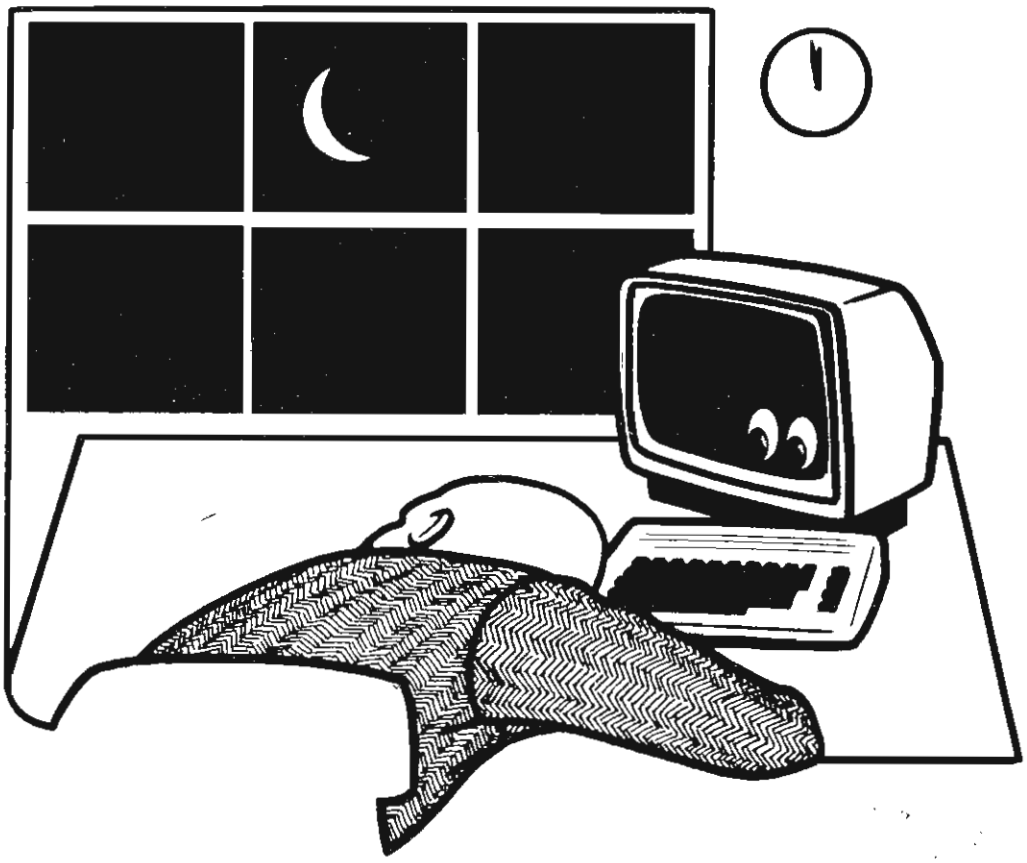
$2.00 US

ISSUE 28

OCTOBER 1985

Midnite Software Gazette

The First Independent U.S. Magazine for users of Commodore brand computers.



Published by: Micro-PACE Computers Inc.
Robert Wolter
Editor-in-Chief: Jim Oldfield Jr.
Editor: Tim Sickbert
Art Direction: Art Lewis Kimball
Assc. Editors: Jim & Ellen Strasma
Robert Baker
Dr. Richard Immers
Art Lewis Kimball
Subscriptions: 1-800-362-9653
One Year Price: $23 US $28 CDN
$33 S $43 AIR
Questions, back issues, Problems: 1-217-356-1885
BBS Number: 1-217-356-8056
(300 BAUD)
Address for Subs. And Reviews: Post Office Box 1747
Champaign, IL. 61820
Other Correspondence: 1510 N. Neil St.
Champaign, IL. 61820
Issue 27,1985 September

takes a floating point number in accumulator #1 and converts it to a 16 bit
positive integer. An error is printed if the number is not positive. The value
of the integer is returned in standard Lo Byte/Hi Byte format in three
locations:

Hex	Decimal	Label
$14/$15	20/21	LINNUM
$63/$62	99/98	FACHO(The first two bytes only of this location hold the signed result of a floating point to integer conversion)
.Y/.A	782/780	SYREG/SAREG

Since in the above application the numbers dealt with are always less than 40,
only the Lo Byte of the integer is of interest. All in all, kind of a neat
trick.
Happy Cursoring!!!!!!!!! Stephen R. Gast

HEXADECIMAL FILE DUMP UTILITY
by Robert W. Baker

Here's a handy utility program for looking at the contents of sequential data files on disk or tape, as well as program files on disk. It displays the hexadecimal value of each byte in the file so you can easily see the exact contents. Each display line also indicates the decimal offset from the start of the file so you have some idea where the data is located.

Although written for the C-64, if you make the indicated changes in line 180, the program will also work on the VIC-20. By changing the values of BC and CL, the display line will be changed to fit the VIC's 22 column dislplay. If you have a printer, you can get an optional printed copy of the data displayed.

Normally, each line displays 10 bytes and a 5 digit decimal offset. If you should try to dump a file with more than 99,999 bytes, only the five least significat digits of the decimal offset will be displayed. To fit the 22 column VIC-20 display, only five bytes per line can be displayed and the decimal offset is restricted to the three least significant digits.

When run, the program first asks if a printed copy is desired (11. 250-270). Answering 'Y' will cause any data displayed to be printed as well as displayed. Hitting RETURN alone enters a default response of 'N' so that data will be printed to the screen only. When using a non-Commodore printer you may have to modify the OPEN statement for the printer in line 390.

Next, the program askes where the file to be read resides: on tape or disk (lines 280-300). The appropriate tape or disk should be inserted in the drive before answering this question. When reading data from tape, the first file found on the tape will be used. This tape file must be a data file, not a program file.

If the file is on disk, the program asks for the name of the desired file (1. 320). The program assumes the selected disk file is a sequential data file and attempts to open the file (1. 330). If an error #64 (File Type Mismatch) is returned, the program will then assume the file is a program file

and attempt to open the file again (1. 340-80). Random access and user defined files cannot be read.

Once the proper file has been opened, the printer is opened if a printed copy was selected (1. 390). The file is then read byte by byte, with each byte being converted to hex and displayed (11. 420-470). At the end of each line the decimal offset from the start of the file is displayed at the beginning of the next line (11. 480-500).

After displaying each byte, a check is made for keyboard input to allow pausing the display or terminating the program before reaching the end of the file (11. 510-50). While data is being displayed, simply press any key except 'D' to pause the display. Once paused, press any key except 'D' to resume the display. Hitting 'D' at any time will terminate the program prior to reaching the end of the file.

When the end of the file is reached, the program will terminate automatically. If any disk or tape errors are encountered while reading the file, the program will terminate after indicating the error detected. Whenever the program terminates, all the files are closed properly.

```
120 rem                hex dump
140 rem         by: robert w. baker
180 bc=10:c1=5
182 rem change to bc=5:c1=3 for vic
190 h$="0123456789abcdef"
200 printchr$(147)"hex dump":gosub600
210 print"hit any key to "
212 print"hold/continue display"
220 print:print"hit 'd' when done -"
222 print"to stop before"
230 print"end of input file"
240 print:gosub600
250 input"want printed copy";c$
260 f=0:ifc$="y"thenf=1:goto280
270 ifc$<>"n"then630
280 input"file on disk or tape";d$
290 ifd$="t"thenopen1,1
292 f$="** tape file **":goto390
300 ifd$<>"d"then630
310 open15,8,15
320 input"filename";f$:iff$="."then630
330 open1,8,5,"0:"+f$+",s,r"
340 input#15,en,em$,et,es
342 ifen=0thenf$=f$+"(seq):goto390
350 ifen<>64then620
360 close1:open1,8,5,"0:"+f$+",p,r"
370 input#15,en,em$,et,es
375 ifen<>0then620
380 f$=f$+" (prg)"
390 if f then open 4,4
392 print#4,"hex dump of file: ";f$
400 printchr$(147)+"file: "f$
410 b=0:goto490
420 get#1,c$:ss=st:ifd$="d"thengosub610
430 ifss<>0then560
440 a=0:ifc$<>""thena=asc(c$)
450 n=int(a/16)
460 print mid$(h$,n+1,1);
462 mid$(h$,a-n*16+1,1);" ";
470 if f then print#4,mid$(h$,n+1,1);
472 mid$(h$,a-n*16+1,1);" ";
480 b=b+1:if int(b/bc)<>b/bc then 510
482 print
490 printri$("      "+str$(b),c1);": ";
500 if f then print#4
502 print#4,ri$("    "+str$(b),5);": ";
510 getc$:ifc$=""then420
520 ifc$="d" then 550
530 getc$:ifc$=""then530
540 ifc$<>"d" then 420
550 goto630
560 ifss<>64 then print "error..."
562 print"st = ";st:goto630
570 print:print:print" end of file"
580 iffthenprint#4:print#4
582 print#4,"end of file"
590 goto 630
600 print:print"--------------------"
602 print
610 input#15,en,em$,et,es
612 if en=0 then return
620 print:print"disk error..."
622 print en;em$,et;es
630 close1:close15:print
640 iffthenprint#4:close4
```

BASIC VARIABLE CROSS REFERENCE

This handy utility program produces a cross-reference list of every variable found in a BASIC program saved on disk. The program itself was designed to run on the C-128 in either 64 or 128 mode, as well as on the C-64 and older PET and CBM systems. It analyzes all programs written in BASIC 7.0 on the C-128, BASIC 2.0 on the C-64 or VIC-20, or BASIC 4.0 on the PET and CBM.

With the aid of a variable cross-reference list, you can easily control variable assignment and usage within programs being developed. It also makes debugging much easier, as every reference of a particular variable is clearly indicated. You can quickly spot variables reused within subroutines destroying previous values, and other common programming errors. It also makes it easier too, to investigate other people's programs.

The ouput generated by this program is normally printed but can be displayed on screen. Each variable is listed along with the line number of every line that references that variable. Long variable names are reduced to the standard two character name used internally by BASIC. Array variables are indicated by parathenses following the name. Individual array elements are ignored and references to particular elements cannot be indicated.

When you run this program, it first asks for the filename of the BASIC program stored on disk that is to be analyzed. Note that the program file is opened in read mode using the standard BASIC "OPEN" command in line 280. Newer BASIC 7.0 commands are purposly avoided to allow the program to run on a wide range of Commodore systems.

After opening the disk file, the program reads and discards the two-byte load address with the subroutine call at the end of line 290. Remember that the load address is returned as the first two bytes read as input from a program file when it is opened for reading.

The link and BASIC line number are then read, with the line number displayed and saved in LN$ for later reference (lines 300-320). The command at the end of line 310 converts the first character of the string formed in LN$ to a space instead of a cursor right. Whenever the STR$ function is used to convert a number to a string, the first character is always a cursor right for positive numbers.

Each BASIC program line is then scanned for variables while properly skipping data within quotes (lines 370-410), program data (lines 420-500), remarks (lines 510-530), and normal BASIC keywords. Special two-byte tokens used for keywords in BASIC 7.0 on the C-128 are skipped by lines 560-580. For general information, all two-byte tokens created by BASIC 7.0 start with a value of 206 or 254 ($CE or $FE hex). The byte immediately following these values indicates the exact token represented by the two byte code.

When a new symbol is found (lines 540-680) it's added to the current symbol table (SM$) in alphabetical order (lines 690-820). The line number where the variable is first referenced is saved in the LL$ matrix to start the cross reference listing. When a variable is found that already appears in the symbol table, the new line number reference is simply added to the end of the corresponding LL$ entry if that line number has not all ready been entered. Whenever any

entry in LL$ aproaches the maximum string length of 255 characters, another entry is made in both matrices for the same variable.
The current implementation of this utility will only list the individual lines that reference each variable. There is no indication as to how many times the variable may be referenced within each line, so be sure to look at the entire line in the analyzed program when using the cross reference list. If you really need to know multiple reference information, line 740 can be deleted. With his line omitted, every reference will be added to the LL$ entry. Therefore, the line number will be repeated three times if the variable is referenced three times on the same line. I would not use this mode too often, though, since it uses a great amount of memory for so little additional information.
As the program executes, the line number of the current line being analyzed is displayed so you can see how the program is progressing. Be patient, the program can take a while to analyze large programs or those that use a large number of variables. While on the subject, the program is currently limited to handling up to 500 variables as set by the dimensions of SM$ and LL$ in line 190. This seems to be a reasonable limit for most systems but you may run out of space if the program being analyzed contains an abnormally high variable usage. If you are running this program on a C-128 in 128 mode, you could safely raise this value.
Once the data is collected you're given the option of printing or displaying the formatted information. In either case, the first line of ouput indicates the filename of the program that was analyzed. The left column of subsequent lines indicated the variables contained in that program. The numbers following a specific variable name indicate every program line that references that variable. If enough references were found to fill more than one line, the variable name will only appear on the first line shown.
While the output is being displayed or printed, pressing any key on the keyboard will suspend the output. This is especially convenient when using the screen display. When ready to continue, simply press another key on the keyboard and the output will resume. If you press the 'Q' key when the output is suspended, you can terminate the program.
Screen displays are formatted for 40 column lines while printer ouput is formatted for 80 column lines. If you want to run this program on a C-128 with an 80 column display in 128 mode, then change the first value of RM from 25 to 65 in line 900. If your printer has more or less than 80 characters per line, then change the value of RM at the end of line 900 to 15 less than the maximum printer line length. If you need to do anything special for your printer, you can add lines before or after the OPEN in line 880.
One final note, if any errors are detected while reading the program file form disk, the error information returned from the disk will be displayed and the program will terminate with all files properly closed.

For those that don't like to type or would like a copy of the program right away, send $5 to cover costs and I'll send a copy of the program on disk. **Robert Baker.**

```
120 rem         basic program
130 rem  variable cross reference
140 rem
150 rem   by robert w. baker
180 :
190 dimsm$(500),11$(500):sm=0:sp$=chr$(160)
192 printchr$(147);
200 printspc(13);"basic program"
210 print"   variable cross references"
220 print"prints or displays a cross reference"
230 print"table of all variable used within any"
240 print"basic program saved on disk.";chr$(17)
250 print"----------------------------";chr$(17)
260 print"name of basic program on disk:"
270 printchr$(29);chr$(29);chr$(29);sp$;
272 printchr$(157);chr$(157);chr$(157);:inputf1$
275 if f1$=sp$thenend
280 close15:open15,8,15:open5,8,5,"0:"+f1$+",p,r"
282 gosub1170
290 printchr$(17)"ok, scanning program file"
292 printchr$(17)"at line:";:gosub1140
300 gosub 1140:ifv+v1=0then830
310 gosub 1140:ln=v1+(256*v):ln$=" "+mid$(str$(ln),2)
320 printtab(10);ln$;"          ":printchr$(145);
330 rem scan basic line for symbols
340 gosub 1150
350 ifv=0then300
360 ifv<>34then410
370 rem quote--skip chrs till next
372 rem quote or line end
380 gosub1150:ifv=34then340
390 ifv>0then380
400 goto300
410 ifv<>131then500
420 rem data token--skip charactors
422 rem till colon or line end
430 gosub1150:ifv=58then340
440 ifv=0then300
450 ifv<>34then430
460 rem if quote found--skip till
462 rem next quote or line end
470 gosub 1150:ifv=34then430
480 ifv>0then470
490 goto300
500 ifv<>143then550
510 rem rem token-skip chrs to line end
520 gosub1150:ifv>0then520
530 goto300
540 rem check for valid symbol
550 ifv<>206andv<>254then590
560 rem skip 2-byte tokens of basic 7.0
```

```
570 gosub1150:ifv=0then300
580 goto340
590 ifv<65orv>90then340
600 s$=c$:gosub1150
610 ifv<48orv>90then670
620 ifv>57andv<65then670
630 s$=s$+c$
640 gosub1150
650 ifv<48orv>90then670
660 ifv<58orv>64then640
670 ifv=36orv=37thengosub1130
680 ifv=40thens$=s$+"()":gosub1150
690 rem save in alpha order
692 rem with line ref
700 s$=s$+" "
710 z=sm:ifsm=0then810
720 forx=0tosm
730 ifs$<>sm$(x)then780
740 ifright$(l1$(x),len(ln$))=ln$then770
750 iflen(l1$(x))>246thensm$(x)=left$(sm$(x),len(sm$(x))-1)+chr$(1)
752 goto780
760 l1$(x)=l1$(x)+ln$
770 x=sm:nextx:goto350
780 ifs$>sm$(x)thennextx:goto810
790 z=x:fory=smtozstep-1
800 sm$(y+1)=sm$(y):l1$(y+1)=l1$(y):l1$(y)="":nexty
810 sm$(z)=s$:l1$(z)=l1$(z)+ln$
820 sm=sm+1:goto350
830 close5:close15
840 print:pd=3:print"done, want printed output (y/n): ";
850 getc$:ifc$="n"then880
860 ifc$<>"y"then850
870 pd=4:gosub1220
880 open4,pd
890 rem print symbol table in order
900 gosub1200:rm=25:ifpd=4thenrm=65
910 forx=0tosm
920 ifpd=3then940
930 ifpg=56thenfory=1to10:print#4:nexty:gosub1210
940 iflen(sm$(x))=0then1100
950 s$=left$(sm$(x),len(sm$(x))-1)
960 print#4,"   ";left$(s$+"      ",5);
970 b=0:fory=0toint(len(l1$(x))/rm)
980 a=b+1:b=a+rm:ifb>255then1010
990 c$=mid$(l1$(x),b,1):ifc$=""then1010
1000 c=asc(c$):ifc>47andc<58thenb=b+1:goto990:rem break line at space
1010 ln$=mid$(l1$(x),a,b-a):ifln$=""then1040
1020 ify>0thenprint#4,"          ";
1030 print#4,ln$:p=pg+1
1040 nexty
1050 getc$:ifc$=""then1100
1060 ifpd=4thengosub1240
```

GOTO 780

```
1070 getc$:ifc$=""then1070
1080 ifc$="q"then1110
1090 ifpd=4thengosub1220
1100 next x
1110 close4:end
1120 rem subroutines
1130 s$=s$+c$:goto1150
1140 gosub1150:v1=v
1150 get#5,c$:gosub1170:ifc$=""thenv=0:return
1160 v=asc(c$):return
1170 input#15,en,em$,et,es:ifen=0thenreturn
1180 print:printchr$(18)"disk error"chr$(146)enchr$(18)"   trk/sec:";
1185 printchr$(146)et"/"es:printem$
1190 close4:close5:close15:end
1200 ifpd=3thenprintchr$(147);
1210 print#4,"variables in: ";chr$(34);f1$chr$(34):print#4:pg=2:return
1212 printf1$chr$(34):print#4:pg=2:return
1220 printchr$(147)"printing cross reference table"chr$(17)chr$(17)
1230 print"press any key to suspend output":return
1240 printchr$(17)chr$(17)"output suspended"chr$(17)chr$(17)
1250 printchr$(18)"press any key to";
1260 print"continue, q to quit"chr$(146):return
```