"              1=RIGHT OR EVEN"
1350 PRINT"   708      # OF COLUMNS TO ¬
     ¬MOVE"
1360 PRINT"   709      # OF JIFFIES ¬
     ¬DELAY BETWEEN"
1370 PRINT"              ITERATIONS"
1380 PRINT"   710      WRAPAROUND FACTOR"
1390 PRINT"              0=WRAPAROUND IS ¬
     ¬OK"
1400 PRINT"              1=DISAPPEAR OFF ¬
     ¬EDGE"
1410 PRINT"              2=MOVE TO EDGE ¬
     ¬ONLY"
1420 PRINT"   711      # OF ITERATIONS ¬
     ¬BEFORE EXIT"
1430 PRINT"   826      SYS LOCATION FOR ¬
     ¬ROUTINES"
1440 END
```

Listing 2

```
0 :CLR:POKE59468,12
1 REM MACHINE LANGUAGE GRAPHICS DEMO
2 REM PROGRAM BY DAVID MALMBERG
3 REM 43064 VIA MORAGA
4 REM FREMONT, CALIF 94538
5 REM (415) 651-6921
100 QQ=53:KY=151:IFPEEK(50000)=0THENQQ=1
    ¬35:KY=547
110 AA=PEEK(QQ)*256
120 IFPEEK(826)<>32ORPEEK(AA)<>165THENPR
    ¬INT"ĥMACHINE LANGUAGE NOT LOADED":
    ¬END
130 SR=700:LR=701:SC=702:LC=703:FC=704:
    ¬RD=705:RM=706:CD=707:CM=708
140 JD=709:WF=710:IT=711:SY=826
150 R$="h̲↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓":
    ¬C$="→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→
    ¬→"
160 X=RND(-TI):DIMX%(8),A(8),B(8)
170 DEF FNR(N)=INT(N*RND(1))
180 FORI=0TO8:READX%(I):NEXTI
190 DATA209,214,215,223,102,42,127,218,
    ¬219
200 A$=LEFT$(C$,11):B$=LEFT$(C$,5)
210 PRINT"ĥ"LEFT$(R$,6):GOSUB6000
215 REM BUILD BORDERS
220 GOSUB7000:W=4
230 FORI=0TO8:C=X%(I):GOSUB7100:Q=TI
235 IFTI-Q<60THEN235
236 NEXTI
240 C=32:GOSUB7100
250 REM FLASH SCREEN
260 POKEFC,0:N=30:D=15:GOSUB7300
270 REM MAKE REVERSE SQUARES
280 C=0:FORW=1TO4:GOSUB7200:Q=TI
290 IFTI-Q<30THEN290
300 NEXTW
310 W=4:C=32:GOSUB7100
320 C=0:GOSUB7100
330 N=1:D=30:GOSUB7300
340 N=1:D=30:GOSUB7300
350 W=4:C=32:GOSUB7100
360 REM MOVE LEFT:GOSUB7000
370 POKERS,6:POKELR,19:POKESC,5:
    ¬POKELC,35:POKERD,0:POKERM,0
380 POKERM,0:POKECD,1:POKECM,1:POKEIT,
    ¬40:SYS(SY)
390 REM MOVE DOWN
400 POKERD,1:POKERM,1:POKECD,0:POKECM,0:
    ¬POKEIT,25:SYS(SY)
410 REM MOVE UP & OFF
420 POKERD,0:POKECD,1:POKECM,1:POKEWF,1:
    ¬POKEIT,20:SYS(SY)
425 PRINT"ĥ":PRINT
430 REM SHIP & R2D2
440 PRINT"h̲":A$=LEFT$(C$,11):GOSUB5000
450 A$=LEFT$(C$,1):PRINTLEFT$(R$,14):
    ¬GOSUB5200
460 A(0)=0:A(1)=9:A(2)=11:A(3)=39
470 A(5)=1:A(6)=0:A(7)=0:A(8)=1:A(4)=32
475 GOSUB7000:A$=LEFT$(R$,15)+LEFT$(C$,
    ¬9)
480 POKESR,13:POKELR,24:POKESC,0:
    ¬POKELC,13:POKEFC,32:POKERD,1:
    ¬POKERM,0
490 POKECD,1:POKECM,1:DL%=6
495 FORK=1TO2
505 DL%=0.5*DL%:POKEJD,DL%:POKEIT,40
510 FORJ=0TO8:B(J)=PEEK(700+J):POKE700+J
    ¬,A(J):NEXTJ:SYS(SY)
540 PRINTA$"...BEEP BEEP":Q=TI
550 IFTI-Q<60THEN550
552 PRINTA$"              "
553 DL%=1.5*DL%:POKEJD,DL%:POKEIT,40
555 FORJ=0TO8:A(J)=PEEK(700+J):POKE700+J
    ¬,B(J):NEXTJ:SYS(SY)
560 PRINTA$"...SQEEEEEEK":Q=TI
570 IFTI-Q<60THEN570
580 PRINTA$"              "
590 NEXTK
600 POKEJD,0:POKERD,0:POKERM,13:
    ¬POKECM,0:POKECM,0:POKEWF,1:SYS(SY)
610 A$=LEFT$(R$,2)+LEFT$(C$,9)
620 PRINTA$"...BEEP BEEP":Q=TI
630 IFTI-Q<60THEN630
640 PRINTA$"              "
650 POKERM,0:POKECD,1:POKECM,1:POKEIT,
    ¬40:SYS(SY)
700 REM SPACE SHIP
710 GOSUB7000:PRINT"ĥ"LEFT$(R$,19):
    ¬A$=LEFT$(C$,18):GOSUB5100
720 PRINT"h̲↓↓→→USE NUMERIC KEYS TO ¬
    ¬CONTROL DIRECTION"
730 PRINT"↓→→→→→→PUSH ṟ0r̂ WHEN YOU WISH ¬
    ¬TO QUIT"
740 POKESR,1:POKELR,5:POKESC,1:POKELC,
    ¬39:POKEFC,0:POKEIT,9:POKEJD,20:
    ¬SYS(SY)
750 Q=TI
760 IFTI-Q<180THEN760
770 POKEFC,32:SYS(SY)
780 POKESR,19:POKELR,24:POKESC,18:
```

```
      ¬POKELC,22:POKEJD,Ø:POKERD,Ø:
      ¬POKERM,1
790 POKECD,1:POKECM,Ø:L=255
795 REM MOVEMENT USING MATRIX VALUE OF ¬
      ¬KEY BEING PRESSED
800 K=PEEK(KY):IFK=255ORK=34THEN800
805 IFK=LTHEN860
810 IFK=1ØTHEN870
820 IFK>49THENPOKERD,Ø:POKERM,1:GOTO840
830 POKERD,1:POKERM,Ø:IFK<3ØTHENPOKERM,1
840 POKECM,1:POKECD,1:IFK=58ORK=42ORK=26
      ¬THENPOKECD,Ø
850 IFK=5ØORK=18THENPOKECM,Ø
860 SYS(SY):L=K:GOTO800
870 FORI=1TO1Ø:GETZ$:NEXTI:REM EMPTY ¬
      ¬KYBD BUFFER
880 REM ARROW GROWS
890 GOSUB7000:POKESR,1Ø:POKELR,17:
      ¬POKESC,9:POKELR,17:POKEIT,7:
      ¬POKEJD,5
900 POKERM,1:POKECM,1
910 PRINT"ĥ":GOSUB6200
920 FORK=1TO5:POKEJD,5-K:Q=TI
925 IFTI-Q<12ØTHEN925
930 POKEFC,Ø:POKERD,1:POKECD,1:SYS(SY):
      ¬Q=TI:REM GROW
935 IFTI-Q<6ØTHEN935
940 POKEFC,32:POKEIT,1Ø:POKERD,Ø:
      ¬POKECD,Ø:SYS(SY):REM SHRINK
950 POKEIT,1Ø:NEXTK
960 Q=TI:REM SHOOT OFF SCREEN
965 IFTI-Q<12ØTHEN965
970 POKESR,Ø:POKESC,Ø:POKERD,1:POKECD,1
980 POKEJD,Ø:POKEWF,1:POKEIT,26:SYS(SY)
1000 REM REVERSE DESIGN
1010 GOSUB7000:POKEJD,1Ø
1020 PRINT"ĥ":Q=1:K=Ø:GOTO1040
1030 Q=FNR(4)+1:K=FNR(23)
1040 FORI=KTO24STEPQ
1050 POKESR,I:POKELR,24-I:POKESC,I:
      ¬POKELC,39-I:SYS(SY):NEXTI
1060 IFRND(1)>.9THEN1020
1070 GETZ$:IFZ$=""THEN1030
1075 REM SUPER GRAPHICS
1090 A$=LEFT$(C$,11):B$=LEFT$(C$,5)
1100 PRINT"ĥ"LEFT$(R$,6):GOSUB6000
1110 GOSUB7000:POKEJD,2Ø
1120 C=Ø:FORW=1TO4:GOSUB7200:Q=TI
1130 IFTI-Q<3ØTHEN1130
1140 NEXTW
1145 FORI=1TO1Ø:GETZ$:NEXTI:REM EMPTY ¬
      ¬KYBD BUFFER
1150 END
5000 PRINTA$"                    '  "
5010 PRINTA$"          l         'L"
5020 PRINTA$"          l         '  "
5030 PRINTA$"          l         '%"
5040 PRINTA$"        ,r<r̂l         '%"
5050 PRINTA$"         r r̂3 *r r̂    '%"
5060 PRINTA$"     $$$r> r̂≡ 6r r̂     ¬
      ¬r>r̂M$$$$
5070 PRINTA$"   @@""r ..r̂  r  ≤r̂  r. ¬
      ¬..r̂""@@
5080 PRINTA$" ^r ....     .. ..   ..... ¬
      ¬ ," 
5090 PRINTA$"  ^r                       ¬
      ¬,
5095 RETURN
5100 PRINTA$" '> "
5110 PRINTA$" r) ^r̂ "
```

```
5120 PRINTA$" r ≥ f̂ "
5130 PRINTA$"r)USA^f̂"
5140 PRINTA$"Nr)"^f̂M"
5150 RETURN
5200 PRINTA$">r)O:f̂)M"
5210 PRINTA$">r ?f̂)  M"
5220 PRINTA$">r<f̂)     M"
5230 PRINTA$">)  UI  M"
5240 PRINTA$">M  ||   M"
5250 PRINTA$">>M ||    M"
5260 PRINTA$">>>M||     N%"
5270 PRINTA$">>>>||    NN"
5280 PRINTA$">>>>||   NV"
5290 PRINTA$">>>>||MNNMM"
5300 PRINTA$">>>r)((^f̂>>r)((^"
5310 RETURN
6000 PRINTA$"r) ^f̂ r f̂ r f̂ r  ^f̂ r   f̂ ¬
     ¬r  ^f̂"
6010 PRINTA$"r f̂ r f̂ r f̂ r f̂ r f̂ r f̂ r ¬
     ¬f̂   r f̂ r f̂"
6020 PRINTA$"^r^f̂  r f̂ r f̂ r  f̂) r  f̂  ¬
     ¬r  f̂)"
6030 PRINTA$" ^r^f̂ r f̂ r f̂ r f̂   r f̂   ¬
     ¬r ^f̂"
6040 PRINTA$"r f̂ r f̂ r f̂ r f̂ r f̂   r f̂  ¬
     ¬ r f̂^r^f̂"
6050 PRINTA$"^r f̂) ^r f̂) r f̂   r   f̂ r ¬
     ¬f̂ r f̂"
6060 PRINT
6082 PRINTB$"r) ^f̂ r  ^f̂ r) ^f̂ r  ^f̂";
6084 PRINT" r f̂ r f̂ r   f̂ r) ^f̂ r) ^f̂"
6092 PRINTB$"r f̂ r f̂ r f̂ r f̂ r f̂ r f̂";
6094 PRINT" r f̂ r f̂ r f̂ r f̂  r f̂  r f̂ r ¬
     ¬f̂ r f̂ r f̂"
6102 PRINTB$"r f̂   r  f̂) r f̂ r f̂";
6104 PRINT" r  f̂) r   f̂  r f̂  r f̂   ¬
     ¬^r^f̂"
6112 PRINTB$"r ! f̂ r ^f̂  r   f̂ r f̂";
6114 PRINT"  r f̂ r f̂  r f̂  r f̂    ^r^f̂"
6122 PRINTB$"r f̂ r f̂ r f̂^r^f̂ r f̂ r f̂";
6124 PRINT" r f̂   r f̂ r f̂  r f̂  r f̂ r f̂ ¬
     ¬r f̂ r f̂"
6132 PRINTB$"^r  f̂ r f̂ r f̂ r f̂ r f̂ r f̂";
6134 PRINT"  r f̂ r f̂ r   f̂ ^r f̂) ^r f̂)"
6140 RETURN
6200 PRINT"   !\"
6210 PRINT"   ''\"
6220 PRINT"   '''\"
6230 PRINT"$$$NM%%%"
6240 PRINT"\$$M M%%"
6250 PRINT" \$#M M%"
6260 PRINT"  \##M M"
6270 PRINT"   ###M M"
6280 PRINT"        M M"
6290 PRINT"         M M"
6300 PRINT"          M M"
6310 PRINT"           M M  r^f̂"
6320 PRINT"            M M r ^f̂"
6330 PRINT"             M Mr  ^r"
6340 PRINT"              Mr)   f̂"
6350 PRINT"           ^r       f̂"
6360 PRINT"            ^r      f̂"
6370 PRINT"             ^r     f̂"
6380 RETURN
7000 REM SUB TO ZERO PARAMETERS
7010 FORM=ØTO11:POKE700+M,Ø:NEXTM:RETURN
7100 REM SUB TO BUILD A BORDER W WIDE ¬
     ¬WITH CHARACTER C
7110 POKEFC,C:POKESR,Ø:POKELR,24
7120 POKESC,40-W:POKELC,W-1:SYS(SY)
7130 POKESC,Ø:POKELC,39
7140 POKESR,25-W:POKELR,W-1:SYS(SY):
     ¬RETURN
7200 REM SUB TO FILL AN AREA WITHIN A ¬
     ¬BORDER W WIDE WITH CHARACTER C
7210 REM IF C=Ø THEN REVERSE AREA ..IF ¬
     ¬C=32 THEN BLANK AREA
7220 POKEFC,C:POKESR,W:POKELR,24-W
7230 POKESC,W:POKELC,39-W:SYS(SY):RETURN
7300 REM SUB TO REVERSE SCREEN N TIMES ¬
     ¬WITH D JIFFY DELAY BETWEEN ¬
     ¬ITERATIONS
7310 POKEJD,D:POKEIT,N:POKESR,Ø:
     ¬POKELR,24
7320 POKESC,Ø:POKELC,39:SYS(SY):RETURN
```

©

Disk File Recovery Program

David L. Cone, Sunnyvale, CA

Have you ever been working happily along on a program, updating it periodically on your disk, only suddenly to discover that something wierd has happened and you've apparently lost half of the programs on the disk. (I've even had the case where the programs disappeared from the directory while the number of blocks remaining stayed the same). Maybe what happened was that AFTER you scratched the program from the disk you found that the PET had also gobbled up your program in memory — (or you did). Or perhaps you had done a short 'new' of a disk only then to realize that valued programs were on it!

If you've ever been in this frustrating position of knowing your program was just sitting there on the disk with no easy way to get it back, this DISK FILE RECOVERY program will help — it will recover such programs. As long as you can still initiate the disk and have not written a new program over the file you want, you can recover it. It cannot recover programs from a disk which will not initiate or upon which a long 'new' (ie. a 'new' with a disk number) has been performed.

The disk works this way: When a file is placed on a disk, part of the information placed in the directory on track 18 is a pair of pointers giving the track and sector numbers of the block where the file begins. The first two bytes of this block are also pointers giving the next track and sector numbers. This process continues until the last block is reached. For the last block, a 00 is placed in the first byte and nothing appears to be done to the second. Files are stored in a somewhat alternating way below and above track 18. The first file is stored starting at 17,0 (track 17, sector 0). When track 17 is filled, the next new file appears to be started at 19,0 and so on back and forth. If you have lost or destroyed track 18, the problem is then how to find and identify the initial blocks of the lost files and then to recover the files.

This is what the RECOVERY program does! First, it gives you the choice of working with either the lower band (tracks 17 to 1) or upper band (tracks 19 to 35), and on which track you wish to stop. It sets up an integer array [D%(35,20,3)] which can receive for each block the "in" pointers (ie. the track and sector numbers of the block which 'points' at it) and the "out" pointers (ie. the track and sector number of the block at which it points. The program then scans the first track for these pointers. What we are looking for are blocks which have no "in" pointers, for they must be the ones pointed at by the directory and thus the initial blocks for any files. Next the program takes each initial block and follows that file through all its blocks to the end, filling in the array as it goes. Each subsequent track is similarly scanned and as new files are found they are traced. You have the option of stopping this process at any point. Meantime, the program has kept track of the start and end of each file and the number of blocks it uses. This summary is presented on the screen.

The next major problem is the identifying which file is which (since only the disk knows where a file was saved and on which half of the disk). The program offers you a number of appropriate options at this point, and the most useful one for file identification is labeled LOOK. LOOK pulls the initial block of any file out and extracts information that will probably allow you to identify the file. First, it displays in a useful form the first four pairs of bytes. The first pair are the pointers to the next block. If the file is a program the next pair of bytes tell where the program is to be loaded in memory. For Basic programs, this is usually 1024. The third and fourth pairs of bytes are from the program itself. They are the link and line number of the first instruction in the basic program. If the file is a machine language program or a sequential file, then you get weird and meaningless values for the link and line numbers. Next, LOOK gives you the first 48 bytes of the program in hexadecimal form (as if they were being examined by the machine language monitor). Finally, LOOK gives you a printed "translation" of the first 240 bytes. Basic commands are tokenized and appear as reversed characters or symbols. The link and line pointers also can look quite strange. However, numbers, variables, anything between quotes, and REM statements all appear as usual. Thus, if you have some convenient identifications at the beginning of your program, you will be able to recognize them. To see how this "translation" takes place, see lines 1360-1390 and 5090-5095. Eighty characters are scanned at one time and you can go from one set of eighty to another. With this amount of information it is usually quite easy to determine what any file is and if you wish to recover it.

Aside from LOOK, you have the following options: 1) SUMMARY REVIEW — this gives you the start block of any file and the number of blocks in that file. You need to know the start block to either look at or recover a file. Also, the number of blocks in the file may aid in its identification. 2) RETRIEVE A FILE — here is the point of all of this; now you get the program or file back! The program asks for all the essential things: starting track and sector, the name you want for the recovered file

and whether it is a program or a data file. It gives one final chance to abort unless everything is ok and then it is off and running. 3) SCAN OTHER BAND, 4) DIFFERENT DISK, and 5) EXIT PROGRAM are all obvious.

The program itself, while complex in details, is straightforward in construction. It is divided into the following sections:

400- 492 Program description and credits
500- 595 Description of all variables
600- 696 Start of Program — Initial choices
700- 865 Search for initiator blocks
1000-1055 Print summary table
1100-1165 Choices
1200-1415 Performs LOOK option
1500-1655 Retrieves the file
4000-4076 General subroutines
5000-5109 Disk operation subroutines

"REM**" statements are used to show major divisions of the program while "REM @" indicates descriptive statements within these major divisions. I have used REM statements fairly liberally and these should help in tracing through the details of the program. A pair of REM statements (line numbers 1410 and 4003) need a special comment: if you have a machine language screen dump capability, you should SYS to them here. I use a shifted "P" to activate the screen dump.

A couple of final comments: If you search tracks in which there are no programs, you may get a disk read error (22 READ ERROR 13,0). If this occurs, simply type GOTO 1000 and you will be able to go on without any problems. I hope this program is as useful to you as it has been to me. I made it because I really needed it. You may not need it often, but when you do, the situation is likely to be desperate!

```
GOTO5    TRACK  19 SUMMARY TABLE
SEC ▌ TR   IN   SC   |   TR   OUT   SC
 0  ▌ 0         0    |   19         10
 1  ▌ 19        10   |   19         11
 2  ▌ 19        11   |   19         12
 3  ▌ 19        12   |   19         13
 4  ▌ 19        13   |   19         14
 5  ▌ 19        14   |   19         15
 6  ▌ 19        15   |   64         220
 7  ▌ 0         0    |   19         17
 8  ▌ 19        17   |   19         18
 9  ▌ 19        18   |   64         214
 10 ▌ 19        0    |   19         1
 11 ▌ 19        1    |   19         2
 12 ▌ 19        2    |   19         3
 13 ▌ 19        3    |   19         4
 14 ▌ 19        4    |   19         5
 15 ▌ 19        5    |   19         6
 16 ▌ 0         0    |   64         130
 17 ▌ 19        7    |   19         8
 18 ▌ 19        8    |   19         9
 19 ▌ 0         0    |   20         8
       TYPE ANY KEY TO CONTINUE
```

Figure 1. Track 19 Summary Table
A summary table such as this is made for each track scanned. The zeros in the IN column indicate the initial block of a file. The 64 in the OUT column shows where a file ends.

```
GOTO5

          RECOVERED DISK SEQUENCES

                SEQUENCE 1

                          TRACK      SECTOR

START                       17          0

PRESENT BLOCK    11         17          12

END                         17          12

NUMBER OF BLOCKS            11


   (C-CONT  P-PAUSE  H-HALT  S-SUMMARY)
          TYPE ANY KEY TO CONTINUE
```

Figure 2. Recovered Disk Sequence
As each file is traced, this table keeps track of what is happening and summarizes the results.

```
GOTO5        FIRST PAGE DATA

                      TRACK     SECTOR
INITIAL BLOCK          17         0
                BYTES
BLOCK POINTER    0&1   17         10
                      DECIMAL VALUE
PRGRM START      2&3      1025
1ST LINK         4&5      1032
1ST LINE #       6&7       0
HEX VALUES
  00 :   11 0A 01 04 08 04 00 00
  08 :   89 35 00 21 04 01 00 99
  10 :   22 93 11 11 11 3E 49 31
  18 :   9D 9D 9D 11 3E 24 31 13
  20 :   22 3A 80 00 3D 04 02 00
  28 :   99 22 93 3E 53 30 3A 43

CHARACTER VALUES:

QJADHD@@◤5@!DA@▬"⊣QQQ>I1┘┘┘Q>$1S": @=DB@
▬"⊣>S0:COPY DISK+":▬">"@$DC@▬"QS♠"|(34)"

 'C' TO CONTINUE: +/- CHANGE LINE SCAN
```

Figure 3. First Page Data Program File
A typical BASIC program looks this way. Note the following: PRGRM START = 1025; typical 1st LINK and LINE # values; and identifiable features in the CHARACTER VALUES. (Unfortunately, my dump program does not give reversed characters which would assist in identifying BASIC tokens).

```
GOTO5        FIRST PAGE DATA

                         TRACK      SECTOR
INITIAL BLOCK             20          0
                    BYTES
BLOCK POINTER      0&1   20          10
                         DECIMAL VALUE
PRGRM START        2&3      28672
1ST LINK           4&5      1954
1ST LINE #         6&7      9149
HEX VALUES
  00 :   14 0A 00 70 A2 07 BD 23
  08 :   70 95 78 CA D0 F8 A2 0A
  10 :   8E E2 03 A2 00 8E E3 03
  18 :   86 83 86 7C 86 81 CA 86
  20 :   BC A2 64 86 82 86 80 60
  28 :   4C D0 72 00 4C F8 72 20

CHARACTER VALUES:

TJ@0▄G┘#0▌8╲┳▄J┐▄C▄@┐‾C▒‾▒C▒▌ ▒▀▄$▒▄▒
L┐2@L‾2 0@r. rU 3 IH _2Q▄ _2Q ( r[╲, rCLC/ 0

 'C' TO CONTINUE: +/- CHANGE LINE SCAN
```

Figure 4. First Page Data Machine Language Program

This program was put into high memory starting at 28672. Note the rather random CHARACTER VALUES, and FIRST LINK and LINE values.

```
GOTO5        FIRST PAGE DATA

                         TRACK      SECTOR
INITIAL BLOCK             17          1
                    BYTES
BLOCK POINTER      0&1   17          11
                         DECIMAL VALUE
PRGRM START        2&3      21062
1ST LINK           4&5      20805
1ST LINE #         6&7      17749
HEX VALUES
  00 :   11 0B 46 52 45 51 55 45
  08 :   4E 43 59 20 42 59 20 52
  10 :   52 0D 20 37 30 20 0D 20
  18 :   38 20 0D 20 38 20 0D 46
  20 :   52 45 51 0D 20 37 20 0D
  28 :   52 52 0D 20 32 20 0D 53

CHARACTER VALUES:

QKFREQUENCY BY RRM 70 M 8 M 8 MFREQM 7 M
RRM 2 MSVCEM 5 MCMNTSM 24 MUSAGEM 1 MSIG

 'C' TO CONTINUE: +/- CHANGE LINE SCAN
```

Figure 5. First Page Data Sequential Data File

The easiest way to identify this type of file is to observe the data items separated by "M" in the CHARACTER VALUES section. The "M" is the screen representation of CHR$(13) and is in reverse field on the screen.

```
GOTO5
            FILE SUMMARY

SEQ #      BLOCKS      START        FINISH
                      TR   SEC     TR   SEC
  1          11       17    0      17    12
  2          13       17    1      16     5
  3           3       16    0      16    20
  4           7       16    1      16     4
  5          40       16    6      14    20
  6           1       14    2      14     2
  7           2       14   12      14     4
  8           8       14   14      13     7

       TYPE ANY KEY TO CONTINUE
```

Figure 6. File Summary

This table summarizes the completed scan results. The START track and sector numbers are needed to use the LOOK and RETRIEVE options.

```
         RECOVERED DISK SEQUENCES

              SEQUENCE 8

                       TRACK     SECTOR

START                   14         14

PRESENT BLOCK   8       13          7

END                     13          7

NUMBER OF BLOCKS        8


 (C-CONT  P-PAUSE  H-HALT  S-SUMMARY)

DISK ERROR AT PROGRAM LINE  5021

ERROR MESSAGE: 22  READ ERROR  13 , 0

READY.
```

Figure 7. Recovered Disk Sequence

This is what you may see if you try to recover files from a part of the disk where no files have been written. Simply type GOTO 1000 to continue.

```
0 GOTO400
5 PRINT"RUN":LIST500-525
10 INPUT"SAVE ON DRIVE #";A:A$="DISK ¬
      ¬FILE RCVRY":IFA<>0ANDA<>1GOTO10
11 B$=STR$(A)+":"+A$:OPEN15,8,15,"S"+B$:
      ¬CLOSE15:PRINTA$" SCRATCHED
12 SAVEB$,8:VERIFYB$,8:PRINTA$" SAVED ¬
      ¬AND VERIFIED":END
400 REM ******************************
401 REM *
402 REM *    DISK FILE RETRIEVER
403 REM *
404 REM *      BY  DAVID CONE
405 REM *
425 REM ******************************
427 REM * PUT DISK WITH LOST FILES
```

```
429 REM * INTO DRIVE 1.  THIS PROGRAM
431 REM * WILL THEN SEARCH FOR FILES
433 REM * IN TRACKS 17-1 & 19-35 AND
435 REM * RETURN THE START BLOCK, END
437 REM * BLOCK, AND NUMBER OF BLOCKS.
439 REM * THE FIRST 256 BYTES OF ANY
441 REM * FILE MAY BE EXAMINED TO
443 REM * ALLOW IDENTIFACATION.
445 REM * ANY IDENTIFIED FILE CAN BE
447 REM * COPIED ONTO DRIVE Ø.
45Ø REM ******************************
452 REM * USAGE: DRIVES:
454 REM *    DRIVE Ø:  GOOD DISK
456 REM *    DRIVE 1:  DAMAGED DISK
458 REM *  LOGICAL FILES:
47Ø REM *    1: WRITE FROM KEYBOARD
472 REM *    8: READ FROM DISK
474 REM *    9: WRITE TO DISK
476 REM *   15: DISK CONTROL
478 REM *  DISK CHANNEL: 2
48Ø REM *  DISK BUFFER: #2 (19ØØ-19FF)
482 REM ******************************
484 REM *  PROGRAM ENTRY POINTS
486 REM *     1Ø - SAVE & VERIFY PRGM
488 REM *   1ØØØ - PRINT SUMMARY TABLE
49Ø REM *   11ØØ - CHOICES
492 REM ******************************
5ØØ REM ** NUMERICAL VARIABLES
5Ø1 P(Ø)=Ø:REM PRESENT TRACK #
5Ø2 P(1)=Ø:REM PRESENT SECTOR #
5Ø3 L(Ø)=Ø:REM TRACK LINK
5Ø4 L(1)=Ø:REM SECTOR LINK
5Ø5 SR=17: REM START TRACK (17 OR 19)
5Ø6 SP=ØØ: REM END TRACK
5Ø7 SM=2Ø: REM MAX # OF SCTRS IN TRACK
5Ø8 TR=SR: REM VARIABLE TRACK VALUE
5Ø9 :M=Ø1: REM SEQUENCE #
51Ø MM=5Ø: REM MAX # OF SEQUENCES
511 :N=ØØ: REM # OF BLOCKS IN SEQ
512 PS=ØØ: REM START OF BASIC
513 PL=ØØ: REM 1ST LINK POINTER
514 PN=ØØ: REM 1ST LINE #
515 ES=ØØ: REM ERROR IN SECTOR
516 ET=ØØ: REM ERROR IN TRACK
517 EL=ØØ: REM DISK ERROR IN LINE #
524 :     : REM  A,J,K GENERAL VARBLS
53Ø REM **
531 REM ** STRING VARIABLES
532 :      :REM A$,Z$ GENERAL VARIABLES
533 B$="" :REM  CONT RUN VARIABLE
534 F$="" :REM  NEW FILE NAME
535 T$="" :REM  TYPE OF FILE CREATED
536 EN$="ØØ" :REM  DISK ERROR #
537 EM$=" OK":REM  DISK ERROR MESSAGE
538 S$="   <<<" :REM  STRING UTILITY
539 H$="Ø123456789ABCDEF":REM HEX DGTS
54Ø ZG$="ĥGOTO5":REM  STRING UTILITY
575 REM **
576 REM ** MATRIX VARIABLES
577 DIMD%(35,2Ø,3):REM  BLOCK POINTERS
578 : REM  ^  ^ Ø,1 IN TRK & BLK PNTRS
579 : REM  ^  ^ 2,3OUT TRK & BLK PNTRS
58Ø : REM  ^  2Ø-BLOCK NUMBER
581 : REM 35- - -TRACK NUMBER
582 DIMS%(MM,4):REM  SEQUENCE DATA
583 : REM    Ø,1  START TRACK & BLOCK
584 : REM    2    NUMBER OF BLOCKS
585 : REM    3,4  END   TRACK & BLOCK
```

```
590 REM **
595 OPEN15,8,15:EL=595:GOSUB5100
600 REM ** PAGE 1/INTRODUCTION,TRACKS
605 OPEN1,0:PRINTZG$:PRINTTAB(9)"↓↓rDISK
    ¬ RECOVERY PROGRAM
610 PRINT"↓↓   PUT DISK FOR RECOVERY IN ¬
    ¬ rDRIVEr̂ rl
615 PRINT"↓↓↓   HIT ANY KEY WHEN rDISKr̂ ¬
    ¬IS IN PLACE.
620 PRINT"↓     (DISK WILL THEN BE ¬
    ¬INITIALIZED)":GOSUB4000
625 PRINT#15,"I1":EL=625:GOSUB5100
630 PRINT"↓↓ START: TRACK r17r̂ (DOWN) ¬
    ¬OR r19r̂ (UP)? r17r̂←←r";:INPUT#1,
    ¬SR:PRINT
635 IFSR<>17ANDSR<>19THENPRINT"↑↑↑↑":
    ¬GOTO630
640 PRINTTAB(7)"↓END SEARCH AT TRACK:
    ¬        ←←←←r";:INPUT#1,SP:PRINT
645 IFSR=17THENIFSP<1ORSP>16THENPRINT"↑↑
    ¬↑":GOTO640
650 IFSR=19THENIFSP<20ORSP>35THENPRINT"↑
    ¬↑↑":GOTO640
655 CLOSE1:FORJ=0TO1000:NEXT
660 REM ** PAGE 2/DESCRIPTION
662 POKE59468,14:PRINTZG$:PRINT"↓TRACKS"
    ¬SR"TO"SP"WILL NOW BE SEARCHED FOR
664 PRINT"h↓↓↓FILES.  THE INITIAL AND ¬
    ¬ENDING BLOCKS
666 PRINT"AND THE LENGTH OF EACH ¬
    ¬RECOVERED FILE
668 PRINT"ARE RECORDED.  (TRACK ¬
    ¬SUMMARIES ARE
670 PRINT"ALSO DISPLAYED).
672 PRINT"↓THIS SEARCH CAN RUN CONTINUOU
    ¬SLY, BE
674 PRINT"HALTED AFTER EACH OPERATION,
    ¬ HAVE A
676 PRINT"PAUSE AFTER EACH OPERATION,
    ¬ OR BE ENDED
678 PRINT"WITH A JUMP TO THE SUMMARY.
680 PRINT"↓↓       TYPE rCr̂ FOR CONTINUOUS
    ¬ RUNNING
682 PRINT"       TYPE rHr̂ FOR HALT IN ¬
    ¬OPERATION
684 PRINT"       TYPE rPr̂ FOR PAUSES IN ¬
    ¬OPERATION
686 PRINT"       TYPE rSr̂ TO ESCAPE TO ¬
    ¬SUMMARY
688 PRINT"↓↓OPERATIONAL MODE MAY BE ¬
    ¬CHANGED DURING
690 PRINT"THE SEARCH BY TYPING THE ¬
    ¬ABOVE COMMANDS AT ANY TIME.
692 GOSUB4000:B$=A$
694 PRINTZG$:POKE59468,12:TR=SR:M=1
696 OPEN8,8,2,"#2":EL=695:GOSUB5100
700 REM ** FIND INITIATOR BLOCKS
705 P(0)=TR:GOSUB5010:REM @ GET NUMBER ¬
    ¬OF BLOCKS IN TRACK TR
710 FORK=0TOSM:P(1)=K:REM @ SEARCH TRK
715 :IFD%(P(0),P(1),2)<>0GOTO735
720 :GOSUB5020:REM @ GET LNKS/OUT PTRS
725 :IFL(0)=0THEND%(P(0),P(1),2)=64:
    ¬GOTO735 :REM @ END OF FILE FOUND
730 :GOSUB5040:REM @ IN PTRS TO NXT BLK
735 NEXT
740 GOSUB4060:REM @ TRK TABLE
745 REM ** FOLLOW LINKS FOR EACH START
```

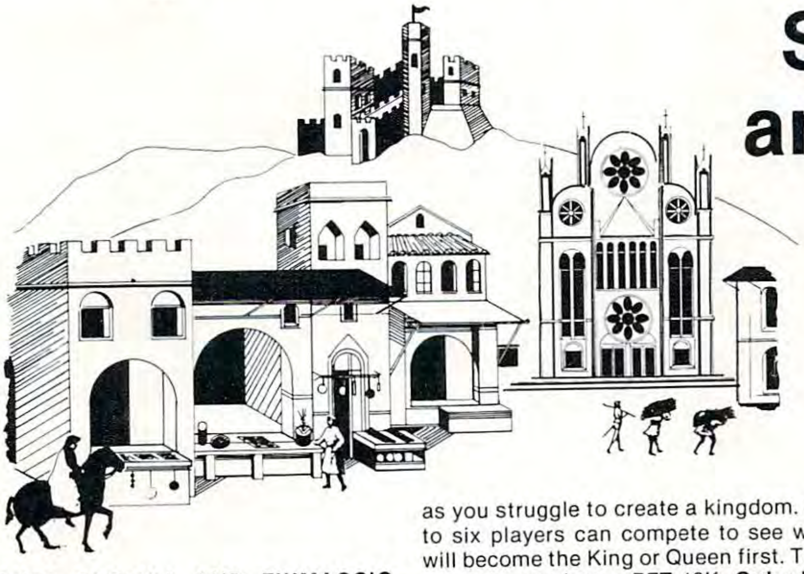
```
750 P(Ø)=TR
755 FORK=ØTOSM:P(1)=K:N=1
760 :IFD%(P(Ø),P(1),Ø)<>ØGOTO835
765 :S%(M,Ø)=P(Ø):S%(M,1)=P(1)
770 :GOSUB4Ø1Ø:REM @ PRINT DISPLAY
775 :PRINT"↑"TAB(14)S$N,S$P(Ø),S$P(1)
780 :GOSUB5Ø2Ø:REM @ GET LINKS
785 :IFL(Ø)=ØTHEND%(P(Ø),P(1),2)=64:
     ¬GOTO8Ø5
790 :GOSUB5Ø4Ø:REM @ IN PTRS TO LNK BLK
795 :P(Ø)=L(Ø):P(1)=L(1):N=N+1
800 :GOTO775
805 :REM @ CLOSE END OF LNK SEQUENCE
810 :S%(M,2)=N:S%(M,3)=P(Ø)
815 :S%(M,4)=P(1)
820 :GOSUB4Ø2Ø:REM @ COMPLETE DISPLAY
825 :IFB$="S"THENK=SM
830 :M=M+1:P(Ø)=TR:REM @ RESETS
835 NEXT
840 IFB$="S"GOTO1ØØØ
845 TR=TR-1:IFSR=19THENTR=TR+2:REM @ GO ¬
     ¬ON TO NEXT TRACK
850 REM @ TEST FOR END TRACK
855 IFSR=17ANDTR<SPGOTO1ØØØ
860 IFSR=19ANDTR>SPGOTO1ØØØ
865 GOTO7ØØ
1ØØØ REM ** PRINT OUT SUMMARY
1ØØ5 K=Ø:B$="H"
1Ø1Ø PRINTZG$;:PRINTTAB(11)"↓rFILE ¬
     ¬SUMMARY"
1Ø15 PRINT"↓rSEQ #","BLOCKS"," START ",
     ¬" FINISH
1Ø2Ø PRINT,,"TR  SEC","TR  SEC"
1Ø25 FORJ=1+15*KTO15+15*K
1Ø3Ø :IFJ>M-1THENJ=15+15*K:GOTO1Ø45
1Ø35 :GOSUB4Ø4Ø:REM @ FORMAT NUMBERS
1Ø4Ø :PRINT" "J," "S%(J,2),A$,Z$
1Ø45 NEXT
1Ø5Ø K=K+1:PRINT"↓":GOSUB4Ø3Ø
1Ø55 IFJ<MGOTO1Ø1Ø
11ØØ REM ** CHOICES
11Ø5 PRINTZG$:PRINT"↓↓↓rCHOICESR:
     ¬     rLROOK: FIRST 24Ø BYTES
111Ø PRINTTAB(12)"↓rSRUMMARY REVIEW
1115 PRINTTAB(12)"↓rRRETRIEVE A FILE
112Ø PRINTTAB(12)"↓rBR SCAN OTHER BAND
1125 PRINTTAB(12)"↓r^R DIFFERENT DISK
113Ø PRINTTAB(12)"↓rERXIT PROGRAM
1135 GOSUB4ØØØ:IFA$="L"GOTO12ØØ
114Ø IFA$="S"GOTO1ØØØ
1145 IFA$="R"GOTO15ØØ
115Ø IFA$="B"THENCLOSE8:GOTO6ØØ
1155 IFA$="^"THENGOSUB4Ø5Ø:RUN4ØØ
116Ø IFA$="E"THENGOSUB4Ø5Ø:PRINTZG$:END
1165 GOTO1135
12ØØ REM ** GET 1ST PAGE OF FILE DATA
12Ø5 INPUT"↓↓↓INPUT TRACK, SECTOR";J,K
121Ø IFJ>35ORJ<1GOTO11ØØ
1215 P(Ø)=J:GOSUB5Ø1Ø:IFK<ØORK>SMGOTO11Ø
     ¬Ø
122Ø P(1)=K:GOSUB5Ø2Ø:REM @ LNKS L(Ø) ¬
     ¬L(1)
1225 A=2:GOSUB5Ø7Ø:REM @ READ NEXT ¬
     ¬BYTES  (SET BP)
123Ø GOSUB5Ø5Ø:PS=A:REM @ START BASIC
1235 GOSUB5Ø5Ø:PL=A:REM @ 1ST PROG LINK
124Ø GOSUB5Ø5Ø:PN=A:REM @ 1ST LINE #
1245 A=2:GOSUB5Ø7Ø:REM @ GET STRING
```

```
1250 Z$=""
1255 FORJ=ØTO25Ø
1260 :GET#8,A$:GOSUB5Ø6Ø:Z$=Z$+A$
1265 NEXT
1270 Z$=CHR$(L(Ø))+CHR$(L(1))+Z$
1275 PRINTZG$TAB(12)"ɼFIRST PAGE DATA":
     ¬REM @ PRINT DATA
1280 PRINT,,"↓ɼTRACK","SECTOR
1285 PRINT"INITIAL BLOCK",P(Ø),P(1)
1290 PRINTTAB(15)"ɼBYTES
1295 PRINT"BLOCK POINTER"TAB(16)"Ø&1",
     ¬L(Ø),L(1)
1300 PRINTTAB(2Ø)"ɼDECIMAL VALUE
1305 PRINT"PRGRM START"TAB(16)"2&3"TAB(2
     ¬3)PS
1310 PRINT"1ST LINK"TAB(16)"4&5"TAB(23)P
     ¬L
1315 PRINT"1ST LINE #"TAB(16)"6&7"TAB(23
     ¬)PN
1320 PRINT"ɼHEX VALUES
1325 FORJ=ØTO5
1330 :A=8*J:GOSUB5Ø8Ø:PRINT"  "A$" :  ";
1335 :FORK=ØTO7
1340 ::A=ASC(MID$(Z$,8*J+K+1,1))
1345 ::GOSUB5Ø8Ø:PRINTA$" ";
1350 :NEXT:PRINT
1355 NEXT
1360 PRINT"↓ɼCHARACTER VALUES:↓"
1365 K=Ø
1370 FORJ=1TO8Ø
1375 :A=ASC(MID$(Z$,J+8Ø*K,1))
1380 :GOSUB5Ø9Ø
1385 NEXT
1390 PRINT"↓ 'C' TO CONTINUE: +/- ¬
     ¬CHANGE LINE SCAN"
1395 GOSUB4ØØØ:IFA$="C"GOTO11ØØ
1400 IFA$="+"ANDK<2THENPRINT"↑↑↑↑";:
     ¬K=K+1:GOTO137Ø
1405 IFA$="-"ANDK>ØTHENPRINT"↑↑↑↑";:
     ¬K=K-1:GOTO137Ø
1410 GOTO1395
1500 REM ** RETRIEVE A FILE
1505 INPUT"↓↓↓STARTING TRACK, SECTOR";J,
     ¬K
1510 IFJ>35ORJ<1GOTO11ØØ
1515 P(Ø)=J:P(1)=K:GOSUB5Ø1Ø:IFK<ØORK>SM
     ¬GOTO11ØØ
1520 PRINTZG$SPC(1Ø)"ɼFILE RETRIVAL
1525 PRINT"↓↓RETRIEVING THE FILE ¬
     ¬STARTING AT:"
1530 PRINTTAB(3)"TRACK ɼ"P(Ø)"r̂"TAB(2Ø)"
     ¬SECTOR ɼ"P(1)
1535 PRINT"↓NAME FOR RETRIEVED FI←↓↓^↑↑L
     ¬E:":INPUT"    ";F$
1540 PRINT"↓IS THIS A ɼPr̂ROGRAM OR A ¬
     ¬ɼSr̂EQUENTIAL FILE"
1545 GOSUB4ØØØ:IFA$="P"THENT$="PRG":
     ¬PRINT"    ɼPROGRAM":GOTO156Ø
1550 IFA$="S"THENT$="SEQ":PRINT"    ¬
     ¬ɼSEQUENTIAL":GOTO156Ø
1555 GOTO1545
1560 PRINT"↓PUT GOOD DISK WITH SUFFICIEN
     ¬T BLOCKS         IN DRIVE ɼØ
1565 PRINT"↓INITIALIZE? (Y/N)":GOSUB4ØØØ
1570 IFA$="Y"THENPRINT#15,"IØ":EL=1535:
     ¬GOSUB51ØØ
1575 GOSUB4Ø3Ø
1580 PRINT"↓ɼIS EVERYTHING OKr̂?  TYPE ¬
     ¬'C'  TO GO!
1585 PRINT"    ↓(ANY OTHER LETTER WILL ¬
     ¬ABORT)↓":GOSUB4ØØØ
1590 IFA$<>"C"GOTO11ØØ
1595 REM ** RETRIEVE PROGRAM
1600 OPEN9,8,4,"Ø:"+F$+","+T$+",WRITE":
     ¬EL=16ØØ:GOSUB51ØØ
1605 EL=1555:GOSUB51ØØ
1610 PRINT"COPYING TRACK:"P(Ø)TAB(21)"SE
     ¬CTOR:"P(1)
1615 GOSUB5Ø2Ø:P(Ø)=L(Ø):P(1)=L(1):
     ¬REM @ GET LINKS
1620 A=255:IFP(Ø)=ØTHENA=P(1)
1625 FORJ=2TOA
1630 :PRINT#15,"M-R";CHR$(J);CHR$(19)
1635 :GET#15,A$:GOSUB5Ø6Ø
1640 :PRINT#9,A$;
1645 NEXT:EL=157Ø
1650 IFP(Ø)<>ØGOTO161Ø:REM @ GET NEXT ¬
     ¬BLOCK
1655 CLOSE9:PRINT"↓ɼFILE RECOVERED↓":
     ¬GOSUB4Ø3Ø:GOTO11ØØ
4000 REM ** GET AND HOLD
4001 FORA=ØTO1Ø:GETA$:NEXT
4002 GETA$:IFA$=""GOTO4ØØ2
4003 REM @ SCREEN DUMP:IFA$="P"THENGOSUB
     ¬'SCREEN DUMP'
4005 A=VAL(A$)
4009 RETURN
4010 REM ** PRINT DISPLAY OF RECOVERED ¬
     ¬SEQUENCES
4011 PRINTZG$:PRINTTAB(7)"↓RECOVERED ¬
     ¬DISK SEQUENCES
4012 PRINTTAB(12)"↓SEQUENCE"M
4013 PRINT,,"↓TRACK","SECTOR":PRINT"↓ɼST
     ¬ART",,S%(M,Ø),S%(M,1)
4014 PRINT"↓PRESENT BLOCK":RETURN
4020 REM ** BOTTOM OF DISPLAY
4021 PRINT"↓ɼEND",,S%(M,3),S%(M,4)
4022 PRINT"↓NUMBER OF BLOCKS",S%(M,2)
4023 PRINT"↓↓ (C-CONT  P-PAUSE  H-HALT  ¬
     ¬S-SUMMARY)
4024 GOSUB4Ø7Ø:RETURN
4030 REM ** TYPE ANY KEY---
4031 PRINTTAB(7)"TYPE ANY KEY TO ¬
     ¬CONTINUE":GOSUB4ØØØ:RETURN
4040 REM ** FORMAT NUMBERS
4041 A$="ɼ"+RIGHT$(STR$(S%(J,Ø)),
     ¬2)+"    "+RIGHT$(STR$(S%(J,1)),
     ¬2)+"r̂
4042 Z$=RIGHT$(STR$(S%(J,3)),2)+"    ¬
     ¬"+RIGHT$(STR$(S%(J,4)),2)
4043 RETURN
4050 REM ** CLOSE FILES
4051 PRINT#15,"B-P,2,Ø":CLOSE8:CLOSE15:
     ¬RETURN
4060 REM ** PRINT TRACK SUMMARY TABLE
4061 PRINTZG$:PRINTTAB(8)"TRACK ¬
     ¬ɼ"TR"r̂SUMMARY TABLE
4062 PRINT"SEC ⊥ TR  ɼINr̂   SC  ⊥    TR ¬
     ¬ ɼOUTr̂  SC
4063 FORJ=ØTOSM:PRINTJ;:FORK=ØTO3
4064 PRINTTAB(5+9*K)D%(P(Ø),J,K);
4065 NEXT:PRINT:PRINT"↑"TAB(4)"⊥"SPC(14)
     ¬"⊥":NEXT
4066 GOSUB4Ø7Ø:RETURN
```

```
 4070 REM ** PAUSE/STOP CONTROL
 4071 GETA$:IFA$="H"ORA$="C"ORA$="P"ORA$=
      ¬"S"THENB$=A$
 4072 IFB$="H"THENGOSUB4030:RETURN
 4073 IFB$="P"THENFORJ=0TO5000:NEXT
 4074 GETA$:IFA$="P"THENGOSUB4003:RETURN
 4075 IFA$="H"ORA$="C"ORA$="S"THENB$=A$
 4076 RETURN
 5010 REM ** NUMBER OF BLOCKS IN TRACK
 5011 SM=16:IFP(0)<31THENSM=17
 5012 IFP(0)<25THENSM=19
 5013 IFP(0)<18THENSM=20
 5014 RETURN
 5020 REM ** GET LNKS--P(0)P(1)IN:
      ¬L(0)L(1)OUT--OUT PTRS SET
 5021 PRINT#15,"U1:";2;1;P(0);P(1)
 5022 EL=5021:GOSUB5100
 5023 FORJ=0TO1:REM @ GET LINKS
 5024 :PRINT#15,"M-R";CHR$(J);CHR$(19)
 5026 :GET#15,A$:GOSUB5060
 5028 :L(J)=ASC(A$):D%(P(0),P(1),
      ¬J+2)=L(J)
 5030 NEXT:A=2:GOSUB5070:RETURN
 5040 :REM ** IN POINTERS FOR LINK BLOCK
 5041 :FORJ=0TO1
 5042 ::D%(L(0),L(1),J)=P(J)
 5043 :NEXT:RETURN
 5050 REM ** GET DEC VALUE-2 BYTES
 5051 GET#8,A$:GOSUB5060:REM @ LO
 5052 A=ASC(A$)
 5053 GET#8,A$:GOSUB5060:REM @ HI
 5054 A=ASC(A$)*256+A:RETURN
 5060 REM ** WHEN A$=""
 5061 IFA$=""THENA$=CHR$(0)
 5062 RETURN
 5070 REM ** SET B-P
 5071 PRINT#15,"B-P,2,";A
 5072 EL=5071:GOSUB5100:RETURN
 5080 REM ** DEC>HEX
 5081 A$=MID$(H$,1+(240ANDA)/16,1)+MID$(H
      ¬$,1+(15ANDA),1):GOSUB5060:RETURN
 5090 REM ** ASC > CHARACTERS
 5091 IFA<32THENPRINT"ɼ"CHR$(A+64)"f̂";:
      ¬RETURN
 5092 IFA=34ORA=98THENPRINTCHR$(34)CHR$(3
      ¬4)CHR$(20);:RETURN
 5093 IFA<128GOTO5095
 5094 IFA<160THENPRINT"ɼ"CHR$(A+32)"f̂";:
      ¬RETURN
 5095 PRINTCHR$(A);:RETURN
 5100 REM ** CHECK DISK ERROR
 5101 INPUT#15,EN$,EM$,ET,ES:IFEN$="00"TH
      ¬ENRETURN
 5102 PRINT"↓↓ɼDISK ERRORf̂ AT PROGRAM ¬
      ¬LINE "EL
 5105 PRINT"↓ERROR MESSAGE: "EN$"  "EM$,
      ¬ET","ES
 5107 IFEN$="22"GOTO1000
 5109 END
READY.
```

©

Captivate Yourself.



Santa Paravia and Fiumaccio

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

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

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

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

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

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

CHIMERA

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

•**Reflex**—Round and round the little white ball rolls. Only fast reflexes can guide it into the center of the maze.

•**Dragon**—You'll have to shoot down those pesky, fire-breathing dragons with your cannon. If you succeed your castle will be safe, if not it will mean a call to your fire insurance company. For one player.

•**Dungeon**—A very punctual guard comes down to the dungeon every day to torture you. This means that you have only thirty seconds to dig your way under the castle to freedom. For one player.

•**Dragon Hunt**—You must go forth and slay a fire-breathing dragon. The only thing that will protect you from the flames is your shield, if you know when to use it. For one player.

•**Dropoff**—You must make your opponent's men "dropoff" the board by moving and firing your own men. For one or two players. **Order No. 0110P. $9.95.**

PET DEMO I

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

•**Slot Machine**—You won't be able to resist the enticing messages from this computerized one-armed bandit.

•**Chase**—You must find the black piece as you search through the ever-changing maze.

•**Flying Pheasant**—Try to shoot the flying pheasant on the wing.

•**Sitting Ducks**—Try to get your archer to shoot as many ducks as possible for a high score.

•**Craps**—It's Snake Eyes, Little Joe, or Boxcars as you roll the dice and try to make your point.

•**Gran Prix 2001**—Drivers with experience ranging from novice to professional will enjoy this multi-leveled race game.

•**Fox and Hounds**—It's you against the computer as your four hounds try to capture the computer's fox.

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

DOW JONE$

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

We Guarantee It!

Instant Software Guarantee

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

Prices subject to change without notice.

TO ORDER

SEE YOUR LOCAL INSTANT SOFTWARE DEALER

OR

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

Instant Software™

*A trademark of Commodore Business Machines

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

Pet Exec Hello

Gordon Campbell
Toronto, Canada

When you turn on your Pet, what do you do next? I found that there was a standard set of commands, which went along with the particular diskette being used. For example, for program development, the commands I use are:

POKE 59468, 14 (set upper and lower case)
POKE 59458, 62 (this may damage YOUR machine)
OPEN 1,8,15,"IO" (because DOS is first)
LOAD "BASIC AID", 8 (extensions to BASIC
SYS 7*4096 (invoke BASIC AID)
DISK "$0" (directory)
REPEAT (turn on auto-repeat keyboard)

Eventually, you get tired of issuing the same old commands over and over. So I did something about it. PET EXEC HELLO is a suite of three small programs which allow you to use a 'HELLO' file on disk.

The HELLO file consists of a set of direct commands which are executed when you 'boot' from the disk. The first file on the disk must be the SIGNON program. It prints a greeting, and invokes a machine-language program called EXEC HELLO. EXEC HELLO reads in the whole HELLO file, and feeds it to the keyboard buffer one character at a time. At the end of the commands, EXEC HELLO disengages. BUILD HELLO is a program which helps you create HELLO files.

```
SIGNON    - NOV 26, 1980        PAGE  1

 100   IF PEEK (13) THEN 160
 110   PRINT "QQQpet 'exec hello' in operation
 120   PRINT "Q  for upgrade rom - 32k disk
 130   PRINT "QQ
 140   POKE 13,1
 150   LOAD "exec hello",8
 160   POKE 13,0
 170   SYS 6 * 16 ↑ 3 + 4 * 256
```

```
                0010 ;    PET EXEC-HELLO
                0020 ;   FOR 'UPGRADE (3.0) ROM'
                0030 ;  AS OF NOV 26, 1980
                0040 ;  SAVED AS 'PEH ML V5'
                0050 ;
                0060 ;  COPYRIGHT (C) 1980
                0070 ;  BY GORDON CAMPBELL
                0080 ;     36 DOUBLETREE ROAD
                0090 ;     WILLOWDALE, ONTARIO
                0100 ;                 M2J 3Z4
                0110 ;     PHONE (416) 492-9518
                0120 ;
                0130 ; PERMISSION TO MODIFY OR COPY FOR
                0140 ; NON-COMMERCIAL PURPOSES IS HEREBY
                0150 ; GRANTED, PROVIDED THAT THE COPYRIGHT
                0160 ; AND THIS NOTICE IS RETAINED.
                0170 ;
                0180 ; THIS PROGRAM IS INVOKED BY THE BASIC
                0190 ; PROGRAM 'SIGNON'. IT RUNS IN UNPROTECTED
                0200 ; MEMORY BY DESIGN, SO IT SHOULD BE
                0210 ; INVISIBLE TO OTHER OPERATIONS. ANY REALLY
                0220 ; BIG PROGRAM WILL SMEAR IT. ANYTHING
                0230 ; WHICH USES INTERRUPTS (EG. AUTO REPEAT
                0240 ; KEYBOARD) WILL EITHER COME TO GRIEF
                0250 ; OR SIMPLY DISENGAGE IT. THATS OK IF
                0260 ; IT'S THE LAST COMMAND.
                0270 ;
                0280 ; THE EXCEPTION IS 'LOAD', WHICH I WATCH
                0290 ; FOR. IF THERE IS A LOAD, I GENERATE
                0300 ; A 'SYS 0' TO RE-ENGAGE.
                0310 ;
                0320 ; TRY TO AVOID DOS COMMANDS WHICH MAY
                0330 ; CAUSE TROUBLE.  FOR EXAMPLE, USE:
                0340 ;   LOAD"$0",8
                0350 ;   LIST
                0360 ;   NEW       RATHER THAN >$0
                0370 ;
                0380              .BA $6400
                0390              .OS
                0400 ;
6400- 4C 17 64  0410              JMP ENTRY     ;SKIP AROUND FILENAME
                0420 ;
                0430 ;
6403- 48 45 4C  0440 FNAME        .BY 'HELLO,P,R' 0 0 0
6406- 4C 4F 2C
6409- 50 2C 52
640C- 00 00 00
640F- 00 00 00  0450              .BY 0 0 0 0 0 0 0 0
6412- 00 00 00
6415- 00 00
                0460 ;
6417- A2 00     0470 ENTRY        LDX #0
6419- BD 03 64  0480 SHLOOP       LDA FNAME,X   ; FIND LENGTH
641C- F0 03     0490              BEQ LENFND    ; OF FILENAME
641E- E8        0500              INX
641F- D0 F8     0510              BNE SHLOOP    ; JUMP
                0520 ;
6421- 8E 7A 65  0530 LENFND       STX FNLEN
                0540 ;
6424- A9 0F     0550              LDA #$0F      ; OPEN
6426- 85 D2     0560              STA *FNUM     ; CONTROL
6428- A9 08     0570              LDA #8
642A- 85 D4     0580              STA *DEV      ; CHANNEL
642C- A9 0F     0590              LDA #$0F      ; 15,8,15
642E- 09 60     0600              ORA #$60
6430- 85 D3     0610              STA *SCNDRY
6432- A9 00     0620              LDA #0
6434- 85 D1     0630              STA *OPLEN    ; NO FILENAME
6436- 85 96     0640              STA *ST
6438- 20 24 F5  0650              JSR OPEN      ; ROM ROUTINE
                0660 ;
643B- A9 08     0670              LDA #8        ; OPEN
643D- 85 D2     0680              STA *FNUM     ; TEXT
643F- 85 D4     0690              STA *DEV      ; FILE
6441- 09 60     0700              ORA #$60      ; 8,8,8
6443- 85 D3     0710              STA *SCNDRY
6445- AD 7A 65  0720              LDA FNLEN     ; LENGTH OF
6448- 85 D1     0730              STA *OPLEN    ; FILE NAME
644A- A9 64     0740              LDA #H,FNAME
644C- 85 DB     0750              STA *FNPTR+1          ; AND IT'S
644E- A9 03     0760              LDA #L,FNAME          ; ADDRESS
6450- 85 DA     0770              STA *FNPTR
6452- A9 00     0780              LDA #0
6454- 85 96     0790              STA *ST
6456- 20 24 F5  0800              JSR OPEN      ; ROM ROUTINE
                0810 ;
6459- 20 C1 64  0820              JSR ERRCHK
                0830 ;
645C- A9 02     0840              LDA #2        ; SKIP PAST
645E- 8D 7B 65  0850              STA NCHRS     ; LOAD-ADDRESS
```

Professional Business Software

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

General Ledger

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

Accounts Payable

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

Accounts Receivable

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

Payroll

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

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

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

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

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

CMS Software Systems

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

Program Operation

As I said earlier, the first program on the disk must be SIGNON. Thus, after turning on the Pet, key the following:

CLR 4 spaces "*",8 Home RUN

This causes the first program on the disk to be loaded and run. Very quickly, you see the commands which were entered earlier using BUILD HELLO, appear on the screen. At the end of the HELLO file you regain control at the keyboard. (Or earlier if one of the commands disengages EXEC HELLO).

Program Details

SIGNON

This tiny program uses one trick. Since a LOAD command issued by a program will cause a restart, location 13 is used as a flag that we are restarting after loading the machine-language program. This location normally contains a zero.

BUILD HELLO

This program does very straightforward text entry. I chose to save the HELLO text as a program file on disk, so the text is poked into memory, and the machine-language monitor invoked to save the results. The cursor-control keys are thus all active, and characters such as double-quote and comma cannot cause any problem. The only checking done in the program is to ensure that the text is not too large for the area allocated to it in EXEC HELLO.

EXEC HELLO

The first thing done in EXEC HELLO is to count the number of characters in the filename. This allows the name to be changed by POKE's without having to reassemble the program. Next the error-channel and the text file are opened. The error-channel is interrogated to make sure there is a HELLO file on the disk. If not, the message *ERROR* is printed on the screen and the program breaks into the monitor. The next file is read into memory, and both channels are closed. The part of the

```
6461- A2 08                    LDX #8        ; SET INPUT CHANNEL
6463- 20 C6 FF                 JSR SETIN     ; FOR TEXT FILE
6466- 20 E4 FF     PASSJK      JSR GET
6469- CE 7B 65                 DEC NCHRS
646C- D0 F8                    BNE PASSJK
                   ;
646E- AD 78 65                 LDA MYPTR     ; SET UP FOR
6471- 85 01                    STA *PTR      ; INDIRECT
6473- AD 79 65                 LDA MYPTR+1   ; ADDRESSING
6476- 85 02                    STA *PTR+1
6478- A0 00                    LDY #0
647A- 8C 7F 65                 STY SVY
                   ;
647D- 20 E4 FF     CHRGET      JSR GET
6480- C9 FC                    CMP #252      ; END OF FILE CHARACTER
6482- F0 11                    BEQ DONE
6484- AC 7F 65                 LDY SVY
6487- 91 01                    STA (PTR),Y
6489- 88                       DEY
648A- 8C 7F 65                 STY SVY
648D- C0 FF                    CPY #$FF
648F- D0 EC                    BNE CHRGET
6491- C6 02                    DEC *PTR+1
6493- D0 E8                    BNE CHRGET    ; JUMP, OR I'M DEAD
                   ;
6495- AC 7F 65     DONE        LDY SVY
6498- 91 01                    STA (PTR),Y   ; STORE EOF
649A- 20 CC FF                 JSR RESCHN    ; RESTORE CHANNEL
                   ;
649D- A9 08                    LDA #8
649F- 20 AE F2                 JSR CLOSE
64A2- A9 0F                    LDA #15
64A4- 20 AE F2                 JSR CLOSE
                   ;
64A7- A5 90        CONECT      LDA *INTHND   ;          A BIT OF
64A9- 8D E8 64                 STA GOBACK+1            ; ROM INDEPENCE
64AC- A5 91                    LDA *INTHND+1
64AE- 8D E9 64                 STA GOBACK+2
64B1- 78                       SEI   ; NO INTERRUPTS
64B2- A9 E3                    LDA #L,INTRTN
64B4- 85 90                    STA *INTHND   ; CONNECT
64B6- A9 64                    LDA #H,INTRTN
64B8- 85 91                    STA *INTHND+1
64BA- 58                       CLI
64BB- A9 00                    LDA #0
64BD- 8D 7E 65                 STA SVX
64C0- 60                       RTS   ; THATS ALL FOLKS
                   ;
                   ; EXEC FILE NOW RUNNING
                   ;
                   ;
64C1- A2 0F        ERRCHK      LDX #15       ; FILE NUMBER
64C3- 20 C6 FF                 JSR SETIN     ; SET INPUT CHANNEL
64C6- 20 E4 FF     GER         JSR GET
64C9- C9 30                    CMP #$30      ; ZERO?
64CB- F0 F9                    BEQ GER       ; OK, GET ANOTHER
64CD- C9 2C                    CMP #$2C      ; COMMA?
64CF- D0 04                    BNE ERR       ; NO - MUST BE BAD
64D1- 20 CC FF                 JSR RESCHN
64D4- 60                       RTS
                   ;
                   ;
64D5- A2 13        ERR         LDX #ERMSG+256-ERMEND
                   ; PRINT *ERROR*
64D7- BD 78 64     ERLOOP      LDA ERMEND-256,X
64DA- 20 D2 FF                 JSR PRINT
64DD- E8                       INX
64DE- D0 F7                    BNE ERLOOP
64E0- 00                       BRK
64E1- 00                       BRK
64E2- 00                       BRK
                   ;
                   ;
                   ; INTERRUPT ROUTINE
                   ;
64E3- A5 9E        INTRTN      LDA *KBUFNO
                   ; LAST CHARACTER PROCESSED?
64E5- F0 03                    BEQ SENCHR    ; YUP; GIVE HIM ANOTHER
64E7- 4C 00 00     GOBACK      JMP $0000
                   ;
                   ; ABOVE ADDRESS WILL BE FILLED IN
                   ; BY THE PROGRAM DURING EXECUTION
                   ; AS THE NORMAL INTERRUPT HANDLER
                   ;
64EA- A5 01        SENCHR      LDA *PTR      ; SAVE 'USR'
64EC- 8D 7C 65                 STA SVPTR     ; VECTOR
64EF- A5 02                    LDA *PTR+1    ; (PROBABLY DON'T
64F1- 8D 7D 65                 STA SVPTR+1   ; HAVE TO)
64F4- AD 78 65                 LDA MYPTR
64F7- 85 01                    STA *PTR      ; SET UP MY
64F9- AD 79 65                 LDA MYPTR+1   ; INDIRECT
```

program which feeds characters into the keyboard buffer is hooked in to the interrupt processor, and control is returned to BASIC. The interrupt routine sees if there are any characters in the buffer, and if not, deposits one. It looks at the text being passed, and if the word LOAD appears, sets a flag. At the end of a line, if the flag is set, then the USR vector is pointed at the re-connect routine in EXEC HELLO, and a SYS 0 added to the content of the keyboard buffer. At the end of the text everything is restored as it was.

EXEC HELLO tries to be transparent to the rest of the Pet, so it sits in unprotected memory. This means it could be clobbered if the commands RUN a program, and cause it to process far enough to build variables on top of EXEC HELLO. Note also the warnings in the comments at the start of the listing.

What next

EXEC HELLO could be modified to handle just about any purpose where running a program would cause problems but direct commands will work. Several of these cases (such as dumping the contents of a tape to disk) are handled by utility programs, but with EXEC HELLO the only thing you have to do to handle a new requirement is to change the direct commands on the HELLO file.

If you wish to obtain a disk containing PET EXEC HELLO along with a number of other programs, please send $12 to the author. If you do key it in or send for a disk, please give it to all your friends.

```
64FC- 85 02                      STA *PTR+1     ; ADDRESS
64FE- A0 00                      LDY #0
                      ;
6500- B1 01                      LDA (PTR),Y
6502- C9 FC                      CMP #252       ; END OF FILE?
6504- F0 65                      BEQ UNHOOK     ; YES
6506- 8D 6F 02                   STA KEYBUF
6509- A2 01                      LDX #1         ; PRETEND IT CAME
650B- 86 9E                      STX *KBUFNO    ; FROM KEYBOARD
                      ;
650D- AE 7C 65                   LDX SVPTR      ; RESTORE
6510- 86 01                      STX *PTR       ; 'USR' VECTOR
6512- AE 7D 65                   LDX SVPTR+1
6515- 86 02                      STX *PTR+1
                      ;
6517- C9 0D                      CMP #$0D       ; CR?
6519- D0 24                      BNE NOCR       ; NOPE
651B- AD 80 65                   LDA LFLG       ; DID WE SAY 'LOAD'?
651E- F0 1F                      BEQ NOCR       ; NOPE
                      ;
6520- A9 00                      LDA #0         ; RESTORE FLAG
6522- 8D 80 65                   STA LFLG
6525- A2 05                      LDX #5         ; SAY SYSO $0D
6527- BD 85 65        SYSLOP     LDA SYS,X
652A- 9D 6F 02                   STA KEYBUF,X
652D- CA                         DEX
652E- D0 F7                      BNE SYSLOP
                      ;
6530- A2 06                      LDX #6         ; 6 CHARACTERS
6532- 86 9E                      STX *KBUFNO
6534- A9 A7                      LDA #L,CONECT           ; SET UP
6536- 85 01                      STA *PTR       ;          RESTORE
6538- A9 64                      LDA #H,CONECT           ; HOOK
653A- 85 02                      STA *PTR+1
653C- 4C 6B 65                   JMP UNHOOK
                      ;
653F- AE 7E 65        NOCR       LDX SVX        ; WATCH
6542- DD 81 65                   CMP LOAD,X     ; OUT FOR
6545- F0 07                      BEQ CHKLD      ; 'LOAD'
6547- A9 00                      LDA #0
6549- 8D 7E 65                   STA SVX
654C- F0 0D                      BEQ ENCHKL     ; JUMP
                      ;
654E- E8              CHKLD      INX
654F- 8E 7E 65                   STX SVX
6552- E0 04                      CPX #4         ; WHOLE WORD?
6554- D0 05                      BNE ENCHKL     ; NOT YET
6556- A9 01                      LDA #1
6558- 8D 80 65                   STA LFLG       ; SET FLAG
                      ;
655B- A9 FF           ENCHKL     LDA #$FF
655D- CE 78 65                   DEC MYPTR      ; DOUBLE
6560- CD 78 65                   CMP MYPTR      ; DECREMENT
6563- D0 03                      BNE BACK
6565- CE 79 65                   DEC MYPTR+1
6568- 4C E7 64        BACK       JMP GOBACK     ; SEE YOU SOON
                      ;
656B- AD E8 64        UNHOOK     LDA GOBACK+1            ; RESTORE
656E- 85 90                      STA *INTHND    ;          INTERRUPT
6570- AD E9 64                   LDA GOBACK+2            ; VECTOR
6573- 85 91                      STA *INTHND+1
6575- 4C E7 64                   JMP GOBACK
                      ;
                      ERMEND
                      ;
                      ;
                      ; WORK AREA
                      ;
6578- 00 67           MYPTR      .SE $6700      ; TOP OF TEXT AREA
657A-                 FNLEN      .DS 1          ; LENGTH OF FILE NAME
657B-                 NCHRS      .DS 1          ; # CHARS TO SKIP
657C-                 SVPTR      .DS 2          ; POINTER SAVE AREA
657E-                 SVX        .DS 1          ; X REG SAVE AREA
657F-                 SVY        .DS 1          ; Y REG SAVE AREA
6580- 00              LFLG       .BY 0          ; =1: THIS LINE
                                 ;                CONTAINED A 'LOAD'
6581- 4C 4F 41        LOAD       .BY 'LOAD'
6584- 44
6585- 20 53 59        SYS        .BY ' SYSO' $0D
6588- 53 30 0D
658B- 2A 45 52        ERMSG      .BY '*ERROR*'
658E- 52 4F 52
6591- 2A
                      ;
                      ;
                      ; EQUATES
                      ;
                      PTR        .DE 1          ; INDIRECT ADDRESS
                      ;
                      INTHND     .DE $90        ; INT HANDLER
```

```
2620 ST        .DE $96      ; STATUS
2630 KBUFNO    .DE $9E      ; # CHARS IN BUFFER
2640 OPLEN     .DE $D1      ; LEN OF FILE NAME
2650 FNUM      .DE $D2      ; FILE NUMBER
2660 SCNDRY    .DE $D3      ; SECONDARY ADDRESS
2670 DEV       .DE $D4      ; DEVICE NUMBER
2680 FNPTR     .DE $DA      ; ADDRESS OF NAME
2690 KEYBUF    .DE $26F     ; KEYBOARD BUFFER
2700 ;
2710 ;
2720 ; TWO NON-STANDARD ROM ADDRESSES
2730 ;
2740 CLOSE     .DE $F2AE
2750 OPEN      .DE $F524
2760 ;
2770 SETIN     .DE $FFC6    ; SET CHANNEL
2780 RESCHN    .DE $FFCC    ; RESET IT
2790 GET       .DE $FFE4
2800 PRINT     .DE $FFD2
2810           .EN
```

```
LABEL FILE:  [ / = EXTERNAL ]

FNAME=6403          ENTRY=6417          SHLOOP=6419
LENFND=6421         PASSJK=6466         CHRGET=647D
DONE=6495           CONECT=64A7         ERRCHK=64C1
GER=64C6            ERR=64D5            ERLOOP=64D7
INTRTN=64E3         GOBACK=64E7         SENCHR=64EA
SYSLOP=6527         NOCR=653F           CHKLD=654E
ENCHKL=655B         BACK=6568           UNHOOK=656B
ERMEND=6578         MYPTR=6578          FNLEN=657A
NCHRS=657B          SVPTR=657C          SVX=657E
SVY=657F            LFLG=6580           LOAD=6581
SYS=6585            ERMSG=658B          /PTR=0001
/INTHND=0090        /ST=0096            /KBUFNO=009E
/OPLEN=00D1         /FNUM=00D2          /SCNDRY=00D3
/DEV=00D4           /FNPTR=00DA         /KEYBUF=026F
/CLOSE=F2AE         /OPEN=F524          /SETIN=FFC6
/RESCHN=FFCC        /GET=FFE4           /PRINT=FFD2

//0000,6592,6592
```

```
100  UL = PEEK (59468):
      POKE 59468,14
110  PRINT "Si help you create 'hello'
120  PRINT "files on disk (drive zero).
130  PRINT "Qsorry, i'm not a full text editor;
140  PRINT "use 'stop' if you change your mind.
150  PRINT "Quse shift @ to signal the end.Q
160  SL = 3 * 16 ↑ 3:
     MX = SL + 350
170  POKE 170,0
180  GET A$:
      IF A$ = "" THEN 170
190  IF A$ = "@" THEN 260
200  POKE SL, ASC (A$)
210  X = FRE (0)
220  SL = SL + 1
230  IF SL > MX THEN PRINT:
      PRINT "Qsorry, this hello file is
      too big.":
      POKE 59468,UL:
      STOP
240  PRINT A$;
250  GOTO 170
260  OPEN 15,8,15,"s0:hello"
270  CLOSE 15
280  POKE SL,252
290  SL = SL + 1
300  DIM X$(15)
310  FOR J = 0 TO 15:
      READ X$(J):
      NEXT
320  A1 = INT (SL / 16 ↑ 3)
330  SL = SL - 16 ↑ 3 * A1
340  A2 = INT (SL / 256)
350  SL = SL - 256 * A2
360  A3 = INT (SL / 16)
370  A4 = SL - 16 * A3
380  PRINT "QQQQQQQQQ.s " CHR$ (34)"0:
     hello";
390  PRINT CHR$ (34)",08,3000,"X$(A1)X$
     (A2)X$(A3)X$(A4)
400  PRINT ".x"
410  PRINT "QQQQQQQ"
420  POKE 59468,UL
430  POKE 623,13:
      POKE 624,13:
      POKE 158,2
440  SYS 64785
450  END:
      REM NEVER EXECUTED
460  DATA 0,1,2,3,4,5,6,7
470  DATA 8,9,A,B,C,D,E,F
```

©

Agricultural Software from Cyberia™

FOR PET/CBM SYSTEMS

Cyber-Farmer™ The complete accounting package for today's farm operations

- Records, sorts, combines and prints the results of the farm operation.
- Account headings and numbers are pre-assigned for nearly every type of farm income or expense, but any account may be deleted, altered or added.
- Keeps personal, family and house-hold accounts as well.
- Cyber-Farmer management tools include cash-flow analyses, depreciation schedules, inventory and budget reports.
- No computer experience is necessary to operate this system.

Farrow-Filer™

Record-keeping and management system for the pork producer

- Helps the pork producer to lower their cost-per-pig by producing more animals.
- High visual impact of reports allows for better culling of freeloaders.
- Alerts the operator to possible medical problems in the herd.
- Forecasts farrowings for a 3 month period.
- Keeps individual records on each animal.

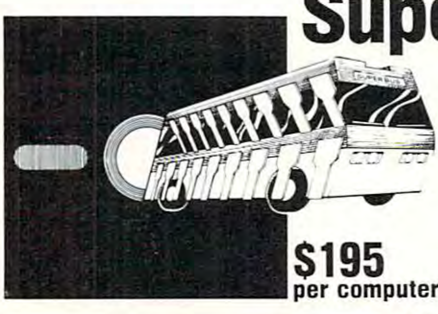
Farmer's Workbook™

The most powerful management tool ever for the agricultural producer. Farmers Workbook combines the power of Visicalc® * with the knowledge of a major midwestern university. The Farmers Workbook is a collection of templates that are designed to be run on the Visicalc® program. The templates include lables, formulas, sample data, test cases and full documentation. Template titles include: Cattle Feeder, Pig Production, Sheep Production, Grain Management, Loan Payments, Market Average, Land Purchase and many others.

*Visicalc is a trademark of Personal Software, Inc.

Also...Two Other Enhancements for PET/CBM Systems

SuperBus™ Greatly multiplies system capabilities



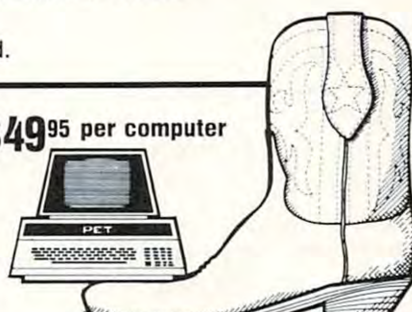
$195 per computer

This is an active, integrated system—not just a passive network. SuperBus gives schools, banks, laboratories and businesses control of the way their networks operate.

- Up to 18 computers, disk drives or printers can be interconnected.
- Complete file security (program and data) and BASIC security.
- Built-in error detection and convenience features.
- Can both read and write to disk.
- All BASIC commands can be used.

Auto-Boot™ Simplifies PET/CBM operation

$49.95 per computer



- Auto-Boot is a ROM that automatically loads and runs the first program on the disk (initializing if necessary).
- Completely compatible with most other programs.
- Just insert the disk in the drive, turn on the computer, and Auto-Boot does the rest.

MasterCard, VISA and C.O.D. orders accepted. Specify computer model when ordering SuperBus or Auto Boot. Dealer inquiries invited.

CYBERIA INC. 515-292-7634
2330 LINCOLN WAY, AMES, IOWA 50010

A Flexible Input Subroutine

Glenn M. Kleiman Research Triangle Park, NC

Many interactive programs require a variety of types of input from the user. For example, in my own programs written for classroom use by children, each of the following four types of input are often required:

1. Alphabetic strings, such as the user's name or answers to questions.
2. Numbers, such as the user's age or the answers to math problems.
3. Single digits or letters from a restricted set, such as when the user is asked to make a selection from a menu.
4. Y or N, in response to questions such as "Do you want to continue? (Y/N)".

A program designed for unsophisticated users must have checks that the user's input is appropriate. For example, the programmer must guard against the uncertain user who, when given a Yes/No question, presses M for "maybe". Often, particularly in programs to be used by children, one should also control the number of characters that can be input. A program should not accept a name consisting of 100 letters, nor should it accept 15 digits in answer to a math problem that calls for a 3 digit answer. Furthermore, the user should be able to erase mistakes, and inappropriate responses should not stop program execution.

I have written a general purpose input subroutine to handle all of the above. It is written for the PET, but most of the routine is compatible with other BASICs, so it can be easily revised for other microcomputers.

Within a program using this subroutine, the accepted inputs are specified by assigning values to variables before the subroutine is called. The main variable is UF, which can have any one of four values. If UF = 0 (the default value), any letters, but no other characters, will be accepted. If UF = 1 then only numbers will be accepted. For both letters and numbers, UM controls the maximum number of input characters. The default value for UM is set to 1 in line 300.

In order to restrict the accepted characters, as for menu selection responses, UF is set to 2, and the first and last characters to be accepted are assigned to variables F$ and L$, respectively. For example, the following line in a program will set the subroutine to accept only the letters M, N, O, and P:

UF = 2:F$ = "M":L$ = "P":GOSUB300

Finally, to accept only Y or N, UF is set to 3. If UF = 2 or 3, UM is set to 1 automatically.

In all cases, inappropriate input is ignored. Input characters can be erased by pressing the DEL key and a completed input is signaled by pressing RETURN. DEL and RETURN are not accepted until at least one character has been input. Once UM characters have been input, only DEL and RETURN will be accepted.

When RETURN is pressed, UF and UM are reset to their default values. Input strings are then available in the program as variable IN$, input numerics as IN.

A few other notes. I use a flashing ? as a cursor, but any character can be substituted in line 420. In line 430, UT = TI + 35 controls the rate of cursor flashing. The flashing rate of 35 jiffies is slower than most cursors, but seems to be less annoying to many people than the usual speed. The technique of flashing the cursor is based on the INP routine from CURSOR #4. This subroutine, and any other frequently used one, should be placed at the beginning of the program. The reason is that whenever a GOSUB (or a GOTO) occurs, the sequential search for the referenced line number begins at the first line of the program. An input subroutine placed at the end of a long program may be noticeably slow in accepting responses.

This subroutine, written to be easily readable rather than compact, uses 406 bytes (without the REMs).

```
100 REM     FLEXIBLE INPUT SUBROUTINE
101 REM
102 REM          GLENN M. KLEIMAN
103 REM           TEACHING TOOLS:
104 REM       MICROCOMPUTER SERVICES
105 REM           P.O. BOX 12679
106 REM       RESEARCH TRIANGLE PARK
107 REM             N.C. 27709
110 REM
120 REM     VARIABLES TO BE SET:
130 REM  UF=0 FOR ALPHABETIC INPUT
140 REM  UF=1 FOR NUMERIC INPUT
150 REM  UF=2 FOR RESTRICTED INPUT
160 REM  UF=3 FOR Y OR N INPUT
170 REM
180 REM  IF UF=0 OR 1 SPECIFY:
190 REM     UM = MAXIMUM NUMBER OF INPUT CHARACTERS
200 REM     (DEFAULT UM SET IN LINE 300)
210 REM  IF UF=2, SPECIFY:
220 REM     F$=FIRST CHARACTER ACCEPTED
230 REM     L$=LAST CHARACTER ACCEPTED
240 REM
250 REM     OUTPUT VARIABLES
260 REM  IN$ = INPUT STRING
270 REM  IN = VAL(IN$)
280 REM
300 IFUM=0THENUM=1
310 IFUF=0THENF$="A":L$="Z"
320 IFUF=1THENF$="0":L$="9"
330 IFUF>1THENUM=1
340 IN$="":UT=TI:UC=1
400 GETU$:IFU$<>""GOTO440
410 IFUT>TIGOTO400
420 PRINTMID$(" ?",UC,1);"←";
430 UC=3-UC:UT=TI+35:GOTO400
440 UL=LEN(IN$):IFUL=UMGOTO510
450 IFUF<>3GOTO480
460 IFU$="Y"ORU$="N"GOTO490
470 GOTO500
480 IFU$<F$ORU$>L$GOTO500
490 IN$=IN$+U$:PRINTU$;:GOTO400
500 IFUL=0GOTO400
510 IFU$=CHR$(20)THENIN$=LEFT$(IN$,UL-1):PRINT" ←← ←";
520 IFU$<>CHR$(13)GOTO400
530 PRINT" ":UF=0:UM=0:IN=VAL(IN$):RETURN
READY.
```

Universal Tape Append For PET/CBM

Roy Busdiecker
Woodbridge, VA

Many times we run into the situation where we'd like to combine programs which have been SAVEd separately. Typical examples include subroutines which can be used in many different programs; interest calculations for mortgage, loan, or savings programs; complex arithmetic for math or engineering programs; and sorting routines for data processing applications.

Owners of Commodore PET/CBM 2001-series computers have several alternatives. Several firms offer a plug-in ROM (read-only memory) in the $50 to $100 range, which adds an APPEND command to the normal instruction repertoire.

The program described here allows owners of Version 1 (BASIC 2.0), Version 2 (BASIC 3.0), or 8000 series (BASIC 4.0) PET/CBM computers automatically to combine two or more programs which were saved on cassette from either one of those two computers.

In use, the program is extremely simple. First, LOAD this APPEND program from its cassette. Next, take the APPEND cassette out of the recorder, and replace it with the cassette that has the first program you want to append to it. DO NOT LOAD the second cassette, but RUN the APPEND program that is already in the computer. You'll be instructed to "Press Play on Cassette #1", and once you do that, the program takes over. After the first program has been added, take out the cassette and insert the second one you want to add. When you RUN the APPEND program again, it will once more ask you to "Press Play...", then add the second program at the end of the first. After you've combined all the programs you want to join, delete the BASIC Append routine (type each line number, 0 through 29, pressing 'RETURN' after each), and use the BASIC SAVE command to store the combined version.

Preparing Programs to be Combined

A few rules must be observed with regard to the programs which are to be joined. In general, you must assure that there is no overlap in line numbers between the two (or more) programs. For example, if you have two programs where one contains lines numbered from 100 to 500 and the other contains lines 300 to 700, the computer's operating system will not react 'normally' if the two are appended. An easy 'fix' is to renumber one program or the other, so that none of its line numbers fall in the range of numbers used in the other program. An exception to this rule is the Append routine itself, because it will be deleted before you start using the combined programs.

When programs are appended, the one(s) with lower line numbers should be done first, to avoid problems.

Some programs, especially those prepared commercially, were SAVEd from the Monitor rather than BASIC, and contain machine language instructions ahead of the BASIC routines. These may not be combined using this program unless the BASIC and machine language sections are "broken apart" and stored separately. You may be able to figure out how to do this by careful study of this article and some experimentation...but be sure that you have backup copies of everything critical before you start! If the APPEND program detects one of these (relatively unusual) programs, it will give you an error message and stop without trying to do the APPEND.

As long as you have sufficient room left in the computers free memory, you may keep adding programs. If you try to add a program which requires more than the remaining free space, the program will print an error message and not attempt to APPEND.

How the Program Works

Actually there are two separate programs which work together to do the job. The first (Figure 1) is a machine-language routine, loaded in the second cassette buffer, which inspects the program in the BASIC text area and calculates where the BASIC program ends.

```
C*
      PC   IRQ  SR AC XR YR SP
.;   C6FB E62E 3A 9E 36 34 FA
.
.:   033A B8 08 A9 01 8D 55 03 69
.:   0342 01 8D 4F 03 A9 04 8D 50
.:   034A 03 8D 56 03 AD B9 08 F0
.:   0352 18 AA AD B8 08 8D 55 03
.:   035A 69 01 8D 4F 03 8A 8D 56
.:   0362 03 69 00 8D 50 03 4C 4E
.:   036A 03 AD 55 03 8D 3A 03 AD
.:   0372 56 03 8D 3B 03 60 FE B7
.
```

Figure 1a. Machine Language Program Listing (Monitor Version, for entering in computer).

The BASIC Append routine (Figure 2) uses the machine-language routine to find the end of the current program in memory. Then it uses one of the built-in ROM (read-only memory) routines to find the "header" on the cassette tape at the beginning of the SAVEd program. That header contains the start-

```
.,   033C A9 01      LDA #$01
.,   033E 8D 55 03   STA $0355
.,   0341 69 01      ADC #$01
.,   0343 8D 4F 03   STA $034F
.,   0346 A9 04      LDA #$04
.,   0348 8D 50 03   STA $0350
.,   034B 8D 56 03   STA $0356
.,   034E AD B9 08   LDA $08B9
.,   0351 F0 18      BEQ $036B
.,   0353 AA         TAX
.,   0354 AD B8 08   LDA $08B8
.,   0357 8D 55 03   STA $0355
.,   035A 69 01      ADC #$01
.,   035C 8D 4F 03   STA $034F
.,   035F 8A         TXA
.,   0360 8D 56 03   STA $0356
.,   0363 69 00      ADC #$00
.,   0365 8D 50 03   STA $0350
.,   0368 4C 4E 03   JMP $034E
.,   036B AD 55 03   LDA $0355
.,   036E 8D 3A 03   STA $033A
.,   0371 AD 56 03   LDA $0356
.,   0374 8D 3B 03   STA $033B
.,   0377 60         RTS
```

Figure 1b. Machine Language Program (Disassembly Listing)

ing and ending addresses from which its program was saved, and knowing those values allows us to calculate the length of the program on tape.

Armed with knowledge of the end of the current program in memory, and the length of the program to be appended, we can calculate new starting and ending locations for loading the program from tape. By changing those values before we bring the program in from the computer, we can start loading the new program right where the old one ends.

Complications

Back when there was only one operating system (set of ROMs) for the PET, the APPEND routine was much simpler. The second version (BASIC 3.0) made several changes which increased the challenge in designing an APPEND program to run on either version and APPEND a tape which had been created on either version. Appearance of BASIC 4.0 in the 8000 series complicated matters further. There are now nine possible combinations, as depicted in Figure 3.

The first problem, and most obvious, is that the various "built-in" routines used by the program are in different locations in the two versions. Furthermore, some "fixing-up" which is done automatically in BASIC 2.0 by the tape load routine requires calls to other routines in BASIC 3.0 and 4.0. A summary of these differences is shown in Figure 4.

```
0 REM-SUPER APPEND-FOR PET/CBM, COPYRIGHT OCT 79 BY ROY BUSDIECKER
1 P=256:SYS828:PRINTPEEK(826)+P*PEEK(827):PV=PEEK(50003):PRINT"■■■■■■■"
2 GOSUB12:A1=PEEK(826)+P*PEEK(827)-1:A2%=A1/256:A3=A1-P*A2%:IFPV=160THEN16
3 ONPV+1GOTO14,15
4 P=256:B1=PEEK(635)+P*PEEK(636):B2=PEEK(637)+P*PEEK(638):IFPEEK(636)<>4THEN27
5 IFPEEK(635)=0THEN8
6 IFPEEK(635)=1THENA1=A1+1:A2%=A1/256:A3=A1-P*A2%:GOTO8
7 GOTO27
8 B3=B2-B1+A1:C1%=B3/256:C2=B3-P*C1%:POKE635,A3:POKE636,A2%:POKE637,C2
9 POKE638,C1%:IFC1%>PEEK(53)ORC1%=PEEK(53)ANDC2>=PEEK(52)THEN29
10 IFPV=160THEN23
11 ONPV+1GOTO17,18
12 IFPV=160THENPOKE158,9:BU=623:FORI=BUTOBU+8:POKEI,13:NEXT:RETURN
13 POKE525-PV*367,9:BU=527+PV*96:FORI=BUTOBU+8:POKEI,13:NEXT:RETURN
14 PRINT"SYS62894■■■■■■■":PRINT"GOTO4IIIIIIIIIIIIIII]":STOP
15 PRINT"SYS62886■■■■■■■":PRINT"GOTO4IIIIIIIIIIIIIII]":STOP
16 PRINT"SYS62949■■■■■■■":PRINT"GOTO4IIIIIIIIIIIIIII]":STOP
17 GOSUB12:PRINT"■■■■■SYS62403":PRINT"IIIIIIIIII]":STOP
18 GOSUB12:PRINT"■■■■■SYS62393■■■":PRINT"GOTO19IIIIIIIIIIIII]":STOP
19 SYS50242
20 SYS828
21 POKE42,PEEK(826)+2:POKE44,PEEK(826)+2:POKE46,PEEK(826)+2
22 POKE43,PEEK(827):POKE45,PEEK(827):POKE47,PEEK(827):END
23 GOSUB12:PRINT"■■■■■SYS62456■■■":PRINT"GOTO24IIIIIIIIIIIII]":STOP
24 SYS46262
25 SYS828
26 GOTO21
27 PRINT"■■ERROR.■  TAPE TO BE APPENDED IS NOT A    SIMPLE BASIC PROGRAM."
28 PRINT"MACHINE-LANGUAGE SEGMENTS MUST BE SAVED SEPARATELY.":END
29 PRINT"■■ERROR.■  NOT ENOUGH MEMORY SPACE LEFT TO APPEND THIS PROGRAM.":END
```

Figure 2. BASIC Program Listing

APPEND Program running on	Tape SAVEd by
Version 1 PET (BASIC 2.0)	Version 1 PET Version 2 PET/CBM 8000 Series CBM
Version 2 PET/CBM (BASIC 3.0)	Version 1 PET Version 2 PET/CBM 8000 Series CBM
8000 Series CBM (BASIC 4.0)	Version 1 PET Version 2 PET/CBM 8000 Series CBM

Figure 3

BASIC 2.0	BASIC 3.0	BASIC 4.0	Routine
SYS 62894	SYS 62886	SYS 62949	Find program header
SYS 62403	SYS 62393	SYS 62456	Load program
*	SYS 50242	SYS 46262	Fix chaining of program link pointers

*done automatically by BASIC 2.0 "load program" routine

Figure 4. Differences in Built-in Routines

One other difference is that on BASIC 3.0 & 4.0 machines, it is necessary to reset the pointers for variable storage to the new end-of-program. This fix, too, was done by the "Load program" routine on BASIC 2.0 machines.

Alignment

More subtle is the problem of properly aligning the appended program to the one already in the computer. Version 1 PETs start the SAVE process at location 1024 (0400 in hexadecimal notation), which fortunately always contains a zero. BASIC 3.0 & 4.0 PET/CBM computers, on the other hand, start SAVEing at location 1025 (hexadecimal 0401).

At the end of each line of BASIC program text, there is one byte which contains a value of zero to mark the place (not the same as the ASCII character zero, which is stored as a value of 48). Following each of these line-end markers, except the last one, are two bytes containing a line pointer, then two bytes containing the value of the program line number. The last line-end marker is followed by two zero-value bytes, so this series of three zero-bytes may be thought of as an end-of-program marker. Figure 5 illustrates this scheme.

1024 1025

0	text	0	text	0	text	0	0	0

end of program

Figure 5

Focusing in on the end-of-program area, Figure 6 shows how each of the two types of SAVEd program must be lined up with the program in the machine, if a successful APPEND is to occur. Notice that the leading byte of the BASIC 2.0 tape (which is always zero since it originated in byte 1024) can be overlaid on the last end-of-line marker, since both values are zero. All we have to do, then, is detect whether the program was SAVEd on a BASIC 2.0 or BASIC 3.0/4.0 machine, and adjust the location for starting the LOAD, if necessary.

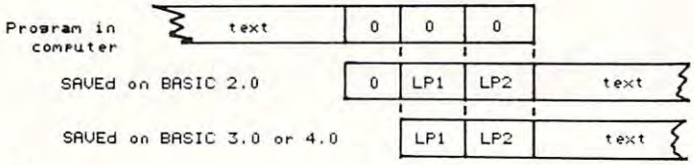


Figure 6

Determining Source

When the tape header is found, the starting and ending locations of the SAVE operation which created the tape are retrieved and stored in decimal locations 635 through 638 (hexadecimal 027B-027E). The starting location is in the first two bytes (low order byte first, followed by high), and the ending address is the last two. If the SAVE was done by a BASIC 2.0 computer, then the starting address in location 1024 (0400 in hexadecimal notation), which 636 would contain values of 1 and 4 respectively, after the header was found.

Byte 635, then, holds the key. A value of zero indicates a BASIC 2.0 SAVE, while a value of one indicates BASIC 3.0 or 4.0. The test for this value is in lines 4-6 of the program, Figure 2. If byte 635 contains neither 0 nor 1, or if byte 636 contains other than 4, then the routine on the tape is not a 'normal' BASIC program, and special steps must be taken before it can be appended.

Entering the Program

While the BASIC portion of the program may be typed in quite simply, the easiest way to enter the machine language segment is to use the monitor, so it is a good idea to do that part first.

If you are using a BASIC 2.0 PET, it will be necessary to load the monitor from tape, then tell it to RUN. On the newer 2001 series computers, simply enter SYS 64785. On the 8000 series, enter SYS 54386. Either machine will then give a display of register contents, similar to that at the top of Figure 1, a dot at the beginning of the current line, and the cursor just after the dot.

Simply type in the locations and contents as they are shown in Figure 1, ending each line with a carriage return. When you are finished, type

M 033A 0377

and press 'RETURN', and your entries will be displayed so you can check them. Should any mistakes be found, simply move the cursor to the appropriate location, type in the correct value, and press 'RETURN' to correct them.

When you are satisfied that the program has been entered correctly, enter X, and the monitor will transfer control back to the BASIC operating system.

Before you start typing in the BASIC part of the program, if you are using a BASIC 2.0 PET, be sure to type NEW to clear out the monitor.

Saving the Program

When both programs have been entered, be sure to SAVE a copy (or two) for security, to avoid the embarrassment of discovering a machine language error by losing control of the computer.

In BASIC 3.0 enter SYS 64785, or in BASIC 4.0 use SYS 54386, to return to the monitor. Then enter

S "APPEND", 01, 033A, 08BA

You will get the standard "Press Play and Record" messages. When you're finished, enter X to return to BASIC.

On Version 1 machines, enter

POKE 247,58:POKE 248,3:POKE 229, 186:POKE 230,8

then enter SYS 63153. A second copy may be saved by simply entering SYS 63153 again.

Testing

To see if the program works correctly, first LOAD one of the copies you have just made (if you've done any SAVE or VERIFY, you'll always have to do another LOAD to make the program work correctly.

Then remove the APPEND tape from the cassette unit, insert a tape on which another program

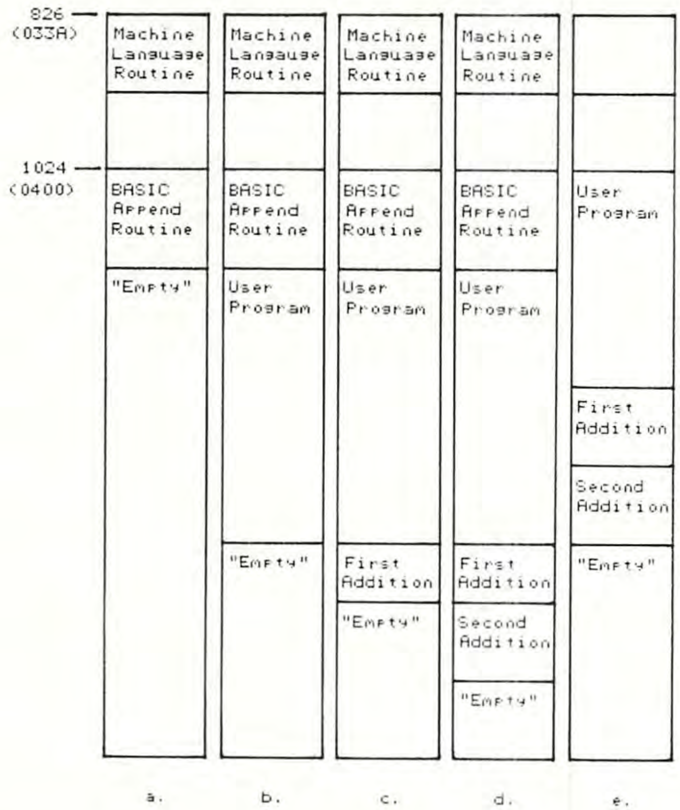


Figure 7. Allocation of Memory during Append Operations

a. At beginning of process, append program loaded
b. After User Program has been appended
c. After first addition to User Program
d. After second addition to User Program
e. End of process, after append routine is deleted

A BRILLIANT FUTURE FOR YOUR AIM-65 WITH THE BANKER MEMORY ™

Your 36K of free address space is the AIM's most valuable and limited resource. With today's large capacity RAM boards, ROM boards, disk systems, video boards, and other expansion accessories it is easy to deplete this resource before the application requirement is satisfied. MTU has solved this problem.

THE BANKER MEMORY contains 32K of RAM, 4 PROM sockets for 2716/2732/2332, a PROM programmer, 40 bits of parallel I/O, and 4 timers from two 6522 I/O chips. Addressing is extremely flexible with the RAM independently addressable in 4K blocks, PROM's independently addressable, and I/O addressable anywhere on a 64 byte boundary (even in AIM's I/O area at AXXX by adding a single jumper to the AIM).

This may sound familiar, but read on! Unlike other AIM compatible memory boards, THE BANKER MEMORY has on-board bank-switching logic! The four 8K blocks of RAM plus the 4 PROM sockets make up 8 **resources**, each associated with a bit in an Enable Register. Through this Enable Register resources may be turned on and off under software control. When a resource is off, its address space is freed for other uses. You can even put BANKER resources at the same address and switch among them for virtually unlimited RAM and PROM expansion! You can even have multiple page zero's and stacks! Do you need 160K byte of memory? It only takes 5 of THE BANKER MEMORY boards and you end up with 5 page zeros and stacks to boot!

There's more! The BANKER MEMORY also incorporates 18 bit addressing which allows for the 256K address spaces of the future. RAM, PROM, and I/O each has its own full 18 bit address decoder which allows these resources to be in different 64K banks. This board and other MTU products, such as our 320 by 200 dot VISIBLE MEMORY and Floppy Disk Controller with 16K DMA RAM, can turn your AIM into a truly powerful 6502 computer that far surpasses the packaged systems in functional performance.

INTRODUCTORY SPECIAL K-1032-1 32K BANKER MEMORY FULLY ASSEMBLED AND TESTED $395.00 ($450.00 as of March 1, 1980) or the K-1032-2 16K RAM only with bank switching and 18 bit address bus only $295.00

Isn't it time you took a closer look at MTU — we offer you power now with an eye to the future.

WRITE OR CALL TODAY FOR OUR 48 PAGE FALL 1980 6502 CATALOG

International requests include $1.00

VISA and MASTERCARD accepted

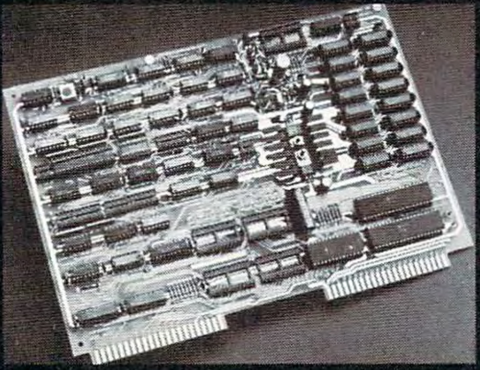


Photo credit:
SUPERNOVA CRAB NEBULA:
Palomar Observatory,
California Institute of Technology

MTU
Micro Technology Unlimited
2806 Hillsborough Street
P.O. Box 12106
Raleigh, NC 27605, U.S.A.
(919) 833-1458

4. Receiver acknowledges receipt of data by raising its **BUSY** flag.
5. Sender removes (raises) **DATA READY.**
6. Receiver lowers **BUSY** when ready for new data.

This sequence can be seen to be equivalent to that suggested by Eric, except that his **DATA READY** is a high-true signal. The choice of signal polarity given here is consistent with the operation of programmable port chips. I/O pins on such chips come up from reset as inputs and are high. Thus the **DATA READY** naturally comes up in the false state. For the receiver, the **BUSY** comes up naturally high, so that no sequence can be started until the receiver program is started and its **BUSY** line is cleared. It will not matter, therefore, whether the receive program or the send program is started first.

Just as the data direction registers come up zeroes from reset defining inputs, the data registers come up zeroes as well. Therefore, it is a good idea to write output data to port bits *BEFORE* configuring the bits as outputs. If the port bits are made outputs first, they will immediately fall to zero, since the reset line zeroes the data register. Even if the program immediately writes ones to the outputs, all output lines will experience a momentary glitch to ground (for the duration of an instruction) until the new data is written. It is important to understand that data can always be written to ports as outputs, even if they are programmed as inputs. Making a bit an input bit merely disconnects the flip-flop from the I/O pin. Even though you will not be able to read the data that you have written to an output bit, it is still in the flip-flop. A representation of a programmable port I/O bit (PAO) and the corresponding data direction register bit is shown in Figure 3, and is worth discussing.

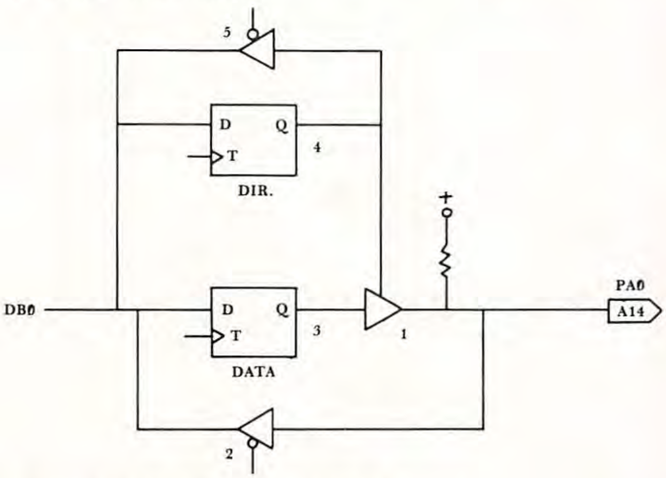


Figure 3. Programmable Port Structure

The flip-flop (item 3) has its "D" input connected to the data bus bit DBO. It can serve as an output bit if and only if it is connected to the port pin PAO via the three-state gate (item 1). With this gate enabled, anything written to the zero bit of a port A will appear on the I/O pin. If the gate is disabled, however, we are now free to use the I/O pin as an input. Note, however, that programming the bit as an input *DOES NOT* PREVENT US FROM WRITING TO THE **DATA** flip-flop. While we will not be able to read the data back, the data is still in the flip-flop, and it will appear on the I/O pin if this bit is subsequently made an output. When the port is read it is the condition of the I/O pin that is being read, regardless of whether the bit is programmed as an input or output. (This is not true of B ports, where the data read back when programmed is the latched data. That is, a bit can be programmed as an output and a one and the I/O pin shorted to ground and have it read back as a one). The three-state gate (item 1) is controlled by a second flip-flop (item 4) called a data direction flip-flop. This condition of this flip-flop may be read via its three-state gate (item 5). (Note that what we have called a three-state gate is in actuality implemented with MOS open-drain technology).

The purpose of this discussion was to convince the reader, that it is possible to successfully write output data to a port while it is still programmed as input. Not only is it possible, but it is recommended as good port software technique, to avoid unnecessary output "glitches".

Getting back to handshaking, it is now necessary to look at handshaking software. We would like to consider both the transmit and receive programs. Figure 4 shows a flowchart for a transmit program. First the ports must be set up. Then before transmitting, we must be certain that the receiver is not **BUSY,** and wait until **BUSY** is false. Then data is loaded and sent to the port. Next the **DATA READY** flag is lowered. The program now waits for the handshake response from the receiver, that is, for **BUSY** to become true. As soon as that has been verified, the **DATA READY** flag is cleared and the memory pointer is incremented and compared with the end pointer. If the end has not been reached, the process is repeated for another byte. Otherwise the program returns to the monitor.

A flowchart for a receive program is shown in Figure 5. After initialization, a wait is made for a **DATA READY** indication, then the data is tucked away, the **BUSY** flag raised, and the pointer incremented. At this point, the pointer may be compared with an endpoint for a completion test. If done, the **BUSY** flag is lowered and a return made to the monitor. If not, the **BUSY** flag is lowered and another data byte is fetched.

Both transmit and receive programs can vary, depending upon whether speed or code conciseness is the most desirable feature. This discussion will be continued in the next column with an analysis of typical transmit and receive programs.

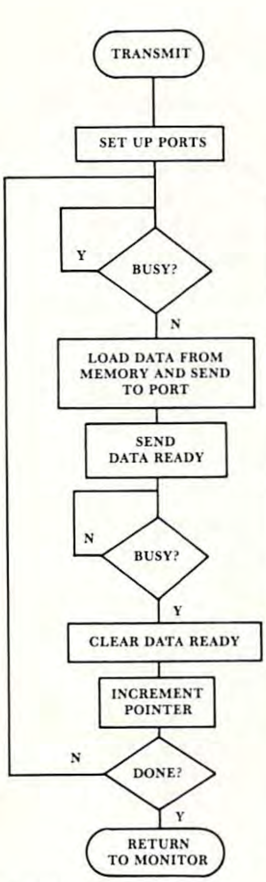


Figure 4. Transmit Handshake Flowchart

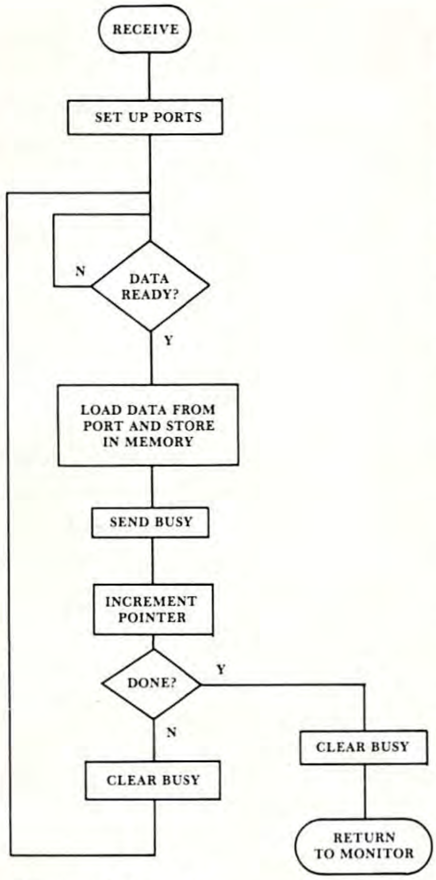


Figure 5. Receive Handshake Flowchart

©

Experimenting With The 6551 ACIA

Marvin L. De Jong
Department of Mathematics-Physics
The School of the Ozarks
Pt. Lookout, MO

There is a growing interest in data communications, computer networks, time-sharing services and other forms of intercomputer communications. An important element in many data communications systems is an Asynchronous Communications Interface Adapter (ACIA). Both Rockwell International and Synertek manufacture a 6502 family device known as the 6551 ACIA. The purpose of this article is to provide information about interfacing this device to 6502-based microcomputers and to provide information about operating and controlling the 6551 with software.

Since I was not familiar with this chip, I decided to do some simple experiments with it to supplement the meager (in my opinion) information supplied by the specification sheets from Rockwell and Synertek. In particular, I decided to use my AIM 65 microcomputer as a smart terminal for a KIM-1, operating the latter in its "teletypewriter" mode. Although this may seem ridiculous, the idea might be useful in a laboratory where various student stations have a KIM-1 that is used for experiment control or data acquisition. If all the KIM-1s were connected to a central terminal, one could load a program into all of them simultaneously or, with the appropriate switching circuitry, one could collect and process data from each of them. In any case, the experiment taught me what a KIM-1 is like when operated in its TTY mode rather than from its keypad, and more importantly, I learned some things about the 6551 ACIA.

A circuit that can be used to interface the 6551 to a 6502-based microcomputer is shown in Figure 1. The connections to the microcomputer are on the left-hand side of the figure. The advantage of using a family-type chip is the ease with which the device may be interfaced to the microcomputer. Thus, the connections IRQ, RES, Ø2, R/W, and the data bus connections are all straightforward. If the lines between the microcomputer and the 6551 are kept short, a few inches or so, no buffering is required. This circuit was built on a protoboard attached to the expansion connector of an AIM 65. The four registers on the 6551 are selected with address lines AØ and A1 connected to the register select pins RSØ and RS1, respectively.

The chip select (or device select) signals, CSØ and $\overline{\text{CS1}}$, can be obtained in several ways depending on your microcomputer system. The AIM 65 provides a $\overline{\text{DS9}}$ device select pulse that is active at logic zero for all addresses $9000 through $9FFF. This signal is available at the expansion connector, and it was used in this circuit. The CSØ pin could have been connected to +5V, but we chose to connect it to address line A1Ø to save half of the address space between $9000 and $9FFF for other possible I/O functions.

If you have a SYM-1 you can use device select $\overline{\text{18}}$, which is available on the expansion connector, for $\overline{\text{CS1}}$. Pin CSØ can either be tied to logic one (+5V) or connected to an address line to divide the device select $\overline{\text{18}}$ address space in half.

If you have a KIM-1 you can use one of the device selects K1 to K4 with suitable pull-up resistors, say 1000 ohms. Tie CSØ to +5V.

If you have an APPLE II you can build the interface shown in Figure 1 on a peripheral card and plug it into one of the eight card slots. However, you

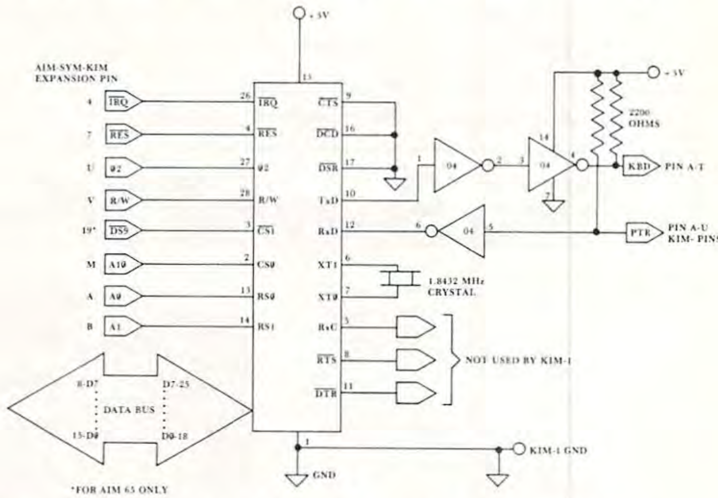


Figure 1. The circuit to interface the 6551 ACIA to a 6502-based microcomputer to control a KIM-1.

must build your own decoding circuit because you cannot use either the device select ($\overline{\text{DS}}$) or the I/O select signals. 6502 family chips such as the 6522 and the 6551 generally require that the address lines and the chip selects are stable some time (approximately 200 nanoseconds) before the rising edge of the Ø2 clock signal. The device select signals generated by the APPLE II address decoding circuitry have been ANDED with Ø2 (actually ØØ), and consequently they cannot be used. This is really unfortunate since it would be very easy to interface 6502 family chips to the APPLE's peripheral bus if it were not for this fact. One could use a 74LS04 inverter and a 74LS30 eight input NAND gate to generate a device select for some page of memory not used by the APPLE II, if you want to interface a 6551 to your APPLE II.

Proceeding to the circuitry on the right-hand side of Figure 1, you will note that you need a crystal whose frequency is 1.8432 MHz. The remaining connections are either input or output pins that connect the 6551 to devices outside of the microcomputer system, such as a modem or, in the experiment described here, another microcomputer. In this application only the data output (TxD) and data input (RxD) pins were used. A 74LS04 was used to provide suitable buffering between the 6551 and the KIM-1 printer and keyboard pins for the teletypewriter. The input pins $\overline{CTS}$, $\overline{DCD}$, and $\overline{DSR}$ are tied to logic zero while the output pins $\overline{RTS}$ and $\overline{DTR}$ are left floating in this application. If the 6551 were connected to a modem, printer, or another terminal, then these pins would be used. The similarity between the names of these pins and RS-232C pin functions is not a coincidence. My modem requires the $\overline{CTS}$, $\overline{DSR}$, and $\overline{DCD}$ connections. A 1488 RS-232C line driver and a 1489 RS-232C line receiver could be used to change the voltages to the appropriate levels for a standard RS-232C interface, but we chose to experiment with a KIM-1, and did not need RS-232C signals.

If A0 and A1 are used as register select signals, as indicated in Figure 1, then the low-order nibble of the address that accesses the 6551 will be $0, $1, $2, and $3 for the data registers, status register, command register, and control register, respectively. For example, if the address decoding scheme shown in Figure 1 is used, the transmitter data register is accessed by WRITING to $9400 and the receiver data register is accessed by READING location $9400. Writing to the status register at $9401 causes a programmed RESET, and the status register is read at $9401. Refer to Figure 2 to identify the meaning of the bits in the status register.

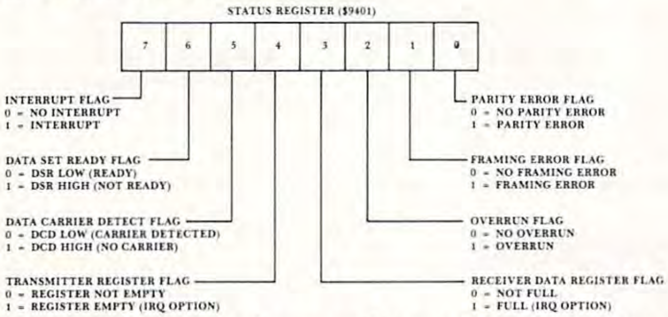


Figure 2. Schematic diagram of the status register of the 6551.

For the simple experiments described here the status register bits of most interest are the **transmitter register flag** and the **receiver register flag.** Writing to the transmitter register clears the register flag, and one should not write data to this register again until the data in the transmitter register has been transformed into a serial bit stream and has been transmitted by way of the TxD pin. When the word has been transmitted, bit four goes to logic one, and the transmitter data register is ready to accept another word. Likewise, when a complete word has been received by way of the RxD pin and the word is in the receiver data register, then bit three of the status register goes to logic one, and the word is ready to be read from the 6551. Both of the events just described may be used to produce interrupt requests ($\overline{IRQ}$ pin goes to logic zero). That is, by programming the command register, one can produce an interrupt request either when the transmitter register is empty or when the receiver register is full.

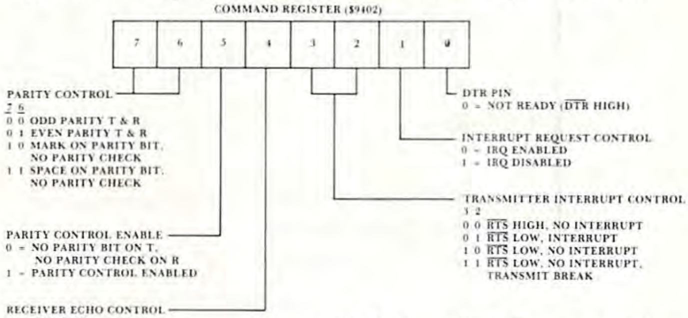


Figure 3. Schematic diagram of the command register of the 6551.

Refer to Figure 3 to identify the meaning of the various bits in the command register, and refer to Figure 4 to identify the meaning of the bits in the control register. The function of these registers will become apparent when we describe the program to use the AIM 65 as a terminal to control the KIM-1. In short, they allow the user to program the 6551 to operate under a large variety of conditions.

The program to test the 6551 with the AIM 65 and KIM-1 is given in Listing 1. The main program reads the keyboard and outputs this character to the 6551 transmit data register. The interrupt routine (remember to load the interrupt vectors if you use this program) reads the 6551 receive data register when the KIM-1 returns information to the AIM 65. In short, the entire program makes the AIM 65 behave exactly like a teletypewriter terminal as far as the KIM-1 is concerned. Since the 6551 is being operated in the interrupt mode, the first instruction in Listing 1 clears the 6502 interrupt flag to allow the 6551 to interrupt it. The next two instructions in the program load the command register. Refer to Figure 3 to see what bits were set or cleared. Since the KIM-1 software in the monitor simply strips the parity bit from any received word, the command register was initialized to disable and disregard any parity bits. Since the 6551 is being operated in an interrupt mode, bit one of the command register is cleared. However, it is the receiver portion of the 6551 that is being allowed to cause an interrupt, thus bit three of the command register is kept at logic one to prevent interrupts from the transmitter. The other bits of the command register control the handshaking signal pins of the 6551, and therefore they were not of any concern in this application.

The fourth and fifth instructions in Listing 1 load the control register. Refer to Figure 4. A "three" in the low-order nibble of the control

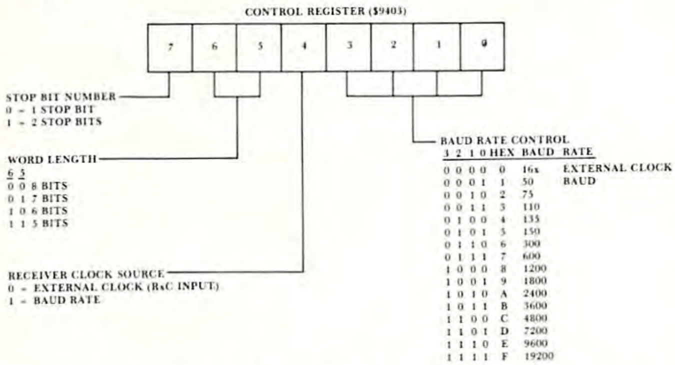


Figure 4. Schematic diagram of the control register of the 6551.

register sets the Baud rate at 110. Higher Baud rates are possible, depending in part on whether the thermal printer on the AIM 65 is used. Without the printer operating, rates as high as 2400 Baud are possible. The usual teletypewriter data format is one start bit, seven data bits, one parity bit, and two stop bits. However, a number of formats will work, and we chose an eight bit word with one stop bit. Note however that the command register was set up for no parity bit, thus our word looks just like a word in the teletypewriter format. If we would have loaded the control register with $BA sending seven bits of word and two stop bits at 2400 Baud, the program would still work. In order to interface the 6551 to any device you must program the command and control registers to match the *protocol* of the devices that are communicating.

Still keeping an eye on Listing 1, note that the next instruction is a subroutine jump to an AIM 65 subroutine that reads the keyboard and returns the ASCII code for the key depressed in the 6502 accumulator. This character is loaded into the 6551 transmit data register, and is promptly sent out on the TxD pin in serial form. Before getting another character, the program waits in a loop until the word is sent. It does this by examining bit four in the status register (refer to Figure 2). When the character has been sent and the transmit data register is empty, then the program loops back to get another character when the keyboard is scanned.

Refer next to the interrupt routine. A PHA instruction saves the accumulator. Next, the receive data register is read. The only time an interrupt occurs is when a new word is received from the KIM-1, and the second instruction of the interrupt routine gets the character in the accumulator of the AIM 65's 6502. Next, it outputs the character to the AIM 65 display. The fourth instruction clears the interrupt signal from the 6551. The accumulator is recalled, and the interrupt routine is concluded.

A future project includes interfacing the 6551 to a Novation Cat modem and trying to send information over telephone lines. Anyone out there care to join this experiment?

Listing 1. Routines to Control the KIM-1 with an AIM 65.

```
CTRLRG  = $9403; Control register of the 6551.
CMNDRG  = $9402; Command register of the 6551.
STATUS  = $9401; Status register of the 6551.
RDWR    = $9400; Read/Write Data register of the 6551.
$0F00 58        START    CLI            Clear interrupt flag.
 0F01 A9 09              LDA #$09       Set up command register.
 0F03 8D 02 94           STA CMNDRG
 0F06 A9 13              LDA #$13       Set up control register.
 0F08 8D 03 94           STA CTRLRG     Baud rate is 110.
 0F0B 20 3C E9  REPEAT   JSR GETKEY     Get input character from the
 0F0E 8D 00 94           STA RDWR       AIM 65 keyboard, output it to
                                        6551.
 0F11 AD 01 94  CHECK    LDA STATUS     Is transmit register empty?
 0F14 29 10              AND #$10
 0F16 F0 F9              BEQ CHECK      No. Then wait here.
 0F18 D0 F1              BNE REPEAT     Yes. Then get another character.

Interrupt Routine
$0E00 48        IRQ      PHA            Save accumulator.
 0E01 AD 00 94           LDA RDWR       Read the receive register and
 0E04 20 7A E9           JSR OUTCHAR    output the result.
 0E07 AD 01 94           LDA STATUS     Read the status register to clear
 0E0A 68                 PLA            the interrupt flag.
 0E0B 40                 RTI            Return to the main program.
INTERRUPT VECTORS: [$A404] = $00; [$A405] = $0E
```

©

A Vocal Hex Dump For The KIM-1

William C. Clements, Jr.
Dept. Of Chemical And Metallurgical Engineering
The University Of Alabama

This article describes a program for the KIM-1 that begins at a given RAM address and pronounces the contents of successive locations, with appropriate pauses inserted for naturalness, just as a person would read off a list of hex words. It uses what is almost certainly the least expensive speech synthesis equipment and software now on the market; for about $100, the single-board computer owner can experiment with computer-generated speech. The program given here is concerned with removing a little of the drudgery from proofreading programs in RAM. The program runs on a KIM-1 to which has been added a 6522 VIA and at least 1K of expansion RAM.

Personal computers surely are the ultimate in modern versatility, making everything from dungeons-and-dragons to home automation to self-instruction in computer science available to nearly everyone. But no matter how much fun it is to use the polished end result of one's programming, the checking of machine code to see if it was entered correctly remains pure drudgery, and the cleverest technology isn't likely to ever place it on a level with playing a rousing game of motorcycle racing with the computer. For those of us with video terminals but no printer, it can be especially irritating; one's eyes move up to the screen dump, down to the written program, up and down, kind of like watching a vertical tennis game, until the eyes have had it.

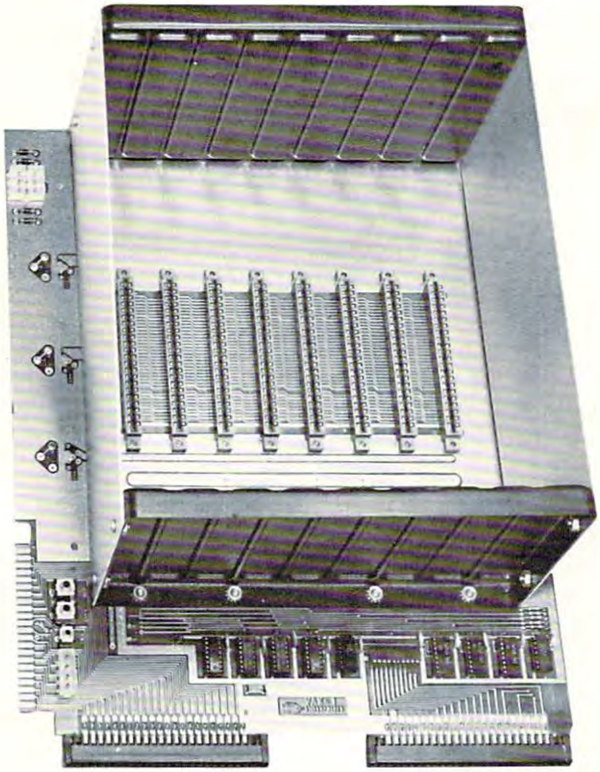
It would help to have someone read off the code from the screen so the programmer can keep his eyes on the paper. But another person isn't always available, and anyhow this is just the kind of work that computers were invented to handle, right? The only trouble is, most speech synthesizers are expensive, and are usually for the S-100 bus, not directly usable with the KIM or similar single-board machine. Now, thanks to Texas Instruments, Inc. and Dave Kemp of East Coast Micro Products, these objections have been neatly removed. The T-I Speak and Spell™, an inexpensive ($50 range) pre-programmed speech synthesizer computer was developed to teach kids to spell.[1] Its internal ROMs contain the coding to vocalize hundreds of words plus several phrases, the letters of the alphabet, and the numerals. But it's more than a toy. The device has an internal edge connector intended for plugging in additional vocabulary ROMs, and the various control lines that operate the speech synthesizer are available there. East Coast Micro Products market a small interface kit, model SP-1[1], that plugs onto the edge connector, and performs the level shifting and parallel-to-serial conversion needed for interfacing the synthesizer to a computer. The whole thing fits into the battery compartment of the Speak and Spell™, making a very neat package. Along with the interface board, you get extensive support software, a detailed explanation of how the synthesizer works, and five demo programs written for the SYM computer. The software includes a program for pronouncing individual hex characters whose ASCII representation has been placed in the accumulator, and uses the 6522 VIA that the SYM uses for I/0.

As mentioned at the start of this article, my immediate goal in purchasing the SP-1 was to use it with a KIM to read out memory words. The listing gives the resulting program. The user begins execution at BEGIN, types the first RAM address on the TTY, and the program reads 256 locations out on the Speak and Spell™. If you're checking fewer locations, just hit the reset key when you're through. If your program is longer, type in the next location and it will read you 256 more.

The comments in the listing should be self explanatory for the most part. Label references not defined in the listing (such as FPNT, OUTSPE, etc.) are mostly labels in the SP-1 software. GETBYT is a routine in the KIM monitor.

The SP-1 software is set up to use the 6522 Versatile Interface Adaptor on the SYM board, so unless you want to re-program extensively, your best bet is to add a 6522 to your KIM; you ought to have one anyhow if you're a serious KIMmer. Mine was already present on a VIDEO PLUS™ board that I use with my system[1]. If you don't have a VIA in your system, I suggest you refer to the articles listed in the footnotes[3,4]. It should not be hard to provide one. The SP-1 software resides entirely in the KIM on-board memory with one exception: the speech data dictionary provided with the software requires 770 bytes of continuous memory in addition to the 478 bytes required by the SP-1 support software and by the vocal dump routine. None of the code is self-modifying, so you can relocate it at will, even into EPROM where it will become a valuable utility. The only memory that has to be RAM is the twelve-byte frame buffer which I located between $17A0 and $17AB. If you do relocate, take care to adjust the entries in TABLE. These are address pointers to entries in the speech dictionary supplied with the SP-1. The accompanying program listing assumes that the dictionary resides between $2000 and $2302 in expansion memory.

VAK-1 MOTHERBOARD



We also carry:

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

We also do custom hardware and software for the 6502 microprocessor

PRICE: **$139.00**

Call or write for shipping charges and our complete catalog.

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

SPECIFICATIONS:

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

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

RNB ENTERPRISES INCORPORATED

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

master charge VISA

```
                 0010          .BA $100
                 0020 ADLO     .DE $A
                 0030 ADHI     .DE $B
                 0035 SPNT     .DE 2
                 0040 GETBYT   .DE $1F9D
                 0050 FPNT     .DE 4
                 0060 SPINIT   .DE $200
                 0070 OUTSPE   .DE $2D0
0100- 20 9D 1F   0080 BEGIN    JSR GETBYT    ;GET START
0103- 85 0B      0090          STA *ADHI     ;ADDRESS
0105- 20 9D 1F   0100          JSR GETBYT    ;FROM
0108- 85 0A      0110          STA *ADLO     ;KEYBOARD
010A- A9 A0      0120          LDA #$A0      ;SET FRAME
010C- 85 04      0130          STA *FPNT     ;POINTER
010E- A9 17      0140          LDA #$17      ;TO
0110- 85 05      0150          STA *FPNT+1   ;$17A0 (12 LOCS NEEDED)
0112- 20 00 02   0160          JSR SPINIT    ;SET UP TIMERS
0115- 20 2D 01   0170          JSR PAUSE     ;PAUSE BEFORE SPEAKING
0118- A0 FF      0180          LDY #$FF      ;USE Y TO COUNT LOCS.
011A- C8         0190 LOOP     INY           ;DUMPED
011B- 98         0200          TYA
011C- 48         0210          PHA
011D- B1 0A      0220          LDA (ADLO),Y ;GET CURRENT CONTENTS FOR DUMP
011F- 20 31 01   0230          JSR SAY       ;PRONOUNCE CONTENTS;
0122- 20 2D 01   0240          JSR PAUSE     ;THEN PAUSE
0125- 68         0250          PLA
0126- A8         0260          TAY
0127- C9 FF      0270          CMP #$FF
0129- D0 EF      0280          BNE LOOP      ;LOOP 256 TIMES
012B- F0 D3      0290          BEQ BEGIN     ;GET NEW START
012D- A2 20      0300 PAUSE    LDX #$20      ;SET POINTER FOR PAUSE
012F- D0 0E      0310          BNE SP1       ;BRANCH TO SPEECH PAUSE
0131- 48         0320 SAY      PHA           ;SAVE CONTENTS
0132- 29 F0      0330          AND #$F0      ;GET HIGH-ORDER NYBBLE
0134- 4A         0340          LSR A         ;FORM INDEX
0135- 4A         0350          LSR A         ;INTO ADDRESS TABLE
0136- 4A         0360          LSR A
0137- 20 3E 01   0370          JSR SPEAK     ;SPEAK FIRST CHARACTER
013A- 68         0380          PLA
013B- 29 0F      0390          AND #$F       ;GET LOW-ORDER NYBBLE
013D- 0A         0400          ASL A         ;FORM INDEX
013E- AA         0410 SPEAK    TAX           ;AND FALL THROUGH
013F- BD 4C 01   0420 SP1      LDA TABLE,X   ;TO SPEAK IT
0142- 85 02      0430          STA *SPNT
0144- BD 4D 01   0440          LDA TABLE+1,X
0147- 85 03      0450          STA *SPNT+1
0149- 4C D0 02   0460          JMP OUTSPE
014C- 00 20 49   0470 TABLE    .BY 0 $20 $49 $20 $76 $20 $9F $20 ;ADDRESS
                               TABLE FOR
014F- 20 76 20
0152- 9F 20
0154- DB 20 11   0480          .BY $DB $20 $11 $21 $52 $21 $86 $21 ;
                               SPEECH DICTIONARY
0157- 21 52 21
015A- 86 21
015C- B7 21 DA   0490          .BY $B7 $21 $DA $21 $16 $22 $36 $22
015F- 21 16 22
0162- 36 22
0164- 61 22 8E   0500          .BY $61 $22 $8E $22 $B5 $22 $D0 $22 $FD $22
0167- 22 B5 22
```

The SP-1 utilities can be used for many other purposes. A great deal of information and some references concerned with speech synthesis using Linear Predictive Coding techniques are given in the literature supplied with the kit. With this material, you can make your KIM as talkative as you wish!

```
016A- D0 22 FD
016D- 22
                    0510                 .EN


LABEL FILE:   [ / = EXTERNAL ]

/ADLO=000A     /ADHI=000B /SPNT=0002
/GETBYT=1F9D /FPNT=0004 /SPINIT=0200
/OUTSPE=02D0  BEGIN=0100  LOOP=011A
PAUSE=012D    SAY=0131    SPEAK=013E
SP1=013F      TABLE=014C
//0000,016E,016E
```

Footnotes

1. Speak and Spell is a trademark of Texas Instruments, Inc. VIDEO PLUS is a trademark of The Computerist, Inc.
2. East Coast Micro Products, 1307 Beltram Ct., Odenton, Md. 21113.
3. See **6502 User Notes**, No. 13, p. 16 for information about adding a 6522 I/O board.
4. See **MICRO**, No. 17, pp. 27-39 for a general description of the 6522. ©

Expanding KIM-Style 6502 Single Board Computers

Hal Chamberlin

The Modified KIM Bus

Part 3 of 3

This leads us to a definition of the "Unbuffered Modified KIM Bus". KIM is part of the name since the bus is essentially what a KIM-1 single board computer presents on its expansion connector. "Modified" is part of the name because not all of the 44 signals on the expansion connector are actually part of the bus. Those signals that are part of the bus are common to the SYM and AIM computers as well as the KIM and thus any of these machines may be plugged into the bus without modification.

Figure 6 gives a signal listing for the bus. Signals marked with an * do not connect to the processor but do connect to all of the other boards in the system. Most of these have different specialized functions on the different processors anyway and are not generally useful in a bus oriented system. Note that RDY is one of the signals that is not bussed. All modern memories are quite fast enough to operate without wait states in 6502 systems and besides, the 6502 will not wait during write cycles anyway. The lines marked (Reserved) are intended for future uses such as memory bank switching signals, etc.

Note that although RAM R/W is listed as a signal (should go low during phase 2 of Write cycles), it should not be used by a bus interface board for general application. The reason is that an AIM-65 printed circuit error makes it go low during *read* cycles rather than write cycles like it should. In any case, one should be able to design any kind of bus interface board using just A0 - A15, D0 - D7, R/W, PHASE 2, interrupt, and power voltages. The additional lines are really just convenience signals.

Two of the signals are important only in KIM systems. DECODE ENABLE must go low whenever addresses between 0000 and 1FFF are on the bus in order to activate KIM's on-board memory. VECTOR FETCH must go low whenever addresses between FFFA and FFFF are on the bus in order for the reset/interrupt vectors stored in the KIM monitor ROM's to be active. Although it is probably best for the motherboard to generate these two signals, many expansion boards generate them anyway so that the bus motherboard can be omitted altogether in systems with just one expansion board.

Note that direct memory access is not supported by the Modified KIM Bus because the address lines from the 6502 cannot be disabled. An approach to DMA in those interfaces that need it, such as video displays and disk controllers, is to provide *two-port memory* on the interface board itself. The big advantage then is that DMA to or from the on-board memory can then proceed at very high speed without slowing the processor at all. A conventional DMA system, such as on S-100 systems, would stop the processor cold at data rates beyond a couple of hundred thousand bytes per second.

Although +5 volts regulated is available on the bus, it is often preferable to use unregulated +8 and an on-board regulator to provide +5 to the logic circuitry of expansion boards. Similarly, +16 unregulated is available for generating +12 power needed by many memory chips. When negative voltages are needed such as for EPROM's or analog circuitry, they may be easily generated from the positive unregulated voltages with a charge-pump circuit and then regulated with IC regulators. The primary advantages of on-board regulation are a smaller and less expensive central power supply and clean, well regulated power on the expansion board itself. The potential problem of additional heat dissipation on the expansion boards is nullified by the very low power consumption of modern LS IC's.

PIN	KIM-1	SYM-1	AIM-65		MODIFIED
E-1	SYNC	SYNC	SYNC		SYNC
E-2	RDY	RDY	RDY		(reserved)
E-3	PHASE 1	PHASE 1	PHASE 1		(reserved)
E-4	IRQ	IRQ	IRQ		IRQ
E-5	SET OVERFLOW	SET OVERFLOW	SET OVERFLOW		SET OVERFLOW
E-6	NMI	NMI	NMI		NMI
E-7	RESET	RESET	RESET		RESET
E-8	DATA BUS 7	DATA BUS 7	DATA BUS 7		DATA BUS 7
E-9	DATA BUS 6	DATA BUS 6	DATA BUS 6		DATA BUS 6
E-10	DATA BUS 5	DATA BUS 5	DATA BUS 5		DATA BUS 5
E-11	DATA BUS 4	DATA BUS 4	DATA BUS 4		DATA BUS 4
E-12	DATA BUS 3	DATA BUS 3	DATA BUS 3		DATA BUS 3
E-13	DATA BUS 2	DATA BUS 2	DATA BUS 2		DATA BUS 2
E-14	DATA BUS 1	DATA BUS 1	DATA BUS 1		DATA BUS 1
E-15	DATA BUS 0	DATA BUS 0	DATA BUS 0		DATA BUS 0
E-16	K6	30	-12 VOLTS REG.	*	(reserved)
E-17	SINGLE STEP OUT	DB OUT	+12 VOLTS REG.	*	(reserved)
E-18	(N.C.)	POWER ON RESET	CS8	*	+7.5 UNREG
E-19	(N.C.)	(N.C.)	CS9	*	VECTOR FETCH
E-20	(N.C.)	(N.C.)	CSA	*	DECODE ENABLE
E-21	+5 VOLT REG.	+5 VOLT REG.	+5 VOLT REG.		+5 VOLT REG.
E-22	GROUND	GROUND	GROUND		GROUND
E-A	ADDR BUS 0	ADDR BUS 0	ADDR BUS 0		ADDR BUS 0
E-B	ADDR BUS 1	ADDR BUS 1	ADDR BUS 1		ADDR BUS 1
E-C	ADDR BUS 2	ADDR BUS 2	ADDR BUS 2		ADDR BUS 2
E-D	ADDR BUS 3	ADDR BUS 3	ADDR BUS 3		ADDR BUS 3
E-E	ADDR BUS 4	ADDR BUS 4	ADDR BUS 4		ADDR BUS 4
E-F	ADDR BUS 5	ADDR BUS 5	ADDR BUS 5		ADDR BUS 5
E-H	ADDR BUS 6	ADDR BUS 6	ADDR BUS 6		ADDR BUS 6
E-J	ADDR BUS 7	ADDR BUS 7	ADDR BUS 7		ADDR BUS 7
E-K	ADDR BUS 8	ADDR BUS 8	ADDR BUS 8		ADDR BUS 8
E-L	ADDR BUS 9	ADDR BUS 9	ADDR BUS 9		ADDR BUS 9
E-M	ADDR BUS 10	ADDR BUS 10	ADDR BUS 10		ADDR BUS 10
E-N	ADDR BUS 11	ADDR BUS 11	ADDR BUS 11		ADDR BUS 11
E-P	ADDR BUS 12	ADDR BUS 12	ADDR BUS 12		ADDR BUS 12
E-R	ADDR BUS 13	ADDR BUS 13	ADDR BUS 13		ADDR BUS 13
E-S	ADDR BUS 14	ADDR BUS 14	ADDR BUS 14		ADDR BUS 14
E-T	ADDR BUS 15	ADDR BUS 15	ADDR BUS 15		ADDR BUS 15
E-U	PHASE 2	PHASE 2	PHASE 2		PHASE 2
E-V	READ/WRITE	READ/WRITE	READ/WRITE		READ/WRITE
E-W	READ/WRITE	READ/WRITE	READ/WRITE		READ/WRITE
E-X	PLL TEST	AUDIO TEST	AUDIO TEST	*	+16 VOLT UNREG.
E-Y	PHASE 2	PHASE 2	PHASE 2		PHASE 2
E-Z	RAM R/W	RAM R/W	# RAM R/W		RAM R/W

* These signals are not bussed to the CPU slot.
\# Signal generated is different from KIM-1 and SYM-1.

Fig. 6 Processor and Modified Expansion Bus Signals.

Cassette I/O With AIM 65 BASIC

Michael Rathbun
Polar Solutions
Kodiak, Alaska

The AIM 65 is one of the few micro systems I have worked with which was packed with PLEASANT surprises. Its monitor, assembler, and BASIC do things I didn't expect from a piece of equipment in its price range. After a while, however, I found myself wishing that the excellent AIM cassette system could be used with the BASIC on the system for data input and output, instead of just for SAVE and LOAD. It turns out that, because BASIC uses certain monitor routines to interface the keyboard and display/printer, BASIC cassette file I/O is not all that difficult.

Monitor Routines

For those who haven't spent an exciting evening or two reading the assembly listing of the monitor which Rockwell provides, here is a brief summary of the I/O routines which BASIC uses.

Most of the AIM 65 functions which get data from the keyboard (i.e. Editor, BASIC, and even Assembler) do so by calling a monitor routine called INALL. INALL, however, is not just for accessing the keyboard. It will get a byte of data from ANY input device. Which device it goes to is determined by the contents of a memory location labelled INFLG, which is located at $A412. If this location contains a RETURN character ($OD) then the input will be from the keyboard. If INFLG contains an ASCII "T" (54), then INALL will look to the cassettes for data.

How does this location come to contain the proper value? The functions which allow a selection of input devices also make use of a subroutine from the monitor called WHEREI; it is this subroutine which displays the familiar "IN = " prompting message after the BASIC LOAD command is entered. If you respond to "IN = " with "T", the WHEREI routine then also asks for a file name ("F = ") and then finds out which cassette you will use ("T = "). From this time on, any time INALL is called, a byte of data from the specified tape file will be returned.

Output works in a similar fashion; there is a subroutine called OUTALL which will output a byte of data to any AIM 65 output device, depending on the contents of a location labelled OUTFLG, which is located at $A413. This location is set to the desired value by a subroutine called WHEREO, which is the one which generates the "OUT = " prompt.

Utilization

Making your BASIC programs read from cassettes is quite simple--most of the work has been done for you by the program logic used by the LOAD command. When you type LOAD and give the cassette file information, BASIC simply takes its input data from the tapes instead of from the keyboard, continuing to do this until a CONTROL Z character ($1A) is read from the tape. The CONTROL Z causes control to return to the keyboard. If your program contains a step with the LOAD command (for example, 100 LOAD) then when this step is executed, you will see the "IN = " message. If you specify input from a cassette file, then from that point on, until a CONTROL Z is read, or until INFLG is changed to a RETURN character, every INPUT statement in your program will take data from the tapes instead of from the keyboard.

When you reach a point in your program when you wish to switch input back to keyboard, simply POKE a RETURN into INFLG. If you want to intermix INPUTs from keyboard and tape, you can change the input device back and forth at any time by changing the contents of INFLG. Remember, though, that if your program bombs with an error while INFLG points to the tapes, the system will go on trying to get its data from the tape file; you will have to use the RESET button to regain control of the situation.

For writing data to cassettes, the procedure is a little more complex; there is no BASIC command which will change OUTFLG. The SAVE command will access the tapes, all right, but all it does is LIST the program and return to keyboard control. However, this sequence of steps will work:

1. POKE the address of the WHEREO routine into locations 4 and 5.
2. Execute a USR(X) statement. This will cause BASIC to call WHEREO.
3. Output data is required using regular PRINT statements.
4. When output is finished, you will need to close the file properly. Do this by PRINTING CHR$(13) and CHR$(26). This puts an AIM Editor end-of-file mark on the tape, followed by a CONTROL Z, just to be safe. Then POKE the address of the routine called DU11 (see table of locations) into locations 4 and 5, and execute a USR(X) statement. This will end the cassette file properly, and also will restore output to the display.

Notes and Cautions

If the OUTFLG is set to send output to tapes, and your program bombs with an error message for some reason, you will never see the error message — it will have been written to tape! For this reason, it is a good idea to debug programs using regular keyboard input and display output before using cassettes; also, it might be wise to "turn off" the cassettes when not actually reading or writing, by POKEing a RETURN into INFLG or OUTFLG after a statement which accesses tape. This allows you to intermix keyboard-display and cassette operation.

You can use both input and output in the same program, but unfortunately, NOT AT THE SAME TIME. The reason for this restriction is as follows: the monitor cassette routines store data on tape in 80-byte blocks. The data going to or from a block on tape is stored temporarily in a buffer area in memory. If INFLG and OUTFLG are both "T", then the cassette write routine uses a different buffer from that used by the read routine. This buffer is located on page zero, right in the middle of the area BASIC uses for its math operations. Therefore, if the same program is going to do both reading and writing, it must finish completely with one operation before it initializes the other. A procedure which eliminates this restriction (but requires assembly-language routines and some memory overhead) was reported in the first issue of Rockwell's new publication INTERACTIVE. The method used here is considerably simpler, but limits you to read-only or write-only at any given instant.

Sample Programs

The two sample programs were developed to fill a need in a project I was working on. The first writes a table of about 600 prime numbers to tape; the second program reads this table from tape into an integer array, and uses this array to print the factors of a number entered from the keyboard. While not elegant examples of the programmer's art, they do show the implementation of the procedures detailed here.

Location Table

Label	Hex	Decimal	Function
INFLG	A412	42002	Defines input device
OUTFLG	A413	42003	Defines output device
WHEREI	E848	59464 Low byte = 72 High byte = 232	Initialize INFLG
WHEREO	E871	59505 Low byte = 113 High byte = 232	Initialize OUTFLG
DU11	E50A	58634 Low byte = 10 High byte = 229	Close active tape file

List

```
0 REM SET UP OUTPUT TAPE FILE.
1 POKE 4,113: POKE 5,232
2 N = USR(N)
5 UL = 600: REM DEFINE TABLE LIMIT HERE
10 DIM X%(UL)
20 X%(1) = 2: X%(2) = 3
30 L = 2
90 N = 3
100 I = 1
110 IF INT(N/X%(I)<>N/X%(I) THEN 200
120 N = N + 2 GOTO 100
200 IF X%(I) = >SQR(N) THEN 300
210 I = I + 1: GOTO 110
300 L = L + 1: X%(L) = N
309 REM OUTPUT TO TAPE.
310 PRINT N
314 REM ALSO SHOW NUMBER ON DISPLAY.
315 POKE 42003,13: PRINT N: POKE 42003,ASC("T)
320 IF L<>UL THEN N = N + 2: GOTO100
321 REM
329 REM WRITE END-OF-FILE MARK ON TAPE
330 PRINT CHR$(13); CHR$(26)
331 REM
339 REM CLOSE TAPE WITH DU11 ROUTINE.
340 POKE 4,10: POKE 5,229
350 N = USR(N)
360 PRINT" DONE."
```

```
10 DIM A%(600)
20 A%(1) = 2
90 REM SET UP TAPE INPUT.
100 LOAD
115 REM READ DATA FROM TAPE TO ARRAY.
120 FOR I = 2 TO 597: INPUT A%(I): NEXT
125 REM TURN OFF TAPE.
130 POKE 42002,13
200 INPUT X
205 PRINT! "**";X
210 Q = 1
220 IF INT(X/A%(Q) = X/A%(Q) THEN 230
225 Q = Q + 1: GOTO 240
230 PRINT! A%(Q): X = A/A%(Q)
240 IF SQR(X) = >A%(Q) THEN 220
250 PRINT! X: GOTO 200
```

©

New Products

Commodore Business Machines Announces Availability Of Emergency Relief Plan Application Program

Commodore Business Machines, a Division of Commodore International Limited has announced the availability of a disaster/emergency plan computer application program.

As a result of the Commodore experiences during the COMDEX 80 exposition and the tragic fire at the MGM Grand Hotel in Las Vegas, November 22, 1980, the striking need for immediate information dissemination on the whereabouts and status of the hotel guests and employees was apparent. With the consent and encouragement of Commodore's Vice Chairman and Chief Executive Officer, Mr. Jack Tramiel, Commodore removed seven complete computer systems from the COMDEX booth and established a computer command center.

Marge Jillett, Director of Public Relations recruited volunteers to man the command center until three a.m., Sunday, November 24, 1980. Brian Padol, representing Micro Search, Inc. adapted a Commodore information list management system program to allow volunteers to type the name, address, MGM room number and the site of relocation of the thousands of guests. Lists were compiled, printed and distributed throughout the night.

Lieutenant Ross of the Las Vegas Metropolitan Police Department stated "We were not equipped to handle a disaster of this magnitude without the computers and personnel". The Commodore "command center" became a vital information source for the police, the fire department, Red Cross and other disaster relief organizations.

Commodore Business Machines Inc. will release to its over 500 dealers this disaster relief program and document its experiences so that should an emergency of this magnitude occur again, the Commodore dealer can assist all local disaster relief organizations within their area, in the continuing concern to assist the public.

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.

Chess And Checkers Programs For Atari Personal Computers

SUNNYVALE, CA — January 22, 1981 — Personal Software Inc. has introduced MicroChess™ and Checker King™ for the Atari ™ 400 and 800 personal computers.

The MicroChess program turns a computer display screen into a chess board, and is the industry's first "gold cassette" software product with sales over $1 million. The board and all its pieces are illustrated in high-resolution color graphics.

MicroChess has eight levels of play, and lets the player pick the appropriate ability level. MicroChess plays by tournament rules and allows no illegal moves, making the program an excellent chess teacher.

Checker King brings the popular game of checkers to Atari home computers. The program turns the computer display screen into a colorful checkerboard, where all pieces are — like MicroChess — illustrated using high-resolution graphics.

Checker King allows single, double and triple jumps, forces jumps and performs according to the tournament rules of checkers. And, again like MicroChess, Checker King allows no illegal moves at any of its eight levels of play.

In both Atari versions of MicroChess and Checker King, tournament excitement is generated by an on-screen, real-time clock that ticks off the seconds while the player and the computer ponder the next move.

IT'S A DAISYWHEEL COMPUTER PRINTER & AN ELECTRONIC TYPEWRITER BUT . . .

IT WON'T WORK!

...UNLESS YOUR PRINTER & YOUR SOFTWARE ARE TOTALLY COMPATIBLE

Only **$2850** Suggested Retail

The TYPRINTER 221™

Is a TYPEWRITER QUALITY, DAISYWHEEL PRINTER that is Totally Compatible with All Word Processors. That's because the TYPEPRINTER 221 may be PROGRAMMED in PLAIN ENGLISH, Imbedded within The Text File of All Word Processing Software!

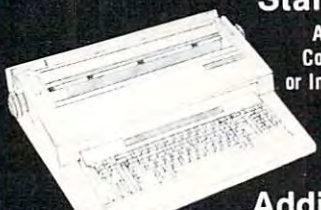
Use the 221 as your...

Electronic Typewriter

When not being used as a Computer Printer, the 221 becomes a fully functional Electronic Typewriter.

Computer Printer

It's a Daisywheel Computer Printer with more standard features and available options than any other machine.



Stand Alone Terminal

Available options allow the 221 to Communicate with Distant Computer or Information Services such as Source, Micronet & others.

Tele-Communications Terminal

Option available to allow your 221 to access the Teletype & Telenex networks.

Telex & Teletype are registered trademarks.

Additional Options

4K or 16K RAM Memory which can be used as INPUT or OUTPUT Buffers. Also use as an Automatic Spooler to your computer. Bi-Directional Communicatons from The 221 to your Pet, Apple or TRS-80. Nothing else to buy. Lawyers, Accountants and others will find our Automatic Strike-Out Type and High Density Spacing options very useful.

Built-in Features

The 221 Centers Copy Automatically, Sets Columns, Prints in Reverse, Bold Face and Underlines Automatically. The 221 also Justifies Right, Types in Three Pitches and does Proportional Spacing. It Types in Spanish, French, German, Italian and Portugese as well as English. And much, much more!

Call 714/778-3443 for the distributor in your area.

HOWARD INDUSTRIES

2031 E. Cerritos Ave. 7K
Anaheim, California 92806

Both MicroChess and Checker King for the Atari are available on cassette for Atari 400 and 800 personal computers and both require 8K bytes of memory. MicroChess was written by Peter Jennings; Checker king is by Michael Marks.

For more information, please contact Jeff Walden, Personal Software, Inc., 1330 Bordeaux Drive, Sunnyvale, CA 94086.

Atari, Atari 400 and Atari 800 are registered trademarks of Atari, Inc. MicroChess and Checker King are trademarks of Personal Software Inc.

Real Estate Analyzer

A new edition of the "REAL ESTATE ANALYZER by HowardSoft" is now available for Apple Computers. This software package is unique in its realistic handling of TODAY'S market conditions for real estate investments: creative financing, negative cash flows, component depreciation, high property inflation, rent control, property tax limitations, high returns on near-term income, and inflationary increases in operating expenses.

The software provides projections of annual cash flows and on-sale return-on-investment, as well as several other measures of profitability, including all the consequences of ordinary and capital gains taxes as well as inflation. Data for your properties are easily filed on disk for later retrieval and alteration. Results are displayed on the video screen or printed with a line printer in a flexible report format with complete itemized tables. The package comes with two disks and a detailed instruction manual in a quality notebook, complete with explanations of the principles of investment analysis. More complete and realistic than packages costing many times more, this product sells for $99 at dealers everywhere. (Requires Apple Computer with 48K, Applesoft ROM, and disk drive.) HOWARD SOFTWARE SERVICES, 7722 Hosford Avenue, Los Angeles, CA 90045, (213) 645-4069.

Cimarron Announces An Attorney Package For Commodore's 8032 Business Computer

Costa Mesa, CA./ Cimarron Corporation has announced a major applications package programmed exclusively for the legal profession. Incorporating both accounts receivable and matter tracking, Legal Time Accounting (LTA) offers law firms with an inexpensive solution to the problems of managing the daily flow of words and information.

LTA proceduralizes daily operations by logging each activity e.g., conference, telephone time, etc., then stores this data by matter and lawyer. The resultant data provides for control of receivables, tracking of attorney activity and revenue and tracking of client and matter activities — all with daily and monthly totals. Reports include aging analysis, attorney billings with ratios, client billings with ratios, activity code analysis and a daily charges and payments journal. Statements can be generated twice monthly allowing for more predictable cash flow. General ledger and accounts payable are also available.

LTA is programmed specifically for the Commodore 8032 computer system utilizing either the 4040 or 8050 twin diskette drives. Compatible printers are the NEC Spinwiter or C. Itoh's Starwriter. Both printers allow for printing of fully formed characters so that the popular WordPro word processing program can be used in conjunction with LTA. In its full hardware configuration, an automatic sheet feeder is added providing for continuous, hands-off operation.

According to Michael C. Miller, developer of LTA and cofounder of Cimarron, the advanced design of the program represents the first time high quality applications software created for minicomputers has been made available on the now more powerful Commodore business computer.

LTA is priced at $900.00 per copy and includes documentation and support materials. For high volume dealers, Cimarron will offer a one time charge. Additionally, Cimarron will pre-package and fully test the entire system for those dealers wishing turnkey installation. Sales and program training are also available.

For more information, please contact Daniel M. Gomez, Cimarron Corporation, 600 Baker Street, Suite 319, Costa Mesa, CA 92626. (714)641-1156

Apartment Management Software Package

Norcross, Georgia — MIN Microcomputer Software, Inc., has announced The Landlord™, an apartment management software package for Apple II® computers. The system can be used by apartment properties of up to 400 units.

The Landlord™ provides property owners and managers with listings of apartments, residents, and past residents, as well as reports on vacancies, lease expirations, intents to vacate, and resident payments. Records of disbursements and other financial transactions are maintained by the system and a monthly property analysis statement is produced.

The Landlord™ allows entry of resident charges and payments using up to 26 different account codes. Security and pet deposits, returned checks, and overpayments are also handled by the system. An outstanding balance report allows expedient follow-up of delinquent residents.

The package is designed to be used by managers who have no prior computer or data processing experience. The manual included with The Landlord™ as well as the

SOFTWARE CITY

presenting the LARGEST SELECTION OF SOFTWARE EVER ASSEMBLED...

for ATARI® • APPLE® • PET® • and other Microcomputers

at SUPER DISCOUNT PRICES!

ATARI

- ☐ PHYSICS (AT) 24.50
- ☐ GREAT CLASSICS (AT) 24.50
- ☐ BASIC PHYSCOLOGY (AT) 24.50
- ☐ PRINCIPLES OF ECONOMICS (AT) 24.50
- ☐ SPELLING (AT) 25.50
- ☐ BASIC ELECTRICTITY (AT) 24.50
- ☐ BASIC ALGEBRA (AT) 24.50
- ☐ 8K RAM MODULE (AT) 95.00
- ☐ 16K RAM MODULE (AT) 170.00
- ☐ KINGDOM (AT) 13.55
- ☐ LEMONADE (AT) 13.55
- ☐ STATISTICS I (AT) 17.95
- ☐ BLACKJACK (AT) 13.55
- ☐ BIORYTHM (AT) 13.55
- ☐ HANGMAN (AT) 13.55
- ☐ SPACE INVADERS (AT) 17.95
- ☐ EUROPEAN CAPITALS (AT) 13.55
- ☐ MORTGAGE LOAN (AT) 13.55
- ☐ STATES & CAPITALS (AT) 13.55
- ☐ EDUCATION SYSTEM (AT) 22.50
- ☐ ATARI BASIC (AT) 53.95
- ☐ ASSEMBLER DEBUG (AT) 53.95
- ☐ BASKETBALL (AT) 35.95
- ☐ VIDEO EASEL-LIFE (AT) 35.95
- ☐ SUPER BREAKOUT (AT) 35.95
- ☐ MUSIC COMPOSER (AT) 53.95
- ☐ COMPUTER CHESS (AT) 35.95
- ☐ 3-D TIC TAC TOE (AT) 35.95
- ☐ STAR RAIDERS (AT) 53.95
- ☐ TELELINK (AT) 22.50
- ☐ PADDLES (AT) 17.95
- ☐ JOYSTICKS (AT) 17.95
- ☐ U.S. HISTORY (AT) 24.50
- ☐ U.S. GOVERNMENT (AT) 24.50
- ☐ SUPERVISERY SKILLS (AT) 24.50
- ☐ WORLD HISTORY (AT) 24.50
- ☐ BASIC SOCIOLOGY (AT) 24.50

ADVENTURE INTERNATIONAL

- ☐ ADVENTURE HINT SHEET 7.95
- ☐ ADVENTURE (1,2,3) [D] (AP) 35.95
- ☐ ADVENTURE (4,5,6) [D] (AP) 35.95
- ☐ ADVENTURE (7,8,9) [D] (AP) 35.95
- ☐ ADVENTURE #10 [D] 18.95
- ☐ ADVENTURE (specify 1-10) (AP) 13.55
- ☐ PLANETOIDS "ASTEROIDS" [D] (AP) 17.95
- ☐ PLANETOIDS "ASTEROIDS" (AP) 13.55
- ☐ POKER (AP) 13.55
- ☐ POKER (AP) [D] 18.95
- ☐ KID VENTURE #1 (AP) 13.55

AVALON HILL

- ☐ MIDWAY (P,AP) 13.50
- ☐ NUKE WAR (P,AP) 13.50
- ☐ PLANET MINERS (P,AP) 13.50
- ☐ CONVOY RAIDER (P,AP) 13.50
- ☐ B1 BOMBER (P,AP) 13.50
- ☐ LORDS OF KARMA (P,AP) 18.00

AUTOMATED SIMULATION

- ☐ TUESDAY QUARTERBACK [D] (AP) 26.95
- ☐ STAR WARRIOR [C,D] (AP) 35.95
- ☐ THREE PACK [D] (AP,P) 45.00
- ☐ STARFLEET ORION [C,D] (AP) 22.50
- ☐ STARFLEET ORION [C] (P) 22.50
- ☐ INVASION ORION [C,D] (AP) 22.50
- ☐ INVASION ORION [C] (P) 22.50
- ☐ APSHAI [D] (AP) 26.95
- ☐ APSHAI [C] (P) 26.95
- ☐ RYN [D,C] (AP) 17.95
- ☐ RYN [C] (P) 17.95
- ☐ MORLOC [C,D] (AP) 17.95
- ☐ MORLOC [C] (P) 17.95
- ☐ RIGEL [C,D] (AP) 22.50
- ☐ RIGEL [C] (P) 22.50
- ☐ HELLFIRE [D] (AP) 26.95
- ☐ HELLFIRE [C] (P) 26.95

QUALITY SOFTWARE

- ☐ 6502 DISASSEMBLER (AT) 10.55
- ☐ ASTRO APPLE (AP) 13.55
- ☐ ASTRO APPLE (AP) [D] 17.95
- ☐ ASTEROIDS IN SPACE [D] (AP) 17.95
- ☐ ATARI ASSEMBLER (AT) 22.50
- ☐ BABBLE (AP) 13.55
- ☐ BABBLE (AP) [D] 17.95
- ☐ BATTLESHIP COMMANDER (AP) 13.55
- ☐ BATTLESHIP COMMANDER [D] (AP) 17.95
- ☐ BENEATH APPLE MANOR (AP) 13.55
- ☐ BENEATH APPLE MANOR (AP) [D] 17.95
- ☐ FASTGAMMON [D] (AP) 22.50
- ☐ FASTGAMMON (AP,AT) 17.95
- ☐ FORTH [D] (AT) 70.00
- ☐ FRACAS ADVENTURE (AP) 17.95
- ☐ FRACAS ADVENTURE [D] (AP) 22.50
- ☐ LINKER (AP) [D] 44.00
- ☐ TANK TRAP (AT) 10.55
- ☐ TANK TRAP (AT) [D] 13.55
- ☐ TARI TREK (AT) 10.55
- ☐ TARI TREK (AT) [D] 13.55

PERSONAL SOFTWARE

- ☐ CCA DATA MGMT [D] (AP) 85.00
- ☐ DESKTOP PLAN [D] (AP) 85.00
- ☐ GAMMON GAMBLER (AP) 17.95
- ☐ GAMMON GAMBLER [D] (AP) 22.50
- ☐ MONTY MONOPOLY [D] (AP) 31.55
- ☐ VISICALC [D] (AP) 125.00
- ☐ VISICALC [D] (AT,P) 170.00

INSTANT SOFTWARE

- ☐ AIR FLIGHT SIMULATION (P,AP) 8.95
- ☐ APPLE FUN [D] (AP) 17.95
- ☐ CASINO (P) 7.25
- ☐ MORTGAGE (P) 7.25
- ☐ PADDLE FUN [D] (AP) 17.95
- ☐ PENNY ARCADE (P) 7.25
- ☐ PET UTILITY (P) 8.95
- ☐ QUBIC 4/GOMOKU (P) 7.25
- ☐ SANTA PARAVIA FIUMACCIO (AP,P) 8.95
- ☐ SANTA PARAVIA FIUMACCIO (AP)[D] 17.95
- ☐ SAHARA WARRIOR (AP) 7.25
- ☐ SKY BOMBERS (AP) [D] 17.95
- ☐ SPACE WARS (AP) 7.25
- ☐ SUPERSHOOTERS (AP) 8.95
- ☐ TREK-X (P) 7.25

STRATEGIC SIMULATIONS

- ☐ COMPUTER AMBUSH [D] (AP) 51.50
- ☐ COMPUTER BISMARCK [D] (AP) 51.50
- ☐ COMPUTER CONFLICT [D] (AP) 35.00
- ☐ COMPUTER NAPOLEONICS [D] (AP) 35.00
- ☐ COMPUTER QUARTERBACK [D] (AP) 35.00
- ☐ AIR COMBAT [D] (AP) 51.50
- ☐ WARP FACTOR [D] (AP) 35.00

SUB-LOGIC

- ☐ 3D GRAPHICS (AP) 40.00
- ☐ 3D GRAPHICS [D] (AP) 48.00
- ☐ A2-FS1 FLIGHT SIMULATOR (AP) 22.00
- ☐ A2-FS1 FLIGHT [D] (AP) 29.00

MICROSOFT SOFTWARE

- ☐ ADVENTURE [D] (AP) 25.50
- ☐ OLYMPIC DECATHALON [D] (AP) 20.00
- ☐ TYPING TUTOR (AP) [D] 17.95
- ☐ TYPING TUTOR (AP) 13.55
- ☐ Z-80 SOFTCARD [D] (AP) 280.00

ON LINE SYSTEMS

- ☐ HI-RES ADVEN. #0 (AP) [D] 17.95
- ☐ HI-RES ADVEN. #1 [D] (AP) 22.50
- ☐ HI-RES ADVEN. #2 [D] (AP) 29.00
- ☐ HI-RES FOOTBALL #1 [D] (AP) 36.00
- ☐ HI-RES CRIBAGGE [D] (AP) 22.50
- ☐ PADDLE GRAPHICS [D] (AP) 36.00
- ☐ TABLET GRAPHICS [D] (AP) 44.95

SIRIUS

- ☐ CYBER STRIKE [D] (AP) 36.00
- ☐ STAR CRUISER [D] (AP) 22.50
- ☐ BOTH BARRELS [D] (AP) 22.50
- ☐ PHANTOM FIVE [D] (AP) 36.00

SYNERGISTIC SOFTWARE

- ☐ DUNGEON & WILDERNESS [D] (AP) 29.00
- ☐ DUNGEON (AP) 13.50
- ☐ DUNGEON [D] (AP) 15.75
- ☐ ODYSSEY [D] (AP) 27.00
- ☐ HIGHER GRAPHICS [D] (AP) 31.50
- ☐ WILDERNESS (AP) 15.75
- ☐ WILDERNESS [D] (AP) 18.00

BORDERBUND

- ☐ EMPIRE GALACTIC (AP) [D] 22.50
- ☐ GALAXIAN (AP) [D] 22.50
- ☐ HYPER HEAD ON (AP) [D] 22.50
- ☐ REVOLUTION GALACTIC (AP) [D] 22.50
- ☐ TANK (AP) [D] 13.55
- ☐ TAWALA'S REDOUBT (AP) [D] 26.95
- ☐ TRADER GALACTIC (AP) [D] 22.50

MUSE COMPANY

- ☐ ABM [D] (AP) 22.50
- ☐ ADDRESS BOOK (AP) [D] 44.50
- ☐ APPILOT II [D] (AP) 80.00
- ☐ BEST OF MUSE (AP) [D] 35.95
- ☐ GLOBAL WAR (AP) [D] 22.50
- ☐ MATH-APPLESOFT (AP) [D] 35.95
- ☐ SUPER TEXT II (AP) [D] 135.00
- ☐ THREE MILE ISLAND (AP) [D] 35.95
- ☐ U-DRAW II (AP) [D] 35.95
- ☐ THE VOICE (AP) [D] 35.95

IRIDIS

- ☐ IRIDIS 1 (AT) 8.95
- ☐ IRIDIS 1 (AT) [D] 11.75
- ☐ IRIDIS 2 (AT) 14.50
- ☐ IRIDIS 2 (AT) [D] 16.95

EDU-WARE

- ☐ COMPU READ (AP) [D] 22.50
- ☐ ESP (AP) [D] 14.50
- ☐ NETWORK (AP) [D] 17.95
- ☐ PRISONER (AP) [D] 26.95
- ☐ SPACE (AP) [D] 26.95
- ☐ SPACE II (AP) [D] 22.50
- ☐ TERRORIST (AP) [D] 26.95

PROGRAMMA

- ☐ MICRO INVADERS (AP) 14.50
- ☐ EXPAND-A-PORT (AP) 53.95
- ☐ JOYSTICK (AP) 35.95
- ☐ TINY PASCAL (AP) [D] 44.50
- ☐ SPACE WARS (AP,P) 8.95
- ☐ WPS STANDARD (AP) [D] 117.00

HAYDEN

- ☐ SARGON II (AP) 25.00
- ☐ SARGON II (AP) [D] 30.00
- ☐ REVERSAL (AP) 25.00

If you don't see it listed, write...we probably have it in stock!

**Check program desired.
Complete ordering information
and mail entire ad.
Immediate Shipments from stock.**

Ship the above programs as checked to:

Mr./Mrs. ______

Address ______

City ______

State ______ Zip ______

I have a ______ (name of Computer)

with ______ K memory

Number of Programs Ordered ......

Amount of order ......

N.Y. residents add Sales Tax ......

Add shipping anywhere in the U.S. 2.00

Total amount enclosed ......

Charge my: ☐ Master Charge ☐ Visa

Signature ______

Card No. ______ Expires ______

KEY:
AT-Atari
AP-Apple
P-Pet
D-on Disc.
C-Cassette

If not marked-Cassette

ATARI is a trademark of ATARI INC.
APPLE is a trademark of APPLE COMPUTER, INC.
PET is a trademark of COMMODORE BUSINESS MACHINES.

Prices subject to change without notice.

Mail to: **SOFTWARE CITY**

a division of DigiByte Systems Corp.

31 East 31st Street, New York, N.Y. 10016 • (212) 889-8975

instructions that appear on the Apple's® screen are completely non-technical in nature.

Suggested retail price for The Landlord™ is $795.00. The software requires an Apple II® computer w/48K RAM, 2 disk drives, and either a Silentype® or Centronics 779 printer. The Landlord™ will be sold exclusively through retail computer outlets.

MIN Microcomputer Software, Inc. specializes in the development of software packages for specific small business applications.

For more information, please contact Art Nacht, MIN Microcomputer Software, Inc., 5835-A Peachtree Corners East, Norcross, GA 30092. (404) 447-4322.

The Landlord is a trademark of MIN Microcomputer Software, Inc. Apple, Apple II, and Silentype are registered trademarks of Apple Computer, Inc.

1981 Tax Preparer

The 1981 Edition of the "TAX PREPARER by HowardSoft" is now available for Apple Computers. The new edition has several improvements over the acclaimed 1980 version, including continuous-stream printing for professional tax preparers, printouts that can be filed directly with the IRS, expanded documentation in a quality notebook, and the addition of Form 2210 to the long list of built-in forms (Schedules A, B, C, D, E, F, G, R&RP, SE, TC, and Forms 1040, 2106, 3468, 4562, 472, 4797, 5695).

Unique features include on-screen facsimiles of IRS forms during preparation, easy creation, filing, and editing of itemized lists to support any entry, automatic computing of all arithmetic, automatic linking of results of various forms, and easy comparison of alternative tax strategies. More complete and easy-to-use than packages costing many times more, this package comes with two disks and professional documentation, and sells for $99 at dealers everywhere. (Requires Apple Computers with 48K, Applesoft ROM, and at least one disk drive.) HOWARD SOFTWARE SERVICES, 7722 Hosford Avenue, Los Angeles, CA 90045, (213) 645-4069.

Super X-10 Mod From CMC For Home/Office Security Systems

The SUPER X-10 MOD, recently introduced by Connecticut microComputer, Inc. allows direct computer control over the basic components in a home/office security system.

Developed for use with most popular microcomputers, including PET, APPLE, TRS-80, and KIM, the MOD controls up to 256 different remote devices by sending signals over house wiring to readily available BSR remote modules. These low cost modules, in conjunction with the SUPER X-10 MOD, allow microcomputer control over lamps, motors, and appliances. With eight digital inputs and eight digital outputs included, the SUPER X-10 module can easily be connected to switches at windows and doors for sensing by the microcomputer. The module can be programmed so that the opening or closing of a window or door initiates a sequence of operations such as turning on lights, radio, and alarm, even if the computer is turned off. Direct, plug-in compatibility and software are available for most microcomputers.

In addition, the SUPER X-10 MOD can put kitchen appliances, stereo systems, television, motors, fans, pumps, and laboratory equipment under computer control. With the module, additional service from microcomputers in business and small industrial applications is now possible. A clock and calendar which can be read by microcomputers are also incorporated into the module. Suggested single unit pricing for the SUPER X-10 is $249, and the module is available from CMC factory stock or from one of a select group of personal computer dealers.

For further information, write: SUPER X-10 MOD, Connecticut microComputer, Inc., 150 Pocono Road, Brookfield, CT 06804

Scientific Plotter for APPLE II

STATE COLLEGE, PA...Interactive Microware, Inc. has announced a program called Scientific Plotter which produces professional-looking graphs. Plotting your results with Scientific Plotter is much easier, faster, neater and more accurate than plotting your data by hand. Data may be input from the keyboard, from the disk or it may be calculated by your own subroutine. In each case, the data may be supplied either as X,Y pairs or as Y values at a constant X interval. Since 20 different plotting symbols are provided, you may plot more than one set of data on the same graph. Also, error bars of variable length may be used to indicate the range of error for each point.

Scientific Plotter gives you complete control of the length and position of each axis, the grid size and the interval between numbers that are printed along the axes. Thus, it is possible to plot data in one, two or four quadrants and different scales may be specified for up to four axes. Any number of labels may be superimposed on the graph, using an alphabet of 76 letters and scientific symbols which can be printed in four different orientations at 90 degree angles. The finished graph may be saved on disk for later review or it may be printed on a graphics printer.

Many features of Scientific Plotter make it easy to use. The program displays the allowable range for input values, based on previous answers, and warns of any errors. At any time, you may erase the graph and replot it with any desired changes. All previous answers become the defaults, so that you can make changes quickly. After the best format for your graph has been selected, that format may be saved on disk for subsequent use with similar data. Five demonstrations are included on the disk so that you will learn quickly how to construct various types of graphs.

Scientific Plotter requires a 48K APPLE II computer with Applesoft ROM. It is supplied on a disk with a 25 page manual for $25. The manual may be purchased separately for $5, refundable with purchase. For further information, contact Paul K. Warme, Interactive Microwave, Inc.; P.O. Box 771; State College, PA 16801 or call (814) 238-8294.

PET Software Vendor Expands

Microphys has announced the conversion of its entire educational software line for use on the Apple II/Bell & Howell microcomputers. Over 160 programs are described in our new Winter catalog. These computer-assisted instruction and individualized-instruction programs have been successfully employed in

Chemistry, Physics, and Calculus classes (on both the high school and college levels) and in junior and senior high school Mathematics and English classes. These programs continue to be available for use with the Commodore PET/CBM systems.

For more information, please contact Microphys Programs, 2048 Ford Street, Brooklyn, NY 11229. (212)646-0140.

Hayden Unveils Gameware™ Series

ROCHELLE PARK, NJ— Hayden Book Company, Inc. has announced a new computer game series, called GAMEWARE™. The GAMEWARE series features high-quality, attractively-packaged computer games.

The first three games in the series are: Hayden's REVERSAL™, winner in the software division of the First International Man-Machine Othello Tournament; BLACKJACK MASTER™, a game that allows players to test their betting and playing strategies over thousands of games in minutes; and the famous SARGON II™ chess game.

According to Steven Radosh, Hayden's Software Games and Entertainment Editor, "Hayden's GAMEWARE features the finest microcomputer games on the market, attractively packaged with four-color art, shrink-wrapped, and suitable for rack or shelf display."

Radosh said Hayden plans an extensive promotion program for the GAMEWARE series.

All three initial games in the GAMEWARE series will be available from Hayden in December 1980. For more information, contact: Steven Radosh, Hayden Book Company, Inc., 50 Essex Street, Rochelle Park, NJ 07662, (201) 843-0550.

*Gameware, Reversal, Blackjack Master and Sargon II are trademarks of Hayden Book Company, Inc.

32K Ram Expansion For Atari 400, 800 Announced

Sunnyvale, Calif.—AXLON, Inc. of Sunnyvale has announced its new memory expansion system for the Atari 400 and 800 personal computers.

According to John Vurich, AXLON's President, the memory modules, called RAMCRAM™, can expand the Atari 400 system to 32K, and the Atari 800 to up to 48K-bytes of random access user memory. RAMCRAM contains 16 memory chips, yielding a total of 32K-bytes of additional user program memory.

In the case of the 400, RAMCRAM is installed by removing the top enclosure of the computer console and unplugging the 8K RAM module supplied by Atari. The RAMCRAM module is then plugged into the same slot.

According to Vurich, this modification allows the user of the 400 system to plug in disk drives, printers, and any other peripheral devices formerly compatible only with the much more expensive Atari 800 product. "It really lets one upgrade a 400 to provide all of the capabilities of the 800 with 32K of RAM," he commented. "Any 32K Atari 800 software on the market will run on a 400 with RAMCRAM."

The advantages of RAMCRAM over the Atari plug in memory modules are a little less obvious when it is used with the 800 system. But users with future expansion in mind will immediately see that putting a full 32K-bytes into one memory slot allows upgrading of the system to 48K with one entire slot left over for future expansion.

Are there any devices that can use the extra slot? According to Vurich, "There are many things in the near future." While somewhat reluctant to discuss future products, he did mention that a bus extender could be plugged into the third slot. Such an extender might terminate on the other end with a series of "slots" for use in plugging in "all sorts of interesting things."

This is reminiscent of Atari's competitors who use built-in slot connectors for connection of printers, modems, terminals, and other devices intended to establish contact between the computer and the outside world.

Developing the logic necessary to make the system "think" that two slots are being used instead of one was a relatively small problem for Vurich and his fellow designers of the RAMCRAM modules. The Atari operating system actually does some bank selecting anyway, and they were able to take advantage of this for their own purposes.

"The whole idea", says Vurich, "is to take the Atari 400 system out of the sophisticated toy category and turn it into a useful computer tool." With the ability to plug in printers, disk drives, and other previously incompatible Atari 800 peripherals, Axlon has certainly accomplished that goal!

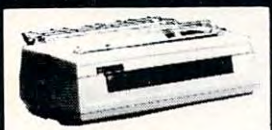
For more information, please contact John Vurich, AXLON, Inc., 170 N. Wolf Rd., Sunnyvale, CA 94086

ΩMEGA SALES CO.

"WHOLESALE COMPUTER PRICES" DIRECT TO THE PUBLIC

East Coast: 12 Meeting St., Cumberland, RI 02864
West Coast: 3533 Old Conejo Rd., #102, Thousand Oaks, CA 91320

Intertec Superbrain
32K Ram - $2449
64K Ram - $2649



NEC spinwriter
5510-5530 - $2449

Apple II - 16K, 48K
Call for Price

Epson MX-80 - $599

Diablo 630 - $1995
(with tractor feed) $2195

Televideo
912 B or C — $699
920 C — $769

PRODUCT SPECIAL of the MONTH!!

Products are NOW IN STOCK AT ΩMEGA Sales Co.

Atari 800 16K - $749
32K - $849

Atari 850 Interface Module $159
Atari 825 Printer $695

Okidata Microline 80 - 525

Centronics - Ltd. quan.

730-3	$ 525
737-1	$ 625
704-9	$1250
730-1	$ 525
753	$1750
703-9	$1550

ΩMEGA OFFERS THE BEST DELIVERY AND PRICE ON:
APPLE • ATARI • TRS-80 Model II • INTERTEC •
DIABLO • EPSON • HEWLETT—PACKARD • SOROC •
COMMODORE • NEC • QUME • CENTRONICS

CALL TOLL FREE FOR ΩMEGA'S PRICE!
EAST COAST: 1-800-556-7586

ΩMEGA sells only quality merchandise to our customers.
ΩMEGA will try to match any current advertised price with similar purchase conditions.
Before you buy anywhere else — be sure to call ΩMEGA Sales Co.
WEST COAST: 1-805-499-3678 EAST COAST: 1-401-722-1027

ΩMEGA ships via UPS, truck, or air. COD's.
Visa, Mastercharge accepted, with no service charge.

VISA **Telex: 952106** master charge

Professional Software Packages

MISSION VIEJO, CA — CompuSoCo has announced the release of three new professional software packages for the Dentist, Attorney, and consultant. The series of "Professional" packages is designed to utilize the popular desk type computer for appointment scheduling, professional time management, private client billing, and management reporting.

The first new package, called Professional I, is for the Dental Professional. The system features preparation of A.D.A. claims forms for third party patients. The system also allows the professional to locate and prepare notices for professional dental checkups automatically on the schedule the dentist feels advisable for his patients.

The second package, called Professional II, is geared to the needs of the legal profession. The system features preparation of special reports for third party legal plans and special accounting plans to analyze court time usage, and work on retainer or contingency engagements.

The third package variation, called Professional III, is a general purpose package for consultants, accountants and contract administrators. This system allows the creation of sub-jobs, special cost centers, overhead accounts, billing under time and materials contracts, fixed priced job cost accounting and many other job set up systems.

All systems include daily cash reports, time utilization, and professional service reporting. Monthly reports include full aged accounts receivables by client and class of client as well as third party payors. Management and analysis package which is so flexible it can be used to manage personal finances or client trust account funds.

All systems require an Apple II or Apple II Plus computer with Applesoft, a 130 column printer, and at least two mini practitioners with client bases of up to 10,000 clients each.

The systems are available from CompuSoCo at a single site license cost of $750.00 for the selected package. Additional information is available from CompuSoCo, 26251 Via Roble, P.O. Box 2325, Mission Viejo, California 92690.

Hellfire Warrior, Sequel To Temple Of Apshai, Now Available

Automated Simulations, is now offering the sequel to the best-selling Temple of Apshai, Hellfire Warrior.

Like the Temple of Apshai, Hellfire Warrior is a fantasy role-playing adventure, but with more magic, more detail and more command options. Hellfire Warrior lets the player take on the role of his favorite hero.

The player must rescue the beautiful warrior maid Brynhild from the depths of a four-level dungeon and bring her back to sun and air.

Hellfire warrior has more than 200 rooms—riddled with trap doors, bottomless pits, and filled with monsters and treasures, and the player must kill the great bat-winged demon, cross bridges of flame, face death itself and live before the adventure is complete.

Hellfire Warrior is a game for experienced fantasy role-playing gamers. Even more challenging than The Temple of Apshai, Hellfire Warrior allows the player to explore four levels of 60 rooms each.

The magical rooms of level five are inhabited by giant insects. On level six, the player must search for the only exit, hidden within the labyrinth. And on level seven, the player must do battle with skeletons, ghouls, mummies and even invisible ghosts.

The culmination of the adventure lies on level eight. But first the player must overcome the legions of the lost souls in an underworld guarded by dragons and riddled with bottomless pits and blasts of hellfire.

Hellfire Warrior includes an armory where the player must bargain with a tight-fisted innkeeper for five types of armor, five kinds of swords and shields in two sizes. He will also find 13 kinds of potions and healing ointments to choose from. At the Magic Shoppe—if the player has enough money, he can have ordinary weapons transformed into Magical ones.

Hellfire Warrior is available on cassette for the PET (32K) and TRS-80 (Level II, 16K), and on disk for the TRS-80 (32K) and the APPLE (48K with ROM Applesoft) for $39.95 from Automated Simulations, P.O. Box 4247, Mountain View, CA. 94040.

Space Wargame

Strategic Simulations has just released its first space wargame, The Warp Factor. The game allows one or two players to choose from among 12 starship designs representing 5 Galactic Empires. The player(s) are placed squarely in the Captain's role, dealing with the critical parameters of interstellar battle such as energy allocation for phasers, shields, disruptor bolts, screens, and warp engines. With an average game lasting between thirty minutes and four hours, the player(s) can create scenarios ranging from space skirmishes to a full-scale, all-out star war. For $39.95 the game comes complete with a Starship Operating Manual, 3 Starship Data Cards, and a Game Selection Card. The Warp Factor is available on disc for a 48k Apple II (Applesoft ROM).

For more information, please contact Stratetic Simulations Inc., 465 Fairchild Drive, Suite 108, Mountain View, CA 94043. (415)964-1353.

Hooray for SYS (Correction)

Harvey B. Herman
Greensboro, NC 27412

There is a problem with the APPEND programs (Jan. 1981 COMPUTE!) for "old" and "new" PETs. I recently learned that there are four kinds of PET cassette tapes. Unfortunately, in my ignorance, I only tested two types, both of which worked. The third very common PET tape, made with "new" ROMs, was ignored and, in fact, does not work. An easy fix which will cover most, but not all, cases is to change line 230, in both APPEND programs to:

```
230 C = C - 3:T = C + 1 : IFPEEK(635) = 0 THEN
C = C - 1:T = T - 2
```

The programs will now work with the PET tapes which users are most likely to encounter. It may be instructive to discuss the remaining problems in more detail as readers may not be aware of it and could come to grief, as I did.

Both versions of APPEND were designed to work with tapes made on "old" and "new" machines. There is a difference in tapes — original ROMs save starting at hex 400 (dec 1024) and upgrade ROMs save starting at hex 401 (dec 1025). The APPEND programs, as published, checked for start save at statement 230 and made a minor correction depending on which machine was used to make the tape. What I did not know was that new machines saved one byte less on either end. A short program which is written and saved on an "old" machine saves, for example, from hex 400 to hex 424 (call this case 1). The same program, if written and saved on a new machine (call this case 2) would be saved from hex 401 to hex 423 (one less on both ends). If the case 1 tape for the example program, is loaded into a "new" machine and saved, we get a tape which I will call case 3. This tape is a hybrid of cases 1 and 2. Locations saved are from hex 401 to hex 424. My tests for APPEND were done unwittingly with case 1 and case 3 tapes. The line 230 correction discussed above, will allow the program to work with case 2 tapes. Hybrid case 3 tapes will not work but can easily be converted to case 2 after loading by decrementing the location pointer at hex 28 and 29 (dec 42 and 43) and resaving the program. Thus, after loading our short example (case 3 or case 1 tape) change location hex 28 (dec 42) from hex 24 (dec 36) to hex 23 (dec 35) and save again. This new tape (now case 2) and the old one (if case 1) will both append properly. There is also a hybrid case 4 which requires the location pointer on old PETs to be incremented but I think you get the idea.

I want to thank Brien L. Wheeler for calling my attention to a possible error in APPEND and apologize to all readers for this inconvenience.

A REMARKABLE MAGAZINE

creative computing

David Ahl, Founder and Publisher of Creative Computing

"The beat covered by Creative Computing is one of the most important, explosive and fast-changing."—Alvin Toffler

You might think the term "creative computing" is a contradiction. How can something as precise and logical as electronic computing possibly be creative? We think it can be. Consider the way computers are being used to create special effects in movies—image generation, coloring and computer-driven cameras and props. Or an electronic "sketchpad" for your home computer that adds animation, coloring and shading at your direction. How about a computer simulation of an invasion of killer bees with you trying to find a way of keeping them under control?

Beyond Our Dreams

Computers are not creative per se. But the way in which they are used can be highly creative and imaginative. Five years ago when *Creative Computing* magazine first billed itself as "The number 1 magazine of computer applications and software," we had no idea how far that idea would take us. Today, these applications are becoming so broad, so all-encompassing that the computer field will soon include virtually everything!

In light of this generality, we take "application" to mean whatever can be done with computers, *ought* to be done with computers or *might* be done with computers. That is the meat of *Creative Computing*.

Alvin Toffler, author of *Future Shock* and *The Third Wave* says, "I read *Creative Computing* not only for information about how to make the most of my own equipment but to keep an eye on how the whole field is emerging.

Creative Computing, the company as well as the magazine, is uniquely light-hearted but also seriously interested in all aspects of computing. Ours is the magazine of software, graphics, games and simulations for beginners and relaxing professionals. We try to present the new and important ideas of the field in a way that a 14-year old or a Cobol programmer can understand them. Things like text editing, social simulations, control of household devices, animation and graphics, and communications networks.

Understandable Yet Challenging

As the premier magazine for beginners, it is our solemn responsibility to make what we publish comprehensible to the newcomer. That does not mean easy; our readers like to be challenged. It means providing the reader who has no preparation with every possible means to seize the subject matter and make it his own.

However, we don't want the experts in our audience to be bored. So we try to publish articles of interest to beginners and experts at the same time. Ideally, we would like every piece to have instructional or informative content—and some depth—even when communicated humorously or playfully. Thus, our favorite kind of piece is acessible to the beginner, theoretically non-trivial, interesting on more than one level, and perhaps even humorous.

David Gerrold of *Star Trek* fame says, "*Creative Computing* with its unpretentious, down-to-earth lucidity encourages the computer user to have fun. *Creative Computing* makes it possible for me to learn basic programming skills and use the computer better than any other source.

Hard-hitting Evaluations

At *Creative Computing* we obtain new computer systems, peripherals, and software as soon as they are announced. We put them through their paces in our Software Development Center and also in the environment for which they are intended—home, business, laboratory, or school.

Our evaluations are unbiased and accurate. We compared word processing printers and found two losers among highly promoted makes. Conversely, we found one computer had far more than its advertised capability. Of 16 educational packages, only seven offered solid learning value.

When we say unbiased reviews we mean it. More than once, our honesty has cost us an advertiser—temporarily. But we feel that our first obligation is to our readers and that editorial excellence and integrity are our highest goals.

Karl Zinn at the University of Michigan feels we are meeting these goals when he writes. "*Creative Computing* consistently provides value in articles, product reviews and systems comparisons . . . in a magazine that is fun to read."

Order Today

To order your subscription to *Creative Computing*, send $20 for one year (12 issues), $37 for two years (24 issues) or $53 for three years (36 issues). If you prefer, call our toll-free number, **800-631-8112** (in NJ 201-540-0445) to put your subscription on your MasterCard, Visa or American Express card. Canadian and other foreign surface subscriptions are $29 per year, and must be prepaid. We guarantee that you will be completely satisfied or we will refund the entire amount of your subscription.

Join over 80,000 subscribers like Ann Lewin, Director of the Capital Children's Museum who says, "I am very much impressed with *Creative Computing*. It is helping to demystify the computer. Its articles are helpful, humorous and humane. The world needs *Creative Computing*."

creative computing

Attn: Barbie
P.O. Box 789-M
Morristown, NJ 07960
Toll-free **800-631-8112**
(In NJ 201-540-0445)

Writing For COMPUTE!

Robert Lock, Editor/Publisher

We are always seeking good material for publication in **COMPUTE!**. I cannot overstress our interest in material for the beginner; in short (e.g. 1 page or so) programming hints; in material that crosses "machine boundaries". We present a mix of long articles and short ones. Length is not a criteria of success. Frequently our most favored articles have been simple, provocative programs.

Remember The Beginner

Every time an issue of **COMPUTE!** goes out, there are new readers, with new machines, trying to get started with documentation that may or may not meet their needs. That's one of the reasons we stress good solid introductory material. Many of our readers are interested in simple programming assistance and support. Many are interested in useful programs that allow them to get more practical use from their machines.

Guidelines for Potential Authors

Take a look at The Readers Feedback column this time. It's devoted to reader comments on content. Then sit down and write up a brief article describing that program you've been using at home for six months that you think nobody else would be interested in. You might be surprised.

Submitting Articles To COMPUTE!

Manuscripts should be double spaced, typed with both upper and lower case (please!). Program listings should be provided in printer output form as well as machine readable form. If you don't have a printer, that shouldn't stop you from submitting an article. I'm sure your local store or a friend would be more than happy to let you run off a listing for **COMPUTE!** If that isn't feasible, send it anyway. Many excellent articles don't even contain programs.

Address your articles to:

The Editor
COMPUTE! Magazine
P.O. Box 5406
Greensboro, NC 27403 USA

The Follow-Up

We pay for accepted articles based on their number of pages in the magazine. You'll receive payment after the article appears. Thanks to you all for writing for **COMPUTE!**

©

Advertiser's Index

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

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**__________
☐ **Don't yet have one...**
During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

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

Name, Address: ________________________

Charge my: ☐ Visa ☐ MC
Number________________ Expires_____/_____/
3 4 5 6 7 8 9 10 11 12

The Editor's Feedback:

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**__________
☐ **Don't yet have one...**
During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

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

____

Other suggestions:

3 4 5 6 7 8 9 10 11 12

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

My computer is:
☐ **PET** ☐ **APPLE** ☐ **ATARI** ☐ **OSI**
☐ **KIM** ☐ **SYM** ☐ **AIM** ☐ **OTHER**__________
☐ **Don't yet have one...**
During the next year I expect to buy:
☐ **computer** ☐ **printer**
☐ **disk drive** ☐ **other peripherals**

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

Name, Address: ________________________

Charge my: ☐ Visa ☐ MC
Number________________ Expires_____/_____/
3 4 5 6 7 8 9 10 11 12

Place
Stamp
Here

COMPUTE.
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

Place
Stamp
Here

COMPUTE!
Post Office Box 5406
Greensboro, NC 27403

IF YOU CAN WAIT A MINUTE, WE CAN SAVE YOU $1,000.

With the Starwriter™ Daisy Wheel 25 cps printer from C. Itoh.

A business letter, written on a 45 cps word-processing printer, might take about two minutes to print.

With the Starwriter, it might take closer to three.

The typical 45 cps printer retails for about $3,000.

But the Starwriter 25 retails for about $1,895—thus saving you about $1,000.

And therein lies the biggest difference between the Starwriter 25 and the more expensive, daisy wheel printers.

The Starwriter 25 comes complete and ready-to-use, requiring no changes in hardware or software. It uses industry-standard ribbon cartridges, and it's "plug-in" compatible to interface with a wide variety of systems, to help lower system-integration costs.

Using a 96-character wheel, it produces excellent letter-quality printing on three sharp copies with up to 163 columns, and offers the most precise character-placement available, for outstanding print performance.

C. Itoh's warranty;

3 months on parts and labor, supported by one of the best service organizations in the industry.

$1,000 OFF

Leading Edge Products, Inc.,
225 Turnpike Street,
Canton, Massachusetts 02021

Dear Leading Edge:
I'd like to know more about the Starwriter, and how spending a minute can save me a grand. Please send me the name of my nearest dealer.

Name ______
Title ______
Company ______
Street ______
State ______ Zip ______
Phone: Area Code ______
Number ______

LEADING EDGE.

Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Dealer inquiries invited. For immediate delivery from the Leading Edge Inventory Bank™ call toll free 1-800-343-6833
In Massachusetts, call collect (617) 828-8150. Telex 951-624

The Great American Solution Machine.

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

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

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

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

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

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

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

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

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

commodore

COMMODORE