ECLECTIC SYSTEMS CORPORATION
Order TOLL FREE 1-800-527-3135

ECLECTIC SYSTEMS

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

ANNOUNCES

EM© for the SuperPET

EM© is ANSI standard MUMPS

Make your SuperPET Super-Load in EM© and Dramatically Improve your Programming Productivity

EM© is more than just a programming language. It is a well integrated data management system combining with one syntax what other operating systems would call 1) an application programming language; 2) a job control language; 3) a linkeage editor; 4) a database management system; and 5) a communications monitor.

PROGRAM MANAGEMENT:
EM© provides all programming management facilities needed to manage programs and program files. Programs can be created, edited, cataloged and debugged from within EM©. Programs can be as large as disk capacity. A resident algorithm rids memory of least frequently used variables and program modules so that what you need off-disk normally resides in memory.

STRING POWER:
EM© makes string handling easy with its extensive set of string operations and functions. Variable length strings can be used routinely without the obstacles presented by most other programming languages.

PATTERN MATCHING:
EM© can "filter" user input with a useful pattern matching that will result in fewer user or device errors. For example: dates, zip codes and names can be tested for validity with a single statement.

GLOBALS:
EM© obviates the need for traditional read and write operations on secondary storage devices by allowing data elements to be directly referenced as a set of subscripts; all the details of file organization and retrieval are handled by the system.

TIMING:
EM© enables a programmer to associate timing constraints with several operations. This feature allows testing for terminal malfunctions as well as prompting users in time-critical dialogue.

DATA BASE MANAGEMENT:
Sorts and merges are not necessary as EM© automatically stores data in a dynamically allocated balanced tree structure. Random access to any data item requires at most three disk reads.

EM© UNMATCHED IN PROGRAMMING PRODUCTIVITY:
System houses that program in EM© (MUMPS) find that their costs are lower than those of their competitors using other languages. Fewer lines of code are necessary per application. Dimension statements are not required. Subscripts may be alpha, numeric or any legal string. Data types need not be defined and can change freely throughout as EM© can recognize when it is dealing with alpha, numeric, integer or floating-point data types. EM© gives the professional programmer a full set of software tools designed for real-life tasks and problems he consistently encounters in the production and maintenance of application software. EM© adheres rigidly to ANSI MUMPS standards, which make it transportable to larger processors manufactured by DEC, TANDOM, DATA GENERAL, HARRIS and others. Additionally EM© gives the less-experienced programmer the tools to do a professional job on formidable programming applications.

You may order EM© or SuperPET by calling ECLECTIC SYSTEMS toll-free at 1-800-527-3135 from 10 AM to 4 PM CDT Monday through Friday, or you can order by mail using the form below. Texas residents call 1-214-661-1370.

ECLECTIC SYSTEMS CORPORATION
P.O. Box 1166, 16260 Midway Road, Addison, Texas 75001

Here's my order for EM©@ $299 plus $3.75 for shipping and handling (UPS surface unless specified otherwise). Residents of Texas, Louisiana, Oklahoma City and Tulsa, Oklahoma must add applicable taxes.

☐ My certified check or money order is enclosed.
☐ Please charge my VISA # ______________________ or
MasterCard # ______________________ Expiration date ________

Name ______________________ Signature ______________________

Address ______________________

City ______________ State ______________ Zip ______________

```
230 CLOSE0:REM RESETS ST TO 0
240 IFST=64THEN280
250 J=J+1:INPUT#8,A$(J)
260 IFJ<MAXGOTO240
270 PRINT"{DOWN}FILE TOO LARGE":CLOSE8:END
280 CLOSE8
290 K=0:SK=1
300 PRINT"{CLEAR}":K=SK-1
310 K=K+1:IFK>JTHENK=0:GOTO360
320 IFPEEK(216)=1THENSK=K
330 N=K:GOSUB1300:PRINTN$;A$(K)
340 IFPEEK(216)>=20THEN360
350 GOTO310
360 PRINT:PRINT"COMMAND? ";
370 GETCM$:IFCM$=""THENZ=1-Z:PRINTMID$("{RE
    REV}{OFF}",Z+1,1);" {LEFT}";:GOTO3
    70
380 SH=0:CM=ASC(CM$):IF(CMAND127)<>CMTHENSH
    =1:CM$=CHR$(CMAND127)
390 PRINT"{OFF}";CM$
400 IFCM$=CHR$(13)THENPRINT"{CLEAR}":GOTO31
    0
410 IFCM$<>"I"THEN460
420 NS=1:PRINT"{UP}INSERT AFTER WHICH ITEM"
    ;:GOSUB1310:IN=VAL(IN$):IFRTTHEN45
    0
430 J=J+1:FORI=JTOIN+2STEP-1:A$(I)=A$(I-1):
    NEXT
440 PRINT"ENTER NEW LINE:":GOSUB1310:IFRT=0
    THENA$(IN+1)=IN$
450 GOTO300
460 IFCM$="X"THENPRINT"{CLEAR}":GOTO830
470 IFCM$<>"K"THEN510
480 INPUT"{UP}ARE YOU SURE?__{03 LEFT}";X$:
    IFX$<>"Y"THEN290
490 PRINT"{CLEAR}NOW KILLING THE FILE...":P
    RINT#15,"S"+FIL$
500 PRINT FIL$" HAS NOW BEEN SCRATCHED.":GO
    TO1420
510 IFCM$<>"D"THEN590
520 NS=1:IFSH=0THENPRINT"{UP}DELETE WHICH I
    TEM";:GOSUB1310:DL=VAL(IN$):D2=0
530 IFSH=0ANDDL=0THEN300
540 IFSH=1THENINPUT"{UP}DELETE WHICH ITEMS?
    __{03 LEFT}";IN$,D2:DL=VAL(IN$)
550 IFDL=0THEN300
560 Z=1:DI=D2-DL:IFDI<0THENDI=1:Z=0
570 J=J-DI-Z:FORI=DLTOJ:A$(I)=A$(I+DI+Z):NE
    XT
580 GOTO300
590 IFCM$<>"S"THEN640
600 PRINT#15,"S"+FIL$
610 OPEN8,8,8,FIL$+",S,W"
620 FORI=1TOJ:PRINT#8,A$(I);CHR$(13);:NEXT:
    CLOSE8
630 GOTO300
640 IFCM$<>"G"THEN680
650 PRINT"{UP}GOTO WHICH LINE";:NS=1:GOSUB1
    310:IFRTTHEN300
660 L=VAL(IN$):IFL<1ORL>JTHEN640
670 K=L-1:PRINT"{CLEAR}":GOTO310
680 IFCM$<>"R"THEN730
690 PRINT"{UP}REPLACE WHICH ITEM";:NS=1:GOS
    UB1310:R=VAL(IN$):IFRTTHEN300
700 IFR<1ORR>JTHEN690
710 PRINT"ENTER NEW LINE:":GOSUB1310:IFRT=0
    THENA$(R)=IN$
720 GOTO300
730 IFCM$<>"P"THEN800
740 IFSHTHENFR=1:TU=J:GOTO760
750 DN=4
760 INPUT"FROM LINE,TO LINE";FR,TU
770 IF(FR<1ORTU<1)OR(FR>JORTU>J)OR(FR>TU)TH
    EN750
780 OPEN1,DN:FORI=FRTOTU:N=I:GOSUB1300:PRIN
    T#1,N$;A$(I):NEXT:CLOSE1
790 GOTO300
800 IFCM$<>"1"ANDCM$<>"0"THEN840
810 PRINT"{CLEAR}":IFCM$<>D$THEND$=CM$:GOTO
    180
820 GOTO180
830 GOTO1420
840 IFCM$<>"F"THEN920
850 PRINT"{UP}FIND WHAT";:GOSUB1310:S$=IN$:
    PRINT"{CLEAR}":Z=0:IFRTTHEN300
860 FORI=1TOJ:FORK=1TOLEN(A$(I))
870 IFMID$(A$(I),K,LEN(S$))=S$THENN=I:GOSUB
    1300:PRINTN$;A$(I):Z=1:GOTO890
880 NEXTK
890 GETA$:IFA$<>""THENI=J
900 NEXTI:IFZ=0THENPRINT"NOT FOUND."
910 GOTO360
920 IFCM$<>"E"THEN970
930 PRINT"{UP}EDIT WHICH LINE";:NS=1:GOSUB1
    310:L=VAL(IN$):IFRTTHEN300
940 IFL<1ORL>JTHEN930
950 V=1:Z$=CHR$(SH*29):PRINT"? ";Z$;A$(L);"
    {UP}":GOSUB1310:IFRT=0THENA$(L)=IN
    $
960 GOTO300
970 IFCM$="L"THENPRINT"FILE LENGTH:";J;"LIN
    ES.{DOWN}":GOTO1290
980 IFCM$="Q"THEN1430
990 IFCM$<>"A"THEN1020
1000 PRINT"ENTER NEXT LINE:":GOSUB1310:IFRTT
    HEN300
1010 J=J+1:A$(J)=IN$:GOTO300
1020 IFCM$<>"H"THEN1270
1030 UL=PEEK(59468):POKE 59468,12
1040 PRINT"{CLEAR}"TAB(7)"{REV}LIST OF FIXFI
    LE COMMANDS{DOWN}"
1050 PRINT"A = APPEND"
1060 PRINT"D = DELETE"
1070 PRINT"D = (SHIFT) DELETE RANGE"
1080 PRINT"E = EDIT A LINE"
1090 PRINT"F = FIND A STRING"
1100 PRINT"G = GO TO LINE #"
1110 PRINT"I = INSERT LINE"
1120 PRINT"K = KILL (SCRATCH) FILE"
1130 PRINT"L = PRINT LENGTH OF FILE"
1140 PRINT"P = PRINT FILE TO PRINTER"
1150 PRINT"Q = QUICK-FIND"
1160 PRINT"R = REPLACE LINE"
1170 PRINT"S = SAVE FILE (DON'T FORGET!)
1180 PRINT"X = EXIT FIXFILE"
1190 PRINT"Z = SWITCH UPPER/LOWERCASE
1200 PRINT"0 = EDIT A FILE FROM DRIVE 0"
1210 PRINT"1 = EDIT A FILE FROM DRIVE 1
1220 PRINT"{DOWN}HOLD DOWN SHIFT WITH COMMAN
    DS":PRINT"A,E,F,I,Q,R"
1230 PRINT"TO ENTER COMMAS OR COLONS."
1240 PRINT"PRESS A KEY TO CONTINUE..."
1250 GETA$:IFA$=""THEN1250
1260 POKE59468,UL:GOTO300
1270 IFCM$="Z"THENPOKE59468,12+2*(PEEK(59468
    )=12):PRINT"{02 UP}";:GOTO360
1280 PRINT"{REV}TRY{OFF} A,D,E,F,G,I,K,L,P,Q
    ,R,S,X,Z,1 OR 0":PRINT"OR {REV}H{O
    OFF} FOR HELP"
1290 PRINT"{03 UP}COMMAND?     {UP}";:GOTO360

1300 N$="{REV}"+RIGHT$("     "+MID$(STR$(N),2
    ),4)+"{OFF} ":RETURN
1310 RT=0:Z$=CHR$(160+SH*126*(NS=0))
```

```
1320 IFV=1ANDSH=0THENZ$="{RIGHT}":V=0
1330 PRINT"? ";Z$;:POKE205,0:PRINT"{03 LEFT}
     ";
1340 IFNS=0ANDSH=1THENPOKE158,1:POKE623,ASC(
     "{RIGHT}")
1350 INPUTIN$
1360 IFIN$=Z$THENNS=0:RT=1:RETURN
1370 IFSH=0ORNS=1THENNS=0:RETURN
1380 FORI=1TOLEN(IN$):IFMID$(IN$,I,1)<>","TH
     EN1410
1390 REM CHR$(108) IS FAKE COMMA.  SAVEABLE ~
     TO DISK
1400 IN$=LEFT$(IN$,I-1)+CHR$(108)+MID$(IN$,I
     +1)
1410 NEXT:RETURN
1420 CLOSE8:CLOSE15:END
1430 PRINT"{UP}QUICK-FIND WHAT";
1440 FOUND=0:GOSUB1310:IFRTTHEN300
1450 S$=IN$:L=LEN(S$):PRINT"{CLEAR}";
1460 FORI=1TOJ:IFLEFT$(A$(I),L)<>S$THEN1490
1470 N=I:GOSUB1300:PRINTN$;A$(I):FOUND=1
1480 IFPEEK(216)>20THENWAIT158,1:GETA$:PRINT
     "{CLEAR}"
1490 GETA$:IFA$<>""THENI=J
1500 NEXT
1510 IFFOUND=0THENPRINT"NOT FOUND."
1520 GOTO360
```

Program 2. Atari Version

```
100 REM  FIXFILE-File Editor v1.0
110 REM Charles Brannon, 1982
120 REM
130 GRAPHICS 0:SETCOLOR 4,9,4
140 DLIST=PEEK(560)+256*PEEK(561)+4
150 POKE DLIST-1,7+64:POKE DLIST+2,6
160 POSITION 6,0:? "FIXFILE":POSITION 3+
20,0:? "a file editor"
170 DIM TEMP$(36),FILE$(14)
180 MAX=200:REM Maximum number of lines
in the file
190 DIM DISK$(MAX*35),LN(MAX),KEY$(35)
200 OPEN #1,4,0,"K:":POKE 752,1
210 ? :? "(Press |RETURN| alone for dire
ctory.)":?
220 ? :? "Enter Filename - D:
 {12 LEFT}";
230 GOSUB 1340:SETCOLOR 0,10,12
240 IF TEMP$=" " THEN GOSUB 1080:RUN
250 FILE$="D:":FILE$(3)=TEMP$
260 ? :? "{2 UP}Reading ";FILE$,
270 TRAP 280:OPEN #2,4,0,FILE$:GOTO 290
280 CLOSE #2:? :? "{UP}Error - ";PEEK(19
5),,:GOTO 220
290 TRAP 350:LINES=0
300 INPUT #2;TEMP$
310 L=LEN(TEMP$):LN(LINES)=L
320 DISK$(LINES*35+1,LINES*35+L)=TEMP$
330 LINES=LINES+1:IF LINES<MAX THEN 300
340 ? :? "{UP}Error - File too Large!";:
GOTO 0
350 IF PEEK(195)<>136 THEN 280
360 CLOSE #2
370 CURR=0:POKE 82,0:LINES=LINES-1
380 GRAPHICS 0:SETCOLOR 4,9,4:POKE 752,1
:POKE 766,1
390 FOR I=CURR TO CURR+18
400 IF I>LINES THEN POP :GOTO 450
410 TEMP$="000":TEMP$(4-LEN(STR$(I)))=ST
R$(I)
420 FOR J=1 TO 3:TEMP$(J,J)=CHR$(ASC(TEM
P$(J))+128):NEXT J
430 ? TEMP$;" ";DISK$(I*35+1,I*35+LN(I))

440 NEXT I
450 POKE 766,0:POSITION 0,19:? "{40 R}";

460 POKE 703,4
470 ? "{CLEAR}|E|dit |R|eplace |I|nsert
|D|elete |A|ppend |P|rint |W|rite |Q|uit
 |N|ext Page |G|oto |O|ther |START|"
480 ? :? "Which? ";
490 IF PEEK(53279)=6 THEN 1300
500 IF PEEK(764)=255 THEN 490
510 P=PEEK(702):POKE 702,64:GET #1,A:POK
E 702,P
520 IF A=ASC("O") THEN 1450
530 IF A<>ASC("E") THEN 580
540 ? "{CLEAR}|EDIT|: Which Line? ";:GOS
UB 1200:IF N<0 OR N>LINES THEN 540
550 ? "{CLEAR}Enter changes, press |RETU
RN|"
560 ? "    ";DISK$(N*35+1,N*35+LN(N)):?
"{UP}";:GOSUB 1180
570 GOTO 610
580 IF A<>ASC("R") THEN 620
590 ? "{CLEAR}|REPLACE|: Which Line? ";:
GOSUB 1200:IF N<0 OR N>LINES THEN 590
600 ? "Enter new line:":GOSUB 1180
610 LN(N)=LEN(TEMP$):DISK$(N*35+1,N*35+L
N(N))=TEMP$:GOTO 380
620 IF A<>ASC("I") THEN 670
630 ? "{CLEAR}Insert a line at line: ";:
GOSUB 1200:IF N<0 OR N>LINES THEN 630
640 ? "Patience...":LINES=LINES+(LINES<M
AX):DISK$(LINES*35)=" "
650 FOR I=LINES TO N+1 STEP -1:TEMP$=DIS
K$((I-1)*35+1,I*35):DISK$(I*35+1,I*35+35
)=TEMP$:LN(I)=LN(I-1):NEXT I
660 DISK$(N*35+1,N*35+14)="*** INSERT **
*":LN(N)=14:GOTO 380
670 IF A<>ASC("D") THEN 730
680 ? "{CLEAR}Delete which line? ";:GOSU
B 1200:IF N<0 OR N>LINES THEN 680
690 IF N=LINES THEN LINES=LINES-(LINES>0
):GOTO 380
700 ? "Patience...":LINES=LINES-(LINES>0
):DISK$(LINES*35+71)=" "
```

SJB DISTRIBUTORS. THE MOST COMPETITIVE PRICES ON COMMODORE PRODUCTS.

commodore

AUTHORIZED SALES AND SERVICE FOR THE FOLLOWING PRODUCTS —

HARDWARE:	
SuperPET SP9000, 96K	$1,599 (New)
CBM 8032 Computer, 80 Col.	1,069
CBM 8050 Disk Drive, 1 Mb	1,299
CBM 8250 Disk Drive, 2 Mb	1,760 (New)
CBM D9060 Hard Drive, 5 Mb	2,795 (New)
PET 4032 Computer, 40 Col.	950
CBM 4040 Disk Drive, 340K	969
CBM 2031 Disk Drive, 170K	525
CBM Memory Expansion, 64K	399
CBM 4022 Printer, Graphics	595
CBM 8023P Printer, 150 CPS	769
CBM 8300P Printer, Ltr. Qual.	1,750
Commodore 64 Computer, 64K	595 (New)
VIC 20 Computer, 5K	259
VIC Datasette Recorder	69
VIC 1540 Disk Drive, 170K	467 (New)
VIC 1515 Printer, 30 CPS	320
VIC 8K Memory Expansion	50
MONITORS:	
Panasonic CRT, 13" Color	$ 405
Panasonic CRT, 9" B & W	144
PRINTER PAPER:	
9½" x 11" Continuous, Blank 2700 Sheets, Carton	$ 32
SOFTWARE:	
OZZ Data Base System (8050)	$ 289
Silicon Office	995 (New)
Wordcraft 80	289
Wordcraft Ultra	429 (New)
VisiCalc	199
Personal Tax Calculator	59
Dow Jones Portfolio Management System (RS232)	120
WordPro 4+ or 5	299
The Manager	199
Galaxy One	39
Legal Time Accounting	449
I.R.M.A.	380
B.P.I. A/R, G/L, Job Cost, Inventory, Payroll	325/pkg
VIC Games Cartridges	23 ea
VIC Cassette Recreation Pack	44
VIC Cassette Calculation Pack	44
DISKETTES:	
Verbatim #18158, 5¼", SS/DD, Hub Ring, Box of 10	$ 30
3M "Scotch" #744-0, 5¼", SS/SD, Hub Ring, Box of 10	33
Memorex #3481, 5¼", SS/DD, Hub Ring, Box of 10	28

SJB carries many other lines of equipment and products. Call us for a free catalogue.

REGISTER FOR OUR NEW SOFTWARE SUPPORT SERVICE!

ORDER TOLL-FREE

800-527-4893 (Outside Texas) **800-442-1048** (Within Texas)

MASTERCARD, VISA, MONEY ORDER, BANK CHECK, COD (ADD $5) ACCEPTED.

Add 3% Surcharge For Credit Cards.
In Stock Items Shipped Within 48 Hours, F.O.B., Dallas, Texas.
All Products Shipped With Manufacturer's Warranty.

SJB DISTRIBUTORS, INC.
10520 Plano Road
Suite 206
Dallas, Texas 75238
(214) 343-1328

Prices are subject to change without notice.

```
710 FOR I=N TO LINES:TEMP$=DISK$((I+1)*3
5+1,(I+1)*35+35)
720 DISK$(I*35+1,I*35+35)=TEMP$:LN(I)=LN
(I+1):NEXT I:GOTO 380
730 IF A<>ASC("A") THEN 790
740 IF LINES=MAX THEN ? "{CLEAR}Not enou
gh room!":FOR W=1 TO 200:NEXT W:GOTO 470

750 CURR=(LINES-17)*(LINES>18)
760 ? "{CLEAR}Enter line to be appended:
":GOSUB 1180:IF TEMP$=" " THEN 470
770 LINES=LINES+1
780 LN(LINES)=LEN(TEMP$):DISK$(LINES*35+
1,LINES*35+LN(LINES))=TEMP$:GOTO 380
790 IF A<>ASC("P") THEN 870
800 ? "{CLEAR}|PRINT|: Turn on printer,
press |RETURN|":GET #1,A:IF A=27 THEN 47
0
810 TRAP 820:CLOSE #2:OPEN #2,8,0,"P:":G
OTO 830
820 ? "{CLEAR}{BELL}Printer not ready.":
? "{DOWN}Press |RETURN|":GET #1,A:CLOSE
#2:GOTO 470
830 ? "Starting Line?";:GOSUB 1200:S=N
840 ? "Ending Line?";:GOSUB 1200:IF N=0
THEN N=LINES
850 FOR I=S TO N:PRINT #2;DISK$(I*35+1,I
*35+LN(I)):NEXT I
860 LPRINT :CLOSE #2:TRAP 0:GOTO 470
870 IF A<>ASC("W") THEN 970
880 ? "{CLEAR}|WRITE|: Save file to disk
."
890 ? "Use same file name? (Y/N)";
900 GET #1,A:IF A<>ASC("Y") AND A<>ASC("
N") AND A<>27 THEN 900
910 IF A=27 THEN 470
920 IF A=ASC("N") THEN ? "{CLEAR}Enter f
ile name: ";:GOSUB 1340:FILE$="D:":FILE$
(3)=TEMP$
930 ? "{CLEAR}SAVING FILE TO ";FILE$
940 TRAP 950:OPEN #2,8,0,FILE$:GOTO 960
950 ? "{CLEAR}{BELL}Disk Error - ";PEEK(
195):CLOSE #2:? :? "Press |RETURN|":GET
#1,A:GOTO 470
960 FOR I=0 TO LINES:PRINT #2;DISK$(I*35
+1,I*35+LN(I)):NEXT I:CLOSE #2:TRAP 0:GO
TO 470
970 IF A<>ASC("Q") THEN 1030
980 ? "{CLEAR}|QUIT|: Are you sure? (Y/N
)";
990 GET #1,A
1000 IF A=ASC("N") OR A=ASC("n") OR A=27
 THEN 470
1010 IF A=ASC("Y") OR A=ASC("y") THEN GR
APHICS 0:END
1020 GOTO 990
1030 IF A=ASC("N") THEN CURR=CURR+19:IF
CURR>=LINES THEN CURR=0
1040 IF A=ASC("N") THEN 380
1050 IF A<>ASC("G") THEN 470
1060 ? "{CLEAR}Go to which line? ";:GOSU
B 1200:IF N<0 OR N>MAX THEN 1060
1070 CURR=N:GOTO 380
1080 REM |Print directory|
1090 ? "{CLEAR}":POSITION 5,0:? "DIRECTO
RY":POSITION 20,0:? "__________________
"
1100 SETCOLOR 0,INT(16*RND(0)),10
1110 TRAP 1120:OPEN #2,6,0,"D:*.*":GOTO
1130
1120 ? "Can't read directory":GOTO 1160
1130 TRAP 1160:INPUT #2;TEMP$
1140 ? "{TAB}  ";TEMP$:IF PEEK(84)>15 TH
EN POSITION 0,1:? "{DEL LINE}":POSITION
2,15:FOR W=1 TO 100:NEXT W
1150 GOTO 1130
1160 ? :? :? "Press |RETURN| to continue
..."
1170 CLOSE #2:GET #1,A:RETURN
1180 POKE 752,0:? "   ";:INPUT TEMP$:POK
E 752,1:IF TEMP$="" THEN TEMP$=" "
1190 RETURN
1200 REM GET A NUMBER
1210 TEMP$="0":FOR I=1 TO 4
1220 ? "|#|{LEFT}";
1230 GET #1,A:? " {LEFT}";:IF A=27 THEN
POP :GOTO 470
1240 IF A=126 AND I>1 THEN ? CHR$(A);:TE
MP$(I-1)=" ":I=I-1:GOTO 1220
1250 IF A=155 THEN POP :GOTO 1280
1260 IF A<48 OR A>57 OR I=4 THEN 1220
1270 TEMP$(I)=CHR$(A):? CHR$(A);:NEXT I
1280 N=0:P=1:FOR I=LEN(TEMP$) TO 1 STEP
-1:N=N+(ASC(TEMP$(I,I))-48)*P:P=P*10:NEX
T I
1290 ? :RETURN
1300 ? "{CLEAR}START OVER: Are you sure?
(Y/N)";:GET #1,A
1310 IF A=27 OR A=ASC("N") OR A=ASC("n")
 THEN 470
1320 IF A=ASC("Y") OR A=ASC("y") THEN RU
N
1330 GOTO 1300
1340 REM GET FILE NAME
1350 TEMP$=" ":FOR I=1 TO 12
1360 FOR I=1 TO 12
1370 ? "_{LEFT}";:POKE 702,64
1380 IF PEEK(764)=255 THEN SETCOLOR 0,IN
T(16*RND(0)),10+2*INT(3*RND(0)):GOTO 138
0
1390 GET #1,A:? " {LEFT}";:IF A=155 THEN
 POP :GOTO 1430
```

```
1400 IF A=126 AND I>1 THEN TEMP$(I-1)="
":I=I-1:? CHR$(A);:GOTO 1370
1410 IF (A<48 OR (A>57 AND A<65) OR A>90
) AND A<>46 THEN 1370
1420 TEMP$(I)=CHR$(A):? CHR$(A);:NEXT I:
I=I-1
1430 POKE 764,255
1440 RETURN
1450 REM OTHER COMMANDS
1460 REM ADD YOUR COMMANDS HERE
1470 ? "{CLEAR}|L|ength |S|earch |N|ex
t Search"
1480 ? "Any other key--Other commands"
1490 ? "Which?";
1500 P=PEEK(702):POKE 702,64:GET #1,A:PO
KE 702,P
1510 IF A<>ASC("L") THEN 1540
1520 ? "{CLEAR}Length of File: ";LINES+1
;" lines."
1530 ? :? "Press |RETURN|":GET #1,A:GOTO
 470
1540 IF A<>ASC("S") THEN 1770
1550 ? "{CLEAR}|SEARCH|: Enter character
s to search for:"
1560 GOSUB 1180:IF TEMP$=" " THEN 470
1570 KEY$=TEMP$:NX=0
1580 KL=LEN(KEY$)
1590 ? "{CLEAR}By |C|haracter or |L|ine?
";:GET #1,A:IF A=27 THEN 470
1600 TYPE=-1
1610 IF A=ASC("C") OR A=ASC("c") THEN TY
PE=1
1620 IF A=ASC("L") OR A=ASC("l") THEN TY
PE=0
1630 IF TYPE=-1 THEN 1590
1640 ? "{CLEAR}Searching..."
1650 FOR I=NX TO LINES
1660 TEMP$=DISK$(I*35+1,I*35+LN(I)):TEMP
$(LEN(TEMP$)+1)="
                   ":REM 35 SPACES
1670 IF TYPE THEN 1700
1680 IF TEMP$(1,KL)=KEY$ THEN 1750
1690 GOTO 1730
1700 FOR J=1 TO 35-KL
1710 IF TEMP$(J,J+KL-1)=KEY$ THEN 1750
1720 NEXT J
1730 NEXT I
1740 ? "{CLEAR}Not found.":? "{DOWN}Pres
s |RETURN|":GET #1,A:GOTO 470
1750 CURR=I:I=LINES+1:NEXT I
1760 GOTO 380
1770 IF A<>ASC("N") THEN 470
1780 IF LEN(KEY$)<1 THEN 470
1790 NX=CURR+(CURR<LINES)
1800 ? "{CLEAR}|SEARCH| for: "
1810 ? KEY$:GOTO 1580
```

©

This extraordinarily useful Atari Disk technique lets you eliminate DUP.SYS and MEM.SAV from most of your disks and to call a function menu directly from BASIC.

MicroDOS

Dennis Keathley
Webster, TX

It's late at night and you've been working on major revisions to a huge BASIC program for four hours. You finally finish and want to save it back on disk as ULTIMATE.003 (versions 001 and 002 are already on the disk). Upon SAVEing it, you get ERROR 162 (disk full). You think to yourself, "I'll save it as version 001 since that version is out of date anyway." You get another error, ERROR 167 (file locked). "Well I'll just type DOS and unlock the file. After all, MEM.SAV will preserve my program." So you type "DOS" and sit back to twiddle your thumbs for a minute or so. Wait! There's something wrong. There was no MEM.SAV file on the disk. You deleted it because you needed more free sectors. Four hours of work are lost forever.

Save Yourself 88 Sectors And 75 Seconds

Would you like to eliminate the need for DUP.SYS and MEM.SAV on most of your disks?

The Disk Operating System, version 2.0, is a powerful and well-designed piece of software. It also allows you more free RAM for programming use than DOS 1 by moving many of the utility functions into a separate file called DUP.SYS. The Disk Utility Programs are loaded into memory only when requested by typing DOS. The trade-off is that the file DUP.SYS (42 sectors) must be on your disk in drive #1, and it requires 15 to 20 seconds to load into RAM.

Another trade-off is that the BASIC program in RAM will be lost unless you have a MEM.SAV file (45 sectors) on your disk in drive #1, and using MEM.SAV requires almost a minute longer. Eliminating the need for DUP.SYS and MEM.SAV saves you 88 sectors and 75 seconds.

When DOS is booted into RAM at power-on, the portion that remains memory-resident at all times is called the File Management System. The FMS allows you to perform some file manipulation and directory modifications. The most common way to do this is to type DOS which gives you the DOS 2.0 Menu contained in DUP.SYS. Some schemes have been developed to access the RMS functions from BASIC, but they require you to either remember the exact XIO command or to have numerous short programs on your disk that remember them for you (more time and sectors wasted).

Wouldn't it be nice to be able to call a function menu from BASIC without using additional memory, losing your BASIC program, or performing disk I/O?

MicroDOS is a machine language program that is loaded into page six of RAM (Decimal 1536 to 1791), an area conveniently left to the user by the wizards that designed the Atari Operating System. When executed, MicroDOS gives you a menu of FMS functions, which allows you to select the function and desired filename. You may perform as many functions as you wish, and then return to BASIC. And your BASIC program is still there, intact.

Program 1 is the assembly listing. For those of you who do not have an assembler, Program 2 provides a BASIC program to accomplish the same result. If using an assembler, you may designate "D:AUTORUN.SYS" as the binary output file. This will perform automatic loading of MicroDOS when DOS is booted into RAM. If you are presently using an AUTORUN.SYS file, you may append it onto this one using the /A (append) feature of the copy (C) option of the DOS 2.0 menu. The comments in the listings make them self-explanatory.

For ease of use, we "steal" the DOS jump vector, DOSVEC ($0A or decimal 10) and point it to MicroDOS. Perform POKE 10,0 and POKE 11,6. When you type "DOS", BASIC jumps to the address specified in DOSVEC and DOSVEC+1 (low byte, high byte). Now it will jump to MicroDOS instead. One more detail: when SYSTEM RESET is pressed, DOS restores DOSVEC to the original value. To prevent this, POKE 5446,0 and POKE 5450,6. DOS will now reload DOSVEC with $0600 (the start of MicroDOS) when SYSTEM RESET is pressed.

Note that the last six lines of the assembler listing before the .END statement will cause this to be done automatically when the file is loaded. In the BASIC listing, the last 16 numbers perform this. Omit them if you desire and subtract 16 from the 342 in line two. If you should desire to reset DOSVEC so that DUP.SYS may be loaded, POKE 10,159 and POKE 11,23. Now, typing "DOS" will load DUP.SYS, but SYSTEM RESET will cause DOSVEC to point to MicroDOS.

Upon typing "DOS", MicroDOS will display the menu:

LOCK
UNLOCK
DELETE
RENAME

SWEET SIXTEEN

FOR 16K ATARI COMPUTERS

FLIP'N' FILE

FLIP'N'FILE is a new functional and attractive storage system for ATARI 400/800 Computer cartridges. Each desktop unit holds 10 cartridges for easy access.
Cat No. 3790 $29.95

DYSAN
5¼" DISKETTES

Dysan diskettes, long known for their exceptional quality, are ideally suited for use on your Atari computer (or any 5¼" soft sector computer). And with HW's low, low price, it's time to stock up. Dysan #104/1
Cat No.3966 Box of 10 $39.95

SYNCRO
WORDMANIA

Learning is not supposed to be this much fun. In WORDMANIA, one or two players compete against a rapidly burning dynamite fuse in order to unscramble anagrams. If you take too long... KABOOM! Random selection from over 1800 words and auto-adjustment of difficulty levels make WORDMANIA a game you won't soon tire of. Requires BASIC cartridge. CAUTION: This game is extremely addicting!
Cat No.3907 cassette $14.95
Disk version (3908) available requires 24K **19.95**

SYNCRO
ALIEN HELL

Lucky you... you're enslaved by Aliens to work in their crystal mines. Escape is your only hope. You must search the vast underground corridors in order to find the Magic Crystals and escape. Beware! Strange creatures and deadly fireballs could be around the next corner. Alien Hell is a real-time adventure played in high resolution graphics. Requires BASIC cartridge.
Cat No.3780 cassette, jystk. $14.95
Disk version (3781) available-requires 24K **19.95**

DATASOFT
CANYON CLIMBER

You're at the bottom of the Grand Canyon and trying to scale the toughest canyon wall. You must face three difficult challenges before you reach the top. Angry sheep, threatening Indians, and attacking birds all stand in your way.
Cat No.3957 cassette, jystk. $29.95
Cat No.3958 diskette, jystk. 29.95

DATASOFT
SHOOTING GALLERY

Some of the most superb graphics ever created on the Atari Computer. With a limited amount of bullets, you must shoot the elephants, ducks, rabbits, happy & sad faces, and finally the bears. If you can make it to the tenth round, a surprize awaits you.
Cat No. 3959 cassette, jystk. $29.95
Cat No. 3960 diskette, jystk. 29.95

DATASOFT
PACIFIC COAST HWY

PCH is a fast action game in which you must safely cross PCH without getting squashed by the rush hour traffic. Once you survive the highway, you must leap from boat to raft in order to cross the mighty Pacific. One or two may play. Features excellent graphics.
Cat No. 3961 cassette, jystk. $29.95
Cat No. 3962 diskett, jystk. 29.95

APOGEE SOFTWARE
ASTROWARRIORS

A fast action player vs. player game of space combat. ASTROWARRIORS features realistic gravity and orbital mechanics as well as extensive use of Atari graphics and sound. Attack with your Photon missiles. Protect yourself with shields. Maneuver with your thrusters or enter hyperspace to avoid damage or destruction. Up to four players may battle in one of four skill levels and battle modes.
Cat No.3886 cassette, jysth. $29.95
Cat No.3885 diskette, jystk. 29.95

GAMMA SOFTWARE
HOCKEY

HOCKEY is a hi-res, fast action game for two to four players. An offensive player can carry the puck, pass and shoot. A defensive player can steal the puck and intercept passes. An advanced feature of HOCKEY is the inclusion of "Smart" players who perform automatically. In case of a tie after regulation time, there is even sudden-death overtime.
Cat No.3588 cassette, jysth. $29.95
Cat No.3587 diskette, jystk. 29.95

SYNAPSE
CHICKEN

There's trouble in the barnyard! Ma Hen and a pesky old fox battle it out, as you try to save the local chicken population. Seems that the fox has found Ma Hen's eggs, and she is trying to save them from his deadly clutches. The action gets faster and faster as eggs turn into chicks, feathers fly, chickens squawk, and all bedlam breaks loose!
Cat No. 3906 cassette, jystk. $29.95
Cat No.3905 diskette, jystk. 29.95

ANALOG SOFTWARE
RACE IN SPACE

RACE IN SPACE is a very different space game: you must not only avoid the myriad of tiny asteroids about you, but you must contend with comets zooming past, negative universes, and missiles shot by your opponent's ship. Written in machine language, RACE IN SPACE features 128 options, multi-color player/missile graphics & "outer space" sound effects.
Cat No.3934 cassette, jystk(2) $21.95
Cat No.3935 diskette, jystk(2) 24.95

COMPUTER CORE SOFTWARE
YAHTMAN

YAHTMAN is the computerized version of a dice game (of a similar name) which has popular for years. One to four players, school age and up, may complete for the highest score. Invite your friends over and see who can beat YAHTMAN. Requires BASIC cart.
Cat No.3405 cassette, jystk. $19.95

HW ELECTRONICS

WHEN IN SOUTHERN CALIFORNIA, VISIT OUR RETAIL STORES

19511 Business Center Dr.
Northridge, Ca. 91324

2301 Artesia Blvd.
Redondo Beach, Ca. 90277

444 S. Indian Ave.
Palm Springs, Ca. 92262

HOW TO ORDER

WRITE OR PHONE. Pay by check, M/C, Visa, or C.O.D. (add $1.50 for COD.)

SEND ORDERS TO:
HW ELECTRONICS, INC.
19511 BUSINESS CTR. DR.
Dept G7
NORTHRIDGE, CA. 91324

TOLL-FREE USA (800) 423-5387
INSIDE CALIF (213) 886-9200

Offer expires July 31, 1982

Mention this ad and we pay shipping (UPS ground only)

FORMAT
MENU
ADOS
BASIC
Note: Underlined letters are inverse video.

Enter the command you desire (only the first letter is necessary). You will then see:

FN?

Enter your filename and press RETURN. You must enter the "D:" or "D2:" at the beginning of your filename so that FMS will know which drive to access (do *not* use quotes, as in BASIC). The function will be performed and the menu will reappear. Note: if you selected RENAME, the correct filename input is "D1:fn1,fn2". Only one "Dn:" is needed and the comma is required.

If you selected FORMAT, instead of FN? you will see:

FD#

to warn you that the information on the disk will be destroyed. Type "D1",RETURN (or D2,D3, etc.) to format, or anything else to abort formatting.

If you selected MENU, you will see:

D#?

as a prompt. Enter the drive number (single digit) from which you wish to see a disk directory, and it will scroll across the screen. Use CTRL 1 to momentarily halt the scrolling, and CTRL 1 to resume. The directory will stay on the screen until you press RETURN once more.

Making Use Of The 6502 Stack

Selecting ADOS will allow you to load DUP.SYS in the normal way if you have a need to do that (and you shouldn't often!). As a safety feature, to select this function you must type the first *two* letters. If you enter an invalid filename or if FMS is unable to perform the function, MicroDOS will just return to the "›" prompt. To return to BASIC, type "B" (or any other letter that is not a selection letter).

The most difficult part about writing MicroDOS was fitting it into just one *page* (256 bytes) of RAM. Due to this constraint, a Disk Directory option could not be included in page six. In order to add this feature, the MicroDOS selection menu and input buffer had to be moved to the beginning of page one. What? You thought page one of RAM was used by the 6502 microprocessor as a stack area? Well, it is, but the 6502 will never need more than the top one-third. So we will use the bottom one-third.

If you spend a portion of your time programming in BASIC, you will probably find MicroDOS well worth the time spent typing it in. It has saved me much time and frustration.

Program 1. Source Listing

```
0100  .TITLE "MICRO-DOS"
0110 REM COPYRIGHT 1982
0120 REM ALL COMMERCIAL RIGHTS
0130 REM RESERVED BY DENNIS KEATHLEY
0140  .OPT NOEJECT
0150 ;
0160 ;        DEFINE EQUATES
0170 ;
0180  *=$0342
0190 ICCMD *=*+2
0200 ICCMD1=ICCMD+16
0210 ICBAL *=*+1
0220 ICBAL1=ICBAL+16
0230 ICBAH *=*+3
0240 ICBAH1=ICBAH+16
0250 ICBLL *=*+1
0260 ICBLL1=ICBLL+16
0270 ICBLH *=*+1
0280 ICBLH1=ICBLH+16
0290 ICAX1 *=*+1
0300 ICAX11=ICAX1+16
0310 ICAX2 *=*+1
0320 CIOV=$E456
0330 OPEN=3
0340 CLOSE=12
0350 GETC=7
0360 PUTC=11
0370 GETR=5
0380 PUTR=9
0390 EOL=155
0400 CLR=125
0410 TAB=127
0420 ;  ******** MICRODOS SOURCE CODE **
******
0430  *=$0600
0440 ;
0450 ;        DISPLAY MENU ON SCREEN
0460 ;
0470 INIT
0480  LDY #MBUF&255
0490  LDX #EOM-MBUF
0500  LDA #PUTC
0510  JSR IOE2
0520 ;
0530 ;        ACCEPT COMMAND
0540 ;
0550 GRAB
0560  JSR INPUT
0570 ;
0580 ;        EXECUTE COMMAND
0590 ;
0600  JSR IOP
0610  BPL INIT
0620  BMI GRAB If CIOV error, Y>127
```

```
0630 ;
0640 ;        INPUT CONTROL ROUTINE
0650 ;
0660 INPUT
0670 ;        PUT PROMPT ON SCREEN
0680  LDY #QBUF&255
0690  LDX #2
0700  JSR Q
0710 ;        GET COMMAND CHOICE
0720  JSR IN
0730  LDY #6
0740  STY ICAX11 To "open directory" lat
er
0750 LOOP
0760  DEY
0770  BEQ EXIT
0780  LDA ASCII-1,Y
0790  CMP KBUF
0800  BNE LOOP
0810  LDA CMD-1,Y
0820  PHA      Save command #
0830  CMP #$FE
0840  BNE CONT
0850 ;        VERIFY FORMAT COMMAND
0860  LDY #QBUF+5&255
0870  LDX #3
0880  JSR Q
0890  JSR IN
0900  LDA KBUF
0910  CMP #68  Is first letter a "D"?
0920  BNE EXIT No, then leave MICRO-DOS
0930  BEQ RT
0940 CONT
0950 ;        PUT PROMPT ON SCREEN
0960  LDY #QBUF+2&255
0970  LDX #3
0980  JSR Q
0990 ;        GET FILENAME
1000  JSR IN
1010 RT
1020  PLA   Retrieve FMS command
1030  RTS
1040 ;
1050 ;        PROMPT ON SCREEN ROUTINE
1060 ;
1070 Q
1080  LDA #PUTC
1090  JSR IOE
1100  RTS
1110 ;
1120 ;        EDITOR LINE IN ROUTINE
1130 ;
1140 IN
1150  LDY #KBUF&255
1160  LDX #40
1170  LDA #GETR
1180  JSR IOE2
1190  RTS
1200 ;
1210 ;        EXIT AND I/O ROUTINES
1220 ;
1230 EXIT
1240  LDA KBUF
1250  CMP #77 Directory Request?
1260  BNE DOS No, go to DOS
1270  LDY #QBUF+6&255
1280  LDX #3
1290  JSR Q Put D#? prompt on screen
1300  LDY #SBUF+1&255
1310  LDX #1
1320  LDA #GETR
1330  JSR IOE2 Get drive #
1340  LDY #58
1350  STY SBUF+2 Restore colon
1360  LDY #SBUF&255
1370  LDA #OPEN
1380  JSR IOP2 Open directory
1390 AGAIN
1400  LDA #19
1410  STA ICBLL1
1420  LDA #GETR
1430  JSR IOP Get directory entry
1440  BMI THRU Last entry? Go to THRU
1450  LDY #KBUF-1&255 Tab is first chara
cter
1460  LDX #20
1470  LDA #PUTR
1480  JSR IOE2 Put directory entry on sc
reen
1490  BPL AGAIN Go get another entry
1500 THRU
1510  LDA #CLOSE
1520  JSR IOP Close directory
1530  JSR IN  Wait for RETURN pressed
1540  JMP INIT Go to selection menu
1550 ;
1560 DOS
1570  CMP #65 Is DOS desired?
1580  BNE BASIC No, exit MICRODOS
1590  LDA #68
1600  CMP KBUF+1 Additional check for sa
fety
1610  BNE BASIC
1620  JMP 6047 Load DUP.SYS
1630 ;
1640 BASIC
1650  JMP $A04D Warm start entry
1660 ;
1670 IOE
1680  PHA
1690  LDA #QBUF/256
1700  STA ICBAH
```

```
1710  PLA
1720 ENT
1730  STY ICBAL
1740  STX ICBLL
1750  LDX #0
1760  STX ICBLH
1770  STX ICBLH1
1780  BEQ IO
1790 IOE2
1800  PHA
1810  LDA #KBUF/256
1820  STA ICBAH
1830  STA ICBAH1
1840  PLA
1850  BNE ENT
1860 ;
1870 IOP
1880  LDY #KBUF&255
1890 IOP2
1900  STY ICBAL1
1910  LDX #$10
1920 IO
1930  STA ICCMD,X
1940  JSR CIOV
1950  RTS
1960 ;
1970 ;       DEFINE MENU AND BUFFERS
1980 ;
1990 ASCII
2000  .BYTE 76,85,68,82,70
2010 CMD
2020  .BYTE $23,$24,$21,$20,$FE
2030 QBUF
2040  .BYTE 29,">FN?",198,"D#?"
2050 ;       PUT THE REST IN PAGE 1
2060  *=$100
2070 SBUF
2080  .BYTE "D1:*.*",EOL,TAB
2090 KBUF
2100  *=*+29
2110 MBUF
2120  .BYTE CLR,204,"OCK",EOL
2130  .BYTE 213,"NLOCK",EOL
2140  .BYTE 196,"ELETE",EOL
2150  .BYTE 210,"ENAME",EOL
2160  .BYTE 198,"ORMAT",EOL
2170  .BYTE 205,"ENU",EOL
2180  .BYTE 193,196,"OS",EOL
2190  .BYTE 194,"ASIC",EOL
2200 EOM
2210  *=10
2220  .BYTE 0,6
2230  *=5446
2240  .BYTE 0
2250  *=5450
2260  .BYTE 6
2270   END
```

Program 2. BASIC Loader

```
1 OPEN #1,8,0,"D:MICRODOS.OBJ"
2 FOR I=1 TO 342
3 READ A
4 PUT #1,A
5 NEXT I
6 END
10 DATA 255,255,0,6,253,6,160,37,162,50,
169,11,32,209,6,32,19,6,32,221,6,16,239,
48,246,160,245,162,2,32,84,6,32,90
20 DATA 6,160,6,140,90,3,136,240,63,185,
234,6,205,8,1,208,245,185,239,6,72,201,2
54,208,19,160,250,162,3,32,84,6,32
30 DATA 90,6,173,8,1,201,68,208,30,240,1
0,160,247,162,3,32,84,6,32,90,6,104,96,1
69,11,32,186,6,96,160,8,162,40
40 DATA 169,5,32,209,6,96,173,8,1,201,77
,208,62,160,251,162,3,32,84,6,160,1,162,
1,169,5,32,209,6,160,58,140,2
50 DATA 1,160,0,169,3,32,223,6,169,19,14
1,88,3,169,5,32,221,6,48,11,160,7,162,20
,169,9,32,209,6,16,233,169,12
60 DATA 32,221,6,32,90,6,76,0,6,201,65,2
08,10,169,68,205,9,1,208,3,76,159,23,76,
77,160,72,169,6,141,69,3,104
70 DATA 140,68,3,142,72,3,162,0,142,73,3
,142,89,3,240,19,72,169,1,141,69,3,141,8
5,3,104,208,228,160,8,140,84,3
80 DATA 162,16,157,66,3,32,86,228,96,76,
85,68,82,70,35,36,33,32,254,29,62,70,78,
63,198,68,35,63,0,1,7,1,68
90 DATA 49,58,42,46,42,155,127,37,1,86,1
,125,204,79,67,75,155,213,78,76,79,67,75
,155,196,69,76,69,84,69,155,210,69
100 DATA 78,65,77,69,155,198,79,82,77,65
,84,155,205,69,78,85,155,193,196,79,83,1
55,194,65,83,73,67,155,10,0,11,0,0
110 DATA 6,70,21,70,21,0,74,21,74,21,6
```

©

tele-communications
for the Atari 400/800

Send E-mail, download programs and files, chat, buy merchandise, access the Source* and other exciting data bases with the MICROCONNECTION, a quality direct connect modem for your Atari 400/800. The Autodial/Autoanswer MICROCONNECTION automatically dials other computers and answers too! The bussdecoding MICROCONNECTION eliminates the need for the 850 Expansion Interface* and the "obsolete" acoustic coupler. Terminal software included.

Bussdecoding MICROCONNECTION Atari 400/800	$249.00
with Autodial/Autoanswer	$279.00
RS-232 MICROCONNECTION for Atari 400/800	$199.00
with Autodail/Autoanswer	$239.00
TSMART (Smart terminal program)	$ 79.95
TYPER-A (Users MICROCONNECTION to drive serial printers to provide printed output and listings.)	$ 19.95

the microperipheral corporation

2643 151st Pl. N.E., Redmond, WA 98052
(206) 881-7544

*Trademark of Reader's Digest, Inc. and Atari, Inc.

ATARI*

UNLOCK YOUR ATARI

"ATARI HAS THE CAPABILITY– ACE GIVES YOU THE KEY"

RIGHT IN YOUR POCKET!!

16 comprehensive pages

- error codes
- basic commands with abbreviations
- peek and poke locations
- internal codes
- machine language aides
- much, much more!!

Get Yours Now only **$9.95**

(dealer ad space available)

ORDER NOW-CASH, CHARGE OR C.O.D.

ADVANCED COMPUTING ENTERPRISES
5516 ROSEHILL
SHAWNEE, KS. 66216
(913)262-2875 • (913)631-4180

ACE

ATARI IS A REGISTERED TRADEMARK OF ATARI INC.

Put any machine language file on a boot cassette, impossible you say?

not any more!

introducing

NUDMEHI SOFTWARE'S BOOT CASSETTE MAKER for the ATARI® 400/800 computers

- **BOOT CASSETTE MAKER** places ANY dos binary loadable machine language program on a boot cassette.
- **BOOT CASSETTE MAKER** converts even MULTIPLE machine language files to boot file format and then places them on a boot cassette.
- **BOOT CASSETTE MAKER** creates a boot cassette on which all run and initialization jumps are executed.
- **BOOT CASSETTE MAKER** is user oriented, fully documented and written in machine language.

$29.95*

(POSTAGE INCLUDED)

To order, send check or money order along with your name and address to:

NUDMEHI SOFTWARE
495 OLD YORK ROAD, SUITE 425
JENKINTOWN, PENNSYLVANIA 19046
(215) 635-2722

master charge

Our order lines are open 24 hours a day.
MasterCard Gladly Accepted.

* Pennsylvania Residents Please Add 6% Sales Tax
ATARI® is a trademark of ATARI INC.

Dealer Inquiries Invited

The past, present, and predicted future of the VIC. Jim Butterfield, an internationally recognized expert on Commodore Computers, discusses VIC software, add-ons, and VIC's place in the world of computing.

Whither VIC?

Jim Butterfield
Associate Editor

VIC has been around for about a year now. It is catching on? Old hat? Will it survive in the turbulent marketplace?

A Little History

The first VIC units had problems. Some were physical in nature. The unit ran too hot – you could hardly touch the ROM plug-in area. The picture was of poor quality. Radiation emission standards were not met. These were cleaned up in a redesign.

VIC had another problem, too. People didn't perceive it as a computer. There were few programs available, and the only peripheral was cassette tape. Memory was limited, and memory expansion was a distant promise.

And to top it all off, VIC was an unknown quantity. The VIC-20 User Guide – the "friendly computer guide" was brightly written and by far the best user documentation that Commodore had yet put out, but it was limited in scope. A Programmer's Reference Guide was hinted at, but seemed slow in making an appearance.

By late summer of 1981, the hardware problems had been resolved, but the VIC had still not established itself as a mature product. Disk, printer, and modem were still promises. Software was still limited.

VICs sold in quite respectable numbers during the Christmas season. A number of games had materialized, allowing the machine to be nicely demonstrated. The economical price and educational potential seemed to appeal to many buyers.

In early spring of 1982, VIC suddenly started to grow up, and a new class of users developed an interest in the machine, not as a game, but as a computer.

New Hardware

The 1540 Single Drive Floppy Disk unit made a dramatic appearance. It looked good and behaved well. It proved to be compatible with the Commodore 4040 units, allowing easy transfer of programs and data files. Speed isn't up to the PET/CBM standards, but for most applications it's quite usable.

Printers are now available in limited quantities. The style is familiar to the Commodore user, but the price is more economical.

Other devices are on the horizon. An IEEE-488 interface, which allows connection to the PET/CBM range of machines, has been demonstrated. An inexpensive modem is almost ready.

Several sizes of memory expansion are on the shelf, from the tiny 3K to the huge 16K. A motherboard system is promised soon which will allow the expansion of the VIC to a full 32K system. On the full system, BASIC will have about 27,500 bytes free; not quite as much as a full PET, but enough for most of us. The remaining memory isn't available to BASIC, but can be used for high-resolution screens and other special applications.

New hardware is appearing from non-Commodore sources. It looks like there will be two types: some will extend the capabilities of the VIC, making new things possible; and others will compete with existing products, hopefully keeping prices interesting.

New Software

Commodore has now made available three major software support modules: a Monitor to allow machine language program writing and debugging; a Super Expander, to make graphics easy; and a Programmer's Aid to help in the writing and debugging of BASIC programs.

These three modules are not essential to the VIC's operation. You can write machine language without the Monitor; you can draw graphics without the Super Expander; and you can certainly write BASIC without the Programmer's Aid. But a job can be made much easier with their help.

The Super Expander, in particular, seems to be very popular among programmers with artistic inclinations. Computer art, geometric figures, and animated sketches have been enthusiastically produced by users.

The initial games library is being beefed up with new offerings. I have seen several prototype games which are nearing release and they look good – an order of magnitude better than the satisfactory, existing games on plug-in ROM cartridges.

I'm pleased to see good quality programs coming from non-Commodore sources. There's a lot of good stuff already out (see Table 1) and high-quality programs are a vote of confidence in the VIC by the software houses.

New Information

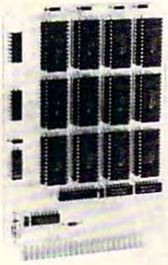
Commodore has now released the *Programmer's Reference Guide*, which contains a great deal of reference information on the VIC. That's the kind

22-40-80 HIKE!

Expand your VIC to 80 columns.

Quantum Data's new Video Combo Cartridge brings you: 40 or 80 column display, plus 16K RAM and PROM socket. $299.00

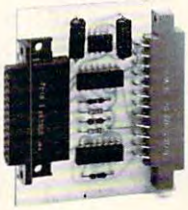
With the Video Combo Cartridge from Quantum Data you can now have 40 or 80 column display, 16K RAM and PROM all in one cartridge. It comes set for 40 column Display compatible with the VIC video modulator and your home T.V. Then, when you are ready to upgrade to 80 columns and a video monitor, just make a simple, no-cost change inside the cartridge. Instructions are provided. Also provided is a socket for a PROM, 16K of memory and AC adaptor. If you don't need memory, then 80 columns can be yours for only $199.50. A listing of the driver software is provided at no charge. A programmed PROM containing this software is also available for $19.95.



QDI expandor:

- Expands Basic user memory up to 24K in 8K steps
- PROMS may be mixed with RAM in 8K blocks
- 8K can be assigned to machine language area
- Plugs directly into VIC expansion port
- Low power, no additional power supply required
- Professional Quality, full buffering on all signals
- Small size: 6 x 4.5 inches.

16K expandor........ $149.95
24K expandor........ $199.95



QDI Printor RS-232 interface:

- Provides RS-232 voltage conversion for VIC serial port
- Allows use of a wide variety of RS-232 peripherals including printers, modems and voice synthesizers
- Low power CMOS circuitry requires no external power supply
- Small size: 2½ x 3 inches

Printor.............. $49.95

QDI Minimother:

- Adds 3 slots to the memory expansion port
- Removable card guides allow either boards or cartridges
- Requires no additional power supply
- Fused to protect VIC power supply from overload
- Simple plug-in installation

Minimother.......... $69.95

VISA and MasterCard accepted.

QUANTUM DATA, INC.
(714) 966-6553
DEALER HOT LINE (714) 754-1945
3001 Redhill Bldg. 4, Suite 105, Costa Mesa, CA 92626

Non-Commodore Software

This is software that I've seen that looks good. There's undoubtedly lots more. All games listed run in the 5K VIC.

Amok – Nicely animated joystick chase game. Similar to the arcade game Berserk.
United Microware Industries, Inc.
3431 H Pomona Blvd.
Pomona, CA 91768

VIC Pics – Many high resolution pictures derived from TV camera. With explanatory booklet. Nice slide show for your friends.
Un-Word Processor – Simple, but does the job. Needs printer, of course.
Midwest Micro Associates
P.O. Box 6148
Kansas City, MO 64110

Snakman – Fast joystick graphics game. Similar to the arcade game Pacman.
Microdigital
752 John Glenn Blvd.
Webster, NY 14580

The Qube – Draws, scrambles, and solves the Rubic's Cube.
Qumax
GRW Laboratories
P.O. Box 17010
Rochester, NY 14617

of resource we need to exploit the capabilities of the machine.

On the learning level, Commodore has released *An Introduction to BASIC*. This is a beginner-level text which comes as a book and two cassette tapes. At first glance, it looks quite attractive.

What Next?

It looks good. Users are now seeing a computer system rather than a toy. There's a new level of interest in the VIC.

Are there clouds on the horizon? Commodore has announced new products – in particular, the Ultimax and the Commodore 64 – which have new capabilities (and some of the new features won't be VIC-compatible). Will VIC be obsolete before it really gest started?

I think not. In their first incarnations, Ultimax and Commodore 64 won't be direct competitors. Ultimax is too small, and the 64 is twice the price. More to the point: it will be some time before these products mature, just as it took the VIC some time to reach its present state.

I suspect that it will be a couple of years before VIC begins to fade away from the retail market. To a buyer interested now, that makes it a worthwhile investment. Less than $300 for a couple of years' computing? A bargain price. Indeed, in this technology-driven era, it would be optimistic to expect any machine to have an economic life that's much longer.

The box isn't the only part of the question, however. If you move on in two years, how compatible will your software be? The Commodore line has been relatively stable for some time. There are occasional grumblings from long-time users (myself included) when Commodore introduces an improved line which calls for adjustments in some specialized programs, but most programs move fairly easily. Commodore's adoption of the "Kernal" standardized program system is likely to help.

Some things aren't likely to transport. Those plug-in games, for example, are in ROM memory, and can't be trimmed up for a new machine. That's OK ... keep the VIC for the games, or delight the kid who eventually buys your VIC by tossing in the games as part of the package.

At a computer club meeting in 1976, I asked a member what type of system he had. He replied, "I'm waiting for prices to drop to zero." I don't know if he's still waiting. But prices are down to the point where obsolescence in a couple of years shouldn't be an important factor in the decision.

Exciting things are happening with the VIC. There are user groups, there are lots of programs, and new hardware is appearing very quickly. It's going to be fun. ©

NEW SOFTWARE

PET CBM

VORTEX™

The enemy is at the center of swirling rings of energy. Your mission – penetrate and destroy.

PHANTOM FIGHTER™

Defend Superstition Mountain from UFO's.

... Each $17.95 on cassette ...

MAGIC CARPET

P.O. Box 35115 Phoenix, AZ 85069

GAMES FOR **VIC-20** ALSO AVAILABLE

VIC-20

TWO GAMES–$7.95

VIC-20 VIC-20

SPACE LANDER*More than just a game. This program presents a "real time" simulation of a spacecraft landing on various heavenly bodies.The computer analyses and scores successful landings. Contains Color, Sound and Graphics.

DOTS WHITE*Nim-like game pits you against the VIC. Flip dots all white and you win. Contains Color, Sound and Graphics.

Send check or money order($7.95 for both games):

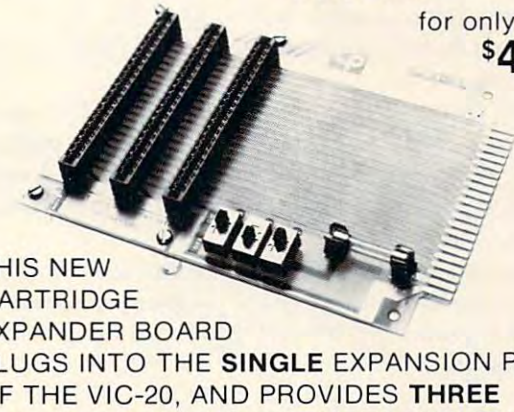
DTC SOFTWARE
P.O. BOX 916
JANESVILLE, WI. 53547

608 752-3434

EXPAND

YOUR VIC-20

for only
$49.95



THIS NEW CARTRIDGE EXPANDER BOARD PLUGS INTO THE **SINGLE** EXPANSION PORT OF THE VIC-20, AND PROVIDES **THREE** SWITCHED EXPANSION PORTS FOR SELECTIVE OR SIMULTANEOUS USE OF ANY VIC-20 CARTRIDGES. ADD MEMORY, PROGRAMMERS AID, SUPER EXPANDER, GAMES, WORD PROCESSOR, OR WHATEVER YOU CHOOSE. SEE YOUR DEALER OR ORDER FROM:

PTI

PRECISION TECHNOLOGY, INC.
COMPUTER PRODUCTS DIV.
2970 RICHARDS ST.
SALT LAKE CITY, UTAH 84115

(801)487-6266

COD
VISA
M/C

VIC-20 SOFTWARE
from MIDWEST MICRO

UN-WORD PROCESSOR ... $14.95
What else do you call a program which does what a word processor does, yet it is just too simple and easy to use? Enter text. Edit ... List ... Save ... just like a program. Send to printer for finished copy. Save paragraphs separately, then merge in any order for printing. Requires only two lines of BASIC to access the M/L routines; over 3K available for text. Can be used with memory expansion. For VIC-1515 or RS-232 printers.*

TERMINAL-40 ... $29.95
NEW, improved TERMINAL-40 program displays 40 uppercase characters per line (3x6 dot matrix) for easier reading. Enables VIC to emulate a standard terminal. Add a BIZCOMP or VIC modem (or RS-232 modem with interface*) and access SOURCE, TELENET, or any of the free Bulletin Boards around the country (list included). 300 baud; full or half duplex; supports control codes; screen dump to printer. Requires VIC-20 and 8K memory expansion.

TERMINAL-22 ($14.95) Same as TERM-40 except 22 character lines and full duplex only. Runs on standard 5K VIC.

GRAFIX MENAGERIE ($11.95). Demonstrate what your $300 miracle can do! Two-program set unleashes VIC's graphics. **SHOWOFF** contains Color Koleidoscope, Arcade Critters, Custom Fonts, Electronics Schematic, and Music Notation. **PLOTTING** uses dot-plot and line-plot routines to make equations perform computer video-art on your screen. Change equation values and create your own interesting patterns. Plot routines may be easily included in your own programs.

BANNER/HEADLINER ($14.95). Two-program set makes GIANT headlines and banners on your printer. **HEADLINER** prints large characters across the page in three sizes. **BANNER** turns the characters sideways, printing continuously down the paper roll. Up to three lines of text, nearly unlimited in length. (How about a ten-foot long "WELCOME HOME"?) For VIC-1515 or RS-232 printers.*

TICKERTAPE ($14.95). Interrupt-driven! Watch your message glide smoothly across the screen in color. Adds motion and interest to any message display. Position on any line, even mix with normal printing. Two built-in character sets: standard and BOLD (or use custom sets from our **LIBRARY VOL. 1**). Message capacity: @ 2K bytes.

LIBRARY VOL. 1 ($12.95). Add style to displays with six full sets of custom character fonts; UPPERCASE, lowercase, numerals, punctuation, etc. Simple to fancy styles. Upper and lower case stored separately; load upper alone to save space ... load both for a full set. May be used with **TICKERTAPE.**

All programs on high quality digital cassette tape

GRAFIX DESIGNER ($14.95). Two-program set helps you design custom graphics characters. **GEN/EDIT** displays an enlarged 8x8 square; move the cursor around in it and turn dots on or off to form a character (holds 100). Erase, edit or recall at random. Load **DATAMAKER** when finished designing. Characters automatically become numbered data statements. Save them on tape just like a program. Instructions included for appending to any new or old program.

Build libraries of graphics ... throw away the graph paper!

VIC-PICS .... digitized pictures! $18.95
Nineteen fascinating high-resolution pictures to display on your VIC screen. Created by digitizing video camera images. Includes portraits, models, scenery, and much more. Over 16K points analyzed in each picture. Three styles: hi-contrast, dithered, and colorized. Compatible with both color and B/W sets.

RS-232 INTERFACE ($49.95). Get more OUT of your VIC. Plug-in interface communicates with most standard serial printers and modems. Simply plug into User Port; needs no external power. Bi-directional operation. 90 day warranty. Full instructions for use. Includes M/L handshake "wedge."

VIC-20 is a trademark of Commodore Business Machines.

Dealer inquiries invited.

*RS-232 printers require an interface. See ours above.

MIDWEST MICRO Associates PO Box 6148 K.C. Mo. 64110

Include $1.25 for postage and handling. Missouri residents add 4.6% sales tax.

*QuadraPET (**COMPUTE!**, June 1981, #13) is a machine language program that lets you "partition" the memory of a 32K PET with Upgrade BASIC into four, 8K areas. A BASIC program can reside in each partition. Each program can be run independently. It's like having four computers in one.*

Super QuadraPET converts QuadraPET to run on 4.0 machines (4032 and 8032) with 32K. Also included is a handy utility program that makes QuadraPET easy to use. It can even relocate itself into one of the partitions – ready to go, but out of the way.

To use QuadraPET 4.0, type in and save Program 1. It places the machine language program in upper memory. Program 1 must be RUN before *you load and RUN Program 2.*

Super QuadraPET

David Sale
Acton, Ontario

Several articles have appeared in **COMPUTE!** dealing with memory partition for the PET computer. I could see the value of this procedure for program development, especially since "Dungeon" could be neatly tucked away in one of the partitions to keep my son happy when I took a break.

Unfortunately, the very useful program by Charles Brannon (**COMPUTE!**, June 1981, #13, p. 102) was written for Upgrade ROMs and will not operate on 4.0 machines. In revising his program, I made several changes:

1) The program has been relocated to the top of memory to leave the cassette buffer available for machine language program development.

2) Calls to BASIC routines are revised for use with 4.0 ROMs.

3) Memory partitions have been changed to make areas more nearly equal in size rather than having the first partition (the most easily accessible) be the smallest.

To create the partitions, load Program 1 and type RUN. Four 8K partitions will be created. By choosing PET #1, you can then load the utility program (Program 2) that will simplify many tasks involved in program development and in "Quadra-PETting." I use the utility program to relocate itself into partition four where it is available when needed to perform the following tasks:

1) Converting decimal numbers to hex and vice versa.

2) Moving programs to and from the first partition.

This is necessary to avoid having to use machine language SAVEs or awkward loading procedures.

3) Printing a directory to show which areas are available for use to avoid accidentally destroying a program.

Moving Around In The Partitions

In relocating programs, the program to be moved is duplicated in the desired partition with all pointers reset to enable it to operate correctly. It is *not* erased from the original area. This makes it possible to revise a program in one area and then compare the results with the original in another.

If you are working in the partition that contains the utility program you can move to another area easily using its routine #6. If you are in another area, you can move using SYS 32513. Once you have moved to a new area, the program contained in it will operate normally. To erase the program or enter a new one, simply type NEW.

One word of caution: the relocation routines will move only BASIC programs. Programs containing machine language must be left in partition one or relocated using "Supermon" (**COMPUTE!**, Dec. 1981, #19, p. 134). If you want to pack "Supermon" into the top of partition four follow this procedure:

1) Load and run QuadraPET.
2) Choose PET #1.
3) POKE 53,127 to temporarily remove the partition.
4) Type NEW.
5) Load and run "Supermon."
6) POKE 53,35 to restore the partition.
7) POKE 32627,121 to protect "Supermon."

If you have saved "Supermon" at the top of memory rather than at 0600 you will need to follow a slightly different procedure:

1) Load "Supermon."
2) Relocate it to 7933 hex using its transfer function.
3) Load QuadraPET and run it choosing PET #1.
4) POKE 32627,121 to protect "Supermon."

"Supermon" is now safely tucked away at the top of memory under the QuadraPET program ready for use with SYS 31027.

No doubt you will discover more uses for these programs once you begin to explore their possibilities.

References:

Brannon, Charles, "QuadraPET: Multitasking On Your PET?," **COMPUTE!**, June 1981, #13, p. 102.

Butterfield, Jim, et al, "Supermon: A Primary Tool For Machine Language Programming," **COMPUTE!**, Dec. 1981, #19, p. 134.

Program 1.

```
10 REM
20 REM  QUADRA PET HIGH MEMORY
30 REM       VERSION 4.0
40 REM     COMPUTE!JUNE 81
50 REM  REVISED BY DAVID SALE
60 REM       NOV.17 1981
70 REM
80 POKE 52,255: POKE 53,127
100 FOR I = 32513 TO 32645
110 READ A
120 POKE I,A
130 NEXT
140 SYS 32586
150 PRINT"{CLEAR}{05 RIGHT}QUADRA PET DIVID
    ES THE AVAILABLE"
160 PRINT"{DOWN}MEMORY INTO FOUR 8K AREAS T
    HAT CAN EACH"
170 PRINT"{DOWN}STORE A SEPARATE PROGRAM."
180 PRINT"{02 DOWN}{05 RIGHT}YOU ARE NOW IN
     PET"PEEK(32611)+1". TO MOVE TO"
190 PRINT"{DOWN}ANOTHER AREA TYPE {REV}SYS ~
    32513{OFF}."
200 PRINT"{02 DOWN}{05 RIGHT}PROGRAMS MAY B
    E LOADED OR SAVED IN"
210 PRINT"{DOWN}THE FIRST AREA IN THE NORMA
    L MANNER."
220 PRINT"{02 DOWN}{05 RIGHT}PROGRAMS IN OT
    HER AREAS MUST EITHER"
230 PRINT"BE SAVED BY MACHINE LANGUAGE OR M
    OVED"
240 PRINT"{DOWN}TO PET #1 USING THE UTILITY
     PROGRAM."
250 END
995 REM  CLEAR END OF STRING
1000 DATA 169,0,141,133,127
1005 REM  STORE PRESENT POINTERS
1010 DATA 174,99,127,165,42,157,104,127,165,
    43,157,108,127
1015 REM  SELECT NEW AREA
1020 DATA 169,116,160,127,32,29,187,32,228,2
    55,41,15,240,249,201,5,176,245
1030 DATA 170,202,142,99,127
1035 REM  SET ALL POINTERS & CLR
1040 DATA 169,1,133,40,189,100,127,133,41,18
    9,104,127,133,42,189,108,127,133
1050 DATA 43,169,0,133,52,189,112,127,133,53
    ,32,238,181,96
1055 REM  INITIALIZE EACH AREA FIRST
1060 DATA 162,0,169,0,24,157,0,4,157,0,35,15
    7,0,66,157,0,97
1070 DATA 232,224,3,144,238,76,1,127
1075 REM  POINTER DATA
1080 DATA 0,4,35,66,97,3,3,3,3,4,35,66,97,35
    ,66,97,127
1085 REM  WHICH PET QUESTION
1090 DATA 147,87,72,73,67,72,32,80,69,84,63,
    32,40,49,45,52,41,0
```

Program 2.

```
100 PRINT"{CLEAR}PET UTILITY PROGRAMS"
110 PRINT"####################"
120 REM  VERSION  JANUARY 10, 1982
130 REM         DAVID SALE
140 REM       ACTON  ONTARIO
150 REM   FOR USE WITH QUADRA PET
160 REM   VERSION 4.0 HIGH MEMORY
170 PRINT"{DOWN}{REV}1{OFF} DECIMAL TO HEX"
180 PRINT"{DOWN}{REV}2{OFF} HEX TO DECIMAL"
190 PRINT"{DOWN}{REV}3{OFF} RELOCATE TO PET
     #1"
200 PRINT"{DOWN}{REV}4{OFF} RELOCATE FROM P
    ET #1"
210 PRINT"{DOWN}{REV}5{OFF} PRINT DIRECTORY
    "
220 PRINT"{DOWN}{REV}6{OFF} MOVE TO ANOTHER
     PET"
230 PRINT"{02 DOWN}TYPE IN THE NUMBER OF YO
    UR CHOICE"
240 GET A:IF A<1 OR A>6 THEN 240
250 ON A GOTO 260,440,590,1040,1490,1670
260 PRINT"{CLEAR}DECIMAL HEX CONVERSION  TY
    PE {REV}M{OFF} FOR MENU"
270 PRINT"#################################
    ######"
280 PRINT"{DOWN}TYPE IN DECIMAL NUMBER (MAX
     65535){DOWN}"
290 INPUT A$:IF A$="M"THEN 100
300 A=VAL(A$)
310 IFA>=16^4THENPRINT"ILLEGAL QUANTITY":GO
    TO280
320 IFA>=16^3THENB(1)=INT(A/(16^3)):A=A-B(1
    )*(16^3):GOTO340
330 B(1)=0
340 IFA>255THENB(2)=INT(A/256):A=A-B(2)*256
    :GOTO360
350 B(2)=0
360 IFA>15THENB(3)=INT(A/16):A=A-B(3)*16:GO
    TO380
370 B(3)=0
380 B(4)=A
390 C$="0123456789ABCDEF"
400 PRINTTAB(8)"{UP}= {REV}";
410 FORX=1TO4:B=B(X)+1:B$(X)=MID$(C$,B,1)
420 PRINTB$(X);:NEXT
430 PRINT"{OFF} HEX":GOTO290
440 PRINT"{CLEAR}HEX TO DECIMAL  TYPE {REV}
    M{OFF} FOR MENU"
450 PRINT"###############################"
460 PRINT"{DOWN}TYPE IN HEX NUMBER (MAX FFF
    F){DOWN}"
470 DC=0:INPUT HX$:IF HX$="M"THEN 100
480 IF LEN(HX$)>4 THEN 530
490 IF LEN(HX$)<4 THEN TM$="000"+HX$:HX$=RI
    GHT$(TM$,4)
500 FOR I=1 TO 4:IN=ASC(MID$(HX$,I,1))
510 IF IN>47 AND IN<58 THEN IN=IN-48:GOTO 5
    40
520 IF IN>64 AND IN<71 THEN IN=IN-55:GOTO 5
    40
530 PRINT"ILLEGAL QUANTITY":GOTO 460
540 DC=DC+IN*(16^(4-I))
550 NEXT
560 TM$="     "+STR$(DC):DC$=RIGHT$(TM$,5)
570 PRINT"{UP}"TAB(8)"= {REV}"DC$TAB(16)"{O
    OFF}DECIMAL"
```

```
580 GOTO 470
590 PRINT"{CLEAR}RELOCATION PROGRAM TO PET ~
    #1"
600 PRINT"############################"
610 IF PEEK(1026)=0 THEN 670
620 PRINT"{DOWN}PET #1 CONTAINS A PROGRAM"
630 PRINT"DO YOU WANT TO DESTROY IT ?"
640 GET R$:IF R$="" THEN 640
650 IF R$<>"Y" THEN 100
660 PRINT TAB(30)"{UP}{REV}YES{OFF}"
670 PRINT"{DOWN}WHICH PET HAS PROGRAM TO BE
    MOVED ?"
680 GET N$:IF N$="" THEN 680
690 NO=VAL(N$):IF NO<2 OR NO>4 THEN 680
700 PRINT TAB(37)"{UP}{REV}"NO
710 PRINT"{DOWN}ARE YOU SURE ?"
720 GET R$:IF R$="" THEN 720
730 IF R$<>"Y"THEN 100
740 PRINT TAB(17)"{UP}{REV}YES{OFF}"
750 PRINT"{DOWN}TRANSFERING CONTENTS OF PET
    #"NO
760 PRINT"INTO PET #1"
770 REM  SAVE POINTERS IF IN PET NO
780 IF PEEK(32611)<>NO-1 THEN 810
790 POKE 32616+NO,(PEEK(42))
800 POKE 32620+NO,(PEEK(43))
810 REM  CORRECT POINTERS IN PET #1
820 NO=NO-1:V1=PEEK(32616+NO)
830 POKE 32616,V1:  REM  VAR. LOW
840 V2=PEEK(32620+NO)-31*(NO)
850 POKE 32620,V2:  REM  VAR. HIGH
860 MH=PEEK(32624+NO)-31*(NO)
870 POKE 32624,MH:  REM  MEMORY LIMIT
880 SU=NO*7936:     REM  SUBTR. FACTOR
890 SA=SU+1025:     REM  START ADDRESS
900 REM  CORRECT POINTER TO NEXT ADDRESS
910 PA=PEEK(SA)+256*PEEK(SA+1)
920 BY=PEEK(SA):POKE(SA-SU),BY
930 SA=SA+1:BY=PEEK(SA):POKE(SA-SU),(BY-SU/
    256)
940 REM  TRANSFER REST OF LINE
950 SA=SA+1:FOR A=SA TO (PA-1)
960 BY=PEEK(A):POKE(A-SU),BY
970 NEXT
980 REM  CHECK FOR END OF BASIC
990 IF PEEK(PA+1)<>0 THEN SA=PA:GOTO 910
1000 POKE PA-SU,0:POKE PA+1-SU,0
1010 PRINT"{DOWN}PROGRAM TRANSFERRED"
1020 FOR A=1 TO 3000: NEXT
1030 GOTO 100
1040 PRINT"{CLEAR}RELOCATE FROM PET #1"
1050 PRINT"####################"
1060 PRINT"{DOWN}TO WHICH PET ? (2-4)"
1070 GET N$:IF N$="" THEN 1070
1080 NO=VAL(N$):IF NO<2 OR NO>4 THEN 1070
1090 PRINT TAB(22)"{UP}{REV}"NO
1100 PRINT"{DOWN}ARE YOU SURE ?"
1110 GET R$:IF R$="" THEN 1110
1120 IF R$<>"Y" THEN 100
1130 PRINT TAB(16)"{UP}{REV}YES{OFF}"
1140 IF PEEK(1026+(NO-1)*7936)=0 THEN 1200
1150 PRINT"{DOWN}PET"NO"CONTAINS A PROGRAM"
1160 PRINT"DO YOU WANT TO DESTROY IT ?"
1170 GET R$:IF R$=""THEN 1170
1180 IF R$<>"Y"THEN 100
1190 PRINT TAB(30)"{UP}{REV}YES{OFF}"
1200 PRINT"{DOWN}TRANSFERING PROGRAM TO PET ~
    #"NO
1210 PRINT"FROM PET #1"
1220 REM  SAVE POINTERS IF IN PET 1
1230 IF PEEK(32611)<>0 THEN 1260
1240 POKE 32616,(PEEK(42))
1250 POKE 32620,(PEEK(43))
1260 REM  CORRECT POINTERS IN PET NO
1270 NO=NO-1:V1=PEEK(32616)
1280 POKE 32616+NO,V1:  REM  VAR. LOW
1290 V2=PEEK(32620)+31*(NO)
1300 POKE 32620+NO,V2:  REM  VAR. HIGH
1310 MH=PEEK(32624)+31*(NO)
1320 POKE 32624+NO,MH:  REM  MEM. LIMIT
1330 AD=NO*7936:        REM  ADD. FACTOR
1340 SA=1025:           REM  START ADDRESS
1350 REM  CORRECT POINTER TO NEXT ADDRESS
1360 PA=PEEK(SA)+256*PEEK(SA+1)
1370 BY=PEEK(SA):POKE(SA+AD),BY
1380 SA=SA+1:BY=PEEK(SA):POKE(SA+AD),(BY+AD/
    256)
1390 REM  TRANSFER REST OF LINE
1400 SA=SA+1:FOR A=SA TO (PA-1)
1410 BY=PEEK(A):POKE(A+AD),BY
1420 NEXT
1430 REM  CHECK FOR END OF BASIC
1440 IF PEEK(PA+1)<>0 THEN SA=PA:GOTO 1360
1450 POKE PA+AD,0:POKE PA+1+AD,0
1460 PRINT"{DOWN}PROGRAM TRANSFERRED"
1470 FORA=1TO3000:NEXT
1480 GOTO 100
1490 PRINT"{CLEAR}DIRECTORY FOR PROGRAMS IN ~
    PARTITIONS"
1500 PRINT"#################################
    ###"
1510 FOR A=1 TO 4
1520 PRINT:PRINT"{DOWN}PET #"A
1530 PRINT"########"
1540 READ B
1550 IF PEEK(B)=0 THEN PRINT "EMPTY";:GOTO 1
    610
1560 PRINT "CONTAINS PROGRAM";
1610 NEXT
1620 DATA 1025,8961,16897,24833
1630 RESTORE
1640 PRINT:PRINT"{02 DOWN}PRESS {REV}M{OFF} ~
    TO RETURN TO MENU"
1650 GET R$:IF R$=""THEN 1650
1660 GOTO 100
1670 SYS 32513
1680 PRINT"{CLEAR}"TAB(3)"{02 DOWN}YOU ARE N
    OW IN PET #" PEEK(32611)+1 "{LEFT}
    ."
1690 PRINTTAB(3)"{02 DOWN}ANY PROGRAM IN THI
    S PARTITION IS"
1700 PRINT"{DOWN}UNDISTURBED AND MAY BE RUN ~
    OR MODIFIED."
1710 PRINTTAB(3)"{02 DOWN}TO ENTER A NEW PRO
    GRAM FIRST TYPE {REV}NEW{OFF}"
1720 PRINT"THEN PROCEED IN THE NORMAL FASHIO
    N."
1730 PRINTTAB(3)"{02 DOWN}TO MOVE TO ANOTHER
     PARTITION TYPE"
1740 PRINT"{DOWN}{REV}SYS 32513{OFF}. THE PR
    OGRAM IN THIS AREA"
1750 PRINT"{DOWN}WILL BE PROTECTED UNTIL NEE
    DED LATER."
```

©

A disaster-prevention technique for Apple Disk systems. Teachers should find this especially valuable. For DOS 3.3 with 48K RAM.

Apple DOS Changer

Robert Swirsky
Cedarhurst, NY

If there is an Apple in a classroom, there is usually a student who gets immense pleasure out of typing "INIT" and watching the drive whir and click as it erases the disk!

There is a strange phenomenon which occurs when you mix microcomputers and young people. Most of them will treat the computer with reasonable care, but there is always someone who feels compelled to try to "mess up" the computer's files.

I know from experience; I used to be such a student. What can be done to stop people from tampering with dangerous DOS commands? The answer is simple – just *change* the names of commands that you don't want available to users. Since the DOS is stored in RAM, it is relatively easy to find the table of commands, and modify it accordingly.

This program will change the DOS commands. When run, it displays each command followed by a prompt. If you want to change the command, just type in your revision. To leave it unmodified, hit the RETURN key. The next DOS command will be displayed, and the same procedure is followed.

A few words of caution are needed. First of all, don't make the new commands so that they match the beginnings of other commands. For example, if you change "CATALOG" to "C", conflicts will arise if you try to use the CLR, CLOSE, or other commands that begin with C. Second, don't put spaces in your command. This tends to make strange things happen. Also, avoid making your commands excessively long. There is only so much room in memory for the command table. Note, however, that the commands can be of different lengths from the original ones.

How To HAHA A Disk

The program will work on DOS 3.3 with 48K of RAM. It will not work with less than 48K, without changing the addresses that the changes are POKEd to. Since the vast majority of Apple owners have the 48K, this is hardly a limitation.

After the program is RUN, the changes can be saved as a permanent part of DOS. Insert a *blank* disk, and type the INIT command, followed by the correct parameters. Of course, if you have changed the INIT *itself* to something else, you would type that in instead. When the initialization is completed, the disk will have your DOS modifications stored on it. Whenever DOS is booted from that disk, your custom commands will be the ones that the Apple knows.

A great deal of protection is offered with this program. If you decide to change INIT to HAHA, nobody can init the disk. Anyone who tries will be greeted with a ?SYNTAX ERROR (beep). However, when you go to HAHA your disk, you will be successful.

```
0 ONERR  GOTO 1000
1 DIM CO$(28)
2 P = 1
10 FOR X = 43140 TO 43271
20 CO$(P) = CO$(P) + CHR$ ( PEEK (X))
31 IF  PEEK (X) > 128 THEN P = P + 1
32 IF P = 28 THEN 70
40 NEXT
70 HOME
80 PRINT "DOS CHANGER BY ROBERT A. SWIRSKY
   "
160 VTAB 24: PRINT "HIT 'C' TO CONTINUE";: ~
    GET R$: IF R$ < > "C" THEN 160
170 HOME
180 PRINT : PRINT "YOU WILL SEE ALL 28 DOS ~
    COMMANDS"
190 PRINT "IF YOU WISH TO LEAVE IT UNCHANGE
    D,       PRESS RETURN"
200 PRINT "TO CHANGE THE COMMAND, SIMPLY RE
    TYPE IT WITH YOUR CHOICE."
210 PRINT : PRINT
220 LL = 43139
221 HL = 43271
230 FOR Q = 1 TO 28
235 PRINT CO$(Q);
240 PRINT  TAB( 20);" --> ";
250 INPUT R$
260 IF R$ = "" THEN R$ = CO$(Q)
261 IF  LEN (R$) = 1 THEN  PRINT "PLEASE --
     TWO OR MORE LETTERS": GOTO 240
270 FOR W = 1 TO LEN (R$) -1
280 POKE LL + W, ASC ( MID$ (R$,W,1))
290 NEXT
295 IFASC(RIGHT$(R$,1))>127THENPOKELL+W,ASC
    (RIGHT$(R$,1)):GOTO 501
300 POKE LL + W,ASC(RIGHT$(R$,1))+128
501 REM
510 LL=LL+W
520 IF LL > HL THEN 600
530 W=0
599 NEXT
600 PRINT "CHARACTER LIMIT EXCEEDED"
1000 PRINT "RUN TERMINATED"
```

©

A Monthly Column

This month, columnist Michael Day tackles some of the practical aspects of adding a modem to your computer. A modem will let your computer talk to other computers over the telephone lines. If you're shopping for a modem, you'll need to answer some questions first. Will you want the computer to be able to "pick up the phone" by itself and answer incoming calls automatically? To place its own calls *automatically? How can you use a modem with a "Princess" phone? Will you need an extra interface to connect the modem to your computer or will it just plug in?*

Telecommunications:

Choosing A Modem

Michael E. Day
Chief Engineer, Edge Technology

There are a number of different types of modems on the market. In order to understand each modem's capability in each application, we need to break them down into different groups.

One aspect to consider is the way the modem attaches to the phone line. There are three basic methods:

1. Acoustic coupler: a modem that communicates through a telephone by sound.
2. Inductive coupler: a modem that communicates through a telephone by electromagnetic induction.
3. Direct connect: a device which allows a modem to connect directly to the phone line instead of using the telephone to communicate through.

The acoustic coupler and direct connect are the most commonly used methods. The inductive coupler is actually a modification of the acoustic coupler, and generally still requires that the transmitter portion of the modem be acoustically coupled. The usually stated advantage of the inductive coupler is its avoidance of the distortion and interference caused by the extra translation to sound and back occuring in the acoustic coupler.

This is of little help, however, since the majority of problems with acoustic coupling involve the transmitter portion, which needs to be acoustically coupled anyway. The inductive coupler also has its own set of problems which can often make it more difficult to work with if care has not been exercised in its design.

Each telephone has its own quirks with regard to inductive coupling, and this can sometimes make it difficult to get a modem to work properly. Also, because of the sensitivity involved in properly picking up the weak electromagnetic signals, the inductive coupler is more sensitive to stray magnetic fields, fields generated by a computer or terminal can interfere with the operation of the modem.

Portability Of The Acoustic Coupler

The acoustic coupler modem has been around for some time. The advantage of the acoustic coupler is its portability. It can be used with any telephone that has a standard handset (referred to as a *500 style phone*). This means that you are not limited as to the electrical connections to the phone line, only to the physical construction of the phone handset, which must fit into the rubber or plastic "cups" of the acoustic coupler.

This can be a problem if you have a princess or flip phone or some other style which does not use the standard 500 series handset. Other problems with the acoustic coupler are a result of the acoustic coupling itself. Because of the signal's conversion from electrical to acoustic and back to electrical again, some amount of distortion is generated. This can be held to a minimum with proper modem design, but cannot be eliminated entirely. The telephone itself is a major contributor to this problem. A great portion of the distortion can be cured simply by replacing the carbon microphone in the telephone handset with a condensor (capacitor) microphone.

The standard acoustic coupler used to communicate over the telephone is referred to as a 103 compatible modem. This is a full duplex type modem – that is, it can send and receive data at the same time. Because of the method used to encode and decode the data to be transmitted by this type of modem, the maximum communication speed that can be used reliably is 300bps (30 characters per second).

This limitation is the result of many things, including the bandwidth of the phone line, the encoding scheme, and the signal distortion caused by the acoustic coupling. Although it is possible to push the speed a bit higher, reliability will suffer. A very good modem might let you push it to 400 or 450 bps, but, because of acoustic distortion, this is about the maximum.

Another problem with the acoustic coupler is its limited sensitivity. Most are not able to operate at a receive level of less than -30db to -36db. This is equivalent to a whisper. About 90% of the phone calls placed will provide a communications link allowing a signal through stronger than this. Therefore, this is not a great problem. However, the strength of the transmitted signal is of major importance, since the weaker the transmitted signal is, the weaker the received signal at the other end is going to be. Generally, it is desirable to have a transmitted signal strength of between -6db and -10db. This is equivalent to loud talking.

Direct Connect Method

The other method of attaching a modem to the phone line is the direct connect method. Here you attach the modem directly to the phone line, rather than acoustically through a telephone handset. This eliminates the problem of distortion by not going through the translation to sound and back.

There are many types of direct connect modems. The type that most computer users need is the 103 compatible type. It uses the same conversion scheme to communicate as the 103 compatible acoustic coupler, although the translation to sound and back is, of course, not necessary.

The main advantages of the direct connect modem are reduced signal distortion and improved sensitivity, (since the sound translation is eliminated). Some direct connect modems are able to communicate at up to 600bps (60 characters per second). However, some reduction in reliability should be expected at these high speeds. Other modems are designed so that they cannot operate at speeds above 300bps – in order to achieve a higher degree of reliability at the normal speeds.

Another attractive aspect of the direct connect modem is that it allows the computer to access the phone line directly to answer, and even make, calls *by itself*. This cannot easily be done with the acoustic coupler because it goes through a phone which must be physically lifted off a receiver to start dialing. The disadvantage of a direct connect modem, however, is that it must be physically attached to the phone line. This can be a problem with a business phone setup, pay phone, etc. As a result, the direct connect method is usually used in a fixed location where the phone line is easily attached. Often a phone line is solely for the use of a computer.

Three Types Of Direct Connect Modems

The three main types of direct connect modems are the "auto answer," "auto answer/manual originate," and "auto answer/auto originate."

An auto answer modem can answer calls, but is unable to place any. An auto answer/manual originate modem can answer calls as well as allowing you to manually make a call (after which control is transferred to the modem). Finally, an auto answer/auto originate modem lets the computer answer calls automatically as well as place them by itself without operator intervention.

While most acoustic couplers are separate devices that attach to a computer or terminal through some sort of interface (usually an RS232 serial interface), a few are built into the equipment. If the acoustic coupler is to be purchased as a separate device, be sure to find out if the separate interface will be needed and include it in the cost of the modem. Often a computer already has an RS232 port available to which to attach the modem. If this is the case, the additional interface may not be needed.

Direct connect modems come as either a separate unit that attaches to the computer, or as a device that can be mounted inside the computer. If it is a separate external unit, the interface must be taken into account. Generally, modems which are designed to be installed inside the computer provide their own interface to the computer, and the extra cost need not be considered.

Although the final decision must be made according to your own needs, a general rule of thumb is that if you have no need for automatic operations, an acoustic coupler will probably serve best. If, however, you do desire automatic functions, then you will need the direct connect type modem. ©

In this, the first of a two-part series on the inner workings of Atari Graphics, the author reviews the computer's system of screen management and defines several important terms including color clock, playfield, mode line, *and* display list. *Next month, the article concludes with techniques for using* color indirection, *a powerful graphics tool, and explores the new GTIA chip in detail. This new chip costs nothing if your Atari is still under warranty. If you have an older machine, the nearest authorized service center should be stocking it by now and will install it for about $60.*

Part I:

Atari Video Graphics And The New GTIA

Craig Chamberlain
Birmingham, MI

The GTIA is an exciting new graphics chip now being shipped in Atari 400/800 computers. Among its special features are a sixteen color mode with a resolution eight times better than the Apple's, and the capability of generating 256 color variations. The GTIA chip provides three new graphics modes in addition to the normal fourteen, totally different, full-screen modes. This article defines a few terms relating to graphics, explains the normal graphics modes, then introduces the new modes provided by the GTIA.

ANTIC Is A Busy Chip

We all know that the Atari 400 and 800 have superior graphics capabilities. This has been achieved by designing special chips to handle video display tasks, taking that burden off the main microprocessor. In Atari computers these special chips are known as ANTIC and CTIA.

The ANTIC chip is actually an advanced DMA (direct memory access) controller that qualifies as a true microprocessor. It has an instruction set (mode lines and "load memory scan" operation), a program (the good 'ole display list), and data (display memory and character sets).

This special chip is a rather busy fellow. Its responsibilities include doing DMA for the display list, the display data (playfields), the character set, and player/missile buffers. Besides that, it sets the playfield width, controls horizontal and vertical fine scrolling, keeps track of the vertical position of the scan beam, and handles NMI interrupts. It also supports a light pen.

The GTIA: Three New Modes

The other chip is the CTIA, or Computer Television Interface Adapted integrated circuit. This is the chip which handles all color and luminance (brightness) information to send to the television screen. This is a complicated process, but the chip designers at Atari got carried away and created whole new functions which we know as the player/missile graphics system. It is the CTIA which processes the horizontal position, size, priority, and color of the players. The CTIA also watches for player/playfield collisions, joystick triggers, and console keys. Like the ANTIC, it is a busy chip.

The new GTIA chip replaces the CTIA. Rumor has it that the "G" stands for George. Apparently some fellow named George was still not satisfied with all the special functions of the CTIA, and gave it the ability to generate three totally new graphics modes. When you find out what these new modes can do, I think you will appreciate "George" and his GTIA.

The three new modes are 9, 10 and 11. The operating system and, therefore, Atari BASIC, supports these new modes. But before describing all the features of these new modes, I want to define a few essential terms and review the normal graphics modes 0 through 8.

In order to fully understand Atari graphics, one must have a solid concept of how a television display is generated. And no discussion on "television theory" would be complete without a definition of the "color clock." The term *color clock* derives from the fact that there is a problem in measuring distances on a television screen. Different television sets have different screen sizes, with 9", 13" and 19" being common diagonal measurements. All television sets, however, have a scanning beam which translates a signal from the computer into a picture on the screen.

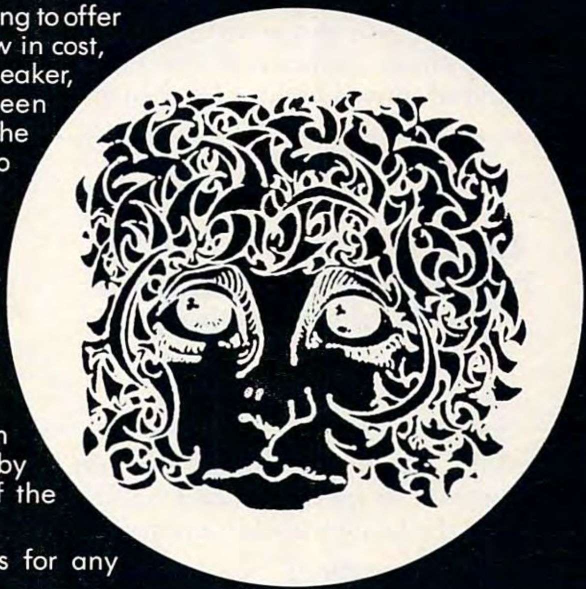
The signal coming from the computer contains two characteristics. It has a frequency, which defines a color, and it has an amplitude, which defines the *luminance* of that color, often referred to as the brightness or intensity. These qualities of the computer signal affect the way in which the scanning beam shoots electrons at the phosphors on a television screen. This electron shooting process is done horizontally, one line at a time, but it is done so quickly that it is not noticeable to the human eye.

When drawing a line, the scanning beam starts at the left edge of the screen and proceeds to the right edge, shooting electrons the whole time. Since the beam has a finite amount of time it can

ATARI SAYS ITS FIRST WORD

WITH A VOICE BOX BY THE ALIEN GROUP!

THE ALIEN GROUP has emerged from the underground, daring to offer a full-featured speech synthesis system that is flexible, low in cost, and needs no accessory devices. No Interface, Cables, Speaker, Amp, or External Power Required! **VOICE BOX** has been designed and programmed by Atari users to become the integral voice of a *400* or *800* computer. Simply plugged into the serial port, **VOICE BOX** automatically routes all speech into the speaker of your television monitor. With the menu-driven operating system supplied, you'll be creating original, intelligible speech within moments after loading disk or cassette. No lengthy or obscure instructions to wade through.



The system includes a dictionary which translates typed text into **VOICE BOX's** phonetic language. The dictionary can be expanded to include as many as *5,000* words of your own custom vocabulary. Unlimited speech can be produced by straightforward phonetic definiton at any time, even if the dictionary should be full.

The speech routines can be called from other programs for any purpose you can conceive. Here are a few suggested uses:

VOICE BOX
Speech Synthesizer

SOUND EFFECTS
- Access 64 phonemes at any of 4 pitch levels to add filtered, contoured sound to the Atari audio repertoire.

GAMES
- Program aliens to hiss threats, moan when destroyed.
- Devise weird, non-human tongues for dungeon dwellers.
- Insert cryptic spoken clues in maze games.

COMPUTER OPERATION
- Code verbal prompts and error messages that command attention and leave the current display intact.

EDUCATION
- Gain an introduction to the principles of phonetics.
- Learn touch typing through spoken feedback from the keyboard.

In addition, the *Random Sentence Generator* included in the operating system, which prints and speaks endlessly startling, amusing, even poetic combinations of words supplied by the user, helps teach school children to identify parts of speech and recognize a variety of sentence structures.

A minimum of 16K RAM is required by the operating system. Either disk or cassette includes both 16K and 32K versions. Try **VOICE BOX** for up to 10 days, and if it isn't the finest value you've ever seen in a computer peripheral, the most challenging and provocative addition you've ever made to your system, return it in its original condition for a full refund.

When ordering specify disk- or cassette-based operating system, and enclose check or money-order for $169, or state your **VISA** or **MASTERCARD** number. Send mail orders to: **THE ALIEN GROUP**
27 West 23rd Street
Dept. PT-1
New York, New York 10010

or telephone orders from
10 AM to 6 PM New York time
(212) 924-5546

ALSO AVAILABLE AT LEADING COMPUTER STORES THROUGHOUT THE WORLD.

spend drawing one line, the beam will seemingly have to move faster to cover more area on a larger screen. Thus the problem of trying to measure horizontal distances is further complicated by the fact that different scanning beams not only travel different amounts, but also at different rates. Our unit of measurement cannot really be a distance; it must be a unit of time. The hint I gave a moment ago was that the scanning beam has a certain amount of time it can spend on one scan line. How fast or how far the beam travels is insignificant.

The ANTIC knows how to handle fourteen different kinds of mode lines.

Understanding Color Clock

The fact that our unit of measurement is based on time explains the word *clock* in the term *color clock*. A color clock is the amount of time the computer needs in order to sufficiently change the frequency of the signal it generates so as to produce a different color. What a mouthful! This is my own personal definition; it has worked for me, but some people may not agree with it. Here's another definition. A scan line is the horizontal path of the scanning beam from the left edge of the screen to the right edge.

Scan lines extend horizontally across the screen, but it takes a lot of them stacked vertically to fill up the screen from top to bottom. Therefore, horizontal resolution is usually expressed in terms of color clocks while vertical resolution is expressed in scan lines. Of course, on different television sets the actual lengths will differ, but the resolution horizontally to vertically is always proportionate. It turns out that, on any screen, one color clock appears to be equal in length to two scan lines.

Now we have to get even more technical for a moment. The scanning beam starts at the upper left corner of the screen and travels horizontally to the right. By the time it hits the right edge it has drawn one scan line that is 228 color clocks wide. The team then shuts off for a short period while it returns to the left edge, only one scan line lower. This period is called the "horizontal blank" for obvious reasons. The beam then turns on again and starts drawing the next scan line. This sequence of drawing scan lines continues 262 times. At that point, the scanning beam, at the lower right corner of the screen, shuts off and returns to the upper left corner of the screen during a period known as the (guess what!) "vertical blank."

This whole process of drawing 262 scan lines, each of 228 color clocks, plus the blanking periods, constitutes one "frame." The television draws sixty of these frames every second, because your home power line is 60 Hz (cycles). The name given to this display method is "raster scan." The fact that your Atari follows a broadcast standard referred to as "NTSC" makes it one of the few home computers that can be video-taped without special equipment.

Just because the scanning beam generates all those scan lines and color clocks doesn't mean that the computer is generating that much display data. Even if the computer did, you wouldn't see the whole image since most television sets display a little less than 200 scan lines of about 170 color clocks. The part where the true picture exists is called the playfield, and now it's time for another definition.

Playfields And Mode Lines

The playfield is the portion of each scan line for which data read from memory can produce colors and luminances. The background exists at the ends of each scan line; the playfield is in the middle. From the viewpoint of one frame, the playfield appears as a rectangular region which extends to the sides of the screen.

Two things control the size of this playfield area. The height in scan line is controlled by the display list as you will see in a moment. Recall that the width in color clocks is set by the DMA control register of the ANTIC.

SDMCTL	**$022F**	**559**	**shadow**
DMACTL	**$D400**	**54272**	**hardware**

D5		**1 display list DMA enable**
		0 display list DMA disable
D1,D0	**00**	**playfield DMA disable (no playfield)**
	01	**narrow playfield (128 color clocks)**
	10	**standard playfield (160 color clocks)**
	11	**wide playfield (192 color clocks)**

The OS screen handler always uses a standard width playfield. The advantage of the narrow playfield is that less DMA is required, so programs execute faster. Unfortunately, the screen handler routines do not work properly when the playfield width is other than the standard. The wide playfield generates more data than the television can display; its uses are rather limited. It's even possible to turn off playfield completely, in which case ANTIC fills the screen with scan lines of the background color. As will be shown in a moment, the playfield also requires a "display list" so bit five must be set for any playfield type to be generated.

Remember that a byte is made up of eight

binary "bits." If playfield and display list DMA is enabled, bits may be read from the computer memory during the course of one scan line. The bit pattern determines the frequency and intensity changes of the scanning beam, with the result being different color/luminances. The same bit pattern may be repeated for several scan lines. And the bit pattern can be interpreted in different ways. This leads us to yet another definition.

A mode line is a contiguous group of scan lines for which display memory is read only once.

There are two main types of mode lines. In direct memory map modes, the bit pattern produces the same image on each scan line. Text modes are a more complicated mode type which use a character set.

The ANTIC knows how to handle fourteen different kinds of mode lines. Each mode line corresponds to a different method for interpreting a bit pattern. A full screen graphics mode is actually just a series of identical mode lines.

The display list is merely a sequence of bytes in memory that, among other things, tells ANTIC the proper sequence of mode lines for one screen.

Whenever the screen is opened (accomplished in Atari BASIC with the GRAPHICS statement), the screen handler establishes a display list of many mode lines to produce a screen of the desired mode. Modes can be mixed by manually changing the display list. Display lists produced by the screen handler always contain the proper number of mode lines for exactly 192 scan lines of playfield. Altering the display list can affect the total number of scan lines, which is how the vertical size of the playfield is controlled.

The display list also has other functions, such as control of fine scrolling, horizontal blank interrupts, and loading the memory scan counter of the ANTIC so it knows where to start reading memory.

A mode line divided into several parts forms pixels, which are single plotting points somewhere within the playfield area. A pixel's vertical resolution is the same as the mode line in which it is displayed, so there can be just as many pixels vertically as mode lines in the display list. The number of color clocks over which one pixel is spread is also determined by the mode line. Here is a little chart to show you the pixel size for the primary mapping modes:

MODE	COLOR CLOCKS	SCAN LINES	RESOLUTION (full/split screens)
3	4	8	40 by 24/20
4,5	2	4	80 by 48/40
6,7	1	2	160 by 96/80

Note that each time the width of a pixel is reduced, its height also decreases, so a single pixel appears to be square in shape regardless of the graphics mode.

Some Observations About Memory

Now to talk about memory. In the one-color modes, one pixel is represented in memory by one bit. If the bit is on, playfield zero shows. If the bit is off, the background shows. Modes 4 and 6 are the one-color modes. For more color, modes 3, 5 and 7 allow three colors. The tradeoff is that a single bit is no longer sufficient. Two bits, a pair, are required. The total value of the two bits selects either one of three playfields or the background:

BIT PATTERN	COLOR	PLAYFIELD TYPE
00	0	background
01	1	playfield zero
10	2	playfield one
11	3	playfield two

Playfield zero is the same thing as COLOR 1 in Atari BASIC. Playfield one is really COLOR 2, and so on, with COLOR 0 being the background.

Although modes 4 and 5 both have the same resolution, or pixel size on the screen, mode 5 will require twice as much memory. In the lower resolution modes which require little memory in the first place, the additional memory needed is rather insignificant. You might have noticed that mode 3 had no single color counterpart. Consider that in a 48K system it is possible to have about 150 different mode 3 screens in memory simultaneously. The chip designers probably decided it wasn't worth the effort or memory savings to provide a one color mode with such low resolution.

Therefore, the size of a pixel on the screen is determined by two things: how many scan lines high, and how many color clocks wide. The amount of memory required for a mode is also determined by two things: how many separate pixels to one mode line, and how many color possibilities per pixel. The only real connection between pixel size on the screen and size in memory is that bigger pixels fill up a screen faster, so there are fewer of them, and less memory is needed.

Now, three colors means two bits must be used. Does that mean we are always stuck with only three colors which can't be changed? No. The CTIA is capable of generating 128 color/luminance variations. It can produce sixteen different colors, each in eight different degrees of luminance. But 128 possibilities means seven bits would be required, and, in most cases, seven bits per pixel is simply not feasible. There is a limit to how much memory can be devoted to a screen. The solution to this problem is a sort of compromise, but it also presents some powerful and flexible advantages, too. The solution is to use *color indirection*. ©

All About PET/CBM Character Sets

Louis F. Sander
Pittsburgh, PA

Commodore's unique assortment of graphics characters, combined with their numerous ROM sets and keyboard configurations, make the various PET and CBM character sets maddeningly hard to comprehend. Occasional inaccuracies in published documentation have confused the situation even more. But as of today, the mystery is over – this article describes all the PET/CBM character sets in specific detail, and shows how they relate to each other and to the standard ASCII character set used by many other manufacturers. Such information will be useful to any PET/CBM owner wanting to get past the beginning stages of programming, and will be invaluable to anyone using the IEEE bus or the user port to communicate with a non-Commodore device.

First, some definitions. Many computer devices can display a group of symbols, or *characters*, on paper or on a CRT. The symbols so displayed are called *printing characters*, and they consist of letters, numbers, punctuation marks, special characters, etc.

Within a given piece of hardware, each character is represented by a pattern of bits, which can be stored, manipulated, and transmitted electrically. The binary numbers corresponding to these bit patterns are called *character codes*. In the PET and CBM, all character codes are 8-bit binary numbers, and they are usually referred to by their decimal equivalents.

For example, the code for a PRINTed asterisk (*) is 0010 1010 binary, or 42 decimal. Eight bits allow 256 different codes, which can be represented as decimal numbers in the range 0-255 inclusive. A given code can represent different characters in different machines, or even within one machine, depending on context. In the PET/CBM, for example, different codes are used to put a given character on the screen by PRINTing or by POKEing.

Some character codes do not represent a *printed* character at all. Instead, they instruct the hardware to take a certain non-printing action. These codes are called *control codes* or *control characters*. RETURN, CURSOR DOWN, and RVS are some familiar PET/CBM control actions. If you have ever made your machine do a RETURN by executing the statement PRINT CHR$(13), you have used a control code (the 13) to generate a control action (the RETURN).

A device's *character set* is its complete set of printing and control characters, along with their associated codes. Many computer devices use a standard character set called *ASCII*, pronounced *ask-ee*, which stands for American Standard Code for Information Interchange.

ASCII and the PET/CBM character sets have quite a bit in common, but there are large differences between them which have to be resolved whenever a PET/CBM is to communicate with an ASCII device. The information in this article will allow you to resolve these differences quickly and accurately in your own programs.

The Printed Set

Now let's look in depth at the PET/CBM character sets. To keep things simple, we'll first investigate the *printed character set*, or the complete set of symbols that PET/CBM can display on its screen. The Character Set Demo program will allow us to do just that. Type it in and RUN it right now, being sure to include the semicolon at the end of line 210. If you have an 80-column machine, you need to substitute line 310 for line 200.

If everything has been entered properly, you'll see 256 evenly-spaced characters on the screen. You'll also see the notation "59468 = 12" (or 14), indicating the current contents of memory location 59468. Press any key several times, and observe that the notation alternates between 12 and 14, and, as it does, some of the displayed characters alternate as well. As you press a key, the demo program is changing the contents of 59468, and PET/CBM is changing certain printed characters as that happens. No character codes are being altered at all.

We are demonstrating that every PET/CBM has *two* sets of printing characters. A given character code will produce characters from one set or the other, depending on a number POKEd into 59468. A 12 in that location produces what is often called the "standard" set of printing characters. It is the same in all PET/CBMs, and we will call it Character Set S, for "standard." POKEing 59468 with a 14 produces what is often called the "alternate" character set. This nomenclature is ambiguous, because there are two *different* alternate character sets. Which one you have depends on the ROM set installed in your machine. In this article, we'll call the alternate character set installed in the Original

TABLE 1 – SCREEN POKE CHARACTER SETS FOR PET/CBM

POKEs shown are made to locations 32768-33767 for 40-column screens, and 32768-34767 for 80-column screens. Character set selection: S - POKE 59468,12 in all machines. AO - POKE 59468,14 in PETs with Original ROMs. A - POKE 59468,14 in all other machines.

POKE	S	A	AO	POKE	S	A	AO	POKE	S	A	AO	POKE	S	A	AO
0	@	@	@	32	Space			64	─	─	─	96	Sh Space		
1	A	a	A	33	!	!	!	65	♠	A	a	97	▌	▌	▌
2	B	b	B	34	"	"	"	66	│	B	b	98	▄	▄	▄
3	C	c	C	35	#	#	#	67	─	C	c	99	▔	▔	▔
4	D	d	D	36	$	$	$	68	─	D	d	100	▁	▁	▁
5	E	e	E	37	%	%	%	69	─	E	e	101	▏	▏	▏
6	F	f	F	38	&	&	&	70	─	F	f	102	▒	▒	▒
7	G	g	G	39	'	'	'	71	│	G	g	103	▕	▕	▕
8	H	h	H	40	(	(	(	72	│	H	h	104	▄	▄	▄
9	I	i	I	41	)	)	)	73	╮	I	i	105	◤	▨	▨
10	J	j	J	42	*	*	*	74	╰	J	j	106	▕	▕	▕
11	K	k	K	43	+	+	+	75	╯	K	k	107	├	├	├
12	L	l	L	44	,	,	,	76	└	L	l	108	▗	▗	▗
13	M	m	M	45	-	-	-	77	╲	M	m	109	└	└	└
14	N	n	N	46	.	.	.	78	╱	N	n	110	┐	┐	┐
15	O	o	O	47	/	/	/	79	┌	O	o	111	▂	▂	▂
16	P	p	P	48	0	0	0	80	┐	P	p	112	┌	┌	┌
17	Q	q	Q	49	1	1	1	81	●	Q	q	113	┴	┴	┴
18	R	r	R	50	2	2	2	82	_	R	r	114	┬	┬	┬
19	S	s	S	51	3	3	3	83	♥	S	s	115	┤	┤	┤
20	T	t	T	52	4	4	4	84	│	T	t	116	▎	▎	▎
21	U	u	U	53	5	5	5	85	╭	U	u	117	▍	▍	▍
22	V	v	V	54	6	6	6	86	╳	V	v	118	▐	▐	▐
23	W	w	W	55	7	7	7	87	○	W	w	119	▔	▔	▔
24	X	x	X	56	8	8	8	88	♣	X	x	120	▀	▀	▀
25	Y	y	Y	57	9	9	9	89	│	Y	y	121	▃	▃	▃
26	Z	z	Z	58	:	:	:	90	♦	Z	z	122	┘	✓	✓
27	[	[	[	59	;	;	;	91	┼	┼	┼	123	▖	▖	▖
28	\	\	\	60	<	<	<	92	▒	▒	▒	124	▝	▝	▝
29	]	]	]	61	=	=	=	93	│	│	│	125	┘	┘	┘
30	↑	↑	↑	62	>	>	>	94	π	▒	▒	126	▘	▘	▘
31	←	←	←	63	?	?	?	95	◥	▧	▧	127	▚	▚	▚

POKE	S	A	AO	POKE	S	A	AO	POKE	S	A	AO	POKE	S	A	AO
128	[illegible]	[illegible]	[illegible]	160	█	█	█	192	[illegible]	[illegible]	[illegible]	224	[illegible]	[illegible]	[illegible]
129	[illegible]	[illegible]	[illegible]	161	[illegible]	[illegible]	[illegible]	193	[illegible]	[illegible]	[illegible]	225	[illegible]	[illegible]	[illegible]
130	[illegible]	[illegible]	[illegible]	162	[illegible]	[illegible]	[illegible]	194	[illegible]	[illegible]	[illegible]	226	[illegible]	[illegible]	[illegible]
131	[illegible]	[illegible]	[illegible]	163	[illegible]	[illegible]	[illegible]	195	[illegible]	[illegible]	[illegible]	227	[illegible]	[illegible]	[illegible]
132	[illegible]	[illegible]	[illegible]	164	[illegible]	[illegible]	[illegible]	196	[illegible]	[illegible]	[illegible]	228	[illegible]	[illegible]	[illegible]
133	[illegible]	[illegible]	[illegible]	165	[illegible]	[illegible]	[illegible]	197	[illegible]	[illegible]	[illegible]	229	[illegible]	[illegible]	[illegible]
134	[illegible]	[illegible]	[illegible]	166	[illegible]	[illegible]	[illegible]	198	[illegible]	[illegible]	[illegible]	230	[illegible]	[illegible]	[illegible]
135	[illegible]	[illegible]	[illegible]	167	[illegible]	[illegible]	[illegible]	199	[illegible]	[illegible]	[illegible]	231	[illegible]	[illegible]	[illegible]
136	[illegible]	[illegible]	[illegible]	168	[illegible]	[illegible]	[illegible]	200	[illegible]	[illegible]	[illegible]	232	[illegible]	[illegible]	[illegible]
137	[illegible]	[illegible]	[illegible]	169	[illegible]	[illegible]	[illegible]	201	[illegible]	[illegible]	[illegible]	233	[illegible]	[illegible]	[illegible]
138	[illegible]	[illegible]	[illegible]	170	[illegible]	[illegible]	[illegible]	202	[illegible]	[illegible]	[illegible]	234	[illegible]	[illegible]	[illegible]
139	[illegible]	[illegible]	[illegible]	171	[illegible]	[illegible]	[illegible]	203	[illegible]	[illegible]	[illegible]	235	[illegible]	[illegible]	[illegible]
140	[illegible]	[illegible]	[illegible]	172	[illegible]	[illegible]	[illegible]	204	[illegible]	[illegible]	[illegible]	236	[illegible]	[illegible]	[illegible]
141	[illegible]	[illegible]	[illegible]	173	[illegible]	[illegible]	[illegible]	205	[illegible]	[illegible]	[illegible]	237	[illegible]	[illegible]	[illegible]
142	[illegible]	[illegible]	[illegible]	174	[illegible]	[illegible]	[illegible]	206	[illegible]	[illegible]	[illegible]	238	[illegible]	[illegible]	[illegible]
143	[illegible]	[illegible]	[illegible]	175	[illegible]	[illegible]	[illegible]	207	[illegible]	[illegible]	[illegible]	239	[illegible]	[illegible]	[illegible]
144	[illegible]	[illegible]	[illegible]	176	[illegible]	[illegible]	[illegible]	208	[illegible]	[illegible]	[illegible]	240	[illegible]	[illegible]	[illegible]
145	[illegible]	[illegible]	[illegible]	177	[illegible]	[illegible]	[illegible]	209	[illegible]	[illegible]	[illegible]	241	[illegible]	[illegible]	[illegible]
146	[illegible]	[illegible]	[illegible]	178	[illegible]	[illegible]	[illegible]	210	[illegible]	[illegible]	[illegible]	242	[illegible]	[illegible]	[illegible]
147	[illegible]	[illegible]	[illegible]	179	[illegible]	[illegible]	[illegible]	211	[illegible]	[illegible]	[illegible]	243	[illegible]	[illegible]	[illegible]
148	[illegible]	[illegible]	[illegible]	180	[illegible]	[illegible]	[illegible]	212	[illegible]	[illegible]	[illegible]	244	[illegible]	[illegible]	[illegible]
149	[illegible]	[illegible]	[illegible]	181	[illegible]	[illegible]	[illegible]	213	[illegible]	[illegible]	[illegible]	245	[illegible]	[illegible]	[illegible]
150	[illegible]	[illegible]	[illegible]	182	[illegible]	[illegible]	[illegible]	214	[illegible]	[illegible]	[illegible]	246	[illegible]	[illegible]	[illegible]
151	[illegible]	[illegible]	[illegible]	183	[illegible]	[illegible]	[illegible]	215	[illegible]	[illegible]	[illegible]	247	[illegible]	[illegible]	[illegible]
152	[illegible]	[illegible]	[illegible]	184	[illegible]	[illegible]	[illegible]	216	[illegible]	[illegible]	[illegible]	248	[illegible]	[illegible]	[illegible]
153	[illegible]	[illegible]	[illegible]	185	[illegible]	[illegible]	[illegible]	217	[illegible]	[illegible]	[illegible]	249	[illegible]	[illegible]	[illegible]
154	[illegible]	[illegible]	[illegible]	186	[illegible]	[illegible]	[illegible]	218	[illegible]	[illegible]	[illegible]	250	[illegible]	[illegible]	[illegible]
155	[illegible]	[illegible]	[illegible]	187	[illegible]	[illegible]	[illegible]	219	[illegible]	[illegible]	[illegible]	251	[illegible]	[illegible]	[illegible]
156	[illegible]	[illegible]	[illegible]	188	[illegible]	[illegible]	[illegible]	220	[illegible]	[illegible]	[illegible]	252	[illegible]	[illegible]	[illegible]
157	[illegible]	[illegible]	[illegible]	189	[illegible]	[illegible]	[illegible]	221	[illegible]	[illegible]	[illegible]	253	[illegible]	[illegible]	[illegible]
158	[illegible]	[illegible]	[illegible]	190	[illegible]	[illegible]	[illegible]	222	[illegible]	[illegible]	[illegible]	254	[illegible]	[illegible]	[illegible]
159	[illegible]	[illegible]	[illegible]	191	[illegible]	[illegible]	[illegible]	223	[illegible]	[illegible]	[illegible]	255	[illegible]	[illegible]	[illegible]

BATTERIES INCLUDED

village by the grange, 71 mccaul st. (f6) toronto m5t 2x1 telephone 596-1405

ARBITER 1.4 MULTI-USER DISK SYSTEM FOR COMMODORE 4.0 COMPUTERS

OVER THREE HUNDRED IN USE ACROSS ONTARIO

Since September 1981 **BATTERIES INCLUDED** has been installing the ARBITER system in classrooms of Commodore BASIC 4.0 computers. The computers are connected to CBM Disk Drives and printers. All users have access to all disk drives and printers plus a host of commands to make this system configuration really usable!

THE ARBITER 1.4 SYSTEM IS READY TO GO!

FEATURES

1) Easy installation.
2) Uses no RAM or Utility Sockets.
3) Up to 32 computers in one system.
4) System self initializes on power up.
5) Operation is completely transparent to the user.
6) Extended commands allow a friendly multi-user environment.
7) System design virtually eliminates interleaved printer output.

SPECIAL COMMANDS

@S – Allows students to protect files with a five character password. A three character user ID is forced into the file name.

@L – Allows the students to load protected files if the password code is known.

LISTC – Used to produce program listings with a Commodore printer. Clumsy OPEN, CMD, LIST, PRINT#, CLOSE sequence not needed. It overcomes the listing problems found on other multi-user hardware systems.

LISTP – Used to get program listings on systems which have an ASCII printer. The cursor control characters are expanded and displayed in brackets. e.g. ‹home›

ALL FILE TYPES ARE SUPPORTED – During relative or sequential file access a delay has been built in so the computer will retain control of the system until the file is closed.

TEACHER UTILITY – A utility is supplied on disk to allow the teacher to produce a hardcopy listing and output from any of the protected or unprotected files selected. Once the files are chosen from the disk directory the teacher may do other tasks while the job is completed.

IF YOUR CLASSROOM WAS DESIGNED TO TEACH COMPUTER LITERACY OR STRUCTURED BASIC THEN THIS SYSTEM WAS DESIGNED FOR YOU.

Arbiter and Arbiter 1.4 are copyrights of Batteries Included.

NEW PRODUCT

SWARM-100

JUST PLUG IT IN

• No soldering • No messy wires

SOFTWARE SELECTABLE

1. Software select one of two operating systems. (BASIC 2.0 / BASIC 4.0)
2. Software select utility ROMs at conflicting addresses.

$150.00

Master Charge and VISA accepted.

For 24 Pin ROM Machines Only.

BATTERIES INCLUDED

Village by the Grange
71 McCaul Street
Toronto, Ontario
Canada M5T 2X1
(416) 596-1405

There are over 500 reasons to own this reference encyclopedia.

Here's one of them.

Programming the PET/CBM — -31- — 4: Effective BASIC

```
x   Input and validate item to be searched for (say, K$ = key item).
    N1 and N2 set to current low and high record numbers
    R = INT((N1+N2)/2)                               :REM CALCULATE NEW MID-POINT
y   Read the appropriate field of record no. R; say R$
•   IF R$=K$ GOTO z                                  :REM FOUND IT!
    IF N1>=N2 THEN PRINT "RECORD NOT ON FILE": GOTO x:REM NON-EXISTENT
    IF R$>K$ THEN N2=R-1: GOTO y                     :REM REVISE UPPER LIMIT DOWN
    N1=R+1: GOTO y                                   :REM REVISE LOWER LIMIT UP
z   Continue processing the record
```

This schematic program of the binary chop search is, I hope, self-explanatory. N1 and N2 converge, sandwiching the correct value of R between them. Note that records needn't be disk-based; they could as easily be a sorted array in RAM, in which case the test line would read IF R$(R)=K$ GOTO z. Try out this technique before implementing a large system, generating test-data with a program, and timing the result. It may be too slow, depending on the disk system and size of file.

4.1.14 Sorting is an important operation in commercial data processing. (COBOL has a SORT verb). Chapter 5 has a collection of routines, mostly in BASIC, with notes. The first example, the 'tournament' sort, is unlike all the others in computing individual results singly, so that results can be printed continually, before all the values are ordered. Most sorts wait until the entire batch of data has been ordered, and this can be irritating to wait for and slightly worrying, as the machine may appear to do nothing for long periods. The 'bubble' sort has achieved fame through being very slow. It operates by checking neighbouring values in the array, interchanging those which are out of sequence, and repeating this process until the sort is guaranteed, or until any pass takes place without a transposition, depending on the algorithm. That in Chapter 5 (section 5.3) has a test in line 620 which uses a 'finished' flag. The sort is assumed to be in ascending order, and after every pass another value is positioned at its correct value at the 'top' of the heap, unless, with a partly-sorted set of data, many items are simultaneously sorted. To illustrate the idea, seven figures in the left-hand column are shown sorted (in five passes) in the right-hand column.

4	<u>7</u>	7	7	7
7	4	<u>6</u>	6	6
1	6	4	<u>5</u>	5
3	1	5	4	<u>4</u>
5	3	1	3	3
2	5	3	1	2
6	2	2	2	1

Starting at the bottom of the set of data, each item is compared with its immediate neighbour and interchanged if it is out of sequence. The process is repeated to a distance up the data which depends on the previous number of passes; the underlined digit represents the top limit in each pass. With n items of data, a maximum of n + (n-1) + (n-2) + ... passes is required, making about $\frac{1}{2}n^2$ in all. On this basis it is often said that the bubble sort takes time proportional to the square of the number of items to be sorted. However, the correct time is very sensitive to partial ordering of the data. The graph at the end of SORT shows that new items, added to an already sorted array, then bubble sorted together, is very fast; in fact, under these circumstances, the bubble sort is one of the fastest possible, since it does little more than check that each item is correctly related to its neighbour, which is necessary in any sorting system. The machine-code sort operates on string arrays, changing the pointers where appropriate, and using the identical comparison to that of BASIC, for consistency. It does not sort the zeroth element, which can therefore be used as a title or reminder. If new items are to be sorted in, keep a number of null or blank elements at the start of the array. As the diagram illustrates, high values (e.g. 6) can rise quickly from the bottom, but low values (e.g. 1) are slow in descending. Note finally that the machine-code can be made to sort from the second, third, ..., characters of the string, rather than the first, by changing $FF in $032E (BASIC 1), or $7FB6 (BASIC>1) to 0 (second), 1 (third),.... A demonstration BASIC routine is provided with the machine-code. Of the other sorts, the Shell-Metzner and Quicksort are well-known; the former performs many small bubble sorts on longitudinal subsets of the data; the latter compares data with a 'pivot value', putting the result into one or other 'stack' depending on the result. It may run out of space; if so, dimension the array in line 40 with a larger value. The 'scatter' sort is an attempt to mimic human sorting: a subsidiary array is used, into which data is first roughly sorted, on some a priori basis, for example with the As at the beginning, Zs at the end, and others in between. Then this array is sorted thoroughly. Its use of RAM is too great to permit the method to be very useful on micros.

Dealer inquiries are invited.

"Your book is EXCELLENT!"

JIM STRASMA,
From a letter to the author.

A **COMPUTE! Books** Publication
In conjunction with Level Limited.

TABLE 2 – CHR$ CHARACTER SETS FOR PET/CBM

The CHR$ function prints all characters shown. Business keyboards do not print codes 161-191. No keyboard prints codes 96-127 or 224-255. *Control Codes in ITALICS apply only to 80-column machines and the newest 40-column machines.* Character set selection is as follows: S - POKE 59468,12 in all machines. AO - POKE 59468,14 in PETs with Original ROMs. A - POKE 59468,14 in all other machines.

CHR$	ACTION	CHR$	S	A	AO	CHR$	S	A	AO	CHR$	S	A	AO
0		32	Space			64	@	@	@	96	Space		
1		33	!	!	!	65	A	a	A	97	!	!	!
2		34	"	"	"	66	B	b	B	98	"	"	"
3	Stop	35	#	#	#	67	C	c	C	99	#	#	#
4		36	$	$	$	68	D	d	D	100	$	$	$
5		37	%	%	%	69	E	e	E	101	%	%	%
6		38	&	&	&	70	F	f	F	102	&	&	&
7	*Bell*	39	'	'	'	71	G	g	G	103	'	'	'
8		40	(	(	(	72	H	h	H	104	(	(	(
9	*Tab*	41	)	)	)	73	I	i	I	105	)	)	)
10		42	*	*	*	74	J	j	J	106	*	*	*
11		43	+	+	+	75	K	k	K	107	+	+	+
12		44	,	,	,	76	L	l	L	108	,	,	,
13	Return	45	–	–	–	77	M	m	M	109	–	–	–
14	*Text*	46	.	.	.	78	N	n	N	110	.	.	.
15	*Set Top*	47	/	/	/	79	O	o	O	111	/	/	/
16		48	0	0	0	80	P	p	P	112	0	0	0
17	Crsr Down	49	1	1	1	81	Q	q	Q	113	1	1	1
18	Reverse	50	2	2	2	82	R	r	R	114	2	2	2
19	Home	51	3	3	3	83	S	s	S	115	3	3	3
20	Delete	52	4	4	4	84	T	t	T	116	4	4	4
21	*Delete Line*	53	5	5	5	85	U	u	U	117	5	5	5
22	*Erase End*	54	6	6	6	86	V	v	V	118	6	6	6
23		55	7	7	7	87	W	w	W	119	7	7	7
24		56	8	8	8	88	X	x	X	120	8	8	8
25	*Scroll Up*	57	9	9	9	89	Y	y	Y	121	9	9	9
26		58	:	:	:	90	Z	z	Z	122	:	:	:
27	*Escape*	59	;	;	;	91	[	[	[	123	;	;	;
28		60	<	<	<	92	\	\	\	124	<	<	<
29	Crsr Right	61	=	=	=	93	]	]	]	125	=	=	=
30		62	>	>	>	94	↑	↑	↑	126	>	>	>
31		63	?	?	?	95	←	←	←	127	?	?	?

CHR$	ACTION	CHR$	S	A	AO	CHR$	S	A	AO	CHR$	S	A	AO
128		160	Sh	Space		192	─	─	─	224	Sh	Space	
129		161	▌	▌	▌	193	♠	A	a	225	▌	▌	▌
130		162	▄	▄	▄	194	│	B	b	226	▄	▄	▄
131	Run	163	▔	▔	▔	195	─	C	c	227	▔	▔	▔
132		164	▁	▁	▁	196	─	D	d	228	▁	▁	▁
133		165	▏	▏	▏	197	─	E	e	229	▏	▏	▏
134		166	▒	▒	▒	198	─	F	f	230	▒	▒	▒
135		167	▕	▕	▕	199	│	G	g	231	▕	▕	▕
136		168	▒	▒	▒	200	│	H	h	232	▒	▒	▒
137	*Toggle Tab*	169	◤	▨	▨	201	╮	I	i	233	◤	▨	▨
138		170	▐	▐	▐	202	╰	J	j	234	▐	▐	▐
139		171	├	├	├	203	╯	K	k	235	├	├	├
140		172	▗	▗	▗	204	└	L	l	236	▗	▗	▗
141	Sh Return	173	└	└	└	205	╲	M	m	237	└	└	└
142	*Graphic*	174	┐	┐	┐	206	╱	N	n	238	┐	┐	┐
143	*Set Bottom*	175	▂	▂	▂	207	┌	O	o	239	▂	▂	▂
144		176	┌	┌	┌	208	┐	P	p	240	┌	┌	┌
145	Crsr Up	177	┴	┴	┴	209	●	Q	q	241	┴	┴	┴
146	Rvs Off	178	┬	┬	┬	210	▁	R	r	242	┬	┬	┬
147	Clear Scrn	179	┤	┤	┤	211	♥	S	s	243	┤	┤	┤
148	Insert	180	▎	▎	▎	212	│	T	t	244	▎	▎	▎
149	*Insert Line*	181	▍	▍	▍	213	╭	U	u	245	▍	▍	▍
150	*Erase Begin*	182	▐	▐	▐	214	╳	V	v	246	▐	▐	▐
151		183	▀	▀	▀	215	○	W	w	247	▀	▀	▀
152		184	▬	▬	▬	216	♣	X	x	248	▬	▬	▬
153	*Scroll Down*	185	▃	▃	▃	217	│	Y	y	249	▃	▃	▃
154		186	┘	✓	✓	218	♦	Z	z	250	┘	✓	✓
155		187	▖	▖	▖	219	┼	+	+	251	▖	▖	▖
156		188	▝	▝	▝	220	▒	▒	▒	252	▝	▝	▝
157	Crsr Left	189	┘	┘	┘	221	│	│	│	253	┘	┘	┘
158		190	▘	▘	▘	222	π	▒	▒	254	▘	▘	▘
159		191	▚	▚	▚	223	◥	▧	▧	255	π	▒	▒

YES! We said "SORT VISICALC"!

versacalc™

A UNIQUE VISICALC(tm) ENHANCEMENT

NOW AVAILABLE FOR PET

If you use Visicalc(tm) but are bumping into its limitations, you need Versacalc(tm)! Versacalc runs within **Visicalc but uses no extra memory; in** fact, it effectively increases memory by letting you call in modules from disk as needed.

A Tutorial section makes clear such features as @LOOKUP, DIF, @NA, @ERROR, which are not well explained in the Visicalc manual.

A Utilities section makes it easy to create your own menu-driven modules which condense hundreds of commands into four keystrokes. You can build in sophisticated error checking (e.g. Is the input value between certain limits?). Now it is possible for people untrained in Visicalc to perform the weekly updating without constant instruction.

for Commodore Pet & CBM 2040 - DOS 1.0
available for 8050 on request

Retail $125

SORT Visicalc, and Other Useful Wonders!

NOW YOU CAN:

- SORT a Visicalc screen on any column, ascending or descending, and all related formulas and labels are sorted too.
- put the entire disk CATALOG on the screen at once!
- easily do Year-To-Date accumulations!
- "pound" formulas to expose the full formulas in place on the screen!
- append two Visicalc files!
- print the contents of a /SS file!
- print the contents of a /PF file!
- AND our EASEL BINDER is so nice that you will put your other manual in it!

Distributed by:

Anthro-Digital Software
P.O. Box 1385
Pittsfield, MA 01202
413-448-8278

Versacalc is a trademark of Versacalc Enterprises, Inc.
Visicalc is a trademark of Personal Software, Inc.

SM-KIT

for Commodore Computers

A Programming Productivity Tool

ONLY $40

NEW !

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

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

ERROR DETECTION: the SM-KIT automatically indicates the erroneous line and statement for any BASIC program error.
LINE NUMBERING: the SM-KIT automatically numbers BASIC statements until you turn the function off.
SCREEN OUTPUT: the commands FIND, DUMP, TRACE and DIRECTORY display on the CRT while you hold the RETURN key (display pauses when the key is released). Continuous output is selected with shift-lock.
OUTPUT CONTROL to DISK or PRINTER: in addition to displaying on the CRT, you can direct output to either disk or printer.
HARDCOPY: allows screen displays to be either printed or stored on disk.
FIND: searches all or any part of a program for text or command strings or variable names. Either exact search or wild card search supported.
RENUMBER: the SM-KIT can renumber all or any part of a program. The selective renumbering allows you to move blocks of code within your program.
VARIABLE DUMP: displays the contents of floating point, integer, and string variables (both simple and array). Can display all variables or any selected variables.
TRACE: SM-KIT can trace program execution either continuously or step by step starting with any line number. Selected program variables can be displayed while tracing.
DISK COMMANDS: as in DOS Support (Universal Wedge), the "shorthand" versions of disk commands may be used for displaying disk directory, initializing, copying, scratching files, load and run, etc.
LOAD: SM-KIT can load all or part of BASIC or machine language programs. It can append to a program in memory, overwrite any part of a program, load starting with any absolute memory location, and load without changing variable pointers.
MERGE: allows merging all or any part of a program on disk with a program in memory.
SAVE and VERIFY: SM-KIT provides one step program save and verification. It also allows you to save any part of a program, or any address range.
DELETE: allows any program segment to be deleted.
REPEAT KEYS: allows repeating functions if not already available on specific PET/CBM model.

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

252 Bethlehem Pike
Colmar, PA 18915
215-822-7727
A B Computers

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

PET ROMs Character Set A0, for "alternate, original," and the alternate character set in all other ROMs Character Set A. These two alternate sets contain the same characters, but in a different order, as we will see later on.

About 75% of the characters in all three sets are identical. Sets S, A, and AO differ only in the characters produced by the alphabetic keys A through Z, and in four other characters, all graphics. At "power on," graphics keyboard PET/CBM's have Character Set S enabled, while machines with business keyboards have Character Set A.

Character sets can be switched by POKEing 59468 with 12 or 14, or with other numbers as well. Numbers having a binary representation of the form XXXX 110X will produce character set S, while any other number will produce your machine's alternate character set. In machines having the GRAPHIC and TEXT commands, these can also be used to switch character sets.

Now back to the demo program. Without touching the keyboard, study the characters displayed on your screen. Notice that there are 256 characters, all different, and that the first 128 of them are repeated in reverse field to make up the second 128. (There may seem to be two identical SPACE characters, but there aren't – the second one is SHIFTED SPACE, and your computer treats it as a separate character altogether.) This is the complete set of printing characters from the currently activated set. In other words, you are looking at every character your machine can display at this moment.

Now press any key and study the characters in the other set. Again, there are 256 unique symbols, 128 regular and 128 in reverse field. Press a key several times, and notice which characters change as the character sets are toggled. If you count them, you'll find 60 characters that change – 30 regular and 30 reverse field. Note which ones they are, and notice that certain combinations of characters can never be on the screen at the same time, the HEART and the lowercase "s," for example.

You have now seen every character that your machine is able to display. All other PET/CBM's have the same printing characters, but in some machines they are gotten at in a slightly different way. Altogether, there are 316 different characters, 256 of them available at any one time.

Since we've now looked at the complete repertoire of printing characters, let's look further into character codes, the other part of the character set. A character can be displayed on PET/CBM's screen in one of three ways: by POKEing a code into a screen memory location, by pressing a key, or by executing a PRINT statement. Additionally, your machine can send characters to, or receive them from, devices connected to the IEEE, user, recorder, and memory expansion ports. In every case, character codes are used to specify which character is to be displayed, recorded, or transmitted.

The Screen And CHR$ Sets

Our demonstration program POKEd characters to the screen, using the 256 character codes from 0 to 255 inclusive, which produced 256 different printed symbols. POKEing a 1 gave an A, a 2 gave a B, and so on through all the printed characters. This particular combination of codes and characters is valid *only* for screen POKEing, and is summarized in Table 1. We'll call it the *Screen POKE Character Set.*

All other character manipulation in the PET/CBM uses a completely different group of codes to print these same characters, and it is summarized in Table 2. Many of the printing characters and control functions in this set can be activated directly from the keyboard, and all of them can be activated by using the CHR$ function. We will call this the *CHR$ Character Set.*

Some people call it PET ASCII, but that terminology is misleading – PET/CBM's CHR$ character set has twice as many codes as ASCII, and only about half of the 128 ASCII codes have the same meaning in the ASCII and CHR$ character sets!

All PET/CBM keyboard and PRINT operations use the CHR$ character set; it is also used whenever characters are sent to or received from external devices such as printers, files, or modems. If you tell PET to send an asterisk to your printer, it will, in fact, send 0010 1010, or 42 in decimal notation. And whenever PET receives a 42, whatever the 42 may have represented in the sending device's character set, PET interprets it as an asterisk.

There are 256 CHR$ codes, numbered from 0 to 255 inclusive, and the CHR$ character set differs substantially from the POKE set, although both can be used to display the same symbols. Here are the essential differences:

- Very few characters have identical POKE and CHR$ codes.
- There are no CHR$ codes for reverse field characters. Instead, the RVS ON/OFF key or its corresponding CHR$ codes are used to produce them.
- The CHR$ set includes 14 control characters (28 in 80-column machines and newer 40-column machines) in addition to its 128 printing characters.
- Since there are 256 CHR$ codes, and only

TABLE 3 – AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII)

The ASCII Codes in Decimal Form:

0	NUL	32	SPC	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	¦
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

English Names of the Special Characters:

33 - Exclamation point
34 - Quotation mark
35 - Number sign
36 - Dollar sign
37 - Percent
38 - Ampersand
39 - Apostrophe
40 - Opening parenthesis
41 - Closing parenthesis
42 - Asterisk
43 - Plus
44 - Comma
45 - Hyphen (Minus)
46 - Period (Decimal point)
47 - Slant
58 - Colon
59 - Semicolon
60 - Less than
61 - Equals
62 - Greater than
63 - Question mark
64 - Commercial at
91 - Opening bracket
92 - Reverse slant
93 - Closing bracket
94 - Circumflex
95 - Underline
96 - Grave accent
123 - Opening brace
124 - Vertical line
125 - Closing brace
126 - Tilde

Key to Control Code Abbreviations:

ACK - Acknowledgement
BEL - Bell
BS - Backspace
CAN - Cancel
CR - Carriage return
DC1 - Device control #1
DC2 - Device control #2
DC3 - Device control #3
DC4 - Device control #4
DEL - Delete
DLE - Data link escape
EM - End of medium
ENQ - Enquiry
EOT - End of transmission
ESC - Escape
ETB - End of transmission block
ETX - End of text
FF - Form feed
FS - File separator
GS - Group separator
HT - Horizontal tab
LF - Line feed
NAK - Negative acknowledgement
NUL - Null
RS - Record separator
SI - Shift in
SO - Shift out
SOH - Start of heading
SPC - Space
STX - Start of text
SUB - Substitute
SYN - Synchronous idle
US - Unit separator
VT - Vertical tab

128 + 14 = 142 CHR$ characters (156 in some machines), many of the CHR$ codes have no meaning at all in the PET/CBM, and in many cases one printed character has two different CHR$ codes!

Table 3 shows the standard ASCII character set. It is presented in a similar format to Table 2, to facilitate comparison of the ASCII and PET/CBM character sets. Study the two tables carefully, and you'll see that PET/CBM has all but seven of the ASCII printed characters, (94-96 and 128-126), but often with different character codes.

You'll also notice that ASCII, being a seven-bit code, has no character codes above 127, and lacks many of PET/CBM's printing characters.

Because there are so many ASCII control codes, most ASCII keyboards use a special CONTROL key, similar to the SHIFT key, to generate them. CTRL A often sends a 1 code (SOH), CTRL B a 2 code (STX), CTRL C a 3 code (ETX), etc. Also, the meanings of the ASCII control codes, established with commercial message traffic in mind, are almost completely foreign to PET/CBM.

No wonder it's sometimes hard to use non-Commodore devices with your machine! But now that you have Tables 2 and 3, you can write programs for perfect conversions between ASCII and PET/CBM codes. Table 2 shows you exactly what code PET/CBM sends when a given character is transmitted, and Table 3 shows you exactly how an ASCII device will interpret that code. Conversely, Table 3 shows you the intended character representation of every ASCII code your machine receives from outside, while Table 2 shows which code it has to be converted to to have the same representation inside your PET/CBM.

Some Example Conversions

A few examples will illustrate the conversions. Suppose that your PET, with Character Set A enabled, is connected through a modem to an ASCII terminal, and that you are sending messages back and forth. The ASCII terminal sends the lowercase letter "a." Table 3 shows that the code actually transmitted will be 97 decimal, or 0110 0001. If your PET receives that code and displays it on the screen as a PRINTed character, Table 2 shows that it will be displayed as an exclamation point!

So you'll need some software in your PET that converts received ASCII input to CHR$ format before displaying it. In this case, whenever PET receives a 97, the program should convert it to a 65 before PRINTing it. Of course, the program should also be smart enough to convert (or not convert) any of the other ASCII codes between 0 and 127 so that they give the proper display on your PET.

Going the other way, suppose that you press the unshifted "b" key on your PET, and want the distant ASCII terminal to see it as a lowercase "b." Table 2 tells us that your PET will send a 66, which Table 3 tells us the ASCII terminal will interpret as an *uppercase* "B," which is not at all what you want. So your program has to convert the 66 to a 98 before transmitting it, and to make conversions on any other transmitted characters where it's appropriate.

If you study Tables 2 and 3, you'll be able to determine every sending and receiving conversion, and to write your programs accordingly. If the remote device has a character set different from standard ASCII (many of them do), all you need to do is compare it to Table 2, and you'll be able to program the conversions.

```
100 REM   *** CHARACTER SET DEMO ***
120 REM
130 REM SHOWS EVERY PET/CBM CHARACTER
140 REM (KEY PRESS CHANGES CHAR. SET)
150 REM
160 PRINT"{CLEAR}"
170 FORCH=0TO255
180 POKE(32768+2*CH+40*INT(CH/20)),CH
190 NEXTCH
200 FORI=1TO23:PRINT:NEXT
210 PRINTTAB(32)"59468=1";
220 IFPEEK(59468)=14THEN250
230 POKE59468,12:POKE33767,50
240 GETA$:IFA$=""THEN240
250 POKE59468,14:POKE33767,52
260 GETA$:IFA$=""THEN260
270 GOTO230
280 REM
290 REM ** LINE 200 FOR 80 COL. CMB'S:
300 REM
310 FORI=1TO11:PRINT:NEXT
```

©

In the May issue of **COMPUTE!**, *Steve Steinberg's article "Language Lab" was inadvertently published with the wrong program accompanying it. It is reprinted here in its entirety.*

An ATARI Learning Program

Language Lab

Steve Steinberg
Washington, DC

Language Lab is a program to use your ATARI to help you build vocabulary in a foreign language. It is basically a computerized version of that old standby of language education, the flash card set, and I have found it extremely simple and effective to use. It is structured so that you can drill and score yourself on as many words as you like, but I find it most useful if you display a fifteen or twenty word vocabulary drill on the screen, spot the errors and review them, then try again. If you have enough memory, you can also use it to create a fair sized foreign language dictionary.

I have used a handful of French words in the program example but you can easily change this to any language you want. Simply change line 55 LANG$ = "FRENCH" to LANG$ = "GERMAN", "SPANISH", "NAVAJO" or whatever you like and enter the appropriate word pairs in DATA.

The DATA, beginning on line 1000 is easy to expand as your language skill increases and can be used in conjunction with either a self teaching or school language course. Just enter the DATA in word pairs, the first in English, the second in whatever language you are working with.

The key to the vocabulary drills is the random word subroutine, lines 500 through 550. In line 510 X = (1 + INT(RND(1)*25)) the 25 is equal to the number of word pairs entered as DATA. As you increase the number of word pairs by adding new DATA this number should also be appropriately increased. You can also alter this line to drill yourself on only part of your total foreign language vocabulary.

Let's assume, for example, that you have 600 word pairs in DATA but only want to drill yourself on the last 100 words you have entered. In that case, change line 510 to X = (500 + INT(RND(1)*100)).

Lines 160,180,260,280,330 and 430 use the ATARI cursor advance and line "up" arrow keys to provide a format that will display as much of your language drill or translations on the screen at one time as possible, but you can replace these with just "PRINT" statements if you prefer. This would be useful if you want to use the program for drill and translation of whole phrases and sentences instead of just single words. Don't forget, however, to increase the size of the appropriate string dimensions (ENGLISH$,WORD$,TRANSLATE$) in line 50.

One final note; if you happen to own IRIDIS 2 (and if you have an ATARI computer I don't think you can find a better bargain in software) you can easily add the appropriate subroutine to use Language Lab for Russian, Greek, Hebrew or whatever you wish by adding the foreign alphabet in lower case. I am currently using the program to teach myself Classical Greek. I hope this program will be useful for budding language students and in any case good luck with it, bon chance, and auf wedersehen.

```
40 REM LANGUAGE LAB
41 REM
45 REM BY STEVE STEINBERG
50 DIM LANG$(15),ENGLISH$(20),WORD$(20),
TRANSLATE$(20),Q$(2)
55 LANG$="FRENCH"
60 PRINT "{CLEAR}":PRINT "        ";LANG
$;" LANGUAGE LAB"
65 PRINT :PRINT
70 PRINT "1) ";LANG$;" TO ENGLISH VOCABU
LARY DRILL"
75 PRINT :PRINT "2) ENGLISH TO ";LANG$;"
 VOCABULARY DRILL"
80 PRINT :PRINT "3) ";LANG$;" TO ENGLISH
 TRANSLATOR"
85 PRINT :PRINT "4) ENGLISH TO ";LANG$;"
 TRANSLATOR"
90 PRINT :PRINT "ENTER NUMBER OF DESIRED
 PROGRAM";:INPUT CHOICE:IF CHOICE>4 THEN
 GOTO 90
95 GOTO CHOICE*100
100 REM LANGUAGE TO ENGLISH DRILL
110 PRINT :PRINT :PRINT "HOW MANY WORDS"
;:INPUT N:COUNT=0:SCORE=0
120 ? "{CLEAR}":? "TRANSLATE THE FOLLOWI
NG WORDS INTO     ENGLISH":PRINT
125 GOSUB 500
130 IF COUNT=N THEN GOSUB 600
135 IF COUNT=N THEN GOTO 100
140 PRINT WORD$,
```

```
150 INPUT TRANSLATE$,
160 IF TRANSLATE$=ENGLISH$ THEN SCORE=SC
ORE+1:COUNT=COUNT+1:PRINT "{UP}{30 RIGHT
}CORRECT"
170 IF TRANSLATE$=ENGLISH$ THEN GOTO 125

180 IF TRANSLATE$<>ENGLISH$ THEN COUNT=C
OUNT+1:PRINT "{UP}{11 RIGHT}WRONG! IT'S
";ENGLISH$:GOTO 125
200 REM ENGLISH TO LANGUAGE DRILL
210 PRINT :PRINT :PRINT "HOW MANY WORDS"
;:INPUT N:COUNT=0:SCORE=0
220 ? "{CLEAR}":? "TRANSLATE THE FOLLOWI
NG WORDS INTO     ";LANG$:PRINT
225 GOSUB 500
230 IF COUNT=N THEN GOSUB 600
235 IF COUNT=N THEN GOTO 200
240 PRINT ENGLISH$,
250 INPUT TRANSLATE$,
260 IF TRANSLATE$=WORD$ THEN SCORE=SCORE
+1:COUNT=COUNT+1:PRINT "{UP}{30 RIGHT}CO
RRECT"
270 IF TRANSLATE$=WORD$ THEN GOTO 225
280 IF TRANSLATE$<>WORD$ THEN COUNT=COUN
T+1:PRINT "{UP}{11 RIGHT}WRONG! IT'S ";W
ORD$:GOTO 225
300 ? "{CLEAR}":PRINT LANG$;" TO ENGLISH
 TRANSLATOR":PRINT :PRINT "ENTER ";LANG$
;" WORD"
310 ? :INPUT TRANSLATE$
320 READ ENGLISH$,WORD$
330 IF WORD$=TRANSLATE$ THEN PRINT "{UP}
{15 RIGHT}";:PRINT ENGLISH$:RESTORE :GOT
O 410
340 IF ENGLISH$<>TRANSLATE$ THEN GOTO 32
0
400 ? "{CLEAR}":PRINT "ENGLISH TO ";LANG
$;" TRANSLATOR":PRINT :PRINT "ENTER ENGL
ISH WORD"
410 ? :INPUT TRANSLATE$
420 READ ENGLISH$,WORD$
430 IF ENGLISH$=TRANSLATE$ THEN PRINT "{
UP}{15 RIGHT}";:PRINT WORD$:RESTORE :GOT
O 410

440 IF WORD$<>TRANSLATE$ THEN GOTO 420
500 REM RANDOM WORD SUBROUTINE
510 X=(1+INT(RND(1)*25))
520 RESTORE
530 FOR M=1 TO X:READ ENGLISH$,WORD$
540 NEXT M
550 RETURN
600 REM SCORE SUBROUTINE
610 PRINT :PRINT "OUT OF ";N;" VOCABULAR
Y WORDS YOU HAVE    CORRECTLY TRANSLATED
 ";SCORE;"."
620 PRINT :PRINT "YOUR SCORE IS ";INT(SC
ORE*(100/N));" PER CENT"
630 PRINT :PRINT "GO AGAIN (Y OR N)";:IN
PUT Q$:IF Q$="Y" THEN RETURN
640 IF Q$="N" THEN END
1000 DATA ONE,UN,TWO,DEUX,THREE,TROIS,FO
UR,QUATRE,RED,ROUGE,BLUE,BLEU,GREEN,VERT
,MAN,HOMME,CHILD,ENFANT
1010 DATA HAT,CHAPEAU,PENCIL,CRAYON,HAM,
JAMBON,EGG,OEUF,CITY,VILLE,COUNTRY,PAYS,
OF,DE,UNDER,SOUS,MONDAY,LUNDI
1020 DATA TUESDAY,MARDI,WEDNESDAY,MERCRE
DI,SATURDAY,SAMEDI,SUNDAY,DIMANCHE,YES,O
UI,NO,NON,OLD,ANCIEN
```

©

FORTH programmers:
COMPUTE! is
looking for screens
and applications
articles.

A map of the significant machine language routines in the VIC Super Expander. You can translate these hexadecimal numbers into decimal, then SYS to them and watch the effects.

VIC Super Expander Memory Map

Chuan Chee
St. Catharines, Canada

General Input/Output Routines

A000-A001 **Vector:** RESET ($A044)
A022-A003 **Vector:** NMI ($A077)
A004-A008 ROM identification ('a0CBM')
A009-A010 **Table:** function key numbers
A011-A043 **Table:** initial function key definitions
A044-A076 RESET routine
A077-A08A NMI routine
A08B-A0BE Parse KEY (get parameters and check syntax)
A0BF-A131 Display all function key definitions
  A110- A11C Print '°+chr$(34)' and an optional '+'
  A11D- A131 Print '°+chr$(13)' and an optional '+'
A132-A135 **Table:** ASCII string for output ('key' backwards)
A136-A13F **Table:** ASCII string for output ('°+chr$(13)' backwards)
A140-A149 **Table:** ASCII string for output ('°+chr$(34)' backwards)
A14A-A17A Delete current function key string (key number in .X)
A17B-A1B0 Insert string into function key definition area
A1B1-A1BE Locate function key definition (key number in .X, return index in .Y)
A1BF-A213 **Table:** new BASIC keywords in ASCII form
A214-A237 **Table:** vectors corresponding to new BASIC tokens ($CC to $DD)
A238-A2A1 Initialize kernal vectors, I/O, RAM
A2A2-A2C1 **Table:** kernal vectors (L,H)
A2C2-A2C7 Warm start routine
A2C8-A317 Output a character to device 3 (char in .A)
A318-A336 End music mode
A337-A365 Interpret keyboard matrix input
A366-A369 **Table:** keyboard matrix code for function keys
A36A-A371 **Table:** conversion pattern for function keys
A372-A394 IRQ routine
A395-A3A5 Input a char from any device (device number in $99)
A3A6-A3B3 Output a char to any device (char in .A, device num in $9A)
A3B4-A3F1 Input each char from keyboard buffer
  A3B4- A3E7 Handle 'RUN' key
  A3E8- A3F1 Handle 'RETURN' key
A3F2-A3FC Input from device 0
A3FD-A406 Print an error message in GRAPHIC 0 mode (error token in .A)
A407-A4B9 Lexically analyse BASIC source line (translate to tokens)
A4BA-A503 Print BASIC tokens in ASCII form
A504-A529 Start new BASIC statement
  A515- A523 Handle new tokens ($CC to $D6)
A52A-A58A Get and evaluate an expression
  A558- A58A Handle new function tokens ($D7 to $DD)
A58B-A596 **Table:** BASIC vectors for RAM
A597-A5A4 Change BASIC vectors during RESET

Music Routines

A5A5-A5D0 Save current sound table (address of table in .X,.Y)
A5D1-A601 IRQ music driver
A602-A625 **Table:** conversion for note index to frequency
A626-A6E5 Interpret music mode characters (char in .A)
  A629-A643 Execute 'O' command, default 3
  A644-A65D Execute 'T' command, default 0
  A65E-A674 Execute 'S' command, default 4
  A675-A686 Execute 'V' command, default 7
  A687-A693 Execute 'R' command
  A694-A69B Execute 'P' command
  A69C-A6A7 Execute 'Q' command
  A6A8- Play new note (note index in .Y)
  A6AB-A6B3 Save new sound table when previous note finishes
  A6B4-A6B9 Common return routine
  A6BA-A6CD Play notes 'A' to 'G'
  A6CE-A6DA Execute '#' command
  A6DB-A6E5 Execute '$' command
A6E6-A6EC **Table:** conversion for notes to note index
A6ED-A6EF **Table:** conversion for octave to base note index
A6F0-A6F9 **Table:** conversion for tempo to duration (jiffies)

Parsing New Command Routines

A6FA- Look for and evaluate first 1-byte and two 2-byte parameters
A6FD- Look for and evaluate two 2-byte parameters
A700- Look for and evaluate one 2-byte parameter
A714-A71B Save one 1-byte parameter (parameter in .A, index in .Y)
A71C- Look for and evaluate two 1-byte parameters
A71F-A72B Look for and evaluate one 1-byte parameter
A72C-A73F Parse GRAPHIC (get parameters and check syntax)
A740-A762 Parse CIRCLE
A763-A7A4 Parse DRAW
A7A5-A7BC Parse POINT
A7BD- Parse COLOR
A7C8-A7CE Go to execute commands after parsing
A7CF-A7D8 Parse REGION
A7D9-A7DC Parse SCNCLR
A7DD-A7E9 Parse SOUND
A7EA-A809 Parse CHAR
A80A-A810 Parse PAINT

A811-A817	Parse RPOT
A818-A81B	Parse RPEN
A81C-A81F	Parse RSND
A820-A823	Parse RCOL
A824-A827	Parse RGR
A828-A842	Parse RDOT
A843-A846	Parse RJOY
A847-A84E	Look for first 1-byte parameter
A84F-A866	Indirect jump to execute new commands (pointer to parameter save area in .X,.Y, command index in .A)
A867-A878	**Table:** vector to execute new commands (H)
A879-A88A	**Table:** vector to execute new commands (L)

Execute New Command Routines

A88B-AA22	Execute GRAPHIC
A8AB- A94E	Handle GRAPHIC 1,2,3 if previous was 0.
A8D4-A942	Transfer BASIC program to above $2000 and execute CLR
A943-A94E	Make screen at $1E00 and character set at $1000
A94F-A9AB	Handle GRAPHIC 4
A967-A9AB	Transfer BASIC program down to old location and execute CLR
A9AC- A9B7	Handle GRAPHIC 0 if previous was 1,2,3
A9B8- AA22	Set up proper GRAPHIC screen
AA23-AA28	Execute RGR
AA29-AA6A	Execute COLOR
AA6B-AA84	Execute REGION
AA85-AA8B	Execute RCOL
AA8C-AAE6	Execute RDOT
AAE7-AAF1	Execute POINT
AAF2-AB12	Execute SCNCLR
AB13-AB22	Execute DRAW (c TO x,y ...)
AB23-AB34	Execute DRAW (c,x,y TO x,y ...)
AB35-AB54	Execute SOUND
AB55-AB69	Execute RSND
AB6A-AB76	Execute RPOT
AB77-AB7D	Execute RPEN
AB7E-	Plot a single point from parameter save area
AB86-ABE4	Plot a single point from beginning scaled X,Y coordinates
ABE5-	Set up pointers to character and colour memory
ABFA-AC0A	Set up pointer to colour memory
AC0B-	Draw a line with a new starting coordinate
AC11-AC92	Draw a line starting from previous coordinate (using a version of Bresenham's DDA algorithm)
AC93-AD12	Execute CIRCLE (using principal of digital differential analyser (DDA))
AD13-	Convert starting angle to radians
AD19-AD22	Divide FAC#1 by 16
AD23-	Calculate new scaled X and Y coordinate on locus
AD39-AD6B	Calculate unit offset * scaled radius
AD6C-ADDE	Execute PAINT
ADDF-	Check for possible new lower bound pivot coordinate
ADE8-AE01	Save pivot coordinate
AE02-AE0B	Check for possible new upper bound pivot coordinate
AE0C-	Check if able to PAINT a coordinate
AE0F-AE1E	Check if able to PAINT a coordinate (X,Y in .A,.Y)
AE1F-AE23	Move beginning scaled X,Y coordinate to .A,.Y
AE24-AE3B	Check if coordinate has been already plotted on
AE3C-AE44	Move beginning scaled X coordinate to the right
AE45-AE51	Move beginning scaled X coordinate 2 to the left
AE52-AE56	Flag 'FORMULA TOO COMPLEX' error message
AE57-AED9	Execute CHAR
AEDA-AF13	Execute RJOY
AF14-AF33	Set up correct VIC chip screen registers
AF34-	Save number of coordinates and colour register
AF39-AF3E	Save colour register
AF3F-AF47	Copy beginning from ending scaled X,Y coordinate
AF48-AF75	Scale X and Y coordinates
AF76-AFB0	Scale X or Y coordinate to the range 0 to 159 (.X = .A*coordinate*2/256) (number of columns or rows in .A)
AFB1-AFBA	**Table:** vector to map Y coordinate to colour memory (L)
AFBB-AFCE	**Table:** vector to map X coordinate to character memory (L)
AFCF-AFE2	**Table:** vector to map X coordinate to character memory (H)
AFE3-AFE5	**Table:** bit set for colour memory
AFE6-AFE6	(not used – contains $00)
AFE7-AFEE	**Table:** bit mask for highres mode
AFEF-AFF6	**Table:** bit mask for multicolour mode
AFF7-AFFA	**Table:** bytes to plot in multicolour mode
AFFB-AFFE	**Table:** conversion factor for VIC chip screen registers
AFFF-AFFF	(not used – contains $AA)

Note:

(H):	high byte of a two-byte address
(L):	low byte of a two-byte address
Vector:	two-byte address used for indirection of execution
Pointer:	two-byte address for data
Index:	one-byte offset for a table

General RAM Area

0024	Number of coordinates
0024	**Flag:** colour register mode ($FF = multicolour, $00 = highres)
0024-0025	**Pointer:** New start of variables / start of BASIC memory
0026	Temp area for building VIC chip registers / for building character byte / for saving start of BASIC (L)

Current Coordinates

0062	Ending scaled X coordinate (0 to 159)
0063	Beginning scaled X coordinate (0 to 159)
0064	Scaled X difference (absolute value)
0065	Ending scaled Y coordinate (0 to 159)
0066	Beginning scaled Y coordinate (0 to 159)
0067	Scaled Y difference (absolute value - 1)

World's Most Cost Effective Development System

- Type your programs directly from your Commodore VIC-20 keyboard into the built-in 4 kilobyte ROM emulator
- Jumper direct to target ROM socket
- Test programs in circuit
- Burn them direct to EPROM with built-in EPROM programmer and power supply
- Burns & runs EPROMS for the Commodore VIC-20, too
- Software on EPROM included
- Comprehensive 25 page manual
- Fits EXP PORT or SUPER EXP Leaves user PORT free

PROMQUEEN CARTRIDGE COMPLETE ONLY $169.50

manual available separately for $20.00 credited on cartridge purchase
Jumper Cable, 2 ft: $18.50 with cartridge; 2732A EPROMS $12.50 each with cartridge

Gloucester Computer Bus Co. 6 Brooks Rd., Gloucester, MA 01930
617-283-7719

Shipping charges for all items extra, Mass. residents add 5% tax.
VISA AND MASTERCARD ACCEPTED

MIGHTY "V"
TO THE
RESCUE

VIC 20®
BUSINESS AND HOME SOFTWARE
That'll KNOCK your Mainframe OUT!!
BY TSASA

Now available on cassettes –

☐ 1.	THE MORTGAGE CALCULATOR C	$ 8.95
☐ 2.	THE GASOLINE UN-GUZZLER C	$ 8.95
☐ 3.	THE BILLING SOLVER	$19.95
☐ 4.	THE CLIENT TICKLER C P	$19.95
☐ 5.	THE TAPE GENI C P	$ 9.95
☐ 6.	THE ACCOUNTANT C P	$29.95
☐ 7.	THE BAR CHART C P	$ 8.95
☐ 8.	CASH FLOW C	$13.95
☐ 9.	THE DEPRECIATOR C	$10.95
☐ 10.	BUSINESS INVENTORY C P	$18.95
☐ 11.	THE BIDDER C	$14.95
☐ 12.	THE TEMPERATURE CONVERTER C	$ 8.95
☐ 13.	FIND THE WORDS C P	$10.95
☐ 14.	DEAR DIARY C	$12.95
☐ 15.	PROFIT SHARING PLAN C	$18.95
☐ 16.	LEASE/BUY? C	$14.95
☐ 17.	RATIOS C	$ 8.95
☐ 18.	PRESENT VALUE C	$12.95
☐ 19.	THE SYNDICATOR C	$15.95
☐ 20.	MOTHER'S RECIPES C P	$12.95
☐ 21.	THE TAPE WORM C	$12.95
☐ 22.	THE CLUB LISTER C	$13.95
☐ 23.	AT HOME INVENTORY C	$12.95
☐ 24.	UTILITY BILL SOLVER C	$12.95
☐ 25.	VICKY-CALC C	$12.95
☐ 26.	BIRTHDAY GUESSER C	$ 8.95
☐ 27.	THE METRIC BRAIN C	$10.95
☐ 28.	ROLL DA' DICE C	$12.95
☐ 29.	THE TICKER TAPE C	$16.95
☐ 30.	THE SUPER BROKER C	$12.95
☐ 31.	THE MORTGAGE COMPARER C	$ 8.95
☐ 32.	THE RULE OF '78 C	$ 9.95
☐ 33.	MATH-A-MAGICIAN C	$14.95
☐ 34.	HOME BUDGET C	$12.95
☐ 35.	IT'S TRAVEL TIME C	$10.95
☐ 36.	YOUR MEDICAL RECORDS C P	$14.95
☐ 37.	THE INVESTMENT ANALYST C	$12.95
☐ 38.	THE PHONE DIRECTORY C P	$ 9.95
☐ 39.	THE AMORTIZER P	$24.95
☐ 40.	THE CHECKBOOK BOOKY C P	$12.95
☐ 41.	THE INCOME & EXPENSER C P	$14.95
☐ 42.	THE LOAN REPAYER C	$10.95
☐ 43.	MY CREDIT CARDS C P	$14.95
☐ 44.	GRADE MY KIDS C P	$14.95
☐ 45.	NET WORTH STATEMENT C P	$14.95
☐ 46.	THE SAVINGS ACCOUNT MIZER C	$13.95
☐ 47.	CALENDAR MY APPOINTMENTS P	$16.95
☐ 48.	MIXED UP WORDS C	$ 8.95
☐ 49.	THE BUDGETER C	$15.95
☐ 50.	PRINT MY PROGRAM P	$ 8.95

JUST RELEASED

☐ 51.	JOGGER'S LOGGER C	$11.95
☐ 52.	NUISMATIC PHANATIC C	$12.95
☐ 53.	SUPER SHOPPER C P	$12.95
☐ 54.	P.E.R.T. MY VIC C	$16.95
☐ 55.	LINEAR REGRESS C	$16.95
☐ 56.	STATISTIC SADISTICS C	$16.95
☐ 57.	TRACK MY WEIGHT C	$12.95
☐ 58.	LIGHTNING SORT C	$12.95
☐ 59.	GOLF SCORECARD C	$11.95
☐ 60.	ORDER TRACKER C P	$18.95

© Copyright TSASA 1981

FREE CATALOG

Specify CRT/TV (C) or (P) Printer
Programs available direct from TSASA or at fine Computer Dealers Nationwide

DEALER INQUIRIES WELCOME

Send check or MO with $1.50 Postage & Handling
COD add $1.50

TSASA

NAME ____________
ADDRESS ____________
CITY ____________
STATE ________ ZIP ________
☐ SEND FREE CATALOG

Mail to: 2 Chipley Run
West Berlin, NJ 08091
Telephone (609)346-3063

TAYLORMADE SOFTWARE • TAYLORMADE •

for the COMMODORE VIC-20

AEROBICS POINTS CALCULATOR

Now you can monitor your fitness program. Enter your time and (distance) to learn your exact aerobics points earned and calories used. For distance activities learn your yards/min., M.P.H., and min./mile. Save your weekly, monthly, and year aerobics points totals on the tape datassette. See a bar graph on the screen of your average weekly aerobics points for each month. Easy to use, all selections are made from a menu by a single keystroke.

Two programs come on one tape cassette: One for the basic VIC-20 (3583 bytes) will calculate aerobics points for running, walking, cycling and swimming. The second program requires 3K memory expansion, and will also calculate aerobics points for 25 additional activities.

Cassette and manual...U.S. $24.95
(U.S. and Canada shipping/handling $1.50)
(Foreign shipping/handliung $3.00)

TAYLORMADE SOFTWARE
P.O. Box 5574
Lincoln, NE 68505
(402) 464-9051

MasterCard VISA

VIC is a trademark of Commodore Business Machines, Inc.

• TAYLORMADE SOFTWARE • TAYLORMADE

COMMODORE DEALER

VIC-mendous Software

Clowns	Tank Invader
Air Targets	Break-Away
Depth Charge	Dungeon X
Easy Hi-Res.	Easy Multi-Color
Guided Missile	Gunner
Nuclear Attack	Replace
Robots	Tank Battle

Each For Only $9.95

Chomper $12.95
Brain Warp.......... $12.95
Music Machine $12.95
CW Trainer.......... $15.95

RANDOM ACCESS COMPUTERS
120 Benning Drive
P.O. Box 1453
Destin, Florida 32541
Ph: 904-837-7201

COMMODORE DEALER

For Scaling Coordinates

0069	Multiplicand - 1
006A	16-bit product
006B-006C	10-bit multiplier

For DRAW

0069	Scaled X unit direction - 1
006A	Scaled Y unit direction
006B-006C	Number of scaled Y units left to plot before next scaled X unit (count up)
006D-006E	Number of points left to plot (count up)

For PAINT

0069	**Index:** pivot coordinates save area

For CHAR

0069	Current row (0 to 19)
006A	Current column (0 to 19)
006B	Length of string
006C-006D	**Pointer:** string location

General RAM Area

009B	**Index:** begining of current function key definition
009B-009C	**Pointer:** current char set address / byte in char set / position in screen memory / destination of byte of BASIC program to transfer
009D	**Index:** end of function key definition area
009E	Current function key number / length of current function key string
009F	Length of current function key string (count down)
009E-009F	**Pointer:** byte in colour memory
00AC-00AD	**Pointer:** current byte (function key definition, tape, scrolling)
00C3	**Flag:** 0 = have transferred BASIC program to a new location
00C3-00C4	**Pointer:** kernal set up / current music table / parameter save area ($033C)
00FB-00FC	**Pointer:** top of BASIC memory (usually same as $0284-$0285)

For Key

028F-0290	**Vector:** interpret keyboard input ($A337)
02A1	Number of bytes taken by Super Expander in high memory ($88)
02A2	Number of characters in function key definition
02A3	**Index:** current byte of function key string
02A4	Length of function key string (amount left to output)

For Music

02A5	Previous character in music mode
02A6	Music mode ($80 = on)
02A7	Screen echo ($50 = on, $00 = off)
02A8	Current voice (sound register - 1)
02A9	Current note index
02AA	Current duration (jiffies)
02AB	Current sound amplitude (volume * 2)
02AC	Current octave (base note index)
02AD	Voice 1 note index (+ $80)
02AE	Voice 1 duration count down (jiffies)
02AF	Voice 2 note index (+ $80)
02B0	Voice 2 duration count down (jiffies)
02B1	Voice 3 note index (+ $80)
02B2	Voice 3 duration count down (jiffies)
02B3	Voice 4 note index (+ $80)
02B4	Voice 4 duration count down (jiffies)
02B5-02BF	(for expansion)

For Execution Of New Commands

02C0-02C2	**Jump table:** execute new commands (JMP $A84F)
02C3	Current VIC chip left margin register
02C4	Current VIC chip top margin register
02C5	Current VIC chip number of columns register
02C6	Current VIC chip number of rows register
02C7	Current row of cursor
02C8	Current GRAPHIC mode
02C9	(for expansion)
02CA	Current colour register parameter (while plotting)
02CB	Current screen colour
02CC	Current border colour
02CD	Current character colour
02CE	Current auxiliary colour
02CF	**Index:** parameter save area (while plotting)
02D0	Current character set address page
02D1	Usual character set address page ($80)
02D2-02D3	**Pointer:** old limit of BASIC memory
02D4	Old screen memory page
02D5	Last scaled X coordinate (0 to 159)
02D6	Last scaled Y coordinate (0 to 159)
02D7	**Flag:** $00 = DRAW c,x,y TO, $01 = DRAW c TO / current number of out of range coordinates ($00 = within range)
02D8	Old number of out of range coordinates ($00 = within range)
02D9	**Index:** parameter save area (while getting parameters)
02DA-02FF	(for expansion)

Operating System Vectors

0300-0301	**Vector:** error message	($A3FD)
0302-0303	**Vector:** BASIC warm start	($C483)
0304-0305	**Vector:** lexically analyse BASIC source line	($A407)
0306-0307	**Vector:** print BASIC tokens in ASCII form	($A4BA)
0308-0309	**Vector:** start new BASIC statement	($A504)
030A-030B	**Vector:** get and evaluate an expression	($A52A)
0314-0315	**Vector:** IRQ	($A372)
0316-0317	**Vector:** BRK instruction	($A2C2)
0318-0319	**Vector:** NMI	($FEAD)
031A-031B	**Vector:** BASIC OPEN statement	($F40A)
031C-031D	**Vector:** BASIC CLOSE statement	($F34A)
031E-031F	**Vector:** set input	($F2C7)
0320-0321	**Vector:** set output	($F309)
0322-0323	**Vector:** restore I/O	($F3F3)

0324-0325	**Vector:** input a character	($A395)
0326-0327	**Vector:** output a character	($A3A6)
0328-0329	**Vector:** test STOP key	($F770)
032A-032B	**Vector:** BASIC GET statement	($F1F5)
032C-032D	**Vector:** abort I/O	($F3EF)
032E-032F	**Vector:** user BRK instruction	($A2C2)
0330-0331	**Vector:** BASIC LOAD statement	($F549)
0332-0333	**Vector:** BASIC SAVE statement	($F685)
0334-033B	(for expansion)	

Save Area

033C-03F8	**Save area:** parameter passing / pivot coordinates (PAINT)

For Circle

033C	**Index:** X or Y
0347-0348	Old scaled X coordinate on locus
0349-034A	Old scaled Y coordinate on locus
034B-034C	New scaled X coordinate on locus
034D-034E	New scaled Y coordinate on locus
034F-0353	Floating point unit offset X coordinate
0355-0359	Floating point unit offset Y coordinate

©

This concludes a three-part series on getting started in machine language.

Part III

Machine Language: First Steps

Jim Butterfield
Associate Editor

In the two previous episodes, our intrepid hero, F. R. Vescent, has coded the following bar graph program in machine language:

```
027A  A2  30           LDX   #$30
027C  A0  00           LDY   #$00
027E  E8        YLOOP  INX
027F  E0  3A           CPX   #$3A
0281  D0  02           BNE   SKIP
0283  A2  30           LDX   #$30
0285  8A        SKIP   TXA
0286  20  D2  FF       JSR   $FFD2
0289  C8               INY
028A  CC  00  03       CPY   $0300
028D  90  EF           BCC   YLOOP
028F  A9  0D           LDA   #$0D
0291  20  D2  FF       JSR   $FFD2
0294  60               RTS
```

After finishing the coding job, he enters it into memory using the machine language monitor like this:

```
.:  027A  A2  30  A0  00  E8  E0  3A  D0
.:  0282  02  A2  30  8A  20  D2  FF  C8
.:  028A  CC  00  03  90  EF  A9  0D  20
.:  0292  D2  FF  60  ..  ..  ..  ..  ..
```

With the machine language program in place, F. R. returns to BASIC and writes:

```
200 DATA 15,10,30,35,28,28,15,0
210 READ V:IF V=0 GOTO 300
220 POKE 768,V
230 SYS 634
270 GOTO210
300 END
```

The program runs properly, but the machine language is in the first cassette buffer, and this makes it hard to SAVE. F. R. uses the following piece of trickery to make the ML coding more tractable: He types:

```
FOR J=634 TO 660:PRINT PEEK(J);:NEXT J
```

634 is the address of the start of the ML program ($027A) and 660 is the address of the last byte ($0294). So what F. R. is doing is writing a line to cause the program to be dumped, byte by byte, to the screen. He will get something like:

```
162  48  160  0  232  224  58...
```

There will be two spaces between each pair of numbers. Now F. R. does some clever screen editing. He places the cursor just before the 4 of 48, and presses the DEL key and the comma. Now the two numbers read 162,48 ... and F. R. moves the cursor ahead to just before the 160 and repeats the sequence. Eventually, he gets a long line of numbers with commas between. He does *not* press RETURN, but backs up to the beginning of the line and types several Insert keys; now he has room to enter the extra information "110 DATA". Now he presses RETURN and the bytes of his machine language program are entered as part of a DATA statement.

There's another line left over and he must repeat the editing sequence to create a 120 DATA ... line and complete the DATA recording of his program. Now he types in line 100:

```
100 FOR J=634 TO 660:READ M:POKE J,M:NEXT J
```

The whole program should now read:

```
100 FOR J=634 TO 660:READ M:POKE J,M:NEXT J
110 DATA 162,48,160,0,232,224,58,208,2,162,48,138,
    32,210,255,200,204,0
120 DATA 3,144,239,169,13,32,210,255
200 DATA 15,10,30,35,28,28,15,0
210 READ V:IF V=0 GOTO 300
220 POKE 768,V
230 SYS 634
270 GOTO 210
300 END
```

Now: whenever we say RUN, the machine language program is POKEd into place (line 100) before the main program uses it. We may now safely SAVE the BASIC program to tape or disk without worrying about how to save the machine language part. The BASIC program makes its own machine language where it needs it.

Serious programmers will think of more efficient ways to convert the program into easily SAVEable code. More advanced machine language programmers will find better ways than DATA statements. But this is a start.

We've traced a machine language program through its conception, assembly, and implementation. It's not too hard a job, and there are aids to help you along the way.

This example was picked because it presented a fairly easy challenge. Even so, it took us three installments to see it through. The bigger jobs are not much harder; most of the work was in the housekeeping.

For beginners who have never followed the whole process through, it's a worthwhile exercise. Once you've seen the trick done, you can see how to do your own. ©

This is the safety net you need to avoid losing hours of programming time when a power failure or a keyboard lockout strikes your Atari.

Atari Variable Table Refresh And Program Backup

Jon Harding
Rochester, NY

Writing new programs, especially long ones, usually requires many hours of editing and typing which can lead to two problems: (1) the variable name table grows too long and (2) the editing session may be catastrophically terminated by a keyboard lockup or power failure. Even commercial software may have variable name tables which are larger than necessary. And power failures occur only at the worst possible time: after typing in the 252nd line of that new game you're so anxious to try! The following routine eliminates the first problem and minimizes the losses of the second.

Type in these two programs (but delete REMs to conserve memory), SAVEing the first as "WRKLOAD.SAV" and LISTing the second as "BACKUP.LST." The meaning of the extensions should be clear. These routines will become part of your program, but can be deleted when you've finished typing. Initiate a new program or add to an existing program, already LOADed, by ENTERing "D:BAKUP.LST". After typing and/or editing five to ten lines, type G.32000 ‹RETURN›. The program in RAM is LISTed to the disk as "WRK," the RAM area cleared, and "WRK" re-ENTERed. Edit or add more code and type G.32000 again.

This time the old "WRK" is RENAMEd "WRK.BAK," the program in RAM becomes "WRK," and is re-ENTERed. The next time, and thereafter, "WRK.BAK" is deleted, "WRK" becomes "WRK.BAK," and the program in RAM becomes "WRK" (and is re-ENTERed).

This procedure provides *father-son (mother-*

daughter) backup and keeps the variable name table as small as possible. When you finish a session, type G.32000 once more to save the latest changes in "WRK." When you start a new session just type ENTER"D:WRK" and you pick up where you left off. If you use this routine to back up other programs on your disk, be sure to save the latest version by its proper name, e.g. "D:GASBILL.UTL." You could also delete the "WRK" files before using the routine on another program, but it's not necessary since they will be replaced anyway.

The beauty of this routine is that it only takes a short command to invoke and can't be started by the program in RAM because of the STOP at 31998. The XIO... code is the special I/O command, explained on page 29 of the *Atari BASIC Reference Manual* which allows you to access DOS utilities without calling DOS, among other things.

Other Possibilities

More generally, this scheme suggests the possibility of a disk with many short utility routines which can be SAVEd for use by themselves or LISTed for use during program writing/editing. Examples might include: listing the disk directory to the screen (without calling DOS), printing the disk directory, a screen viewing of a dumped machine language program, and hex to decimal conversion.

A note of caution: if you write a program which will reside in RAM with the program you're working on, it *must* be complete within itself. In other words, when execution is completed, everything defined must be undefined (variables, dimensions, arrays), all devices OPENed must be CLOSEd, and there must be an END. (The CLR command takes care of undefining anything DIMensioned and clears all variables.) The closing statement of such a routine should look like this: lineno CLOSE #1:CLR:END.

Program 1.

```
32039 REM ****************************
32040 ENTER ":WRK"
32041 REM ****************************
```

Program 2.

```
31997 REM ****************************
31998 ? "SHOULDN'T BE HERE!":STOP
31999 REM *IF 'WRK.BAK' EXISTS, DELETE I
T*
32000 TRAP 32010:XIO 33,#1,0,0,"WRK.BAK"
:TRAP 40000
32005 REM *IF 'WRK' EXISTS, RENAME IT ЮW
RK.BAK'*
32010 TRAP 32020:XIO 32,#1,0,0,"D:WRK,WR
K.BAK":TRAP 40000
32015 REM *LIST PROGRAM TO DISK AS 'WRK'
*
32016 REM *'RUN WRKLOAD.SAV' ERASES RAM
AND RE-ENTERS 'WRK'*
32020 LIST "D:WRK":RUN "D:WRKLOAD.SAV"
32050 REM ****************************
```

©

QUALITY PRODUCTS FOR ATARI COMPUTERS

SWIFTWARE

FROM SWIFTY SOFTWARE

ORDERED BY NASA

Yes, the National Aeronautics and Space Adminsitration selected these three quality products.

HARDWARE
DISK SENTRY TM

SAVE $$

An intellegent digital accessory for your ATARI 810 Disk Drive, lets you selectively write data to both sides of single sided and write protected disks. DISK SENTRY cannot harm your drive or disks. Installs and removes easily; no soldering required. DISK SENTRY's LED signals system status, preventing accidental erasure of data. DISK SENTRY is a convenient push button write-protect override which can pay for itself with your first box of disks. $39.95

SWIFTY TACH MASTER™

An accurate **disk speed diagnostic** utility program designed specifically for ATARI 810 Disk Drives. Provides easy-to-read visual indication of the speed of any drive connected to your system. Using the accuracy of machine language, TACH MASTER displays five RPM readings per second with a working tachometer accurate to ¼ RPM. Allows you to adjust your drive(s) to factory specs easily and at any time in the convenience of your own home. Comes complete with easy to follow user's manual. $29.95

SWIFTY DATALINK TM

High Quality Smart Terminal Communications program. Easy to use Multi-Option, Menu Driven. Full performance uploading/downloading. Works in Duplex or Simplex modes supporting ASCII and ATASCII transmission. Printer Dump, Screen Dump and Disk Search options. Use as remote terminal. Send/receive and **store** programs and data files. Saves connect time charges with commercial services. Requires 24K RAM, 810 Disk Drive, 850 Interface or equivalent, 830 or other 300 Baud modem. (Printer optional) $39.95

NEW GAMES

NEW!

★ **HAUNTED HILL TM**

In this super game you fight bats and ghosts in the dark of a cemetery. This exciting, all machine language game has arcade quality graphics and speed. Requires Joystick. DISK ONLY. $29.95

★ **TRIVIA TREK TM**

A multiple choice game of trivia for one or two players or teams. Comes complete with fifty categories of trivia questions and over two thousand multiple choice answers. A program for creating your own trivia questions and answers is also included. Play it for fun, test your knowledge or entertain friends. Parents and teachers can also make up questions. Great at parties. 32K Disk Only. $29.95

UTILITIES
DISKETTE INVENTORY SYSTEM TM

Use this system to gain control of your expanding disk/program inventory. Quickly get locations of single or multiple copies of your programs and all your valuable files. An invaluable tool, this system is easy and convenient to use and to update. 24K disk system required. $24.95 Printer suggested.

SWIFTY UTILITIES

A valuable collection of programming utilities for the ATARI programmer. This DISK ONLY package includes **all** of Programming Aids I and additional programs designed to make programming time more efficient. Special MENU program runs both saved and listed programs. REM REMOVER eliminates REM statements so programs take less core and run faster. PRINT 825 and PRINTEPS custom print programs prepare condensed, indented and paginated program listings on your ATARI 825 or EPSON MX-80 printer. Listings identify machine code, graphics and inverse video characters. VARIABLE LIST and VARIABLE PRINT programs help you prepare alphabetized annotated list of your program variables. A delete lines utility provides convenience of line deletion while a DOS CALLER gives you convenient access to many DOS utilities while your program is in core. Disklist prepares disk jacket labels. Many of these programs work coresident with each other and with your program. Disk Drive and minimum of 24K required. $29.95

PROGRAMMING AIDS PACKAGE I TM

Four utility programs to help increase programming efficiency and learn more about your computer. RENUMBER handles references and **even** variables. Generates Diagnostic Tables for programming error detection. PROGRAM DECODER, DECIMAL to BCD and BCD to DECIMAL programs give you a practical way of studying internal program representation and ATARI number-conversion procedures. Comes with comprehensive user's manual. 16K cassette $14.95; 24K disk $19.95

PERSONAL DATA MANAGEMENT

★ **FILE-IT 2 TM**

This is Jerry White's popular database system for filing and managing personal and financial information. Create, store, manipulate and retrieve all types of data. Special financial entry and report programs create a powerful personal accounting system. Monthly Bar Graph program shows financial data on the screen and/or printer. Supports up to four disk drives as well as the Axlon Ramdisk (if you have one). Mailing list program generates mailing labels in one or two across format. Extensively documented in a ring binder. 24K Disk and Printer. $49.95 + $3.50 for shipping and handling.

FUN "n" GAMES #1 TM

WORDGAMES, POSSIBLE and LEAPFROG giving you hours of fun, challenge and entertainment. WORDGAMES, two games in one, contains GUESSIT - a deductive alphabetic reasoning game for one or two players and WORDJUMBLE - a multiple word descrambling puzzle with play-on-word hints and mystery answers. Instructions show how you can substitute your own words. Use POSSIBLE to help descramble word jumble puzzles or to create your own. All letter/number combinations or permutations of input are printed to screen or optional printer. LEAPFROG is a Chinese-Checker type jumping game in which you try to position two sets of animated jumping frogs in a minimum number of moves. 16K Cassette $19.95; 24K Disk $24.95. Disk version of GUESSIT works with VOTRAX Type "n" TALK. A real crowd pleaser.

COMING SOON! Space Shuttle Adventure Series TM
Real-time Space Flight Simulations

COMING SOON! The Family Financier TM
AN easy to use financial package.

ACCESSORIES
VINYL DUST COVERS

Custom sewn upholstery grade vinyl dust covers for ATARI 800 and 400 computers and the 810 and 825 peripherals. Protecting your equipment when not in use, these crisp black covers go well with any decor. Specify model(s) when ordering. $14.95 each.

send check or money order including $2.50 Shipping and Handling to:

SWIFTY SOFTWARE, INC.
64 BROAD HOLLOW ROAD
MELVILLE, N.Y. 11747
(516) 549-9141

MasterCard VISA

N.Y. Residents add 7¼ % sales tax

send for free catalogue dealer orders and c.o.d.'s accepted

GREAT NEW PROGRAMS FOR ATARI COMING FROM SWIFTY!

ATARI® A Warner Communications Company

SPACE SHUTTLE ADVENTURE
HAUNTED HILL
MAILIFT
FILE-IT C
MORE!

FOR DETAILS SEE OUR AUGUST COMPUTE! AD

How To Use The 6560 Video Interface Chip

Dale Gilbert
Henegar, AL

The 6560 Video Interface Chip, the VIC chip, provides low cost, high resolution, color video to a color monitor or a color television, and it also incorporates a sound generator, A/D converters [*analog/digital*], and even a light pen feature.

The 6560, some RAM, a crystal, a few bus drivers, and a little decode logic is all the hardware that is required to add color and sound to a microprocessor that has an expansion bus.

The VIC capabilities include on-chip sync generation, screen grid size of up to 192 horizontal dots by 200 vertical dots, two character sizes, three independent programmable tone generators, a white noise generator, an amplitude modulator, screen centering, on-chip DMA address generation, and two modes of color operation.

The 6560 VIC is manufactured by MOS Technology, Inc.. Commodore Business Machines incorporates the 6560 in their VIC-20 Computer. I purchased my chip from Falk-Baker Associates, 382 Franklin Avenue, Nutley, New Jersey 07110, for $14.95.

6560 Software

To produce colored characters, VIC addresses two blocks of memory at the same time. This address method produces twelve bits of data. The eight bit block of memory is called the Character Pointer Block (called the *screen memory* on the VIC-20).

The second block of memory is called the Character Color Block (called the *color nibble area* on the VIC-20). This block contains four-bit character color data.

VIC takes the character pointer data, left shifts it three times, and adds the result to the character cell base address contained in bits zero through three of register five.

VIC then puts the result on the address bus which addresses another block of memory called the Character Cell Block (called character bit maps on the VIC-20). This block of memory is eight bits wide. The data obtained from this address is video information on an 8 x 8 character matrix. The matrix is eight bytes high and eight bits wide.

VIC takes the four-bit character color data and, if the MSB is 0, the character matrix will be displayed in high resolution mode. If the MSB is one, the character matrix will be displayed in the multicolor mode.

When the high resolution mode is selected and when bit-3 of register-F is a zero, all one bits of the character cell data will be displayed in the background color and all zero bits will be in the foreground color. The three remaining bits of the character color data specify the color of the foreground. The color of the background is specified by bits four through seven of register-F. If bit-3 of register-F is 1, the one bits of the character cell data will be displayed in foreground color and the zero bits will be displayed in background color.

If bit-3 of register-F is one, all the character cell matrix will have the same color background. If bit-3 is a zero, all the character cell matrix will have common character colors.

When the multicolor mode is selected (MSB of the character color data is one), there is a pairing of bits of the character cell data. The character matrix now is a 4 x 8 dot matrix with each dot's color determined by the code of each pair. The code has four possibilities: 00,01,10,11. If a dot code is 00, its color is the background color specified by bits four through seven of register-F. If the code is 01, the dot color is the same as the external border color specified by bits zero through two of register-F. If the code is ten, the dot color is the foreground color specified by the three bits of the character color data. If the code is 11, the color of the dot is specified by bits four through seven of register-E.

VIC produces a TV raster of up to twenty-two columns by up to twenty-three rows of character matrix surrounded by a border. The base address of the character pointer block contains the first upper left pointer for that character matrix. The base address plus one of the character pointer blocks contains the pointer for the next right character matrix. It is the responsibility of the MPU to manipulate the pointers in the character pointer block of memory. A whole raster of repeated characters (character matrix) can be obtained by just repeating the pointers in the character pointer block.

6560 Hardware

The 6560 VIC has fourteen address pins (A0-A13) and twelve data pins (D0 - D11). When the 02 clock is high, the MPU can place an address on the address pins and read or write data into any of the sixteen eight-bit registers via data pins D0 - D7. VIC decodes the address pins and selects registers zero through F when address 1000 through 100F

(the VIC-20's VIC chip ignores the A 15 line) is placed on the address pins.

The address pins are input pins when 02 is one; if 02 is zero, then these pins are output address pins.

When the 02 clock is one, the MPU can also write or read data to the character pointer RAM, the character color RAM, and the character cell RAM. The character cell memory may be RAM, ROM, or both. The base address of the character cell block and/or the character pointer can be changed by modifying a register in the VIC.

When the 02 clock is low, the VIC addresses memory in such a way that the character pointer RAM and the character color RAM is selected at the same time. VIC must receive character pointer data on D0 through D7 and character color data on D8 through D11 at the same time.

The R/W pin four is an input only pin and must be driven by the MPU when 02 is one and held high when 01 is one.

Pins 38 and 39 are the master clock inputs. The 6560 VIC requires a 14.31818 MHz, two phase, five volt, non-overlapping signals. The master clock uses a standard 14.31818 MHz crystal (4x color) and the delay of 74LS gates to give a non-overlapping signal. Resistors R1 and R2 are used to give extra pull up to CMOS levels. CMOS gates don't seem fast enough for this clock.

Pins 35 and 36 are the system output clock used for system timing and driving the clock of the 6512 MPU (if used). A 6502 MPU can be used by feeding pin 36 to the 00 IN pin on the 6502. Removing the original 00 signal and wiring this new 00 is the only alteration needed on the mother MPU.

Because the 6502 address lines are active when 01 is 1, the expansion address lines to the VIC and its associated memory must be isolated from the MPU bus during this time.

The data bus should also be buffered and gated for this same reason.

The system clocks are five volt, non-overlapping, 1.02 MHz signals.

Pin 19 provides the sound output which must be fed to an amplifier to a drive speaker. The output impedence is approximately 1000 ohms.

Pin 3 is the output pin for the composite sync and the luminance signal. This pin is an open drain which makes it easy to shift to the needed voltage level for a RF modulator, TV first video amplifier, or a black and white CRT monitor.

In the following wiring diagram, diodes K3, K4, K5, and K6; resistors R5, R6, and R7; and C1 make up the level shifter for a TV output or video monitor output.

The VIC is a superior CRT controller for a B/W monitor due to the varying levels of luminance required for a color picture. VIC can produce varying shades of gray.

Pin 2 provides the composite color signal. This signal contains the color phase and amplitude information plus the 3.58MHz burst signal. Pin 2 is a high impedence output buffer which can be applied to the first chroma amplifier of a TV, color monitor, or RF modulator.

Pins 17 and 18 are the input pins for the Pot-X and Pot-Y analog to digital converters. A pot is used to charge an external capacitor tied to the pot wiper and fed to pin 17 or 18. These pins are systematically pulled to ground after each charge voltage reading. The voltage is digitized and deposited in register 8 or 9.

Pin 37 is the light gun/pen pin. The voltage of triggering is approximately 2.5 volts on the falling edge. Holding this pin low clears registers 6 and 7. The values of registers 6 and 7 represent the horizontal and vertical positions of the current dot being scanned. The light gun/pen option is only available on a 6560-101 which is sometimes identified by a white dot on the case of the IC.

Address decoding must be provided so that the character color nibble RAM will be selected, when the character pointer RAM is addressed by the VIC, but not selected when the MPU addresses the character pointer RAM. A data bus transceiver must be provided to isolate the nibble bus from the byte bus along with the logic for this transceiver. The logic must enable the MPU to read or write to the nibble bus when the 02 clock is one.

The expansion 02 may be used as the E(02) if the expansion 02 has no more than one LS gate delay and it is the true MPU 02. If these provisions can't be met, connect E(02) to the V02 (pin 36).

This author used addresses 1000 through 100F for the VIC; 2000 through 23FF for the character pointer block; 2400 through 27FF for the character color block; and 2800 through 2BFF for the character cell block.

The addition of a 6560, a few RAM chips, and a few gates will free up and complement many a MPU.

Color Code

Auxiliary/Background/Border/Foreground

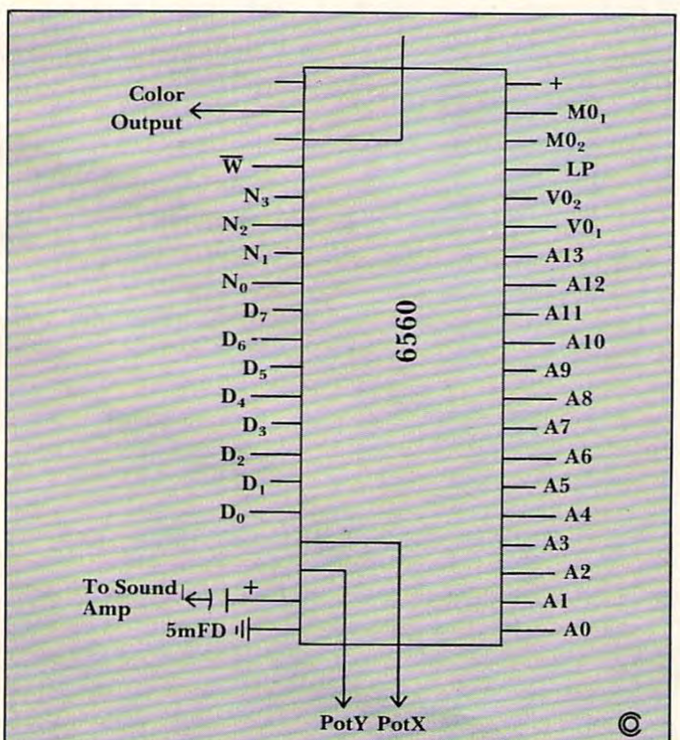
0 BLACK	**8 ORANGE**
1 WHITE	**9 LIGHT ORANGE**
2 RED	**A PINK**
3 CYAN	**B LIGHT CYAN**
4 MAGENTA	**C LIGHT MAGENTA**
5 GREEN	**D LIGHT GREEN**
6 BLUE	**E LIGHT BLUE**
7 YELLOW	**F LIGHT YELLOW**

VIC Control Registers

Register	Data	Address
R0	I = I / O = N | Horiz. Center Vx4 Dots: 6 5 4 3 2 1 0	1000
R1	Vertical Center Vx2 Dots: 7 6 5 4 3 2 1 0	1001
R2	Cp A9 | No. of Character Matrix Columns: 6 5 4 3 2 1 0	1002
R3	RV 0 | No. of Character Matrix Rows: 5 4 3 2 1 0 | 1 = D / 0 = S	1003
R4	Raster Value: 8 7 6 5 4 3 2 1	1004
R5	Base Cp: A13 A12 A11 A10 | Base Char. Cell: A13 A12 A11 A10	1005
R6	LP Horizontal Position: 7 6 5 4 3 2 1 0	1006
R7	LP Vertical Position: 7 6 5 4 3 2 1 0	1007
R8	Pot-X: 7 6 5 4 3 2 1 0	1008
R9	Pot-Y: 7 6 5 4 3 2 1 0	1009
RA	Sw 1 = On | Frequency Osc. -1: 6 5 4 3 2 1 0	100A
RB	Sw 1 = On | Frequency Osc. -2: 6 5 4 3 2 1 0	100B
RC	Sw 1 = On | Frequency Osc. -3: 6 5 4 3 2 1 0	100C
RD	SW 1 = On | Frequency Noise Gen.: 6 5 4 3 2 1 0	100D
RE	Auxiliary Code: 3 2 1 0 | Audio Amplitude: 3 2 1 0	100E
RF	Background Code: 3 2 1 0 | 1 = B / 0 = F | Border Code: 2 1 0	100F

Abbreviations:

I = Interlace	S = 8x8 Matrix
N = None Interlace	D = 16x8 Matrix
CP = Character Pointer Base Address	B = Common Background
R = Raster	F = Common Foreground
V = Value	LP = Light Pen
	Sw = Switch



This Publication is available in Microform.

University Microfilms International

Please send additional information

for ______________________ (name of publication)

Name ______________________

Institution ______________________

Street ______________________

City ______________________

State ____________ Zip ____________

300 North Zeeb Road, Dept. P.R., Ann Arbor, Mi. 48106

NEW COMMODORE ADD-ONS RAM-ROM: GETS RID OF SAFING ROM

MX-910 CBM/PET RAM/ROM;
Allows multi ROM protected programs using the same socket to be put onto diskette/cassette, no need to insert protect ROM in socket after initial load, eliminates need for ROM switch box, write protect in software, decoded for dual ROM socket usage, 4K expandable to 8K, easy internal CBM installation: $119.95

MX-232 CBM/PET TO RS-232C INTERFACE:
Low cost, bidirectional, 50 to 19,200 baud rate, full modem controls, parity allows for two RS-232C CBM ports, installs easily inside CBM: $199.95

SX-100 IEEE-488/PET MODEM SOFTWARE:
Best 810 modem software, by 8010 developer, works with Source/Micronet/CBM to disk/CBM to CBM; Intelligent Terminal Software: $79.95

MX-200 IEEE-488/PET PARITY MODEM/SOFTWARE
Talk to a host computer requiring parity, all features of SX232: $399.95

MX-113 THEFT PROTECTION ROM:
Plug in ROM, displays owner's name, etc. when computer turned on: $49.95

ECX COMPUTER COMPANY

2678 NORTH MAIN ST.
WALNUT CREEK, CA 94596
(415)944-9277

For additional new product information and catalog send self addressed, stamped envelope.

...PET/CBM/VIC? SEE SKYLES...

PET owners everywhere sing *Thanks for the Memories* to good old Bob Skyles

. . . they should . . . because Bob Skyles is the only complete source for memory boards for *any* PET ever sold. Old Bob won't forget you.

And the Skyles memory systems have the highest quality control of any computer product ever. Over 100 million bits of Skyles memory boards are already in the field. First quality static and dynamic RAMS, solid soldered on first quality glass epoxy. That is why they are **guaranteed**—*in spite of the new lower prices*—**for a full two years.**

The boards, inside the PET/CBM, install in minutes without special tools or equipment . . . just a screwdriver.

Because of our new dynamic memory design, and to celebrate old Bob's $30\frac{th}{16}$ birthday, here are the smashing new prices:

8K Memory System	*orig. $250.00*	*now $200.00*	*Save $ 50.00*
16K Memory System	*orig. $450.00*	*now $300.00*	*Save $150.00*
24K Memory System	*orig. $650.00*	*now $400.00*	*Save $250.00*

. . . *For any PET ever made.* When ordering, just describe your PET by model number and indicate the amount and type (or brand) of memory currently in the unit.

Shipping and Handling . . . . . (USA/Canada) **$3.50** *(Europe/Asia)* **$15.00**

California residents must add 6%/6½% sales tax, as required.

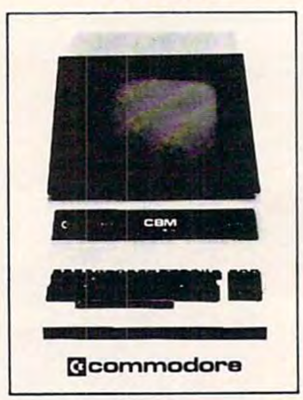
Visa/Mastercard orders: call tollfree **(800) 227-9998** (except California). California orders: please call **(415) 965-1735.**

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

...PET/CBM/VIC? SEE SKYLES...

NEW FOR '82

• PAYROLL SYSTEM •



- OVER 100 EMPLOYEES
- 25 DATA FIELDS PER EMPLOYEE
- MULTIPLE RATES
- OVERTIME
- VACATION
- SUBSISTENCE
- 16 GEN. LEDGER CATEGORIES
- PRINTS CHECKS OR CHECK STUBS
- W2 FORMS
- CALIF. DE3 AND DE3B, 941 AND OTHER REPORTS

$295.00
(INTRODUCTORY PRICE)

FOR MOD 8032 (4040 OR 8050)

OTHER BUSINESS PROGRAMS AND PRICE LIST AVAILABLE

DELTA SOFTWARE CO.
3436 MENDOCINO AVE.
SANTA ROSA, CA 95401
(707) 544-3425

This compactor analyzes and compresses BASIC at lightning speed. For PET/CBM, Upgrade and 4.0 BASIC, with disk drive.

Machine Language Compactor

David L. Evans
Caldwell, ID

When the programs "Compactor" and "Un-Compactor" by Robert Baker appeared in the Sept./Oct., 1980 and May, 1981, issues, respectively, of **COMPUTE!** I immediately typed them in and tried them out on several of my programs.

To my dismay, compacting a 16K program required approximately 48-52 minutes. I am a college student and spend a limited amount of time on my computer (a 16K 4.0 PET) due to study time and my job. That compacting time ate heavily into the time I spend on my PET and I decided to do something about it. This program is the result.

I tried several different BASIC versions of the program, but could come up with no noticeable increase in speed. I then decided to give machine language a try. The result? A 947% increase in speed! Even this phenomenal increase in speed can be improved a bit further by eliminating the error checking in my input subroutine.

The program consists of five different sections. The input section prints the directions and inputs the input and output file names. The scanning section scans the BASIC program for target line numbers and prints them for you to see, while also storing them for use by the compacting section. The compacting section compacts the program (where possible) and prints "DONE" when finished.

A following subroutine section contains an input subroutine, a line number checking routine, and a disk error routine. Following these four sections is the data and work storage area. This program was designed to run on both Upgrade and 4.0 ROMs. It will automatically adjust itself to work on either ROM set, so all the user has to do is to LOAD the PROGRAM and type RUN.

Running the program is fairly simple. Load the program into memory with the LOAD or DLOAD command, depending upon which computer you have. If you do not have DOS 2.0 or higher, make sure that both drives are initialized. Type RUN, and the computer will print a short greeting along with some brief directions.

Typing In Compactor

Below are the step-by-step instructions for entering and SAVEing Compactor. **COMPUTE!** generally provides BASIC loaders so that readers who don't know machine language can easily type in and use machine language subroutines and programs. Though it could be done, we felt that creating a BASIC loader for Compactor would be too complicated to be worthwhile: Compactor sits in memory starting at address 1024 (like a BASIC program). A loader would involve an awkward process of writing over itself and checksums would be equally difficult to use.

Instead, Compactor is presented as *hex dump*. To enter it into the computer:

1. Type: SYS 1024 (to enter the machine language monitor).

2. When prompted by the period (.), type M 0400 0460 and the screen will reveal (in hexadecimal numbers) the contents of the memory cells between address 0400 and 0460. You can type over these numbers and hit RETURN on each line, thereby entering the new numbers into memory. So, for the first line, you type: 00 0B 04 FF FF 9E 31 30 (RETURN). Continue on until you've typed and entered line 0460 as it appears in the magazine listing.

3. Then do the next block of memory. Type: M 0468 0500 and continue on, replacing the values on screen with Compactor's numbers as printed. When you've typed in the last line, (0B70), you are done.

4. Now Compactor is in your computer's memory and it must be SAVEd onto a disk. Type: S "0:COMPACTOR", 08,0400,0B78 (RETURN) and it will now be available for use as a program called "COMPACTOR" on the disk in Drive zero.

5. If the power went off just before you finished (and you lost everything, including your temper), you might prefer to take up the author's offer (at the end of the article) to make a copy for you.

After reading the directions, type in your input file name preceded by the drive number the file is on. Example : 0:COMPACTOR. You will then be asked for the output file name; do the same with it as you did with the input file name. The computer will then scan and compact your program. If any disk error is detected, the program will report it and will return to BASIC. After your program has been compacted, LOAD it into memory and type the CLR command; this will fix all the line links. Be sure to reSAVE the corrected copy.

Clearly, machine language lends itself to input/output programs more readily than BASIC does. Once the basics of how to open files from machine language are learned (see **COMPUTE!**, April, 1981, #11), it is easy to generate input/output programs in machine language which will run at a phenomenal speed.

For those of you who do not want to type this in, I will make you a copy of it if you will send me $3.00 and a self-addressed, stamped mailer and a blank tape or disk. For those who send a disk, I have DOS 2.0, so all disks will be written in DOS 2.0.

David Evans
2202 Ellis Ave.
Caldwell, ID 83605

```
0400 00 0B 04 FF FF 9E 31 30
0408 33 37 00 00 00 AD 50 C3
0410 C9 2E F0 38 A9 24 8D F0
0418 04 8D 30 05 8D EC 06 8D
0420 09 07 A9 AE 8D 69 06 8D
0428 6E 06 8D A7 06 8D 89 09
0430 8D 91 09 8D 99 09 A9 D9
0438 8D 88 06 8D 5E 07 8D DA
0440 08 A9 DC 8D 89 06 8D 5F
0448 07 8D DB 08 A9 78 85 01
0450 A9 0B 85 02 A0 00 A9 00
0458 91 01 C8 D0 F9 E6 02 A5
0460 02 C9 16 D0 EF A9 0D 8D
0468 88 0B A9 00 8D 87 0B A2
0470 00 BD 48 0A C9 00 F0 07
0478 20 D2 FF E8 4C 71 04 A2
0480 00 20 CF FF 9D 50 0C C9
0488 0D F0 08 E8 E0 12 F0 03
0490 4C 81 04 E8 A9 0D 9D 50
0498 0C AD 50 0C C9 30 F0 04
04A0 C9 31 D0 CB AD 51 0C C9
04A8 3A D0 C4 A2 00 BD 09 0B
04B0 C9 00 F0 07 20 D2 FF E8
04B8 4C AD 04 A2 00 20 CF FF
04C0 9D 70 0C C9 0D F0 08 E8
04C8 E0 12 F0 03 4C BD 04 AD
04D0 70 0C C9 30 F0 04 C9 31
04D8 D0 95 AD 71 0C C9 3A D0
04E0 8E A9 0F 85 D2 85 D3 A9
04E8 08 85 D4 A9 00 85 D1 20
04F0 63 F5 20 F7 09 A2 00 BD
04F8 50 0C C9 0D F0 04 E8 4C
0500 F7 04 A9 2C 9D 50 0C 9D

0510 52 9D 53 0C E8 E8 E8 E8
0518 8E 86 0B 86 D1 A9 05 85
0520 D2 85 D3 A9 08 85 D4 A9
0528 50 85 DA A9 0C 85 DB 20
0530 63 F5 20 F7 09 20 CE 09
0538 20 CE 09 18 AD 78 0B 6D
0540 79 0B B0 07 C9 00 D0 03
0548 4C A1 06 20 CE 09 AD 78
0550 0B 8D 7A 0B AD 79 0B 8D
0558 7B 0B 20 D7 09 AD 79 0B
0560 C9 00 F0 D4 C9 89 F0 08
0568 C9 8D F0 04 C9 A7 D0 EA
0570 A9 00 8D 7C 0B 8D 7D 0B
0578 20 D7 09 AD 79 0B C9 3A
0580 B0 DB 90 03 20 D7 09 AD
0588 79 0B C9 20 F0 EA C9 3A
0590 B0 58 C9 30 90 54 A9 0A
0598 85 01 A9 00 8D 7E 0B A2
05A0 08 0A 2E 7E 0B 06 01 90
05A8 09 18 6D 7C 0B 90 03 EE
05B0 7E 0B CA D0 EC 8D 7C 0B
05B8 AD 7D 0B 0A 85 01 0A 0A
05C0 18 65 01 6D 7E 0B 8D 7D
05C8 0B 38 AD 79 0B E9 30 8D
05D0 79 0B 18 AD 79 0B 6D 7C
05D8 0B 8D 7C 0B 90 03 EE 7D
05E0 0B 20 D7 09 AD 79 0B 4C
05E8 8E 05 AD 7C 0B 8D 83 0B
05F0 AD 7D 0B 8D 84 0B 20 9C
05F8 09 AD 85 0B C9 FF F0 71
0600 AD 87 0B C9 00 D0 07 AD

0610 0B 8D 1E 06 AD 88 0B 8D
0618 1F 06 AD 7C 0B 8D FF FF
0620 EE 87 0B AD 87 0B C9 00
0628 D0 0A EE 88 0B AD 88 0B
0630 C9 16 B0 22 AD 87 0B 8D
0638 44 06 AD 88 0B 8D 45 06
0640 AD 7D 0B 8D FF FF EE 87
0648 0B AD 87 0B C9 00 D0 03
0650 EE 88 0B 4C 71 06 A2 00
0658 BD 39 0B C9 00 F0 07 20
0660 D2 FF E8 4C 58 06 A9 05
0668 20 E2 F2 A9 0F 20 E2 F2
0670 60 A2 00 BD 1F 0B C9 00
0678 F0 07 20 D2 FF E8 4C 73
0680 06 AE 7C 0B AD 7D 0B 20
```

```
0688 83 CF AD 79 0B C9 2C D0
0690 03 4C 70 05 C9 20 F0 03
0698 4C 5D 05 20 D7 09 4C 8A
06A0 06 20 CC FF A9 05 20 E2
06A8 F2 A2 00 BD 49 0B C9 00
06B0 F0 07 20 D2 FF E8 4C AB
06B8 06 A2 00 BD 70 0C C9 0D
06C0 F0 04 E8 4C BB 06 A0 00
06C8 B9 70 0B C9 00 F0 08 9D
06D0 70 0C C8 E8 4C C8 06 86
06D8 D1 A9 06 85 D2 85 D3 A9
06E0 08 85 D4 A9 70 85 DA A9
06E8 0C 85 DB 20 63 F5 20 F7
06F0 09 AE 86 0B 86 D1 A9 05
06F8 85 D2 85 D3 A9 08 85 D4
0700 A9 50 85 DA A9 0C 85 DB


0710 09 A2 06 20 C9 FF AD 78
0718 0B 20 D2 FF AD 79 0B 20
0720 D2 FF 20 CC FF A9 00 8D
0728 7F 0B 20 CE 09 AD 78 0B
0730 8D 80 0B AD 79 0B 8D 81
0738 0B A9 00 8D 82 0B 18 AD
0740 78 0B 6D 79 0B B0 07 C9
0748 00 D0 03 4C 63 09 20 CE
0750 09 AE 78 0B 8E 83 0B AD
0758 79 0B 8D 84 0B 20 83 CF
0760 20 D7 09 AD 79 0B C9 20
0768 F0 F6 C9 3A F0 F2 C9 00
0770 F0 0E C9 8F D0 19 20 D7
0778 09 AD 79 0B C9 00 D0 F6
0780 A9 01 8D 82 0B 20 9C 09
0788 AD 85 0B C9 FF D0 9B A2
0790 06 20 C9 FF AD 80 0B 20
0798 D2 FF AD 81 0B 20 D2 FF
07A0 AD 83 0B 20 D2 FF AD 84
07A8 0B 20 D2 FF 20 CC FF A9
07B0 04 8D 7F 0B AD 82 0B C9
07B8 00 F0 12 A2 06 20 C9 FF
07C0 A9 3A 20 D2 FF 20 CC FF
07C8 A9 05 8D 7F 0B A9 00 8D
07D0 82 0B 4C E9 07 A2 06 20
07D8 C9 FF AD 79 0B 20 D2 FF
07E0 20 CC FF EE 7F 0B 20 D7
07E8 09 AD 79 0B C9 89 F0 08
07F0 C9 8B F0 04 C9 A7 D0 05
07F8 A9 01 8D 82 0B AD 79 0B
0800 C9 00 D0 03 4C 9C 08 C9


0810 D7 09 AD 79 0B C9 00 D0
0818 F6 4C 9C 08 AD 79 0B C9
0820 22 D0 46 A2 06 20 C9 FF
0828 AD 79 0B 20 D2 FF 20 CC
```

```
0830 FF EE 7F 0B 20 D7 09 AD
0838 79 0B C9 22 F0 97 C9 00
0840 D0 E1 AD 82 0B C9 00 F0
0848 0D A9 00 8D 79 0B A2 06
0850 20 C9 FF 4C 1C 07 A2 06
0858 20 C9 FF A9 22 20 D2 FF
0860 20 CC FF EE 7F 0B 4C 9C
0868 08 AD 79 0B C9 3A F0 03
0870 4C D5 07 20 D7 09 AD 79
0878 0B C9 20 F0 F6 C9 3A F0
0880 F2 C9 8F F0 8A C9 00 F0
0888 13 A2 06 20 C9 FF A9 3A
0890 20 D2 FF 20 CC FF EE 7F
0898 0B 4C E9 07 AD 82 0B C9
08A0 00 D0 07 AD 7F 0B C9 AB
08A8 90 0D A9 00 8D 79 0B A2
08B0 06 20 C9 FF 4C 1C 07 20
08B8 CE 09 18 AD 78 0B 6D 79
08C0 0B B0 07 C9 00 D0 03 4C
08C8 63 09 20 CE 09 AE 78 0B
08D0 8E 83 0B AD 79 0B 8D 84
08D8 0B 20 83 CF 20 9C 09 AD
08E0 85 0B C9 FF F0 2C 20 D7
08E8 09 AD 79 0B C9 8F D0 03
08F0 4C 0F 08 C9 20 F0 EF C9
08F8 3A F0 EB C9 00 F0 B8 A2
0900 06 20 C9 FF A9 3A 20 D2


0910 E9 07 A2 06 20 C9 FF A9
0918 00 20 D2 FF A9 01 20 D2
0920 FF 20 D2 FF AD 83 0B 20
0928 D2 FF AD 84 0B 20 D2 FF
0930 20 CC FF A9 04 8D 7F 0B
0938 20 D7 09 AD 79 0B C9 20
0940 F0 F6 C9 3A F0 F2 C9 00
0948 F0 04 C9 8F D0 0D A2 06
0950 20 C9 FF A9 3A 20 D2 FF
0958 20 CC FF A9 00 8D 82 0B
0960 4C E9 07 A2 06 20 C9 FF
0968 A9 00 20 D2 FF 20 D2 FF
0970 20 D2 FF 20 CC FF A2 00
0978 BD 65 0B C9 00 F0 07 20
0980 D2 FF E8 4C 78 09 A9 06
0988 20 E2 F2 20 F7 09 A9 05
0990 20 E2 F2 20 F7 09 A9 0F
0998 20 E2 F2 60 A9 0C 85 02
09A0 A9 00 85 01 A8 AA AD 83
09A8 0B D1 01 D0 09 C8 AD 84
09B0 0B D1 01 F0 13 88 C8 C8
09B8 D0 EC E6 02 A5 02 C9 16
09C0 D0 DE A9 00 8D 85 0B 60
09C8 A9 FF 8D 85 0B 60 20 D7
09D0 09 AD 79 0B 8D 78 0B A2
09D8 05 20 C6 FF 20 E4 FF 85
09E0 01 A9 01 8D 89 0B 20 F7
09E8 09 A5 01 8D 79 0B A9 00
```

```
09F0 8D 89 0B 20 CC FF 60 A2
09F8 0F 20 C6 FF A2 00 20 E4
0A00 FF C9 0D F0 07 9D A0 0C


0A10 20 CC FF AD A0 0C C9 30
0A18 D0 08 AD A1 0C C9 30 D0
0A20 01 60 A9 0D 20 D2 FF A2
0A28 00 BD A0 0C 20 D2 FF E0
0A30 00 F0 04 E8 4C 29 0A 68
0A38 68 AD 89 0B C9 00 F0 07
0A40 A9 00 8D 89 0B 68 68 60
0A48 93 20 20 20 20 20 20 20
0A50 4D 41 43 48 49 4E 45 20
0A58 4C 41 4E 47 55 41 47 45
0A60 20 43 4F 4D 50 41 43 54
0A68 4F 52 0D 0D 0D 42 59 3A
0A70 20 44 41 56 49 44 20 45
0A78 56 41 4E 53 0D 0D 0D 45
0A80 4E 54 45 52 20 46 49 4C
0A88 45 20 4E 41 4D 45 20 57
0A90 49 54 48 20 54 48 45 20
0A98 44 52 49 56 45 20 4E 55
0AA0 4D 42 45 52 0D 0D 50 52
0AA8 45 43 45 44 49 4E 47 20
0AB0 49 54 2E 0D 0D 45 58 41
0AB8 4D 50 4C 45 3A 20 20 20
0AC0 30 3A 46 49 4C 45 4E 41
0AC8 4D 45 0D 0D 44 4F 20 54
0AD0 48 45 20 53 41 4D 45 20
0AD8 57 49 54 48 20 54 48 45
0AE0 20 4F 55 54 50 55 54 20
0AE8 46 49 4C 45 20 4E 41 4D
0AF0 45 2E 0D 0D 0D 0D 49 4E
0AF8 50 55 54 20 46 49 4C 45
0B00 20 4E 41 4D 45 20 3F 20


0B10 54 20 46 49 4C 45 20 4E
0B18 41 4D 45 20 3F 20 00 93
0B20 46 4F 55 4E 44 20 54 41
0B28 52 47 45 54 20 4C 49 4E
0B30 45 20 4E 55 4D 42 45 52
0B38 00 93 54 4F 4F 20 4D 41
0B40 4E 59 20 4C 49 4E 45 53
0B48 00 93 43 4F 4D 50 41 43
0B50 54 49 4E 47 20 4C 49 4E
0B58 45 20 4E 55 4D 42 45 52
0B60 2E 2E 2E 2E 00 93 12 44
0B68 4F 4E 45 0D 0D 0D 0D 00
0B70 2C 50 2C 57 00 00 00 00
```

©

Insight: Atari

Bill Wilkinson
Optimized Systems Software
Cupertino, CA

This month I will first respond to some of that unanswered mail; then part six of Inside Atari BASIC [*a continuing series within this column*] will delve further into string and array magic; and finally we will do a preliminary exploration of the depths of Atari's FMS.

Graphics Revisited

Actually, the title of this section might better be "machine language revisited." Probably none of my columns has generated as much response as part four of my Atari I/O series, subtitled "Graphics," in the February, 1982, issue of **COMPUTE!** Unfortunately, most of the response has been of the "I can't make it work" variety. Of course, my first response is "but I *know* it works!" Yet still the letters ask, "How?"

I do not intend to turn this column into a tutorial on machine language. There are several good books available on 6502 machine language (including the Inmans' book specifically for *The Atari Assembler*), and any struggling beginner who is trying to make do without at least one of them is simply a masochist. However, my ego says that it will be better fed if more readers understand my articles.

For the most part, it seemed that those who had trouble with my February article assumed that what was published was some neat program to be used as is. *Not so!* I had simply given you a set of *subroutines* to use with your own programs. For an example, let us take a simple BASIC routine and its machine language equivalent. First, the BASIC:

```
30000  POKE 20,0 : POKE 19,0
30010  IF PEEK(19)=0 THEN 30010
30020  RETURN
```

Now, you would not mistake that for a complete BASIC program. But, if I told you that entering this routine and then executing a GOSUB 30000 from your program would produce a 4.2667-second pause, you would know when and how to use it. So let's do the same thing in machine language:

```
PAUSE
        LDA #0
        STA 20 ; "poke 20,0"
        STA 19 ; "poke 19,0"
LOOP
        LDA 19
        BEQ LOOP ; "if peek(19)=0 then loop"
        RTS        ; "return"
```

Again, this is *not* a complete program! *But* if you enter it (say at the end of your own machine language program) and then execute a JSR PAUSE, it will produce a 4.2667 second pause. Note, then, that JSR in machine language is the equivalent of GOSUB in BASIC.

The graphics routines (Program 5) in my February article are just subroutines, to be placed in your own machine language program and then JSRed to perform their actions. Perhaps the biggest mistake I made was in presenting these as an assembled listing (complete with "*=$660"). I certainly never used them as such. In point of fact, I tested them by .INCLUDEing them in my test programs, which were written with the OSS EASMD Assembler/Editor. And one of the test programs I used was, indeed, the example given on page 77 of that same article.

So how do *you* get these subroutines in and working for you? First and foremost, you obviously must type in all that code. Perhaps the best thing to do would be to type it in exactly as shown, including even the "*= $660" and the ".END". Then assemble it and carefully compare the object code generated with that in the magazine. When all appears correct, remove the "*=" line and the ".END" line, renumber the whole thing (I would suggest REN 29000,5 or something similar), and LIST it to diskette or cassette. Now use NEW and write your mainline code. When you are reasonably satisfied with it, LIST it to disk or cassette also.

Now what? Obviously, if you have OS/A+, I suggest you use .INCLUDE (an assembler pseudo-op which allows you to include one file while assembling another). In fact, I tend to write assembly code structured as follows:

```
.INCLUDE #D:SYSEQU.ASM
<my mainline code>
.INCLUDE #D:library-routine-number-1
.INCLUDE #D:library-routine-number-2
...
.END
```

If my "mainline" code is big enough, I may even break it into two or three pieces and .INCLUDE each of them separately.

But what if you don't have .INCLUDE capability? Well, several assemblers have "FILE" or "CHAIN," which are not quite as flexible (since you don't return to where you left off after you have assembled a chained-to file...thus making the procedure next to useless for zero page equate files, etc.); but the principle is generally the same: put your mainline code first and then CHAIN to the subroutine files.

And what if you have the Assembler/Editor cartridge? (For all of its faults, it is still a remarkably flexible tool, especially considering that it is usable with cassette-based systems.) Again, the principle holds. The only real difference is that you must do the INCLUDEs yourself. How? Via the ENTER command. If you haven't noticed it up until now, get your manual and read up on the ",M" option of ENTER. You can merge two or more machine language program files (including cassette files) via the ",M" option! Just as you can with BASIC, except that BASIC always presumes you want to merge.

Are there things to watch out for? Of course. Would I ever give you a method without a handful of caveats? (1) If you ENTER/merge a file with line numbers which match some (or all) of those in memory, you will overwrite the in-memory lines. (2) If you EVER forget the ",M" option, you will wipe out everything in memory so far. (3) You won't find out about duplicate labels until you assemble the whole thing.

But even with all these cautions, I *strongly* recommend that you store each of your hard-earned routines on its own file/cassette. It then becomes almost easy to write the next program that needs some of those same routines.

By the way, caution number 1 in the previous paragraph is the reason I suggested RENumbering the graphics routines to 29000, or some such out of the way place. If you make notes of what each file (or cassette) does, as well as what line numbers it occupies, you can build a powerful library. And a P.S.: generally, .INCLUDE, FILE, and CHAIN commands do not require unique line numbers, so you need not worry about RENumbering subroutines for use in such environments.

Gozinta and Gozouta

As long as we are on the subject of machine language techniques, I would like to point out the absolute necessity of establishing entry and exit conventions for each and every subroutine. Again, if you will refer to Program 5 from the February issue, you will note that each routine (GRAPHICS, COLOR, POSITION, PLOT, LOCATE, DRAWTO, and SETCOLOR) specified ENTER and EXIT conditions. For example, GRAPHICS requires that the desired graphics mode number be placed in the A-register before the JSR GRAPHICS. Upon return (RTS), the Y-register is guaranteed to contain a completion status.

On machines with more registers, it is good practice to write subroutines in a way that any registers not specifically designated in the ENTER and EXIT conditions are returned to the caller unchanged. On the 6502 microprocessor, though, it is generally hard to write any significant routine that does not affect all three registers. Therefore, I have adopted the opposite convention for this CPU: If the ENTER/EXIT comments don't say otherwise, I presume that all registers are garbage when the routine returns. What convention you adopt doesn't really matter; just be sure to stick to one, and only one, method and you won't go wrong.

FILL From Machine Language

For those of you who are experienced machine language programmers and have not been kept entertained up to this point, take heart. The other question most asked about my February article was something like "so how do you call FILL from assembler?" I guess my comment that FILL from assembly language was exactly the same as from BASIC didn't make a very good impression. So, okay, I know when I'm licked. Herewith is a FILL subroutine, which I would hope you would include with the rest of the graphics routines and keep in your library for future use.

This time, I won't make the mistake of putting in line numbers and using "*=" and ".END" This is a straight subroutine; type it in and JSR to it *only* after you have satisfied its ENTER conditions.

```
FILL H,V

;  ENTER:  Must have previously drawn the right hand edge
;          of the area to be FILLed via JSR's to PLOT and
;          DRAWTO. Just prior to JSR FILL, it must have
;          performed a JSR PLOT to establish the top (or
;          bottom of the line which will define the left edge of
;          the area to be FILLed. FILL presumes that the
;          color to fill with is that which was most recently
;          chosen via JSR COLOR. Finally, on entry, FILL
;          expects the registers to specify the ending position
;          of the line which will define the left edge of the
;          filled area, as follows:
;             h (horizontal) position in X,A registers
;                (X has LSB of position, A has MSB)
;             v (vertical) position in Y register
;
;  EXIT:   Y-register has completion status from OS fill
;          routine
;

FILDAT  =  765   ;where XIO wants the fill color
CFILL   =  18    ;fill is XIO 18
;  rest of equates are from February article and program;
;
FILL
        JSR  POSITION    ; subroutine from Feb. 1982
                         article
        LDA  SAVECOLOR   ; value established via JSR
                         COLOR
        STA  FILDAT      ; see BASIC manual: color used
                         for FILL
        LDX  #6*$10      ; file 6...where S: normally is
```

STATISTICS FOR RESEARCHERS

R-STAT

Friendly, easy to use, well documented.

INCLUDES:

- Database with transformations
- ANOVA N – way
- Analysis of COvariance One – way
- Stepwise Multiple Regression (with Ridge-regression)
- Univariate Statistics
- Mulitple range tests
- Crosstabs
- t – tests
- Graphics
- Menu-driven

DISKETTE & MANUALS **$350.00**

ATARI® DOS II

For a FREE Brochure or further information please call or write.

DSC CO: P.O. BOX 1035 CENTRALIA, WA 98531 (206) 736-7714

VISA & MASTERCHARGE ACCEPTED

WANTED

SOFTWARE AUTHORS!

for Apple, Atari, TRS-80, NEC, Hitachi. . . .

Brøderbund Software is looking for new authors to join its international team of programmers. If you have a product for the micro market, let us show you the advantages of working with our team of design, production and distribution specialists.

Call or write for our free Authors Kit today or send us a machine readable copy of your work for prompt review under strictest confidence.

Brøderbund Software

1938 Fourth Street, San Rafael, CA 94901 • (415) 456-6424

RAM

For ATARI

48K RAM BOARD FOR THE 400

- Increases memory capacity
- Reduces power consumption
- Reduces heat

48K Board	$240
32K Board	$100

INTEC PERIPHERALS CORP

906 E. Highland Ave.
San Bernardino, CA 92404

(714) 881-1533

ATARI, 400, 800 are Trademarks of ATARI, Inc.

A Revolutionary Concept In Software For The ATARI* 400 and 800 Computers

The Interactive Storybook

Sammy The Sea Serpent
A Storybook Program For Children Ages 4 to 7.

Sammy The Sea Serpent
is the story of an imaginery sea creature who is lost and trying to find his way home. The story is read aloud to your child by a professional actress. While the tale is being told, the child uses the joystick to help Sammy out of some tight spots.

The A side of the cassette contains the interactive story; the B side contains games that the child plays with Sammy.

The program uses voice, sound effects, music, color and mixed graphics.

Sammy The Sea Serpent
can be used with either the ATARI 400 or 800 and requires 16K. It is available in cassette format only. Price is $16.95 plus $2.00 shipping and handling.

Also available at fine computer stores.

PDI

Program Design, Inc./11 Idar Court Greenwich, CT 06830
203-661-8799

ATARI is the registered trademark of ATARI, Inc.

```
LDA  #CFILL      ; the fill command (XIO 18)
STA  ICCOM,X     ; ... is specified
LDA  #0
STA  ICAUX1,X    ; remember, XIO 18, #6, 0
JSR  CIO         ; and let the OS do the work
RTS              ; ...and give us status in the Y-reg
```

By the way, did you notice that we didn't actually specify "S:" for the XIO, as specified in the BASIC manual? That's because the BASIC manual doesn't tell the whole truth. If you perform XIO on an already open file, the operating system ignores any filename you give it! Want to save a little space in your BASIC programs? Use 'XIO 18,#6, 0,0,junk$' where 'junk' is *any* string variable you happen to be using for any other purpose in your program.

Inside Atari BASIC: Part 6

Last month, we delved into the hopefully-no-longer-mysterious details on how string and array space is allocated from Atari BASIC and BASIC A+. We showed how to fool BASIC into believing that a perfectly ordinary string was located smack in the middle of screen space. The advantage of such deceptions is that BASIC can move strings of bytes at extremely high speeds, faster than you could ever hope to accomplish with any BASIC subroutine.

We did not discuss one other significant use of such string moves: Player/Missile Graphics. Obviously, if you can move the screen bytes around, you can move the players around just as well, and just as fast. Again, several games and utilities now available on the market use just this technique.

I also promised in the last column to tell of possible uses for multiple variables in the same address space (that is, having a string and an array occupying the same hunk of memory). If the idea interests you, read on.

One thing which BASICs in general lack is a good means of handling record input/output. How many times have you seen programs doing disk I/O using PRINT# and INPUT#? Yuch. (I have several reasons for that "yuch," but the best one is simply that PRINT#ing an item means that the number of disk bytes occupied depends upon the contents of the item.) But what is the alternative? With many BASICs, there is none. With Atari BASIC there is at least GET# and PUT#, but they are slow. So let us examine a way to make PRINT# and INPUT# work for us, instead of against us.

First, we will examine a small program:

```
100 DIM RECORD$(1),NAME$(20),QUANTITY
   ORDERED(0)
110 OPEN #1,8,0,"D:JUNK"
120 VVTP = PEEK(134) + 256*PEEK(135)
130 POKE VVTP+4,27 : POKE VVTP+6,27
140 GOSUB 900
150 PRINT "GIVE NAME AND QUANTITY:"
160 INPUT NAME$
170 INPUT TEMP : QUANTITYORDERED(0) =
   TEMP
180 PRINT #1;RECORD$
190 CLOSE #1
200 REM --- READ FILE WE JUST CREATED
   ---
210 OPEN #1,4,0,"D:JUNK"
220 GOSUB 900
230 INPUT #1,RECORD$
240 PRINT "WE READ BACK IN:"
250 PRINT ,,NAME$
260 PRINT ,,QUANTITYORDERED(0)
270 CLOSE #1
290 END
900 REM --- CLEAR THE VARIABLES ---
910 NAME$="                    ":REM
   20 BLANKS
920 QUANTITYORDERED(0)=0
930 RETURN
```

Surprised? Even though we cleared the variables in line 220, the input of line 230 re-read them from the file. How? Because line 130 set the dimension and length of RECORD$ to 27, which includes the original single byte of RECORD$, the 20 bytes of NAME$, and the six bytes of the single element of the array QUANTITYORDERED. So PRINT# thought it had to print 27 bytes for RECORD$, and INPUT# allowed RECORD$ to accept up to 27 bytes.

Wow! With one fell swoop we have managed to allow fast disk I/O of any sized record, right? Wrong. Unfortunately, there are several limitations

Royal Software

YOUR MARKETPLACE FOR:

* ATARI

WE DEAL EXCLUSIVELY IN PRODUCTS FOR THE ATARI (THE BEST) COMPUTER

ACTION QUEST

A real time Adventure game played with joystick. You, as the ghost, move from room to room in an attempt to capture valuable prizes while eluding capture by strange creatures. $26.90

16K DISK OR TAPE

MEGALEGS

ARCADE QUALITY MACHINE LANGUAGE GAME

16K DISK OR TAPE $31.40

FAST ACTION . . .
Kill the attacking creatures. It's a spectacular arcade game with lots of excitement! Sure to be one of the TOP games in the months to come.

PRO-FOOTBALL

Play against the computer or another player. Over 25 offense and defence plays. The BEST football game yet.

16K
$26.90 TAPE
$31.40 DISK

HARDWARE

800 Computer 16K	$648
800 Computer 48K	$757
400 Computer 16K	$338
810 Disk Drive	$448
850 Interface	$178
410 Recorder	$ 78
825 Printer	$599
16K Ram module	$ 69
32K Ram module	$109
Graphics Tablet	$278

Trade your 400 or older 800 for the new model 800 – Call

SOFTWARE

PAC MAN (cart.)	$39.50
Centipede (cart.)	$39.50
Microsoft Basic (D)	$79.90
Ghost Hunter (T)	$25.50
Ali Baba (D) 32K	$28.90
Protector (T) 32K	$22.45
Apple Panic (D)	$26.90
Threshold (D) 40K	$33.90
GORF (D)	$35.10
Wizard of Wor (D)	$35.10
Text Wizard (D)	$87.90
Dodge Racer (T)	$20.20
Chicken (T)	$20.20
De Re Atari	$17.90
Atari Games Book	$13.90
Personal Finance	$67.50
Zork 1 or II (D)	$35.90
Deluxe Invaders (D)	$31.40
Eastern Front	$25.40
Mousekattack (D)	$31.40
Galactic Chase (T)	$22.40
Bug Attack (T,D)	$26.90
Star Raiders (Cart)	$44.90
Missile Command	$39.50
Midnight Magic (D)	$31.45
Tanktics (T,D)	$24.65
Controller (D)	$27.00

Call or Write for

FREE

ATARI PRODUCTS CATALOG
with nearly 1000 items

Canyon Climber

Smart DONKEYS, boulders and birds dropping rocks try to stop you from reaching the top. A real challenge. Arcade quality.

16K
DISK $26.90 TAPE $26.90

CROSSFIRE

The aliens have landed and it's your job to save the city. (If you can!) TOP RATE GAME, ARCADE QUALITY, HI-RES Graphics & Sound.

32K $26.90
DISK or TAPE

FROGGER

This is the genuine FROGGER game that you see in the arcades. Made by the same people that made Jawbreaker (One of the top ten sellers.)

16K DISK OR TAPE
$31.40

VISA · master charge

TO ORDER CALL TOLL FREE 800-235-5759

FOR INFORMATION CALL 805-967-3889

HOW TO ORDER: Send check or money order or call our toll free number and use your Visa Card. Shipping on software is $2.00 per order anywhere in USA. Hardware shipping call for cost. CA residents add 6% sales tax. Add 3% for VISA or MC. Equipment subject to price change and availability without notice.

Store Hours
8 am — 6 pm
Mon. — Sat.

Royal Software, 5138 Hollister Avenue, Santa Barbara, CA 93111 to this technique. (1) The record cannot be over 255 bytes long or INPUT# won't be able to retrieve it all. And any size over 127 bytes will wipe out routines/data in the lower half of page $600 memory. (2) The record cannot contain a RETURN (155 decimal, 9B hex) character. It will print fine, but the INPUT# will terminate on the first RETURN it sees. (3) The other strings in the record (NAME$ in our example) will *not* have their lengths set properly by the INPUT#, thus necessitating something like the routine at line 900. But if you insert "280 PRINT LEN(NAME$)", you will always get a result of 20.

Well, limitations one and three are easy enough to predict and understand, but how do you insure that your data does not contain a RETURN code? For strings which have been INPUT by a user, that's easy: the RETURN code will never appear in such a string. But what about numbers? Remember that we will be printing the internal form of Atari decimal floating point numbers. Can such numbers contain a byte with a value of 155 ($9B)? Yes, but such a number would be in the range of -1E-74 to -9.E-73, which is unlikely enough to ignore for most purposes.

So, in summary, is this make-a-record technique useful? I'm not sure. Certainly BGET/BPUT or RGET/RPUT from BASIC A+ or their USR equivalents under Atari BASIC are much easier to code and use. And, yet, there is a certain elegance to record-oriented techniques which is not entirely lost to me. I probably will stick with the constructs we invented for BASIC A+, but I would respect a program using the above techniques.

A few last commments: the pokes of line 130 depend on RECORD$ being the first variable defined. Recall my comments from last month about LISTing and reENTERing a program to insure a particular order of definition. Also, if you need to alter a variable other than variable number zero, remember that the formulas are:

VVTP + 8 * VNUM + 4 for the LSB of the length
VVTP + 8 * VNUM + 6 for the LSB of the DIMension

(and, again, see last month's article for fuller explanations).

And, finally, I really would be interested in hearing from anyone who uses the techniques I have devised here to produce a unique, real-world program that does things that can't be done otherwise.

Fun With FMS, Canto The First

Remember that fix for burst I/O I gave you in the May, 1982, issue? Did you try it?. Did it prevent burst I/O errors? Yep. Did it slow down every kind of disk read? Yep. Oooooopsy daisy. Well, you can't be completely right all the time. This month, we will try again.

First, I would like to explain, in terms of the FMS listing and the commentary (Chapter 12 – BURST I/O, *Inside Atari DOS*, **COMPUTE! Books**) why the fix I gave you in the May, 1982, issue worked insofar as it fixed the burst I/O problems.

To begin with, examine the code at locations $09F8-$09FD and $0AD2-$0AD7. These are the locations in PUT-BYTE and GET-BYTE, respectively, where the burst I/O routine is called. But lo! In PUT-BYTE, the JSR to burst I/O is directly preceded by a BCS, meaning that burst I/O won't occur unless carry is clear. But, in GET-BYTE, the JSR to burst I/O is directly preceded by a BCC – burst I/O occurs in read mode only if carry is set!

Now, if you examine the label "WTBUR" at $0A1F, you will note that the first thing that occurs is a test of FCBFLG to find out if we are in update mode or not. If we are updating, we don't burst. But note that GET-BYTE called the label "RTBUR", AFTER the test, and so would always burst, whether in update mode or not. What I tried to do was change the "JSR RTBUR" (at $0AD4) to a "JSR WTBUR" and then use the carry flag to distinguish between the type of request (I changed the BMI at $0A24 to a BCC). *Great!* It worked! Except...it worked *too* well. Unfortunately, FCBFLG is zero (and therefore plus) when we have a file open for read only; so, therefore, the burst I/O was suppressed for *all* reads. Nuts.

We try again, using a slightly different approach. We will still count on the carry being set when called from PUT-BYTE and reset when called from GET-BYTE. This time, though, we will examine the actual I/O mode in use. FMS receives the I/O mode from CIO when the file is opened and places it in FCBOTC. Recall that the only legal values are 4, 6, 8, 9, and 12. Well, burst I/O is only illegal in modes 6 (read directory) and 12 (update). But mode 6 is handled separately (see $0AC5-$0ACB), so 12 is all we are really concerned with. Anyway, without further ado, here's the listing of the FMS patch:

```
          *=$0A1F
;
; first, patch the code where WTBUR used to be
;
WTBUR
BURSTIO
          LDA     FCBOTC,X  ; Open Type Code byte
          EOR     #$0C      ; check for mode 12...
                            ;   update
          BEQ     NOBURST   ; it IS update...don't burst
          ROR     A         ; move carry to MSB of A
                            ;   register
          NOP               ; filler only
TBURST
; ... and the STA BURTYP remains ... but now BURTYP is
```

This Publication is available in Microform.

University Microfilms International

Please send additional information

for ______________________

Name______________________

Institution______________________

Street______________________

City______________________

State______________ Zip________

300 North Zeeb Road
Dept. P.R.
Ann Arbor, Mi. 48106

INTRUDER!

Four nuclear reactors have gone out of control, ready to melt down in only minutes. They are also protected by pre-programmed robots that can destroy anything in seconds. Your mission is to shut down the reactors and capture the robots before a melt down occurs. But before you enter ... remember you are the...

INTRUDER!

16K Cassette . . . . . . 29.95 US funds
24K Disk . . . . 34.95 US funds

ALIEN SWARM

Space Invaders! Asteroids! Meteor Showers and more! You have defended them all. But can you survive the fetish attack of the...

ALIEN SWARM

100% Machine – Joysticks
16K Cassette . . . . . . 29.95 US funds
16K Disk . . . . 34.95 US funds

Now available...ALIEN NUMBER
An educational arcade game.
Cassette $29.95
Disk $34.95

INHOME SOFTWARE

WORLD FAMOUS INHOME SOFTWARE FOR SOFTWARE

CRYPTS OF TERROR

You are in search of the true meaning to life, but first you must battle each level in search of the magic ring. Will you survive this horror? What lurks in these deep and dark...

CRYPTS OF TERROR

Joystick – BASIC required.
16K Cassette . . . . . . 29.95 US funds
16K Disk . . . . 34.95 US funds

MATH PACK + − ÷ ×

Atari brought the computer age home. Now we bring home the educational software.

MATH PACK was written for the educational side of the Atari by professional educators in the field of mathematics. All four independant math programs use Atari's sound and graphics to make learning more fun and enjoyable for your whole family. A final summary at the end of the programs lets you see how well you've done or what area needs to be practiced. Age and skill levels.

1) Math Pack Drill*
2) Equations* **
3) Jumble* **
4) Digit Span*

*BASIC and **Joysticks required.

16K – 2 tapes – 39.95 U.S. funds.
16K – Disk – 42.95 U.S. funds.

Check your local computer dealer or order direct from us.
Ordering: VISA, MasterCard, American Express and Cheques accepted.

Please include all numbers on cards.
Include phone number with all orders.
Add $2.00 for shipping and handling per item ordered.

Please specify which program ordering.

*DEALER INQUIRIES WELCOME

INHOME SOFTWARE
PH. 1-416-961-2760

1728 Avenue Rd.
Toronto, Ontario
Canada M5M3Y6

```
;     negative if BURSTIO was called from GET-BYTE and
;     positive if it was called from PUT-BYTE.
;
              *=$0A41
; so we must patch here to account for the sense of being
; inverted from the original.
              BPL        WRBUR      ; called from PUT-BYTE
              *=$0AD4
; finally, we must patch the GET-BYTE call so that it no
; longer JSR's to RTBUR.
              JSR        BURSTIO    ; call the common burst
                                      routine
;
              .END
```

And for those of you who don't want to type all that in, you might simply use BUG to do the following changes:

```
C A20 < 82,13,49,0C,F0,24,6A,EA
C A41 < 10
C AD5 < 1F
```

And, last but not least, from BASIC you may use the following:

```
POKE 2592,130
POKE 2593,19
POKE 2594,73
POKE 2595,12
POKE 2596,240
POKE 2597,36
POKE 2598,106
POKE 2599,234
POKE 2625,16
POKE 2773,13
```

Fun With FMS, Canto The Second

Not long ago, an OSS customer told me that he couldn't use Atari DOS to SAVE (option K on the menu) the contents of ROM. "How sneaky," cried I, "Best to use the SAVE command under OS/A+. We wouldn't do anything that nasty to you!"

But we did. And we do. And it isn't because we or Atari are sneaky or nasty; it is yet another phenomenon of burst I/O. Recall that when the burst I/O test is passed, FMS calls SIO to transfer the sectors of data directly from the user's buffer space. In order to do so, though, it must write the sector link information (last three physical bytes in a sector) into the correct spot in the user's buffer before calling SIO. Then, when SIO returns, it restores those three bytes and tries to write the next sector the same way. Again, if you have *Inside Atari DOS*, you can follow this happening at addresses $0A52-$0A7A, in the "WRBUR" code.

Ah...but what happens when you try to do burst I/O writes from ROM? FMS blindly tries to put its goodies into those three bytes and call SIO. SIO does what it is told, and FMS thinks that all is OK. Except that all is *not* OK! Those three bytes did *not* get changed, so what was written to the disk is garbage. And even ERAsing the file won't work, because the sector links are badly messed up. Crunchy, crunchy goes the disk, under worst-case circumstances.

Now this restriction is fairly easy to get around: one simply writes a program (in BASIC or machine language) which writes the desired bytes to the disk one at a time, thus preventing burst I/O. So I don't feel that I am giving away deep, dark Atari secrets when I give you an easier method to prevent burst I/O. Simply do either of the following:

```
from BUG:      C A2E < 0
from BASIC:    POKE 2606,0
```

Again, for those of you with the FMS listing, note that what we are doing is changing the AND #$02 which checks for text mode (the read and write text line commands are $05 and $09, neither of which have bit $02 turned on) into an AND #$00 instruction, thus fooling the BEQ that follows into thinking that FMS can't do burst I/O because it's doing text mode I/O. Not too terribly tricky, and it works well.

I cannot recommend that you make this patch a permanent part of most system disks, since it completely disables burst I/O and makes the system load and save files considerably slower. Change it, use it, and then forget it. ©

SO . . . YOU THINK YOU'RE A SPACE ACE!

Can you blast through the deadly asteroid fields?
Can you survive robot space bombs?
Can you handle invisible Silurian space mines?
Can you dodge a rotating death vortex?
Can you defeat the alien battle cruisers?

GOOD LUCK CHARLEY!

You'll need luck and skill to complete your first mission at drive level I. If, by beginner's luck, you progress to Patrolmen at drive level II, the hyper-asteroids and Silurian mines will send you back to the Academy for a refresher course in "Beginning Pilot". Should the incredible happen, and you get your ship to drive level III, and are able to blast your way through the asteroids and mines, the space death bombs will get you—just like all the others. If you get past the bombs, there's more, and more, and more!

SO . . . YOU THINK YOU CAN HANDLE IT?

Try it, if you're right, you're a . . . SPACE ACE!

A great new and challenging space game by Greg Young

- **Hi-res Graphics**
- **Requires 16K and all of the potential of the Atari™ 400/800**
- **Disk/Cassette**
- **Joy stick**
- **100% machine language**
- **Multiple levels of play**
- **Great Audio**
- **Time limit options**
- **Bonus scoring**

$29.95

SEND CHECK OR MONEY ORDER TO:

LONDON SOFTWARE
374 Wildwood Avenue
Piedmont, CA 94611

Ph. orders (415) 893-1090
Visa/Mastercharge
Dealer Inquiries Invited.

Please add $1.50 postage & handling. Calif. residents add additional $1.95 tax.
Atari 400/800 is a registered trademark of Atari, Inc.

COMPUTE!

is looking for applications articles on Sinclair ZX-81, Texas Instruments TI-99/4A, and Radio Shack Color Computer.

BYTE-A-BIT COMPUTING CO.

"The Software Specialists" offer...

★ THE STATE OF THE ART ★
IN COMPUTER SOFTWARE

●NUMBER STUMPER: 24K CAS. $19.95 32K DISK 24.95	AN ARCADE GAME FOR ONE OR TWO PLAYERS. USES JOYSTICKS.
●UTILITY PAK #1: 32K DISK $29.95	DOS UTILITIES, AUTORUN. SYS, RENUMBER, VARIABLE LISTER AND CHANGER, AUTOMENU.
●UTILITY PAK #2: 32K DISK $24.95 (PRESENTLY UNDER DEVELOPMENT)	DISK-SPEED ADJUST, CONVERT DECIMAL-HEX-BINARY #'S, COLOR & SOUND PROGRAMMER'S AID, SAVE-A-SCREEN UTILITY.
●GAME PAK #1: 24K CAS. $14.95 32K DISK $19.95	SCOREFOUR, DOTS, FIFTEEN, PLUS SURPRISE GAME! (A TOTAL OF FOUR GAMES). USES JOYSTICKS.

BYTE-A-BIT OFFERS YOU **FREE:** •PROGRAMMING •CONSULTATION

CALL OR WRITE FOR A CATALOG OF OUR PROGRAMS AND DETAILED INFORMATION ON OUR **FREE OFFERINGS**. SPECIALISTS IN SOFTWARE FOR THE **ATARI*** COMPUTER. GAMES, UTILITIES AND CUSTOM PROGRAMMING AVAILABLE.

DEALERS INQUIRIES INVITED

"Our Software Is Guaranteed For As Long As You Own It" If Any Program **EVER** Fails To Run, We Will Replace It **FREE**!!

BYTE-A-BIT COMPUTING CO.
28 ORCHID ROAD
LEVITTOWN, N.Y. 11756
(516) 796-1212

ALL OUR SOFTWARE MAXIMIZES THE USE OF ATARI GRAPHICS AND SOUND

ADD $2.50 SHIPPING AND INSURANCE. N.Y. RESIDENTS ADD SALES TAX. C.O.D. OR BLUE LABEL SHIPMENT ADD'L.

After you've written a FORTH program, you can metacompile, leaving out words which are not a working part of the application.

The FORTH Page

Headless Metacompilation

Richard Mansfield
Assistant Editor

Somehow, the prefix *meta* always sounds both glamorous and forbidding. Attached to a word, *meta* usually adds a special meaning involving "over" or "beyond" or "transformed" – it can seem paradoxical. A *metatruck* would be a truck that carries other trucks. MetaFORTH, likewise, means using FORTH to compile itself.

"FORTH for PET," an adaptation of FORTH by L. C. Cargile and Michael Riley, supports a FORTH Metacompiler (for Upgrade or 4.0 BASIC with 32K RAM) which can be purchased separately for $30. "FORTH for PET" itself does not require the extra memory.

The goal of metacompilation is to reduce memory size or modify FORTH itself by recompiling the entire FORTH dictionary. Obviously, you cannot eliminate words like DUP and SWAP. They are in the definitions of many other words and will always be central in any working FORTH application.

However, there are applications which will not need keyboard entry, disk access, or other such normal FORTH capabilities. If you write a game which does not involve the disk, you could metacompile a special FORTH which does not have such words as FLUSH, LOAD, etc.

Clear, Step-by-step Instructions

The Metacompiler's instruction booklet is written so clearly that a relative novice in FORTH will have little difficulty with it. The process of metacompilation is essentially straightforward and the manual leads you through it, step-by-step, explaining what's happening along the way. You load your ordinary FORTH into the computer, load screen ten from the Metacompiler disk, and it requests a target address for a new, temporary FORTH. Since regular FORTH starts at hex $0400, you can respond with $4000 as the start of the new version. Both will be resident in memory simultaneously, but after metacompilation, control of the computer is passed to the upper one.

After you've indicated your target address, you can sit back and watch the dictionary being recreated at the new location. The words are put on screen as they are recompiled. Or you can leave and come back about six minutes later to find two FORTHs in the computer. You'll be able to communicate with the upper version and it will respond normally.

The final step is to compile back down, giving $0400 as the target this time, and making any changes you want to the Metacompiler's disk screens. These changes will be reflected in the new FORTH you are creating.

The manual takes you carefully through an entire editing process where you create a standalone "Calendar" program. This calendar application is truly a FORTH transfigured: when run, it uses DUP and other key words, of course, but it will not recognize them from the keyboard. It can do nothing with the disk. In fact, it understands only the few words that relate to the calendar functions to which it is dedicated. What's more, it's *headless.*

Removing Heads

In FORTH, there are four "fields" to an ordinary word: the name, link, code, and parameter fields. The "head" of a word is the name and link fields, taken together. If a word in your new FORTH application will only be used in further definitions and will never be independently executed, you can type DROP-HEADS before it on the Metacompiler disk. That word and any words after it will be compiled without name or link fields when the new FORTH is created. To resume head compilation, type COMPILE-HEADS.

An Excellent FORTH Adaptation

It is gratifying to find this metacompilation capability, a valuable and sophisticated technique, available on PET/CBM computers. Equally pleasing is the excellent FORTH implementation by the same authors, "FORTH for PET" ($50, 16 or 32K, Upgrade or 4.0 BASIC, with disk) to which this Metacompiler is an optional addition.

"FORTH for PET" is a full FIG FORTH system (allowing all FORTH 79 extensions as an option), a complete assembler, several sample programs, many utilities, and introductory and reference manuals. Perhaps most attractive is the fact that the full, powerful Commodore screen editing system works the same way that it works in BASIC. You move anywhere fast and modify screens with great ease. Other versions of FORTH are sometimes encumbered by awkward editing procedures.

"FORTH for PET," and "Metacompiler for FORTH" are both available from A B Computers, 525 Bethlehem Pike, Colmar, PA, 18915. ©

If you've ever wanted to thoroughly document, explore, and understand your computer's BASIC or Operating Systems – the techniques and programs here are your tools. Written for OSI, these ideas can be modified for other computers. Next month this article concludes with additional programs and examples.

Resource Part I:

Mapping Machine Language Code

T. R. Berger
Coon Rapids, MN

Have you ever tried to document your machine software by annotating disassemblies? Have you ever tried to move these programs by reconstructing assembler source listings from disassemblies? If so, you know what a huge investment of time is needed. This article covers a group of BASIC programs which will facilitate regenerating fully documented assembler source listings starting from machine language programs in much less time than the painful direct route.

When I undertook to write these programs, I did not even dream how powerful they would be. I never really anticipated regenerating a source listing of 8K OSI Microsoft Disk BASIC. When I realized that this task could be done, what was one simple program expanded into the four presented here. A much modified and improved version of the single program which started this all off is also included here. If OS65D would allow six buffers to be open at once, these programs could be vastly speeded up and simplified.

These programs are written in BASIC for disk based OSI computers. However, the programs are carefully documented so that those using other 6502 machines with different disassemblers should have no difficulty in copying the idea. The programs accept as input an ASCII file produced by the OSI version of the Apple disassembler (see *Dr. Dobbs Journal,* September, 1976, p. 22). The output is a collection of ASCII files which include the following:

1. An assembly source listing of code which will reassemble at the same location without further editing.

2. Equate files necessary to run the assembly source through an assembler.

3. Separate cross reference files for each of the following:
   a. Zpage addresses,
   b. Jumps and jumps to subroutines,
   c. Memory calls, and
   d. Branches.

A single pass program RESOURCE S in included for resourcing small programs. On a 48K C8PDF it has no trouble handling OS65D. Since only symbols and cross references are kept in memory, a 32K machine should also have no trouble. Cross reference strings in RESOURCE S are of limited size so that the program will crash in attempting to cover 8K BASIC. The Zpage cross references to $AC overrun about halfway through. Since RESOURCE S is a compressed version of the program package presented, I will comment very little on it. When there are a very large number of cross reference strings, the program slows way down due to garbage collection. In Microsoft BASIC, garbage collection times go up approximately as the square of the number of strings in memory (and not their size).

Run Times Approached 24 Hours

I have written this package so that hobbyists can understand their most commonly used language: BASIC. A source file for 8K BASIC is colossal. Therefore, many shortcuts are necessary to complete the resourcing task. I originally tried to enlarge RESOURCE S to cope with the job. OS65D has only two disk buffers requiring that a large amount of information be kept in memory for a single program. So many strings were generated and garbage collection time became so great that run times approached 24 hours. Clearly this is not the way to go. I broke the task into small pieces, each being completed in a reasonable amount of time.

On 8" floppies the BASIC disassembler source ($03A1 - $2300) takes 28 tracks (84K). Those using minifloppies must tackle BASIC in three or more passes, using the cross reference tables to properly join the final product.

The final product, scattered through several files, takes up about 36 tracks. There is no hope of assembling these files without a linking assembler. (Leroy Erickson has written such an extension for the OSI Assembler.) However, printout of the source and the cross reference tables greatly simplifies the annotation and documentation process. After one pass of RESOURCE S over OS65D, it was possible to reassemble OS65D at the same location. After about two hours of editing, a file was obtained which assembles anywhere.

Using my maps of OS65D and Jim Butterfield's maps of BASIC, you should be able to obtain fully documented source listings of both BASIC and OS65D. I would hope to see more articles using specific parts of OS65D and BASIC. Namely, what are some subroutines, how do they work, how does one use them, and how does one resource them?

The entire program package presented here is written in BASIC. This sped implementation and modification time. It also makes the programs easier to understand. The price paid is runtime, which is considerable over 8K BASIC. Efforts have been made to optimize runtimes, especially on inner loops. This adds steps to the process, but significantly reduces program running times.

Of course, one must edit the files generated by these programs. I use a group of utilities which constitute a useful BASIC text file editor and processor. I will describe these utilities in a future article.

The three most useful utilities are a transfer program to move large text files around, a print program to output large files to a printer, and a fast sorter to sort symbol tables. A further useful addition is a large text file single pass character-oriented line editor.

How It Works

The first program (PASS 1) takes the disassembly listing (which I will call SOURCE) and compresses it into a scratch file (which I call SCRATCH). The main working file is SCRATCH. It is about 25% smaller than SOURCE and serves as input to the other programs. A typical line of SOURCE looks as follows:

```
1A3D BD11B0 LDA $B011,X.
```

In SCRATCH this same line would be:

```
1A3D LDA HHB011,X.
```

The code field has been eliminated and $B011 has been changed to a six letter symbol. All four digit operands $XXXX are changed to six letter symbols HHXXXX, which is the maximum size for symbols in OSI's Assembler. Except for immediate operands, two digit operands $YY are replaced by six letter symbols HHZZYY. Further, the first H in every operand is always aligned as the eleventh letter in a line. BASIC is much too slow to search a line for a symbol. Aligning symbols makes them easy to find when editing. For example,

```
MID$(IN$,11,6)
```

removes a symbol from a line IN$. The 'H' in position eleven distinguishes a symbol. The 'Z' in position thirteen distinguishes a Zpage reference.

A line in SOURCE

```
1A40 FF ???
```

would appear in SCRATCH as

```
1A40 .BYTE $FF.
```

This step makes the resource file assembler ready. Bad disassembly of opcodes must be fixed by editing the final file if a true source file is needed. In particular, tables and text are not resourced correctly, only made assembler-ready.

The first program also builds a table of two byte operands (which I will call SYMBOL). SYMBOL is used in PASS 2 to generate labels and an equate file of two byte operands. Since SYMBOL is searched repeatedly in PASS 2, it must be sorted. Sorting SYMBOL means a fast binary search can be used which is many times faster than a sequential search. (For BASIC, this addition reduced line process times in PASS 2 from about 5 seconds per line to less than 1 second per line.) Since BASIC requires 800 symbols, this search method cuts hours off PASS 2. Accordingly, PASS 1 keeps a sorted symbol table.

PASS 2 generates the resource file (which I call OBJECT). It reads one line of SCRATCH:

```
1A3D LDA HHB011,X.
```

It searches SYMBOL for 1A3D. If 1A3D is found, a numbered line

```
10000 HH1A3D LDA HHB011,X
```

is output to OBJECT. Since 1A3D is now defined by a label, it is marked as 'used' in SYMBOL. If 1A3D is not found, a numbered line

```
10000        LDA HHB011,X
```

is output to OBJECT. After OBJECT is complete, the unmarked symbols in SYMBOL are operands which are not defined by labels in OBJECT. Thus, an equate file (which I call EQUATE) is written using these unmarked terms from SYMBOL. For example, if 1A3D is unmarked, it would be written to EQUATE as a numbered line

```
5000 HH1A3D = $1A3D.
```

Except for Zpage labels, OBJECT and EQUATE are ready for the assembler.

PASS 3 generates the various symbol tables. The symbols are picked out of SCRATCH along with their addresses. A symbol HHXXXX is stored in a string SS$(I) as XXXX. A check is run to see if the symbol already appears in the table. If it does not, the counter SN is incremented and the symbol is added. This list is stored as a sorted table.

Suppose that HHXXXX appears in Line YYYY and that SS$(I) = XXXX. Then UYYYY is appended to the right hand end of SA$(I) where U is chosen to give information about the opcode on line YYYY. Some thought went into the choice of U. In the branch table, the middle letter of a branch instruction comes closest to distinguishing all branches. Thus U is the middle letter of the opcode. Again in the JMP and JSR table, the middle letter distinguishes JMP from JSR. Thus U is M or S in this case. The first letter of the opcode is chosen for the memory table.

In decoding programs, I have found that the most important fact to know about Zpage opcodes is their addressing mode. That is, is an opcode indexed or not? Thus, U is the extreme right hand symbol of the disassembly line. This includes), X, and Y. It is not possible from this to tell whether the Y means indexed or indirect indexed. However, given the simplicity of this approach, it is adequate.

If SA$(I) becomes too long, it is written to a cross reference file and SA$(I) is emptied. (In RESOURCE S this step is not performed, the program bombs when SA$(I) becomes too long.) These "long strings" will appear out of order in the file. (The first few cross references may be out of order.) The symbol table can be resorted by most any sorting program. As it stands, the table is "almost in order."

PASS 4 generates the Zpage equate file which I call ZEQUATE. This is done using the Zpage cross reference file generated in PASS 3. The file resembles the EQUATE file.

In resourcing a large program, there will not be enough room on one disk for all the files generated. SCRATCH, and various other files may be moved using a transfer utility. Symbol and cross reference files may be sorted using a sort utility. Final files may be printed using an output utility.

Example 1 shows the OBJECT file (resourced assembly language) for the beginning of the disassembler in the Extended Monitor. Example 2 gives the two equate files. Example 3 gives the output from the Assembler using these three files. Example 4 gives the four cross reference tables. The first address in each row is the symbol. The other addresses following are the cross references, with some indication as to opcode.

Example 1.

```
10000         .BYTE $17
10010         LDA #$16
10020         STA  HHZZC7
10030 HH18DD JSR  HH18ED
10040         JSR  HH19D1
10050         STA  HHZZC5
10060         STY  HHZZC6
10070         DEC  HHZZC7
10080         BMI  HH1922
10090         BNE  HH18DD
10100 HH18ED JSR  HH19BC
10110         LDA (HHZZC5,X)
10120         TAY
10130         LSR A
10140         BCC  HH1901
10150         LSR A
10160         BCS  HH1910
10170         CMP #$22
10180         BEQ  HH1910
10190         AND #$07
10200         ORA #$80
10210 HH1901 LSR A
10220         TAX
10230         LDA  HH17A5,X
10240         BCS  HH190C
10250         LSR A
10260         LSR A
10270         LSR A
10280         LSR A
10290 HH190C AND #$0F
10300         BNE  HH1914
10310 HH1910 LDY #$80
10320         LDA #$00
10330 HH1914 TAX
10340         LDA  HH17E9,X
10350         STA  HHZZC1
10360         AND #$03
10370         STA  HHZZC2
10380         LDA  HHZZC8
10390         BNE  HH1923
10400 HH1922 RTS
10410 HH1923 TYA
10420         AND #$8F
10430         TAX
10440         TYA
10450         LDY #$03
10460         CPX #$8A
10470         BEQ  HH1939
10480         LSR A
10490         BCC  HH1939
10500         LSR A
10510         LSR A
```

How To Use It

STEP 1) Creating a SOURCE file.

If you plan to resource BASIC, you must move the Extended Monitor since it overlays part of BASIC. In another article, I will give explicit instructions on how to do this. I find it handy to have the Extended Monitor available while BASIC is resident.

After trying several methods, I've decided that the following is the easiest way to generate a SOURCE file. It uses the disk output capability of OS65D. The code you are resourcing should not overlay the disk buffer used. (Video with polled keyboard is assumed; otherwise, recheck the I/O flags.)

a) Initialize a fresh disk.

b) Copy the directory Track D onto this disk using OS65D's copy utility (D is Track 8 on 8" floppies).

c) Create files for all empty tracks except Tracks 0 and D. Delete all directory entries on the new disk.

d) Load the machine language program to be resourced.

e) Load and run the Extended Monitor.

We must now set all the various pointers for a disk buffer. To resource BASIC you need a very large file. Let the first available track be N where the directory is on Track N-6 (N=9 on 8" floppies).

f) Choose a first track number N for your SOURCE file. Let M be the last track number on the disk (M=76 on 8" floppies). Do not choose N so that either N=0 or the directory track is included in the range of N to M.

g) Using the "at" (@) sign command, set the following buffer values. (These are valid for OS65D V3.2, i.e. 8" floppies. The correct values for minifloppies are given in the *OS65D User's Guide*.)

ADDRESS ($)	ADDRESS (D)	VALUE	
2326	8998	7E	BUFFER START ADDRESS
2327	8999	31	
2328	9000	7E	BUFFER END ADDRESS
2329	9001	3D	
232A	9002	N	FIRST TRACK OF FILE
232B	9003	M	LAST TRACK OF FILE
232C	9004	N	CURRENT BUFFER TRACK
232D	9005	0	DIRTY BUFFER FLAG
23C3	9155	7E	ADDRESS DISK OUTPUT
23C4	9156	31	(CURRENT BUFFER ADDR.)

h) Mount the fresh disk.

I) From EM type (i.e. turn on disk output)

!IO ,22 <return>.

The next few steps write directly to the disk without error correction. If you make an error, perform step l) and restart at step g. Presumably you know the start ($XXXX) annd the finish ($YYYY) addresses of the code to be resourced. For BASIC these are $XXXX=$03A1 and $YYYY=$2300. For OS65D these are $XXXX=$2336 and $YYYY =$2E1E. For the ROMs these are $XXXX=$FD00 and $YYYY=$FFFA.

j) Commence disassembly with

QXXXX <return>.

k) Put your finger on LINEFEED. Hold it there until $YYYY has been disassembled. Then hit RETURN.

All but the last track of the SOURCE file is on the disk. The last track is still in the buffer. This last part is also missing an "end of file" marking. The next few steps turn off the disk output long enough to make corrections, then turn it back on to write the final track to the disk.

The next step turns off disk output by creating a syntax error, and puts a mark to help find the end of the SOURCE file.

l) Type

!XIT <return>.

m) Search for the end of the file

W!XIT>317E,3D7F.

If all has gone well, you will receive a message

(*) VVVV/21

where VVVV is the address of ! in the expression !XIT. If you do not receive such a message, it is possible (but unlikely) that a "disk write" occurred in the middle of the word !XIT. Go back to step g) and start again. When you reach step k), hit RETURN five times instead of just once, then proceed. If you do not receive the message (*), something is definitely wrong somewhere. Start a careful search (*Beware:* some values given only work for 8" floppies).

n) Using quotes and "at" check the following

```
VVVV/21    "!
VVVV+1/58  "X
VVVV+2/49  "I
VVVV+3/54  "T
```

o) Make the following change.

VVVV/21 0D

Now the "end of file" marker is properly installed. Next we write the buffer to the disk.

p) Make the following pointer change.

```
23C3/YY 7E
23C4/WW 3D
```

q) Write down the value TK

232C/TK

r) Type

!IO ,22 <Return>

The entire SOURCE file is on the disk. It starts on Track N and ends on Track TK. We must now create a directory entry for this file.

s) Load BASIC and CREATE, but do not run CREATE. (You may need a different disk to do this.)

t) Delete line 20290 (which would erase all the work you have done). It reads:

```
20290 DISK!"IN "+T$:DISK!"SA "+T$+",1=
   317/"+P$
```

u) Run CREATE and name the SOURCE file on your new disk.

STEP 2) Create a SCRATCH and SYMBOL file entry.

These files must be on the same disk as SOURCE. Be sure to reload CREATE so that line 20290 is not missing. SCRATCH can be about 25% smaller than SOURCE. SYMBOL can be 1-2 tracks. On 8" floppies, 8K BASIC needs two tracks for

SYMBOL (4K file size).You may put these files anywhere as long as they do not overlap the directory track, Track 0, or the tracks used by SOURCE.

STEP 3) PASS 1.

Run the first resource program. Prompting will tell you what to do. The new disk must be in the drive throughout the run. The screen will display the current status. On large programs, be prepared for several-minute waits for garbage collection. A five minute wait between screen data lines probably means there has been a system crash. This program will not work with ROM BASIC since the garbage collector is defunct. (See *PEEK(65)*, March 1980, p. 3 for a fix.)

The SOURCE, SYMBOL, and SCRATCH file may fill a disk, so you may have to move some files to other disks. SOURCE is no longer needed, but should be saved in case of trouble. Symbol is needed for PASS two and SCRATCH is needed for PASSes two and three. Using a transfer utility you may move SCRATCH and SYMBOL to a new disk.

STEP 4) PASS 2.

The second resource program generates an EQUATE file and the resourced assembly listing OBJECT. Create such files on a disk containing SCRATCH and SYMBOL. EQUATE need not be large, usually much less than a track. OBJECT should be slightly larger than SOURCE.

The next step creates all the cross reference tables. Each table needs its own file. SCRATCH is the input file. The branch table will probably be the largest file.

STEP 5) PASS 3.

Repeat this step until all cross reference tables are complete. Only Zpage cross references are essential. However, I find the Zpage and JSR tables the most useful. You may wish to sort these tables, even though they are "almost sorted."

STEP 6) PASS 4.

Create the Zpage equate file: ZEQUATE. Input to this program is the Zpage cross reference file. This step is the final one which creates the list of Assembler Zpage equates.

Any of the files generated may be dumped to a printer using a printer utility. The process is much simpler than it sounds. The single pass resource program eliminates most steps if only small programs are being resourced.

Moving ASCII Text Files To The Assembler

For small programs, the resource can actually be assembled by the OSI Assembler. The three files (OBJECT, EQUATE, and ZEQUATE) must be merged and the program counter location given (10 * = $XXXX).

The resourced files are ASCII text files with an end of file (EOF) marker:

XIT <return>.

Since OSI's Assembler does not keep an ASCII file, more is needed. We must transfer the disk text files into the Assembler/Editor. In OS65D it is easy to reset output flags with:

IO ,02.

However, only one input is recognized and, if this is not the keyboard, then keyboard input is dead. During disk input, the keyboard is disabled. In particular, OS65D has no way of recognizing the end of a file except by an operating system error. This is a definite deficiency in OS65D. When an operating system error does occur, the IO flags are properly reset to default values.

If a file is on Tracks 2 and 3, inputing these tracks will result in a system error as soon as Track 3 is finished. The trouble is that the actual file may end halfway through Track 3. The rest of Track 3 may contain absolutely destructive information, such as Assembler commands or operating system commands. My favorite is the following. The ASCII character "left bracket" occurs as input opening the Indirect File. This file fills up memory, wiping out everything in the way. It eventually reaches the disk addresses. You hear a thunk and the disk goes dead. If input continues, it next reaches the screen memory filling the screen with jazzy characters. It goes on to the color memory, tone generator, etc. You've probably had this occur and wondered what happened. It's just the Indirect File, filing all the garbage away.

One solution is to remove the destructive information on the track. Another simpler one is to create an operating system error at the end of the file, in this case, midway through Track 3. Input errors to the OSI Assembler do not cause the IO flags to be reset. We must be more subtle than just having an input error. If E<return> is sent to the Assembler, it exits to the operating system. In the operating system command mode, any line which is not a legal command creates a syntax error. For example, another E<return>, will do the job. The following changes to PASS two and PASS four will prepare files for entry into the Assembler. Add the following lines:

PASS Two

```
642 PRINT #7,"E"
644 PRINT #7,"E"

842 PRINT #7,"E"
844 PRINT #7,"E"
```

PASS 4

```
472 PRINT #7,"E"
474 PRINT #7,"E"
```

There is yet another problem. In their normal positions, the disk buffers occupy the same space as program memory. This problem can be solved by moving the buffers. Use the following steps to load first the file ZEQUATE, second EQUATE, and third OBJECT into the Assembler.

a) Load and run the Extended Monitor.

b) Suppose the file we wish to load starts on Track N and ends on Track M. Perform STEP l) g) from "HOW TO USE IT." Be sure to use the values given below (or larger values where you have RAM).

ADDRESS ($)	ADDRESS (D)	VALUE	
2326	8998	00	
2327	8999	50	
2328	9000	00	
2329	9001	5C	
232A	9002	N	
232B	9003	M	
232C	9004	N-1	
232D	9005	FF	
23AC	9132	00	ADDRESS MEMORY
23AD	9133	5C	BUFFERED INPUT

Note that 232C, 232D, 23AC, and 23AD have strange values. These values track the disk into loading the first track of your file into memory. Otherwise you would have to do that job separately.

c) If you have already loaded the first file, skip this step. Initialize the Assembler.

d) Re-enter the Assembler.

e) Get input by

!IO 20 <return>.

f) Repeat a) - e) until all files are loaded.

Your files are now merged in the Assembler. Be sure to inspect them carefully before assembling.

Remarks, Refinements, Additions

Resource will execute on 8K BASIC in a reasonable amount of time. The longest pass (PASS One) will run slightly less than an hour on a 1 MHZ machine.

This package of programs is, in a sense, incomplete. Using the cross reference tables, one could give mnemonic names to all of the various labels and equates. These could be entered into a file. Then one extra pass over OBJECT could exchange address labels with mnemonic labels.

A big file line editor utility could be added to edit any one of the files created. If tables are known at disassembly time, they can be edited into SCRATCH. Incorrectly disassembled code could be corrected. These steps could be performed also on SCRATCH or OBJECT.

If table locations are known in advance, disassembly can be cleaned up considerably by replacing all table bytes with $FF (or any other value not equal to a 6502 opcode). Then, all tables will appear in the resource as a sequence of lines: 10000 .BYTE $FF. Using an editor, it then would be a simple task to replace each $FF by its correct value. I used this procedure on BASIC.

OS65D cannot be changed in this way since it will crash. But there is a simple solution. Move OS65D from addresses 2XXX to addresses, say 5XXX. When SCRATCH and SYMBOL are complete, go through them, changing the leading 5's back to 2's. A program to do this is simple to write. SYMBOL must be resorted and repetitions deleted. This way, I was able to use the trick with $FF in tables to obtain an accurate resource of the code in OS65D.

A simple, but useful, utility would be a commenter. Such a utility would allow the user to add comments to the end of each line of the resourced file or to insert lines into the file. I have used this technique to produce the various listings in this article. I hope to present a future article on this editor.

Even though I am careful to fully document the machine software I write, I still find it useful to run the resource program over my machine programs. The cross reference files often reveal infelicities and logical inaccuracies.

I am still improving these programs. If you think of a nice enhancement, I'd be glad to hear about it.

Example 2.

```
1000  ;EQUATE FILE
1010 ;
1020  ;ZPAGE
1030 ;
1040  HHZZC1 = $C1
1050  HHZZC2 = $C2
1060  HHZZC5 = $C5
1070  HHZZC6 = $C6
1080  HHZZC7 = $C7
1090  HHZZC8 = $C8
1100 ;
1110 ;
1120  ;TWO BYTE
1130 ;
1140  HH17A5 = $17A5
1150  HH17E9 = $17E9
1160  HH1939 = $1939
1170  HH19BC = $19BC
1180  HH19D1 = $19D1
```

Example 3.

```
10 18D8                  * = $18D8
1000                   ;EQUATE FILE
1010                   ;
1020                   ;ZPAGE
1030                   ;
1040 00C1=             HHZZC1 = $C1
1050 00C2=             HHZZC2 = $C2
1060 00C5=             HHZZC5 = $C5
1070 00C6=             HHZZC6 = $C6
1080 00C7=             HHZZC7 = $C7
1090 00C8=             HHZZC8 = $C8
```

```
1100                ;
1110                ;
1120                ;TWO BYTE
1130                ;
1140 17A5=          HH17A5 = $17A5
1150 17E9=          HH17E9 = $17E9
1160 1939=          HH1939 = $1939
1170 19BC=          HH19BC = $19BC
1180 19D1=          HH19D1 = $19D1
10000 18D8 17              .BYTE $17
10010 18D9 A916            LDA #$16
10020 18DB 85C7            STA  HHZZC7
10030 18DD 20ED18  HH18DD JSR  HH18ED
10040 18E0 20D119          JSR  HH19D1
10050 18E3 85C5            STA  HHZZC5
10060 18E5 84C6            STY  HHZZC6
10070 18E7 C6C7            DEC  HHZZC7
10080 18E9 3037            BMI  HH1922
10090 18EB D0F0            BNE  HH18DD
10100 18ED 20BC19  HH18ED JSR  HH19BC
10110 18F0 A1C5            LDA (HHZZC5,X)
10120 18F2 A8              TAY
10130 18F3 4A              LSR A
10140 18F4 900B            BCC  HH1901
10150 18F6 4A              LSR A
10160 18F7 B017            BCS  HH1910
10170 18F9 C922            CMP #$22
10180 18FB F013            BEQ  HH1910
10190 18FD 2907            AND #$07
10200 18FF 0980            ORA #$80
10210 1901 4A      HH1901 LSR A
10220 1902 AA              TAX
10230 1903 BDA517          LDA  HH17A5,X
10240 1906 B004            BCS  HH190C
10250 1908 4A              LSR A
10260 1909 4A              LSR A
10270 190A 4A              LSR A
10280 190B 4A              LSR A
10290 190C 290F    HH190C AND #$0F
10300 190E D004            BNE  HH1914
10310 1910 A080    HH1910 LDY #$80
10320 1912 A900            LDA #$00
10330 1914 AA      HH1914 TAX
10340 1915 BDE917          LDA  HH17E9,X
10350 1918 85C1            STA  HHZZC1
10360 191A 2903            AND #$03
10370 191C 85C2            STA  HHZZC2
10380 191E A5C8            LDA  HHZZC8
10390 1920 D001            BNE  HH1923
10400 1922 60      HH1922 RTS
10410 1923 98      HH1923 TYA
10420 1924 298F            AND #$8F
10430 1926 AA              TAX
10440 1927 98              TYA
10450 1928 A003            LDY #$03
10460 192A E08A            CPX #$8A
10470 192C F00B            BEQ  HH1939
10480 192E 4A              LSR A
10490 192F 9008            BCC  HH1939
10500 1931 4A              LSR A
10510 1932 4A              LSR A
```

Example 4.

```
.  CROSS REFERENCES
.
.  ZPAGE
.
C1   1918
C2   191C
C5   18E3 )18F0
C6   18E5
C7   18DB  18E7
C8   191E
.
.
.  JMP & JSR
.
18ED  S18DD
19BC  S18ED
19D1  S18E0
.
.
.  MEMORY
.
17A5  L1903
17E9  L1915
.
.
.  BRANCH
.
18DD  N18EB
1901  C18F4
190C  C1906
1910  C18F7 E18FB
1914  N190E
1922  M18E9
1923  N1920
1939  E192C C192F
```

```
.PASS 1
.
.     RESOURCE 1 ** BUILD SCRATCH AND
 SYMBOL FILES **
.
.          TWO BUFFERS ARE REQUIRED
.
.
.
.
.
.
100 REM *** RESOURCE 1 ***
110 REM T.R. BERGER 11/80
120 PRINT
.
130 PRINT"RESOURCE ** STEP 1 - BUILD
 SCRATCH AND SYMBOL FILES"
.
140 PRINT
.
.  ** REMOVE COMMA AND SEMICOLON **
150 POKE 2972,13:POKE2976,13
.
160 INPUT"SOURCE FILENAME";SF$
170 INPUT"SCRATCH FILE NAME";JF$
180 INPUT"SYMBOL FILE NAME";SM$
190 PRINT:INPUT"NUMBER OF SYMBOLS";NS
.
.  ** DIMENSION SYMBOL AND POINTER ARRAYS
 **
200 DIM SS$(NS), V(NS)
.
.  ** MAIN PROGRAM **
```

```
.
210 DISK OPEN,6,SF$
220 DISK OPEN,7,JF$
.
.  ** LOOP BACK HERE **
230 INPUT #6,IN$
.
240 IF IN$="XIT" THEN590
250 IF LEN(IN$)<15 THEN230
.
.
.  ** ADJUST SOURCE, PICK UP SYMBOLS **
.
.  A1$=XXXX  ADDRESS
.  A2$=OPCODE +
.  A3$=OPERAND (SYMBOL)
.  A4$=ADDR MODE
.  OU$=A1$+A2$+A3$+A4$
.  IN$=INPUT FROM OSI DISASSEMBLER
.
.
260 A3$="": A4$=""
.
.  ** GET ADDRESS **
270 A1$=LEFT$(IN$,4)
.
.  ** DO ERRORS **
280 IFMID$(IN$,13,1)="?"THENA2$=" .BYTE $"
 +MID$(IN$,6,2):GOTO550
.
.  ** DO REFORMATTTING **
.
.  ** ELIMINATE END SPACES **
290 IN$=MID$(IN$,12): L=LEN(IN$)
300 IF MID$(IN$,L,1)=" " THEN L=L-1: GOTO
 300
310 IN$=LEFT$(IN$,L)
.
.  ** DO IMPLIED AND ACCUMULATOR ADDRESSI
NG **
320 IF L<7 THEN A2$=IN$: GOTO550
.
.  ** DO IMMEDIATE ADDRESSING **
330 IF MID$(IN$,6,1)="#" THEN A2$=IN$:
 GOTO550
.
.  ** ADJUST OPERAND POSITION **
340 IFMID$(IN$,6,1)="$"THENK=7:A2$=LEFT$(
 IN$,5)+" HH":GOTO360
350 K=8: A2$=LEFT$(IN$,6)+"HH"
.
.  ** Z PAGE CHECK **
360 M=K+2
.
.  ** DO Z PAGE OPERANDS **
370 IF M>L THEN A3$=RIGHT$(IN$,2): A2$=A2$
 +"ZZ": GOTO550
.
.  ** Z PAGE, STRIP ADDRESS MODE **
380 IF MID$(IN$,M,1)>"/" THEN400
390 A3$=MID$(IN$,K,2): A2$=A2$+"ZZ": A4$=
 MID$(IN$,M): GOTO550
.
.  ** TWO BYTE OPERAND CHECK **
400 M=K+4
.
.  ** DO TWO BYTE OPERANDS **
410 IF M>L THEN A3$=RIGHT$(IN$,4): GOTO430
.
.  ** TWO BYTE, STRIP ADDRESS MODE **
420 A3$=MID$(IN$,K,4): A4$=MID$(IN$,M)
.
.  ** PUT SYMBOLS IN TABLE **
.
.  ** SEARCH TABLE FOR SYMBOL **
.
.  ** THIS IS A BINARY SEARCH **
430 L=0:R=SN
.
.  ** SYMBOL NOT FOUND, INSERT IT **
440 IF L>R THEN490
450 M=INT((L+R)/2)
.
.  ** SYMBOL IN TABLE SO QUIT **
460 IF A3$=SS$(V(M)) THEN550
470 IF A3$>SS$(V(M)) THEN L=M+1: GOTO440
480 R=M-1: GOTO440
.
.  ** ADD SYMBOL **
490 SN=SN+1: SS$(SN)=A3$
.
.  ** POINT TO ITS PROPER POSITION IN ORDE
RING **
500 IF L=SN THEN540
510 FOR I=SN-1 TO L STEP -1
520 V(I+1)=V(I)
530 NEXT I
540 V(L)=SN
.
.  ** GENERATE LINE FOR SCRATCH FILE **
550 OU$=A1$+A2$+A3$+A4$
560 PRINT #7,OU$
570 PRINT OU$
580 GOTO230
.  ** LOOP BACK NOW **
.
.  ** CLOSE SCRATCH AND SOURCE FILES **
590 PRINT #7,IN$
600 DISK CLOSE,6
610 DISK CLOSE,7
.
.  ** END OF MAIN PROGRAM **
.
.
.  ** WRITE SYMBOL FILE **
.
620 DISK OPEN,7,SM$
.
.  ** WRITE SYMBOLS IN ORDER **
630 FOR I=0 TO SN
640 PRINT #7,SS$(V(I))
650 PRINT SS$(V(I))
660 NEXT I
670 PRINT #7,"XIT"
680 DISK CLOSE,7
.
.  ** OUTPUT DATA **
690 PRINT:PRINT
700 PRINT TAB(9) SN" SYMBOLS USED"
710 PRINT TAB(10) "SCRATCH FILE:  "JF$
720 PRINT TAB(10) "SYMBOL FILE:   "SM$
730 PRINT:PRINT TAB(10) "PASS 1 COMPLETE"
740 PRINT:PRINT:END
```

©

Review:

Caverns Of Mars

Charles Brannon
Editorial Assistant

Caverns of Mars is a new action-packed game from Atari, Inc. Originally sold through the Atari Program Exchange (APX), *Caverns of Mars* is now available through dealers.

The object of the game is to maneuver your spacecraft into the depths of the planet Mars. As you penetrate the five layers of defenses you can blow up fuel silos (to provide additional fuel for your ship), Martian missiles, and "spy stations" that supposedly warn the Martian headquarters of your approach.

Your first challenge is to weave your way through the twisting subterranean (submartian?) tunnels. The screen continuously scrolls upward, so you move left and right mostly, although your craft can also move vertically. When you press the fire button on the joystick, your craft shoots twin photon torpedoes that detonate any target they contact.

After surviving these levels, you face an armada of Martian spacecraft. They don't fire at you; they just try to overwhelm you with sheer numbers, forcing you to dodge them and fire furiously to avoid a fatal collision. If you can make it past this defense, you slowly descend into the very center of the Martian stronghold, where you'll find a giant, glowing, egg-shaped bomb. By landing on it, you "arm" it, and the timer starts ticking at 30 seconds, 29, 28, 27... The destruction of the base is inevitable, but *your* survival is at stake too. If you don't escape from the Martian base before the time is up, you are ruined in a brilliant orange explosion. Assuming that you make it out (not very likely in your first games), you watch the explosion lighting up the mouth of the cavern, and you can then valiantly enter... cavern *two*.

Each new cavern is progressively harder – most noticeably when making your escape – you ascend twice as fast, making a collision with the "ceiling" rather likely. I've played the game for a week, and I still can't escape Cavern two.

You get five ships, each with 99 units of fuel. Here's the catch – if you run out of fuel, your ship explodes. It is not uncommon to meet this fate just moments before arming the bomb, or before rising triumphantly to the surface. If your ship blows up in a certain level, you have to start over at the beginning of the level.

The graphics are impressive. Apparently, the game is constructed using a custom character set in the special four-color IRG graphics mode (a rather unexploited feature of the Atari until now). Most of the animation is performed with vertical fine scrolling, and the ship is a player. It also uses a modified display list to show your status, and I think it even uses display list interrupts. All in all, an excellent use of the Atari's graphics capabilities.

Caverns of Mars is currently only available on diskette. You'll also need a joystick.

A curious feature of the program is that it names its skill levels as *Star Raiders* does: Novice, Pilot, Warrior, and Commander. Will *Caverns of Mars* replace *Star Raiders* in the hearts of Atari space-game fans? If so, we may soon hear a new phrase: "Old Star Raiders never die, they just move to Mars..."

Caverns Of Mars
Available through dealers.
$39.95

©

Review:

Two Programs From The VIC 6 Pack

Harvey B. Herman
Associate Editor

Car Chase

This is a clever game which my kids rated 8 (out of 10). There are two cars (the blue one is yours) traveling in opposite directions in a square, four-lane track. The object of the game is to score the most points before you crash into the other car. You can change lanes or speed up or slow down to avoid a collision. Points are scored by riding over dots (and later diamonds) on the tracks. This game is very similar to the PET program *NAB* sold by *Cursor* magazine.

We (my kids and I) particularly liked several features of the BASIC game. Color and sound are integrated into the program. Effective use of a custom character set makes the track layout much nicer than if only the standard characters had been used. The other car is smart and you must be on your toes to avoid a crash. To score well you must change speeds at appropriate times or you're stuck in the wrong lane, headed for disaster.

The one feature I did not like (and this did not seem to bother my kids) is that the keyboard is used as the car controller. I prefer a joystick or the number pad on the old PETs. The "J" and "K" keys used to change lanes are close together and I was forever confusing which was "move in" and "move out." You don't have to think as much with a joystick.

A helpful hint: the instructions say not to use Car Chase with any memory expansion in place. Bullfeathers! If you have added the 3K memory expansion a special load sequence is necessary. Type:

```
POKE 44,16
POKE 4096,0
NEW
LOAD
```

Before loading other programs, POKE location 44 back to normal or turn the VIC off and on.

Blue Meanies From Outer Space

This is an Invaders-like game which my kids rated 6 (out of 10). The blue meanies (frowning faces) are dropping from the skies trying to knock out your protected power cells inside your starbase. You are able to blast them with fixed laser guns as they fall erratically. Points are added when you kill a meanie or deducted if you hit your own supply ship which is coming to recharge your energy. You must have energy to fire your weapons. Energy is lost when shooting and for various other operations. If the supply ship makes it, you get an energy bonus. It is possible to repair damage (holes in your starbase) using a robot, but at the cost of energy. When the meanies reach the power cells, the game is over.

This BASIC game, like Car Chase, is considerably enhanced by good use of color, sound, and a custom character set. It was fun to shoot the meanies, but it would have been even more fun if they were a little smarter. (However, the instructions do promise "meaner meanies" after 20 are shot.) Again, I prefer a joystick to the use of the keyboard for game control. I get confused if the required actions are not reflexive. My kids commented that the instructions for the repair robot were not clear and they did not care for negative energy or a suspended robot. I believe they have been spoiled by playing Invaders on the PET. Blue Meanies is a good program, but it doesn't quite match up to that very good simulation of the video arcade game.

Note: if you have added a 3K memory expansion, use the same trick as above for Car Chase to LOAD Blue Meanies.

Commodore Business Machines
681 Moore Rd.
King of Prussia, PA 19406
$43.95 (for all 6)

©

Review:

SoftBox — CP/M For PET/CBM

Richard Mansfield
Assistant Editor

A considerable amount of software requires the disk operating system known as CP/M. Originally developed in 1973, it has since become an essential part of many word processor, accounting, data base manager and other commercial software products. CP/M means "Control Program/Monitor" and was intended for use with the 8080 and Z80 microprocessors. "SoftBox," manufactured by Small Systems Engineering, however, allows you to use CP/M-based software packages on the PET/CBM computers.

Attached between your PET and disk drive, this "interface-computer" contains 64K RAM, a Z80 chip, IEEE and RS-232 interfaces with software-definable baud rates, and a Corvus hard disk interface. The unit will work with any PET/CBM using Upgrade or 4.0 BASIC and 8K RAM. An optional cassette program can be used with Original PETs. Any combination of from one to eight Commodore floppy disk drives (2040, 3040, 4040, or 8050) may be attached, resulting in up to a potential eight megabytes mass memory available at one time. Memory can be further increased by attaching Corvus hard disk drives. A total of 80 megabytes is possible.

Simple To Set Up

Adding CP/M to your PET takes only a few minutes. You plug a standard PET-to-IEEE cable between the SoftBox and your disk drive. (Another of these cables goes from the drive to the PET.) When you power-up the three units, everything is normal and, in fact, you can work with the computer and the disk drive as if SoftBox were not connected.

For CP/M, you place the supplied disk (CP/M Version 2.2) into Drive zero, LOAD, and RUN it. That's it. PET is now under the control of CP/M. The disk contains a variety of CP/M utilities, special Corvus files, and a SoftBox memory test diagnostic program.

If you've not worked with CP/M, the SoftBox User's manual is an excellent introduction. It takes you through the major CP/M commands, step-by-step, and clearly demonstrates their uses and terminology. Special commands allow you to control the PET screen (upper/lowercase, graphics/text), the TI$ function, and, when you want to return to normal PET BASIC, simply type "cold" and you're back in the Commodore.

And for the more advanced user, several appendices provide technical information, a map of SoftBox's memory, suggestions on modifying CP/M itself, and I/O details.

SoftBox, $895
Small Systems Engineering
222 B View Street
Mountain View, CA 94041 ©

Review:

SYSRES For PET/CBM

Charles Brannon
Editorial Assistant

SYSRES claims to be the "ultimate program manipulation system." Like its predecessors, the Toolkit and BASIC Aid, SYSRES is a programmer's aid package. SYSRES "wedges" into BASIC and adds 33 commands (including six improved BASIC commands) plus 11 extended DOS Support (the original "Wedge") commands. All of the commands are used for the development and maintenance of BASIC programs.

The primary tools needed by a programmer are: auto line numbering, renumbering, find, and trace. SYSRES supplies "deluxe" versions of all these. Automatic line numbering provides the line numbers when typing in a program. If you start at line 100, it will automatically print 110 when you press RETURN on line 100 (assuming a step size of ten). SYSRES permits as unique variation: automatic statement generation. If you are typing in a long set of DATA statements, SYSRES can provide the line numbers and the word "DATA." SYSRES offers a "fill in the blank" line where you only need to enter a single number. It can even press RETURN for you, giving it the capability for automatic line entry.

Advanced Renumber

Most Renumber commands change the line numbers of a BASIC program by an even increment and update all line references made by GOTO, THEN, GOSUB, etc. The entire program is renumbered. SYSRES permits partial renumber. You can renumber just a subroutine, for example, preserving all other line numbers, and updating any line references if necessary.

Perhaps the most useful command in a package like this is a Find command. Many programmers find it indispensable. Using Find, you can locate any variable, phrase, or keyword used anywhere in your program. Another related command is Change, which permits you to replace any variable, phrase, or keyword with anything else.

For the Change command especially, it is vital that you exactly specify the search string. You may want to change A$ to B$, but you don't want ZA$ to become ZB$. To zero in on whatever you are searching for, SYSRES permits you to use several wild card and limitation symbols (such as V for Variable, B for Beginning of line, or a quote to signal quote mode), which together permit over 700 possibilities.

Debugging a program is much easier with a Trace command. Trace (as implemented by SYSRES) permits you to display program variables as they change or are defined. So, you could display the index of a FOR/NEXT loop, or watch how certain variables interact. (You can also type DUMP in direct mode to display all variables.)

The advanced editor commands let you use the BASIC editor as a general purpose ASCII line editor. The commands GET and PUT let you save this ASCII file to disk. SYSRES even partially tokenizes this file to save disk space. These ASCII files can be used to develop EXEC files, which are a sequence of direct commands that are executed as if they were a program. This gives the PET/CBM the power found formerly only in CP/M's .EXC files, Apple's EXEC files, or Atari's ENTER files. An important feature of EXEC files is that they use no program memory as they execute.

When using SYSRES, you get a feeling that careful design went into the system. All keys repeat, and the cursor keys "take off" quickly. Full up/down scrolling of your BASIC program is supported, and it works perfectly. If you display the directory with the improved "›$0" command, these line numbers cannot be accidentally entered as program lines or activate the scrolling feature.

Full printer support is also included with full ASCII translation. Any command can be sent to the printer if preceded with an asterisk. For example, *LIST would list a program to the printer. You can also dump the screen by holding down both SHIFT keys and pressing RVS. Speaking of keys, with SYSRES you can define any shifted key to equal any sequence of keys or keywords. When you define any key, all the alphabetic keys "come alive" with convenient single-key abbreviations.

Improvements To BASIC Itself

The improved BASIC commands are another example of the care taken to produce a complete "operating system." The LOAD, SAVE, and VERIFY commands default to the disk drive, and the save with replace ("@") works on any disk drive. (It first SCRATCHes, then resaves the program.) Nevertheless, none of the commands are programmable. Therefore, programs developed with SYSRES are fully transportable As the manual says, "SYSRES is designed to be 'addictive to programmers *not* to programs.' "

SYSRES
Solidus International Corporation
Suite 6 – 144 West 15th Street
North Vancouver, B.C.
Canada, V7M 1R5.
$75.

©

Review:

Fun With Microcomputers And BASIC

Fred D'Ignazio
Associate Editor

Do you like puzzles and games?

Are you looking for a quick, painless way to learn how to program a personal computer?

Are you a kid, an adult, or something in between?

If you answered *yes* to all these questions, then this book is for you.

I am a person who has a low pain threshold, when it comes to plowing through "introductory" guides to personal computers. "Ouch!" I cry when I see guides set in a midget-sized typeface that resembles footprints made by a tapdancing flea.

The same goes when I see guides with no pictures, or guides that are too technical, too serious, too somber, or too trivial. These books give me indigestion – the same lumpy discomfort I experience when I swallow a plateful of spaghetti and meatballs I forgot to chew.

Don Spencer's book is different. It's fun, it really teaches, and it's an almost painless introduction to microcomputers and BASIC programming.

The book starts at a lofty level. In the first ten pages, Spencer skims over computers' social impact, computer devices, trends, and careers.

Then you get down to business. You plan a program, learn a little about flowcharting, get a few tips on how to write your first program.

Then you dive into BASIC. For twenty pages, you swim through constants and variables, arithmetic operations and DATA statements. You finish by looping through arrays.

Then come seventy pages of programs including Manhattan Island, Sam's Monkey, The Jolly Green Giant, and Roger Goes to the Circus. If you are math-oriented or a compulsive gambler, you'll really like these programs.

I recommend this book to teachers of introductory programming courses, and to parents, kids, and anxious professionals just getting started on their new computer. The problems start easy and get harder. Almost all come with actual "solution" programs you can study. There are even occasional exercises for you to tackle.

This is what the book offers.

What does it lack?

It is only an *introduction* to microcomputer BASIC, so don't expect to be a programming whiz when you are done – especially with regard to a particular machine. Also, there is almost nothing on sound or graphics – two popular features of the new, low-cost personal computers.

Still, it's a good start. Whether you're old or young. Whether you're cocky about computers or afraid to touch one for fear it might bite off your hand. Or explode.

Fun with Microcomputers and BASIC
by Donald D. Spencer
Reston Publishing Co.
11480 Sunset Hills Rd.
Reston, VA 22090
128 pages
$9.95 paperback

©

COMPUTE!'s Listing Conventions

Many of the programs which are listed in **COMPUTE!** use special keys (cursor control keys, color keys, etc.) To make it easy to tell *exactly* what should be typed in when copying a program into the computer, we have established the following listing conventions.

For The Atari

All the editing and cursor control characters are spelled out and surrounded by brackets in the program listings: {CLEAR} for "clear screen." Other characters, such as CTRL-T (the "ball" character) will be listed as the "normal" character, but it will be within brackets: {T}. A series of identical control characters will be indicated by a number within the brackets: {3DOWN} means type ESC CURSOR-DOWN three times; {12 R} would mean type CTRL-R twelve times. Remember to press the ESC (escape) key before each cursor control key. If you should see {ESC} itself in a program listing, you would press ESC *twice*.

Two of the control characters, {=} and {-}, should be shifted. Any reverse field text will be enclosed within vertical lines. (In other words, any time you see a vertical line within a program listing in **COMPUTE!**, press the Atari logo key {⅄}.)

Atari Conventions

```
{CLEAR}= SHIFT-< (Clear Screen)
{UP}= CTRL-minus (Cursor Up)
{DOWN}= CTRL-equals (Cursor Down)
{LEFT}= CTRL-plus (Cursor left)
{RIGHT}= CTRL-asterisk (Cursor right)
{BACK S}= BACK S (Back space)
{DELETE}= CTRL-DELETE (Delete character)

{DEL LINE}= SHIFT-DELETE (Delete Line)
{INSERT}= CTRL-INSERT (Insert character)

{INS LINE}= SHIFT-INSERT (Insert line)
{ESC}= ESC (ESCape key pressed twice)
{TAB}= TAB (Tab key)
{CLR TAB}= CTRL-TAB (Clear tab setting)
{SET TAB}= SHIFT-TAB (Set tab stop)
{BELL}= CTRL-2 (Ring buzzer)
```

For PET/CBM/VIC

Generally, any PET/CBM/VIC program listings will contain bracketed words which spell out any special characters: {DOWN} would mean to press the cursor-down key; {3DOWN} would mean to press the cursor-down key three times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listing. For example, S̲ would mean to type the S key while holding the shift key. This would result in the "heart" graphics symbol appearing on your screen.

Sometimes in a program listing, especially within quoted text when a line runs over into the next line, it is difficult to tell where the first line ends. How many times should you type the SPACE bar? In our convention, when a line breaks in this way, the ~ symbol shows exactly where it broke. For example:

```
100 PRINT "TO START THE GAME  ~
    YOU MAY HIT ANY OF THE KEYS
    ON YOUR KEYBOARD."
```

shows that the program's author intended for you to type two spaces after the word *GAME*.

For The Apple

Programs listed as "Microsoft" are written for the PET/CBM, Apple, OSI, etc. Although the programs are general in nature, you may need to make a few changes for them to run correctly on your Apple. Microsoft BASIC programs written for the PET/CBM sometimes contain special cursor control characters. The following table shows equivalent Apple words. Notice that these Apple commands are *outside* quotations (and even separate from a PRINT statement). PRINT"[RVS]YOU WON" becomes INVERSE: PRINT"YOU WON":NORMAL

[CLEAR[(Clear Screen) HOME

[DOWN] (Cursor down)
Apple II +: Call -922
POKE 37,PEEK(37) + (PEEK(37)<23)

[UP] (Cursor up)
POKE 37,PEEK(37)-(PEEK(37)>0))

[LEFT] (Cursor left) PRINT CHR$(8);

[RIGHT] (Cursor right)
PRINT CHR$(21)

[RVS] (Inverse video on. Turns off automatically after a carriage return. To be safe, turn off inverse video after the print statement with NORMAL unless the PRINT statement ends with a semicolon.)
INVERSE

[OFF] (Inverse video off) NORMAL

Shifted characters can represent either graphics characters or uppercase letters. If within text, just use the non-shifted character, otherwise substitute a space. Some "generalized" programs contain a POKE such as POKE 59468,14. Omit these from the program when typing it in. One final note: you will probably want to insert a question mark or colon within an INPUT prompt. PET/CBM and many other BASICs automatically print a question mark:

INPUT "WHAT IS YOUR NAME";N$
becomes
INPUT "WHAT IS YOUR NAME?";N$

All Commodore Machines

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

VIC Conventions

Set Color To Black	{BLK}	**Function Two**	{F2}
Set Color To White	{WHT}	**Function Three**	{F3}
Set Color To Red	{RED}	**Function Four**	{F4}
Set Color To Cyan	{CYN}	**Function Five**	{F5}
Set Color To Purple	{PUR}	**Function Six**	{F6}
Set Color To Green	{GRN}	**Function Seven**	{F7}
Set Color To Blue	{BLU}	**Function Eight**	{F8}
Set Color To Yellow	{YEL}	**Any Non-implemented Function**	{NIM}
Function One	{F1}		

8032/Fat 40 Conventions

Set Window Top	{SET TOP}	**Erase To Beginning**	{ERASE BEG}
Set Window Bottom	{SET BOT}	**Erase To End**	{ERASE END}
Scroll Up	{SCR UP}	**Toggle Tab**	{TGL TAB}
Scroll Down	{SCR DOWN}	**Tab**	{TAB}
Insert Line	{INST LINE}	**Escape Key**	{ESC}
Delete Line	{DEL LINE}		

©

COMPUTE! Back Issues

Here are some of the applications, tutorials, and games from available back issues of **COMPUTE!**. Each issue contains much, much more than there's space here to list, but here are some highlights:

January, 1981: Load PET Programs Into The Apple II, Player-Missile Graphics for Atari, The Atari DOS, The Kernel of the OSI Operating System, Fixing LOADing Problems on the PET, Spooling with the PET Disk, Expanding KIM.

February, 1981: Simulating PRINT USING, Using the Atari as a Terminal for Telecommunications, Attach a Printer to the Atari, Double Density Graphing on C1P, Commodore Disk Systems, PET Crash Prevention, A 25¢ Apple II Clock.

March, 1981: Machine Language Programming for Beginners, Getting the Most from your PET Cassette Deck, Apple and PASCAL, Flipping your Apple Disk, Designing your own Atari Character Sets, Renumber for Atari, An Atari Disassembler, Six-gun Shootout Game for OSI C1P, PET Machine Language Graphics.

April, 1981: How to be a VIC Expert, Resolving the Applesoft and Hires Graphics Memory Conflicts, Atari SuperCube, String Arrays in Atari, Memory Partition in PET, Pet Relative Files, Working with BASIC 4.0, Commodore File I/O, ROM Expansion for Commodore PET.

May, 1981: Named GOSUB/GOTO in Applesoft, Generating Lower Case Text on Apple II, Copy Atari Screens to the Printer, Disk Directory Printer for Atari, Realtime Clock on Atari, PET BASIC Delete Utility, PET Calculated Bar Graphs, Running 40 Column Programs on a CBM 8032.

June, 1981: Computer Using Educators (CUE) on Software Pricing, Apple II Hires Character Generator, Ever- expanding Apple Power, Color Burst for Atari, Mixing Atari Graphics Modes 0 and 8, Relocating PET BASIC Programs, An Assembler In BASIC for PET, QuadraPET: Multitasking?

July, 1981: Home Heating and Cooling, Animating Integer BASIC Lores Graphics, The Apple Hires Shape Writer, Adding a Voice Track to Atari Programs, Machine Language Atari Joystick Driver, Four Screen Utilities for the PET, Saving Machine Language Programs on PET Tape Headers, Commodore ROM Systems, The Voracious Butterfly on OSI.

August, 1981: Minimize Code and Maximize Speed, Apple Disk Motor Control, A Cassette Tape Monitor for the Apple, Easy Reading of the Atari Joystick, Blockade Game for the Atari, Atari Sound Utility, The CBM "Fat 40," Keyword for PET, CBM/PET Loading, Chaining, and Overlaying.

September, 1981: The Column Calculator, What is a Modem and Why Do I Need One?, PET, Apple, Atari: On Speaking Terms, A Tape "EXEC" for Applesoft, A Self-altering Program for Apple II, Positioning P/M Graphics and Regular Graphics in Memory, An Atari BASIC Sort, Shoot, an Arcade Game for Atari, Exploring OSI's Video Routine, PET Tape Append and Renumber, All About LOADing PET Cassettes.

October, 1981: Automatic DATA Statements for CBM and Atari, VIC News, Undeletable Lines on Apple, PET, VIC, Budgeting on the Apple, Switching Cleanly from Text to Graphics on Apple, Atari Cassette Boot-tapes, Atari Variable Name Utility, Atari Program Library, Train your PET to Run VIC Programs, Interface a BSR Remote Control System to PET, A General Purpose BCD to Binary Routine, Converting to Fat-40 PET.

November, 1981: SuperPet: A Preview, Japanese Micros: A First Look, Introduction to Binary Numbers, An Apple Primer, Page Flipper for Apple, An Atari Database System, A Program for Writing Programs on the Atari, Atari Textplot, OSI Relocation, The PET Speaks, Inversion Partitioning, A Personal News Service on PET, Bits, Bytes, and Basic Boole.

December, 1981: Saving Fuel $$ (Multiple Computers: versions for Apple, PET, and Atari), Unscramble Game (multiple computers), Maze Generator (multiple computers), Animating Applesoft Graphics, A Simple Printer Interface for the Apple II, A Simple Atari Wordprocessor, Adding High Speed Vertical Positioning to Atari P/M Graphics, OSI Supercursor, A Look At SuperPET, Supermon for PET/CBM, PET Mine Maze Game.

January, 1982: Invest (multiple computers), Developing a Business Algorithm (multiple computers), Apple Addresses, Lowercase with Unmodified Apple, Cryptogram Game for Atari, Superfont: Design Special Character Sets on Atari, PET Repairs for the Amateur, Micromon for PET, Self-modifying Programs in PET BASIC, Tinymon: a VIC Monitor, Vic Color Tips, VIC Memory Map, ZAP: A VIC Game.

February, 1982: Insurance Inventory (multiple computers), Musical Transposition (multiple computers), Multitasking Emulator (multiple computers), Disassemble Apple Programs from BASIC, Plotting Polar Graphs on Apple, Atari P/M Graphics Made Easy, Atari PILOT, Put A Rainbow in your Atari, Marquee for PET, PET Disk Disassembler, VIC Paddles and Keyboard, VIC Timekeeping.

March, 1982: Word Hunt Game (multiple computers), Infinite Precision Multiply (multiple computers), Atari Concentration Game, VIC Starfight Game, CBM BASIC 4.0 To Upgrade Conversion Kit, Apple Addresses, VIC Maps, EPROM Reliability, Atari Ghost Programming, Atari Machine Language Sort, Random Music Composition on PET, Comment Your Apple II Catalog.

April, 1982: Track Down Those Memory Bugs (multiple computers), Shooting Stars Game (multiple computers), Intelligent Input Subroutines (multiple computers), Ultracube for Atari, Customizing Apple's Copy Program, Using PET/CBM In The High School Physics Lab, Grading Exams on a Microcomputer (multiple computers), Atari Mailing List, Renumber VIC Programs The Easy Way, Browsing the VIC Chip, Disk Checkout for PET/CBM.

Back issues are $3.00 each or six for $15.00. Price includes freight in the US. Outside the US add $1.00 per magazine ordered for surface postage. $3.00 per magazine for air mail postage. All back issues subject to availability.

In the Continental US call
TOLL FREE 800-334-0868
(In NC Call 919-275-9809)

Or write to **COMPUTE!** Back Issues, P.O. Box 5406, Greensboro, NC 27403 USA. Prepayment required in US funds. MasterCard, Visa and American Express accepted. North Carolina Residents add 4% sales tax.

CAPUTE!: Modifications Or Corrections To Previous Articles

Further Notes on "Fast Sort For PET/CBM"

The following information from Jim Russo will aid in adapting the PET/CBM machine language sort utility to work within a BASIC program. "Fast Sort" appeared in **COMPUTE!**, May, 1982, #24, pg. 160:

This routine operates on strings of equal length which occupy contiguous memory locations. It does not know about variable names or arrays. The example given in the article uses Program 2 to collect some data, and then Program 3 to read it, in order, into a fresh string data space, without any extra string variables. Both functions could be done in the same program if the data was written into an array as it was collected, and then the following line was used to move the data into a contiguous area at the beginning of string memory:

```
FORI=ØTOTN-1: A$(I)=A$(I):NEXT
```

In a larger program with more than one array, this same technique could be used to move each array to the appropriate place when it was time for it to be sorted.

Line 150 in Program 3 will fail for certain values of length and number of records. A better way to write it is:

```
150 AA=256*HA+LA+LN+2:POKE179,AA AND 255:PO
    KE18Ø,AA/256
```

The two extra bytes used in each string by BASIC 4.0 are a pointer back to the variable data area, where the string length and starting address are stored. After a sort, these pointers are not valid. No harm will be done *as long as none of the sorted strings is redefined by the program.* If some of the sorted strings are redefined by the program, then a subsequent garbage collection will cause chaos. To illustrate this, try adding the following line to Program 3:

```
175 A$(1)="NEW STRING #1":X=FRE(Ø)
```

If there is enough room for two copies of the array data in string memory, then the following two lines will fix the problem:

```
125 IF FRE(Ø) < 27Ø+TN*(LN+2)THENPRINT"NO R
    OOM":STOP
175 FORI= Ø TO TN-1: A$(I)=A$(I): NEXT
```

Line 125 forces a garbage collection, and makes sure there will be enough room for line 175 to execute without causing another garbage collection. Line 175 fixes the pointers by rewriting all of the sorted strings. ©

New Products

Hayden Announces Tetrad: A New Way To Play Tic-Tac-Toe

Tetrad is a game of strategy and mental daring that enables a player to match wits against a computer. Based on the popular game of Tic-Tac-Toe, Tetrad allows the tension to build between opponents as each tries to get four X's or O's in a row in any direction on any or all four playing boards.

Tetrad has a board that offers 16 squares on four different levels. The additional squares and boards allow 76 possible winning combinations – nine times more than conventional, two-dimensional Tic-Tac-Toe.

Players can choose from three levels: novice, intermediate or professional. Colorful graphics accompanied by unique sound effects make Tetrad a new way to play an old favorite.

Tetrad is written by Philip Hess and can be used on an Apple II disk with 32K, 09809, $19.95.

Hayden Book Co., Inc.
50 Essex St.
Rochelle Park, NJ 07662
Toll free (800)631-0856
or NJ: (201)843-0550

Corvus Announces Mass Storage And Network Systems For Atari 800

Corvus Systems, Inc. announced the availability of Winchester disk systems and the Corvus Multiplexer local area network for the Atari 800 microcomputer.

This announcement means that Atari 800 owners can join microcomputer users who enjoy the speed, reliability, and storage capacity of Corvus 6, 11, and 20 megabyte disk systems. It also means that, with the Corvus Multiplexer local network, Atari users can share mass storage as well as printers and other expensive peripherals. Corvus will also offer its Mirror backup system which uses video tape technology for Winchester disk backup. Prices of Corvus disk systems for the Atari 800 start at $3,195 complete with interface and all required software.

Corvus Systems, Inc.
2029 O'Toole Avenue
San Jose, CA 95131
(408)946-7700

Software Publisher Releases P/SAT Preparatory

Edu-Ware has announced two software tutorials for college-bound students on mastering vocabulary and deciphering new words. PSAT Word Attack Skills and SAT Word Attack Skills prepare students specifically for the "antonyms" portion of the Scholastic Achievement Test.

PSAT and SAT Word Attack Skills each contain two diskettes: Prefixes and Roots. Vocabulary words, selected for their frequent appearance on the PSAT/SAT, are grouped into lessons (six per diskette). In each lesson, students first pinpoint a word's meaning by selecting its synonym. The word is then reviewed through definition, sample sentences, analysis of its components (root and/or prefixes), and a test question. A timed test then prepares students to function under pressure.

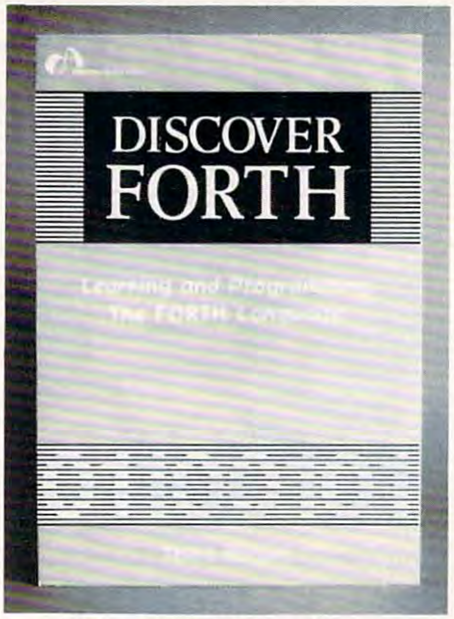
PSAT Word Attack Skills and SAT Word Attack Skills are independent tutorials, yet can be purchased and used together as a comprehensive sequence. Both systems are available for the Apple II and II Plus computers, 48K, DOS 3.3. Each two-disk package retails for $49.00.

Edu-Ware Services, Inc.
P.O. Box 22222
Agoura, CA 91301

Discover FORTH Announced By Osborne/ McGraw-Hill

Osborne/McGraw-Hill has released a new book by Thom Hogan entitled *Discover FORTH: Learning and Programming the FORTH Language.*

This introduction to FORTH, the computer language of building blocks, is written in a friendly, informal style. Beginners will find information on this multi-faceted language and instructions to guide their programming skills up to an intermediate level. More experienced programmers can use Hogan's



book as a reference tool.

In the text, Hogan describes FORTH syntax, specifically applicable to both FORTH-79 and FIG-FORTH. Notes are included on logical extensions and alternatives to the current standard FORTH syntax.

Discover FORTH provides a history of the language and a synthesis of material from programming manuals, independent programmers, and publications of the FORTH Interest Group.

Discover FORTH: Learning and Programming the FORTH Language
by Thom Hogan
$15.00 Paperback
250 pages
Osborne/McGraw-Hill
630 Bancroft Way
Berkeley, CA 94710

To place an order call: (415)548-2805.

Apple Announces Price Drop For Monitor III And Apple II Stand

Apple Computer, Inc. announces a price reduction for its Monitor III video display, and introduces

a stand which permits easy use of the Monitor III with the Apple II personal computer.

The suggested retail price of the Monitor III is now $249.00, a 22% reduction, and applies to both white and green phosphor versions. The new pricing reduces the total system cost for both the Apple II and the Apple III.

The Apple II Stand is designed to the exact width of the Monitor III to allow aesthetic integration into the Apple II system. The stand permits easy accessibility to the back of the Apple II. The suggested retail price is $29.95.

Apple Computer
20525 Mariani Ave.
Cupertino, CA 95014

Sybex Releases Book For Business Managers

Executive Planning with BASIC by X. T. Bui has just been released by Sybex. With this book, the business manager will learn to use a personal computer to accelerate decision-making and planning methods.

Up-to-date BASIC decision model computer programs for business management are presented including: Cost/Volume/Profit, Linear Programming, Inventory Management, Critical Path Analysis and PERT, Exponential Smoothing, Linear and Multi-Linear Regression, Financial Ratio Analysis, Discounted Cash Flow, Portfolio Management and more.

The development, rationale and proper use of each decision model is carefully outlined. True-to-life application examples are provided and problems are followed up and solved in sample runs of the programs. Three appendices complete the text: a summary of BASIC instructions; a collection of subroutines for matrix Algebra; and a summary of the most important business statistics tools along with BASIC programs that perform these algorithms.

Executive Planning With BASIC
by X. T. Bui
$12.95
Ref. B380 ISBN: 0-89588-083-0
Sybex
2344 Sixth Street
Berkeley, CA 94710
(415)848-8233

Intelligent Modem From BIZCOMP

BIZCOMP Corporation announces the immediate availability of Model 1012, a microprocessor-based 1200/300 baud Intelligent Modem. The 1012 features full Bell 212A compatibility and BIZCOMP's keyboard dialing.

An integral serial auto-dialer allows keyboard dialing while the low-power, LSI circuit design gives reliable, cool running operation. A Thompson pre-filter assures excellent data integrity. The built-in microprocessor provides a full complement of powerful commands for controlling dialing, auto-answer, self testing and programming modem parameters.

Front panel controls have been ergonometrically designed for efficient and simple use by nontechnical office personnel. The 1012 is fully FCC registered for direct connection and is packaged in a stylish, low-profile injection enclosure which may be used as a base for a desk telephone.

Applications include remote datastations, store-and-forward electronic mail, computer-computer file transfers and automatic polling of unattended locations.

Inquiries Manager
BIZCOMP Corporation
P.O. Box 7498
Menlo Park, CA 94025
(415)966-1545

EPYX Releases Exciting New Quest Game For The Atari

EPYX has released King Arthur's Heir, a new fantasy quest game for the Atari 400/800 home computer.

In this legendary realm, the player serves at the King's right hand. Arthur, King of the Britons, has named him heir to the throne.

But the player must prove himself worthy to hold the crown of Camelot. Arthur commands that he shall go in quest of the treasured Scroll of Truth, hidden by the great wizard, Merlin.

By moving his joystick, the player's computer displays, in full color graphics, the magical world of Camelot and its mystical surroundings.

Being faithful to the Crown, the player accepts the royal command to retrieve the scroll. Merlin forewarns him of mysterious places, gruesome creatures, evil forces and magical objects that enter his quest at every turn.

Defying all perils, the player journeys through an ominous forest, a supernatural castle, treacherous caves and magical cities – a few of the seven types of realms that entice him. Some hold necessary supplies and even treasures, but in others he is surrounded by evil.

The player must do all in his power to ward off dragons, bargain with wizards, and subdue the devil himself, as more than these will try to waylay him in his quest.

But, amazing treasures will spur him on. A staff that emanates the most powerful of magic and a ring with supernatural powers are among the wonders the player may find.

THE MONKEY WRENCH™ FOR ATARI

$49.95

A BASIC and machine language programmers aid for 800 users. Plugs into right slot and works with ATARI BASIC. Adds 9 new direct mode commands including auto line numbering, delete lines, change margins, memory test, renumber BASIC, hex/dec conversion, cursor exchange, and machine language monitor.

The monitor contains 15 commands used to interact with the 6502. Some are display memory/registers, disassemble, hunt, compare, hex/dec convert, transfer memory, and printer set/clear. Uses screen editing.

CASSETTE BASED MACRO ASSEMBLER/EDITOR

$49.95

"The Compatible Assembler/Editor"

- Macros, Conditional Assembly, String search and/or replace, standard mnemonics, (Ex: LDA (LABLE), Y)
- Long labels, MOVE, COPY, AUTO, DELETE, PUT, GET, etc.

EPROMS – HIGH QUALITY, NOT JUNK

Use with PET, APPLE, ATARI, SYM, AIM, etc. 450 ns. $6.50 for 2716, $12.50 for 2532.

EPROM PROGRAMMER FOR PET AND ATARI COMPUTERS

The BRANDING IRON is an EPROM programmer especially designed for PET and ATARI computers. Programs 2716 and 2532 type EPROMs. The PET version plugs into the cassette and I/O port and comes with software which adds the programmer commands to the PET monitor. The ATARI version plugs into controller jacks and comes with a full fledged machine language monitor which provides 30 commands for interacting with the computer and the BRANDING IRON.

PET – $75.00 ATARI – $119.95

5¼ INCH SOFT SECTORED DISKETTES

Highest quality. We use them on our PETs, APPLEs, ATARIs, and other computers. $22.50/10 or $44.50/20

PET TERMINAL SOFTWARE

A buy you RS-232 users can't pass-up. Includes RS-232 hardware with a sophisticated software package. May be controlled via keyboard or from BASIC. A super buy. $129.95

STARWRITER F-10 DAISY WHEEL PRINTER

PARALLEL – $1495, RS-232 – $1680, TRACTORS – $210

SIGNALMAN MARK I DIRECT CONNECT MODEM – $89.50

Standard 300-baud, full duplex, answer/originate. Powered by long lasting 9-volt battery (not included). Cable and RS-232 connector included.

MAE SOFTWARE DEVELOPMENT SYSTEM FOR PET, APPLE, ATARI

$169.95

"The Compatible Assembler"

- Professional system for development of Machine Language Programs. 31 Characters per label.
- Macro Assembler/Text Editor for Disk-based systems.
- Includes Word Processor for preparation of Manuals, etc.
- Standard Mnemonics – Ex.: LDA (LABEL), Y
- Conditional Assembly, Interactive Assembly.
- Editor has string search/search and replace, auto line numbering, move, copy, delete, uc/lc capability.
- Relocating Loader to relocate object modules.
- Designed with Human Factors Considerations.

BEFORE YOU BUY THAT OFF-BRAND ASSEMBLER, WRITE FOR OUR FREE DETAILED SPEC SHEET.

FLASH!! EHS Management has decided to allow $50.00 credit to ASM/TED owners who want to upgrade to MAE. To get this credit, return ASM/TED manual with order for MAE.

Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!

VISA® MasterCard

Olympic Sales Company

SERVING YOU SINCE 1947

CABLE:"OLYRAV" LSA
Main Showroom & Offices:
216 So. Oxford Avenue
(P. O. Box 74545)
Los Angeles, CA 90004
Telex: 67 34 77

WE HONOR VISA and MASTERCHARGE

ORDER DESKS open 6 days a week!
7:00 am to 7:00 pm Mon – Sat
Order Desks: (213) 739-1130

TOLL-FREE (outside Calif.) 800-421-8045
TOLL-FREE (within Calif.) 800-252-2153

Prices shown are for mail or phone orders; Walk-in slightly higher.

Goods subject to availability; OSC will meet or beat almost any advertised price as long as the competition has the goods on hand; not responsible for typographical errors; prices & specifications subject to change without notice; this ad supercedes all previous ads; min shpg & hndlg $5.95; Send $2 for 112-pg catalog

HEWLETT PACKARD

HP-87 32K Bytes standard $1885.00

82907A	32K Mem. mod.	239.00
82908A	64K Mem. mod.	337.00
82909A	128K Mem. mod.	537.00
HP-85A	Computer	$1985.00
HP-125	Computer	1985.00
HP-41C	Prog. scientific	188.00
HP-41CV	Prog. scientific	248.00
HP-41	Card reader	169.00
HP-41	Printer	297.00
HP-41	Optical wand	99.00
82106A	41C Memory module	26.00
82170A	41C Quad mem. mod.	84.00
82160A	HP-IL interface	109.00
82161A	Cassette drive	437.00
82162A	HP-IL printer	397.00
HP-97	Prog. scien/printer	584.00
HP-67	Hand-held fully prog	309.95
HP-32E	Scientific	58.95
HP-33C	Prog. scientific	93.95
HP-34C	Adv. prog. scientific	114.00
HP-37E	Business	77.95
HP-38C	Prog. financial	119.00
HP-11C	Prog. scientific	114.00
HP-12C	Prog. financial	127.00
82905A	Impact printer	619.00
82901M	5¼" dual drive	1699.00
9895A	8" dual drive	4499.00
2631B	Impact printer	2999.00
2601A	Letter qual. ptr.	2999.00
85-15001	Mass storage ROM	124.00
85-15002	Plot/print ROM	124.00
85-15003	I/O ROM	239.00
85-15004	Matrix ROM	124.00
85-15005	Adv. prog. ROM	124.00
85-15007	Assembler ROM	239.00
82903A	16K mem. module	169.00
82937A	HP-IB interface	328.00
82939A	Serial interface	338.00
82940A	GPIO interface	408.00
82941A	BCD interface	408.00
82949A	Parallel ptr. int.	239.00
HP-7470A	Hi-speed color grphc pltr w/intrfc	1269.00

New TI-99/4A · 48K RAM
100% TI parts w/Extended Basic or TI-LOGO *Re:$1300 Y/C:$*$689.95

Texas Instruments Home Computer

16K

TI-99/4A *Only* $299.95

	RETAIL	Your Cost
10" color monitor high res	374.95	339.95
32K memory module	399.95	314.95
Extended Basic	100.00	75.00
Speech synthesizer	149.95	129.95
Disk memory drive	499.95	394.95
Telephone coupler (modem)	224.95	189.95
Printer (solid state)	399.95	319.95
TI-LOGO	250.00	99.95

SANYO MONITORS

High resolution, number one seller!

		Retail	*Your Cost*
VM4509	9" B & W	190.00	159.95
VM4215	15" B & W (below our cost)	349.00	189.95
DM5109	9" Green *VM4215 limited offer*	200.00	169.95
DM8012	12" B & W	250.00	199.95
DM8112	12" Green	260.00	209.95
DMC6013	13" Color, hi quality	470.00	399.95
DMC6113	13" Color RGB hi res	995.00	799.95
DM2012	(NEW) 12" B & W	179.00	139.95
DM2112	(NEW) 12" Green	199.00	159.95

Free w/purchase of each Sanyo monitor one $49.00 LCD pen watch
(subject to avail. on new models & others)

commodore VIC-20

5K Personal Computer
(Expands to 32K) Works with any TV!
Retail:$300.00 Your SPECIAL Cost:$259.95
FREE with purch of VIC-20 one $49.50 LCD Pen Watch!
Commodore Datasette Recorder for VIC-20 Your Cost:$68.95 We carry all peripherals, software & access. for the VIC-20 (we will beat any price!)

ATARI® IN STOCK COMPUTERS

FOR ATARI 400 & 800–
GHOST HUNTER (similar to Pac-Man)
Disk $34.95 Cassette $29.95

Model		Retail	Your Cost
800	16K	1080.00	689.95
800	48K	1280.00	789.95
800	160K		
(800-32K + 128 RAM disk)			1299.95
810	Disk drive	600.00	469.95
Atari	VisiCalc	200.00	169.95
825	Impact Ptr	1000.00	779.95

ATARI 400™

Personal Computer System

SUPER SPECIAL–ATARI 400 (16K)
Retail $595.00 Your Cost $299.95
400 Lang. card opt. $49.95

ATARI® *Retail* $225.00

Programmable Color TV Game
Your Cost: $139.95

Atari's *game cartridge* **Pac-man** $27.95

I D S Prism Printers–
*& Software for Apple Computers**
**call for our LOW, LOW prices!*

Immediate delivery–	*Retail*	*Your Cost*
132 column (color)	1995.00	1795.00
80 column (B & W)	1196.00	999.00

While supply lasts
CLOSE-OUT *ON PRINTERS*
PAPER TIGER by I D S

Model	*Retail*	*Our Cost*	*We Sell*
460 G	1094.00	733.00	699.95
445 G	894.00	599.00	579.95
445	795.00	556.00	536.00
CENTRONICS			
739-1	895.00	650.00	550.00
737	895.00	610.00	499.00

Epson PRINTERS

	Your Cost
MX70	$299.95
MX80 with graftrax	499.95
MX80 FT " "	599.00
MX100	749.00
Graphtrax chip	95.00

MATTEL ELECTRONICS® INTELLIVISION
$229.95
Retail: $325.00
animated TV game!

AMDEK Leedex High Quality Monitors

Model	Description	Retail	Your Cost
100	12" B/W, 12 MHz	180.00	149.00
300 G	12" Green, 18 MHz	269.00	199.00
Color I	13" Color, NTSC comp. input,audio amp & spkr	499.00	359.00
Color II	13" Color,RGB input,hi res graphics, spkr	1295.00	799.00
Color III	13" Color, RGB 240 line resolution	699.00	459.00

He may discover a magnificent ark, which acts in his favor if, and only if, he has certain magic in his possession.

If he uses his powers wisely, they will hasten his quest and safely return him to Camelot and the Crown.

King Arthur's Heir is available now on disk for the Atari 400/800 (32K) with joystick controller. The suggested retail price is $29.95.

EPYX
P.O. Box 4247
Mountain View, CA 94040

Microcomputer Index To Go Online With Dialog

Microcomputer Information Services, publishers of Microcomputer Index, has announced the availability of Microcomputer Index on the Dialog Information Retrieval Service. The index will begin online service with over 10,000 citations during the summer of 1982.

Microcomputer Index, which is also available in a printed version, began publication in 1980. The index is a subject and abstract guide to microcomputer articles from 32 periodical sources including: BYTE, InfoWorld, Interface Age, Creative Computing and **COMPUTE!**. The information covered includes microcomputer articles, software reviews, hardware reviews, book reviews, new product descriptions and more.

Each citation contains an abstract describing the article, complete bibliographic information and assigned descriptors. The printed version of Microcomputer Index, which is published quarterly, is $30 per year.

For additional information, contact:

Microcomputer Index
2464 El Camino Real, Suite 247
Santa Clara, CA 95051
(408)984-1097

or

Dialog Information Services
3460 Hillview Avenue
Palo Alto, CA 94304
(415)858-3735

Disk-O-Mate For CBM Disk Drives

Optimized Data Systems announces the PH-003 Disk-O-Mate for Commodore 2040/4040 disk drives. With Disk-O-Mate the user now has write protect switches/indicators for each drive, a power ON indicator and audible error beeper. The ease of write protecting drives by switch versus diskette labels encourages good operating procedure to prevent accidentally erased files. The power indicator and error beeper provide valuable disk unit status to the operator.

All switches and indicators are mounted in a small (2"x3"x1") control box that may be placed on the disk unit or near the computer. The control box connects by cable to a circuit board which "piggybacks" onto a 2040/4040 internal connector. Installation involves simple reconnection of internal disk unit cables, which can be accomplished without special tools or skills or any soldering. Full instructions are provided.

Disk-O-Mate is shipped fully assembled and tested. Price is in the $70-$80 range.

Optimized Data Systems
P.O. Box 595
Placentia, CA 92670
(714)966-3201

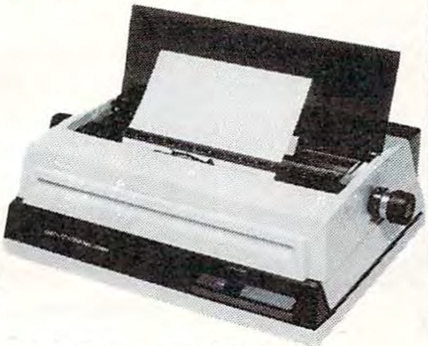
Educational Game Drill From School CourseWare Journal

Students of any age may now begin to find the typewriter and computer keys by touch and without looking, in a new educationally designed game drill by School CourseWare Journal (previously CorseWare Magazine).

"TYPING" presents on the screen the names of the keys for a single typewriter row. This presentation encourages learners to keep their eyes away from their fingers. As the user responds to letter or character cues correctly, the key names begin to disappear from the screen. If a

SMITH-CORONA TP-I

$699



Letter Quality
Daisy Wheel
12 CPS
Low Cost Text Printer

OTHER GREAT PRINTER VALUES

NEC 8023	$479
NEC 7710	$2299
Okidata 82	$479
Okidata 83	$719
Epson MX80	$429
Epson MX80FT	$519
Epson 100	659
Diablo 630	$1975
Citoh Prowriter	$479
CBM/Diablo Special	$1649
Trendcom 100	$269
Centronics 739-I	$479
Centronics 739-III	$579

computer mail order

east
800-233-8950
In Pa. Call (717) 327-9575
477 E. Third St.
Williamsport, PA 17701

west
800-648-3311
In Nev. Call (702) 588-5654
P.O. Box 6689
Stateline, NV 89449

ATARI®

VIC-20 $249

800	(16K)	649
400	(16 K)	319
410	Recorder	74
810	Disk Drive	439
822	Printer (THERMAL)	269
825	Printer (80 COLUMN)	589
830	Modem	149
850	Interface	164
853	Memory (16K)	79

ATARI SOFTWARE

4002	BASIC CARTRIDGE	45
4003	ASSEMBLER EDIT	45
4006	SUPER BREAKOUT	33
4007	MUSIC COMPOSER	45
4008	SPACE INVADERS	33
4009	COMPUTER CHESS	33
4011	STAR RAIDERS	37
4012	MISSILE COMMAND	33
4013	ASTEROIDS	33
483	THE PROGRAMMER	52
484	THE COMMUNICATOR	299
404	WORD PROCESSOR	110
405	PILOT (EDUC)	99
4106	PROGRAMMING 2	21
4117	PROGRAMMING 3	21
8126	MICROSOFT BASIC	66
8121	MACRO ASSEM&EDIT	66
813	CAVERNS OF MARS	29

CALL OR WRITE FOR FREE CATALOG

COSMIC COMPUTERS
UNLIMITED

228 N. PROSPECTORS RD.
DIAMOND BAR, CA.
91765

SHIPPING EXTRA $2 MINIMUM
PRICES SUBJECT TO CHANGE
CALIF. RESIDENTS ADD TAX

OPEN 7 DAYS
8am - 9pm

(714) 861-1265

Tara Computer Products

PUT AN END TO YOUR MEMORY EXPANSION PROBLEMS!

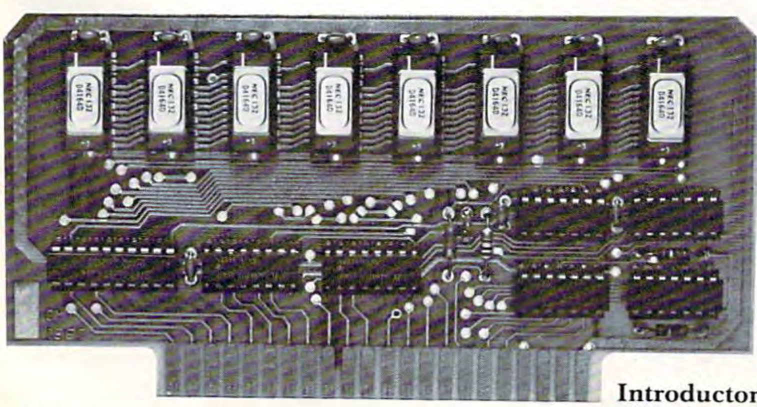
With the only logical choice for 48K memory expansion of the Atari 400.

Upgrade with the new Tara Computer 48K RAM card for the 400.

Features:

- Easy installation.
- Cooler, less power hungry operation compared to standard 16K or 32K products.
- Uses state-of-the-art 64K Dynamic RAMs.
- Extends 400 useable memory to 48K.
- Allows higher performance 800 software to be run on your 400.
- Quality construction with gold edge connector.
- Allows for disk operation.

Increase the performance of your personal system efficiently and economically with the new Tara Computer 48K RAM card. Available direct from Tara Computer or from select dealers.



Introductory Offer $199.00 US PLUS SHIPPING

Tara Computer Products
3648 Southwestern Blvd.
Orchard Park, NY 14127
(716)662-7219

Tara Computer Products
1655 Sismet Rd., Unit 30
Mississauga, Ontario
Canada L4W 1Z4
(416)624-5655

(Dealers' Inquiries Welcome).

cue is missed, the key name will reappear to help the user. The object of the drill is to make all the key names disappear from the screen. A summary of speed, accuracy and score is presented at the end of each game with the previous best speed, accuracy and score also displayed.

Students are encouraged to double the amount of their drill by the question DO YOU WISH TO TRY TO BEAT YOUR SCORE?

Volume 2, Number 3 of School CourseWare Journal, which includes the two programs "TYPING" and "SIGN DRILL," costs $15.95 for the cassette version of the two programs (and $6.50 for each program when included in a school year subscription). The programs are available on cassette or diskette for the Apple II 16K, TRS-80 16K, and PET 8K microcomputers.

For a free catalog:

School CourseWare Journal
4919 N. Millbrook, Suite 222R
Fresno, CA 93726

Data Perfect Atari From LJK

LJK Enterprises, Inc. announced the release of its newest program Data Perfect Atari, for the Atari 400 and 800 computer. Written in machine language, Data Perfect requires no disk swapping, and is fully interactive with LJK Word Processor, Letter Perfect. This user friendly program allows the operator to design his own screen mask. The single-load program, which is menu driven, has incorporated into it a utilities section, as well as a report generator and a mailing label generator. Multiple searches and sorts are allowed. Complete formula operations, as well as mathematical operations, may be performed on and between fields. The program supports one or two disk drives, and requires a minimum of 32K memory. Use with any parallel printer is allowed. The introductory cost for the program is $99.95.

LJK Enterprises, Inc.
P.O. Box 10827
St. Louis, MO 63129
(314)846-6124

Improved Version Of DOS For Atari 800

This command-driven K-DOS is completely compatible with the Atari 2.0S and other related software. In addition, K-DOS supports the Atari 850 handler which allows the use of printers and modems.

K-DOS features: a machine language monitor which allows examination and alteration of memory in hexadecimal and displays ATASCII representation; interception of the break instruction does not crash the system, but takes the user back into K-DOS; new, powerful commands reserve and erase memory and may be executed when the BASIC or Assembler cartridge is in control; K-DOS allows the user to create his own commands.

The 40-page, K-DOS instruction handbook, with disk attached, is easy-to-read and designed to acquaint programmers with all of the K-DOS features, and will give personal computer buffs greater reliability, flexibility and control. The handbook is divided into five individual sections for quick reference and also contains a pocket Command Summary Card.

For additional details and price information on K-DOS and the K-DOS Handbook, please contact:

John Mathies
K-BYTE
1705 Austin, P.O. Box 456
Troy, Michigan 48099
(313)524-9878

New VIC-20 Program By RAK Electronics

Budget II – a new home budget program that maintains a full calendar year's budget data on tape for review and update. The data requires input of each month's NET INCOME and 16 budget categories. After all of