TIPS FROM THE TIGERCUB

#51

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Distributed by Tigercub Software to TI-99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

Over 120 original programs in Basic and Extended Basic, available on cassette or disk, NOW REDUCED TO JUST $1.00 EACH!, plus $1.50 per order for cassette or disk and PP&M. Minimum order of $10.00. Cassette programs will not be available after my present stock of blanks is exhausted. The Handy Dandy series, and Color Programming Tutor, are no longer available on cassette.
Descriptive catalogs, while they last, $1.00 which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $5 postpaid. Each of these contains either 5 or 6 of my regular catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am NOT selling public domain programs - they are a free bonus!
TIGERCUB'S BEST, PROGRAMMING TUTOR, PROGRAMMER'S UTILITIES, BRAIN GAMES, BRAIN TEASERS, BRAIN BUSTERS!, MANEUVERING GAMES, ACTION GAMES, REFLEX AND CONCENTRATION, TWO-PLAYER GAMES, KID GAMES, MORE GAMES, WORD GAMES, ELEMENTARY MATH, MIDDLE/HIGH SCHOOL MATH, VOCABULARY AND READING, MUSICAL EDUCATION, KALEIDOSCOPES AND DISPLAYS

NUTS & BOLTS DISKS
These are full disks of 100 or more utility subprograms in MERGE format, which you can merge into your own programs and use, almost like having another hundred CALLs available in Extended Basic. Each is accompanied by printed documentation giving an example of the use of each.
NUTS & BOLTS (No. 1) has 100 subprograms, a tutorial on using them, and 5 pp. documentation. NUTS & BOLTS No. 2 has 108 subprograms, 10 pp. of documentation. NUTS & BOLTS #3 has 140 subprograms and 11 pp. of documentation. NOW JUST $15 EACH, POSTPAID.

TIPS FROM THE TIGERCUB
These are full disks which contain the programs and routines from the Tips from the Tigercub newsletters, in ready-to-run program format, plus text files of tips and instructions.
TIPS (Vol. 1) contains 50 original programs and files from Tips newsletters No. 1 through No. 14. TIPS VOL. 2 contains over 60 programs and files from Nos. 15 thru 24. TIPS VOL. 3 has another 62 from Nos. 25 through 32. TIPS VOL. 4 has 48 more from issues No. 33 through 41.
NOW JUST $10 EACH, POSTPAID.

```
##############################
#         NOW READY          #
# TIPS FROM TIGERCUB VOL.5 #
# Another 49 programs and  #
# files from issues No. 42 #
# through 50. Also $10 ppd #
##############################
```

TIGERCUB CARE DISKS #1,#2,#3 and #4. Full disks of text files (printer required).
No. 1 contains the Tips newsletters #42 thru #45, etc. Nos. 2 and 3 have articles mostly on Extended Basic programming. No. 4 contains Tips newsletters Nos. 46-52. These were prepared for user group newsletter editors but are available to anyone else for $5 each postpaid.

I believe this word game is totally different from anything you have ever seen, and very challenging if you don't use the AID key. The first time you run it, pick option 3 to create a file of phrases and give it the file name COMPUTE. This will then become the computer's file, option 1, and you can create as many of your own files as you want. Recommend phrases of several to as many as 20 words - short ones are too difficult.

```
100 DIM W$(20):: DIM D$(20)
110 GOTO 150
120 Q$,K,S,Q,F$,E,FLAG,X,J,X
$,Y$,A,B,M$,DY$,V,A$(),C$,CH
,CH$,Y,W$(),L,M,D$(),F,Z,C,R
,H
130 CALL CHAR :: CALL KEY ::
 CALL SOUND :: CALL CLEAR ::
 CALL CHARPAT :: CALL COLOR
:: CALL SCREEN :: CALL VCHAR
 :: CALL SPRITE :: CALL LOCA
TE :: CALL DELSPRITE
140 !@P-
150 CALL CHAR(94,"3C4299A1A1
99423C"):: DISPLAY AT(2,1)ER
ASE ALL:"TIGERCUB SHUTTLESEA
RCH V.1.1":"":"^ Tigercub So
ftware for free":"distributi
on but no price"
160 DISPLAY AT(6,1):"or copy
ing fee to be charged":"":"I
f you should feel moved to":
"send me a few bucks for my"
:"work, I won't be offended!
"
170 DISPLAY AT(12,1):"Jim Pe
terson":"156 Collingwood Ave
.":"Columbus, OH 43213"
180 DISPLAY AT(16,5):"Instru
ctions? (Y/N) N" :: ACCEPT A
T(16,25)SIZE(-1)VALIDATE("YN
"):Q$ :: IF Q$="N" THEN 260
190 DISPLAY AT(2,1)ERASE ALL
:" The computer will display
 a":"phrase or saying concea
led":"within a grid of rando
m":"letters."
200 DISPLAY AT(6,1):" The wo
rds will be horizon-":"tal,
one word per line and":"on c
onsecutive lines, but":"not
necessarily beginning on"
210 DISPLAY AT(10,1):"the to
p line, and the phrase":"may
 'wrap around' from the":"bo
ttom row to the top."
220 DISPLAY AT(13,1):" You c
an find the phrase by":"shut
tling columns of letters":"u
p and down, looking for":"co
nsecutive rows with letter"
230 DISPLAY AT(17,1):"combin
ations that could be":"parts
 of words.":" A cheat key is
 available,":"if you are rea
lly stuck, but"
240 DISPLAY AT(21,1):"try no
t to use it!"
250 DISPLAY AT(23,8):"PRESS
ANY KEY" :: DISPLAY AT(23,8)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 250
260 DISPLAY AT(3,2)ERASE ALL
:"Do you want to - 1":"":" (
1) Solve a saving from my
   file?":"":" (2) Solve a p
hrase from        your file
?"
270 DISPLAY AT(11,2):"(3) Cr
eate a file of":"     phrase
s?":"":" (4) Have someone ty
pe in a      phrase to solve
?"
280 ACCEPT AT(3,19)SIZE(-1)V
ALIDATE(DIGIT):Q :: IF Q<1 O
R Q>4 THEN 280
290 ON Q GOTO 300,310,410,47
0
300 F$="1.COMPUTE" :: E=1 ::
 GOTO 320
310 DISPLAY AT(18,1):"Filena
me? DSK" :: ACCEPT AT(18,14)
:F$ :: E=2
320 ON ERROR 370
330 IF FLAG=1 THEN 350 :: FL
AG=1 :: OPEN #1:"DSK"&F$,FIX
ED,RELATIVE,INPUT :: ON ERRO
R STOP
340 INPUT #1,REC 0:X :: CLOS
E #1 :: FOR J=1 TO X :: X$=X
$&CHR$(J):: NEXT J :: Y$=X$
350 RANDOMIZE :: A=INT(RND*L
EN(Y$)+1):: B=ASC(SEG$(Y$,A,
1)):: Y$=SEG$(Y$,1,A-1)&SEG$
(Y$,A+1,255):: IF LEN(Y$)=0
THEN Y$=X$
```

```
360 OPEN #1:"DSK"&F$,FIXED,R
ELATIVE,INPUT :: ON ERROR ST
OP :: INPUT #1,REC B:M$ :: C
LOSE #1 :: GOTO 490
370 FOR J=1 TO 10 :: DISPLAY
 AT(20,1):"" :: DISPLAY AT(2
0,1):"CANNOT OPEN FILE!" ::
CALL SOUND(-99,110,5,-4,5)::
 NEXT J
380 ON ERROR 390 :: CLOSE #1
390 FLAG=0 :: INPUT "CHECK D
ISK AND DRIVE, PRESS ANY KEY
":DY$
400 IF E=1 THEN RETURN 260 E
LSE IF E=2 THEN RETURN 310 E
LSE RETURN 410
410 DISPLAY AT(8,1)ERASE ALL
:"Filename? DSK" :: ACCEPT A
T(8,14):F$
420 E=3 :: ON ERROR 370 :: O
PEN #1:"DSK"&F$,FIXED 124,RE
LATIVE,OUTPUT :: ON ERROR ST
OP :: X=0
430 DISPLAY AT(12,1):"Enter
END when finished":"":"":"Ty
pe phrases, not more than 20
 words and 124 characters"
440 X=X+1 :: ACCEPT M$ :: IF
 LEN(M$)>124 THEN PRINT "TOO
 LONG!" :: X=X-1 :: GOTO 440
450 IF M$<>"END" THEN PRINT
#1,REC X:M$ :: GOTO 440
460 PRINT #1,REC 0:X :: CLOS
E #1 :: GOTO 260
470 CALL KEY(3,K,S):: DISPLA
Y AT(12,1)ERASE ALL:"Type a
phrase of less than  20 word
s and press Enter"
480 ACCEPT M$ :: CALL CLEAR
490 DISPLAY AT(3,2)ERASE ALL
:"Choose skill level - 1":""
:" (1) All words begin in":"
     first column"
500 DISPLAY AT(8,2):"(2) All
 words begin in same":"
column":"":" (3) Each word m
ay appear in":"     a differ
ent column"
510 DISPLAY AT(14,2):"(4) As
 No. 3 but AID key is":"
 disabled":"":" (5) Quit"
520 ACCEPT AT(3,23)SIZE(-1)V
ALIDATE(DIGIT):V :: IF V<1 O
R V>5 THEN 520 :: IF V=5 THE
N CALL CLEAR :: STOP
530 DISPLAY AT(12,6)ERASE AL
L:"SCRAMBLING......"
540 A$(1)="jkzae klmpr vgaho
 nceci sdufy bqijw astrf urd
sa nvjxe blbig trakv nobth w
ehey vnijo oherq umbmi rtika
 opleg nosve tarkh zeski "
550 A$(2)="!boiu m.fgt krac,
 pjip? tn-un osheg kar,q ibl
.o tons! idrix ?uhig ebarf u
ks,k ,,jhge vifyt kibrn taga
, .!ry lakle ilf.! inst"
560 C$=A$(1)&A$(2)
570 FOR CH=65 TO 90 :: CALL
CHARPAT(CH,CH$):: CALL CHAR(
CH+32,CH$):: NEXT CH :: CALL
 CHAR(42,"8244428281010")
580 CALL CHAR(143,"18243C4A4
A3C2418"):: CALL COLOR(14,16
,1)
590 M$=M$&" " :: Y=1
600 X=POS(M$," ",1):: W$(Y)=
SEG$(M$,1,X):: L=LEN(W$(Y)):
: M=MAX(M,L):: RANDOMIZE ::
W$(Y)=W$(Y)&SEG$(C$,INT(230*
RND+1),20-L)
610 Y=Y+1 :: IF Y=21 THEN 62
0 :: M$=SEG$(M$,X+1,255):: I
F LEN(M$)>0 THEN 600
620 FOR J=Y TO 20 :: W$(J)=S
EG$(C$,INT(230*RND+1),20)::
NEXT J
630 ON V GOTO 670,640,650,65
0
640 X=INT(RND*(20-M))+M+1 ::
 FOR J=1 TO Y :: W$(J)=SEG$(
W$(J),X,255)&SEG$(W$(J),1,X-
1):: NEXT J :: GOTO 670
650 FOR J=1 TO Y :: X=INT(RN
D*(20-M))+M+1 :: W$(J)=SEG$(
W$(J),X,255)&SEG$(W$(J),1,X-
1):: NEXT J :: GOTO 670
660 ! the string
670 FOR J=1 TO 20 :: FOR L=1
 TO 20 :: D$(J)=D$(J)&SEG$(W
$(L),J,1):: NEXT L :: NEXT J
680 IF V=1 THEN F=M ELSE F=2
0
690 FOR J=1 TO F :: Z=INT(20
*RND+1):: D$(J)=SEG$(D$(J),Z
,255)&SEG$(D$(J),1,Z-1):: NE
XT J
700 CALL CLEAR :: CALL SCREE
N(5):: FOR S=1 TO 13 :: CALL
 COLOR(S,5,16):: NEXT S :: C
ALL VCHAR(1,31,1,96)
710 CALL VCHAR(4,5,143,20)::
 CALL VCHAR(4,28,143,20)
720 FOR C=1 TO 20 :: FOR R=1
 TO 20 :: CALL VCHAR(R+3,C+6
,ASC(SEG$(D$(C),R,1))):: NEX
T R :: NEXT C
730 DISPLAY AT(1,1):"s&d to
select, e&x to scrollfctn 7
aid, fctn 8 restart"
740 H=1 :: C=48 :: CALL SPRI
TE(#1,42,7,18,C)
750 CALL KEY(3,K,S):: IF S=0
 THEN 750 ELSE ON POS("EXSD"
&CHR$(1)&CHR$(6),CHR$(K),1)+
1 GOTO 750,800,810,820,830,7
60,840
760 IF V=4 THEN 750
770 FOR S=5 TO 8 :: CALL COL
OR(S,16,5):: NEXT S
780 CALL KEY(3,K,S):: IF S=-
1 THEN 780
790 FOR S=5 TO 8 :: CALL COL
OR(S,5,16):: NEXT S :: GOTO
750
800 D$(H)=SEG$(D$(H),2,19)&S
EG$(D$(H),1,1):: FOR R=1 TO
20 :: CALL VCHAR(R+3,H+6,ASC
(SEG$(D$(H),R,1))):: NEXT R
:: GOTO 750
810 D$(H)=SEG$(D$(H),20,1)&S
EG$(D$(H),1,19):: FOR R=1 TO
 20 :: CALL VCHAR(R+3,H+6,AS
C(SEG$(D$(H),R,1))):: NEXT R
 :: GOTO 750
820 C=C-8-(C=48)*8 :: H=C/8-
5 :: CALL LOCATE(#1,18,C)::
GOTO 750
830 C=C+8+(C=200)*8 :: H=C/8
-5 :: CALL LOCATE(#1,18,C)::
 GOTO 750
840 CALL CLEAR :: FOR J=1 TO
 20 :: D$(J)="" :: NEXT J ::
 M=0 :: CALL DELSPRITE(#1)::
 IF Q=1 OR Q=2 THEN 350 ELSE
 470
```

Here are three screen display subprograms of the type you will find on my Nuts and Bolts disks. Note that subprograms can read DATA from the main program. The double colons in the DATA statement cause input of null strings of data for spacing between the lines. The M$() in the subprogram parameter lists is necessary, even though the array is not passed from the main program, in order to DIMension the array in the subprogram - unless you prefer to place the DIM in the subprogram itself. T is the number of DATA items to be read.

```
100 CALL CLEAR
110 DATA THIS IS A DEMO,,OF
THREE SCREEN PRINTING,,SUBPR
OGRAMS PUBLISHED IN,,TIPS FR
OM THE TIGERCUB,,No. 51,,BY
TIGERCUB SOFTWARE
120 DIM M$(11):: CALL DOWNPR
INT(M$(),11):: FOR D=1 TO 10
00 :: NEXT D :: CALL CLEAR :
: RESTORE 110 :: CALL DIAGPR
INT(M$(),11)
130 FOR D=1 TO 1000 :: NEXT
D :: CALL CLEAR :: RESTORE 1
10 :: CALL INWARD(M$(),11)
1000 SUB DOWNPRINT(M$(),T)
1001 FOR J=1 TO T :: READ M$
(J):: L=INT(LEN(M$(J))+.5)::
 M$(J)=RPT$(" ",14-INT(L/2))
&M$(J):: M$(J)=M$(J)&RPT$("
",28-LEN(M$(J))):: NEXT J
1002 FOR J=1 TO 28 :: FOR L=
1 TO T
1003 DISPLAY AT(L,1):SEG$(M$
(L),1,J):: NEXT L
1004 NEXT J :: SUBEND
2000 SUB INWARD(M$(),T):: FO
R J=1 TO T :: READ M$(J):: N
EXT J :: R=1 :: FOR A=1 TO T
2001 L=INT(LEN(M$(A))):: F=1
3-L/2 :: G=L+F
2002 FOR J=1 TO INT(L/2+.5):
: DISPLAY AT(R,F+1):SEG$(M$(
A),J,1);:: DISPLAY AT(R,G):S
EG$(M$(A),L-J+1,1);:: F=F+1
:: G=G-1 :: NEXT J :: R=R+1
:: NEXT A :: SUBEND
3000 SUB DIAGPRINT(M$(),T)::
 FOR J=1 TO T :: READ M$(J):
: L=INT(LEN(M$(J))+.5):: M$(
J)=RPT$(" ",14-(L/2))&M$(J):
: M$(J)=M$(J)&RPT$(" ",28-LE
N(M$(J))):: NEXT J
3001 FOR J=1 TO 28+L :: FOR
L=1 TO T
3002 IF J<L THEN 3007
3003 DISPLAY AT(L,1):SEG$(M$
(L),1,J-L):: NEXT L
3004 NEXT J :: SUBEND
```

Just in case you didn't know - to jump directly to the first or last line in a TI-Writer file, use FCTN 9 and S(earch) and 1 for the first line or E for the last.

MEMORY ALMOST FULL...

Jim Peterson

FROM: SPIRIT OF 99

THE NONPROGRAMMER'S NOTEBOOK

by David Fink
Nutmeg TI-99ers

Last month we made a start on exploring utilities using CARFAX ABBEY as an example. My goal in doing this is not to derrogate what I believe to be a fine programming effort, but rather to suggest applications for some fine utility programs to the beginning user. Any suggested alterations in programs are meant for the end user of a program, and are not to be substituted for the original program in distribution.

In this month's session CARFAX ABBEY will once again be used, this time to highlight Jim Swedlow's XB-TOOLS. XB-TOOLS consists of 7 programs that allow one to compress program lines, move lines, cross reference variables, compress data statements, resequence blocks of lines within a program, change variable names through out a program, and one may print out all or part of this information. We will be using the compression program, which will save some disk space as well as improve running speed. We will also use the reference program to generate a variable list for the purpose of turning off pre-scan.

First, if you have never run XB-TOOLS before, you will probably need to configure it. This is a much simpler process than configuring FUNNELWEB so don't get scared off by it.

Place a copy of XB-TOOLS into drive one, and run the INSTALL program.

You will be presented with a menu asking for the printer codes needed to turn the listed printer features on and off. This is the trickiest part of the INSTALL program. If you don't know these codes and aren't about to look them up, just replace all the numbers with asterices and change the width to 80. Do this also if you don't have a printer, and for printer name type "DSK1.DATA". When you are done press the key that saves CONFIG to disk.

XB-TOOLS is designed to work with only one drive, so keep your working copy in drive one. There should be space on board to handle some fairly large programs. Now let's get started. Get a copy of CARFAX ABBEY into drive 1 and type OLD DSK1.CARF/LOAD (if you are using the modified disk from last month the file name is "DSK1.LOAD")

Place the TOOLS disk into the drive and type SAVE DSK1.COM,MERGE.

Now type RUN "DSK1.COMPRESS"

You should now see a menu before that will allow you to choose the operations that compress will perform. For our purpose it can be used as is. All you have to do is press "P" to proceed. The program will proceed to replace all variable names with one and two letter variables, all REM and ! statements will be deleted, lines will be combined, subprogram names will be shortened, all references to the number 1 will be changed to "@", and the statement "@=1" will be placed in the beginning of the altered program. These are all the default options from the title screen. If this program intrigues you experiment with different options using a short program. If you choose the print option COMPRESS will report on what it has done giving you a complete cross reference of old and new variables.

The program will be busy for a while have a cup of coffee, you might even have time to make a pot if you don't have one ready. If the wait bothers you buy "PRE-SCAN IT" from ASGARD. This is an assembly language program that performs similar operations to COMPRESS but does them much more quickly. It is also reputed to be easy to use. If you try it and like it write a review or reccomendation for your newsletter.

Once coffee break is done, run the load program. this will catalogue the disk in drive one for you. Press 5 ,and when a file name is requested type in "COM". This will delete the file, but before you do note the difference in length between "COM" and "PRESS". When all the operations dscribed here are done, there will be about a 12%

reduction in program length. *I reccomend testing any compressed program as they don't* always work properly after compression. In this case you type MERGE DSK1.PRESS and then RUN. However, Ican assure you that this particular program will work just fine.

Our next step in this process is to run the REFERENCE program. This program will generate a suggested pre-scan list as well as a thorough cross reference of a program. We will use this variable list and reference to actually do a vary slight and painless bit of programming.

As with COMPRESS, the first thing to appear on the screen will be an option list. This time however, some changes need to be made. 1) Type in a program name and press enter

2) Press D, and F. This will reduce the size of the print out by eliminating line references and line lengths.

3) Press G and type "PRESS" (the file name of our altered program.)

4) Press P to start the program.

Cofee break again, but this time you won't have time to make it since this operation is a bit quicker than compressing. When Tex signals that its done, take your hard copy and look for the suggested pre-scan variable list. It should be listed right after the individual variable listings. (Those of you without printers *will have* to make hard copy by viewing the file DATA and copying it by hand. Use any file viewer e.g. the Type command in DM1000 or the screen print out in LOAD of XBTOOLS.) Once you have a copy in your hand, type NEW or restart Tex from the title screen. This will clear the memory for the next operation.

Make sure the disk cotaining the file PRESS is in drive one and type MERGE DSK1.PRESS to load our program into memory. Once this is done it is time to program.

As you may already know the TI does a house keeping task called pre-scanning before a program is run. This allows the allocation of memory areas for each variable, sub program, and data statement. To do this it goes through the program line by line looking for any of these items. This accounts for the time lag ~~that is often~~ present between typing RUN and the actual start of a program. The to beat this is to insert a list of all variables and subprograms into the beginning of the program and the leaving the command !@P- immediately after the list. This causes the pre-scan to stop at that point and the program runs almost as soon as your fingers leave the keys from the RUN command. The actual difference in time is actually only a few seconds, but if your dealing with a program that loads in segments, all those seconds do add up. Also time spent staring at an unchanging screen seems to grow logarithmically until it seems like forever.

REFERENCE will have generated a list of variables wiht the most frequently used variables listed last. The reason for this is that when variables are looked up by the computer, the last ones named are looked up first, thus these variables are found the fastest. The program actually runs faster when it does not have to seek so far down the list each time a variable is used. Here are additional line numbers that work on my copy fo the program:

```
10 CALL INIT :: GOTO 110 ::
A,B,C,D,E,A$,B$,C$,D$,E$,F$ :: CALL CHAR :: CALL CLEAR :: CALL SCREEN :: CALL
COLOR :: CALL KEY

20 CALL HCHAR :: CALL LOAD

30 !@P-
```

This is the variable list and sub program list, try this one if you like. If you miss a variable or subprogram, the computer will issue a syntax error statement when you run the program. Simply list the line that generated the error and compare it to your list. There will be a variable or subprogram in that line that is not in the *pre-scan list, simply add it in, and you will have solved the problem. Try the program* again and repeat the operation above until you no longer get a syntax error.

With any luck the only error you'll see is DATA ERROR IN 110. The reason for this is that the first data statement in the program must be scanned before the program runs. However the statement in line 30 turns off the pre-scan before any DATA statement is made. There is an easy fix for this. Look up data on your print out from REFERENCE. The first DATA statement should be line 320. Use the statement !@P+ to turn the scan on in a line before 320, then turn it off again in a line after 320 with !@P-. This is what it looks like in my program:

```
310 !@P+
320 DATA 128,139,128,209,134,139,144,149,159,169,179,189,199,213,213,
213,213,213,213,213,128,139

330 !@P-
```

I reccomend using a separate line for the pre-scan statement as it may not work properly if it is tacked on the end of an existing line.

Test the program again. If it seems to be working properly type

```
SAVE DSK1.PRESS,MERGE
```

At READY type

```
NEW
```

Then

```
MERGE DSK1.PRESS
```

When the drive cycles down, remove the disk and place your altered CARFAX disk from last month into the drive and type

```
SAVE DSK1.LOAD
```

If your wondering about the reason for the merge operations, it is a simple speed up operation. The basic interpreter runs a program line by line following the order of the line numbers, however the program is not necessarily stored in memory in that order. If the lines are not in order the interpreter seeks through memory line by line until the appropriate linenumber is found. Saving a file in merge format causes it to be placed in numeric order. When it is merged back in to memory it will be set to run as rapidly as possible.

The operation described here is complicated but not difficult. The only programming knowledge required is rudimentary line editing. Try this out, and if you have luck with it, try doing the CARFAX program on your own. Let me know how it works out.

Dave

FAIRE SCHEDULE

TI-FEST WEST	FEB 18-19	CLARION HOTEL at BALBOA PARK SAN DIEGO, CA.
T.I.C.O.F.F.	MARCH 18	ROSELLE PARK HIGH SCHOOL ROSELLE PARK, NEW JERSEY
BOSTON FAYUH	APRIL 1	RAMADA INN WOBURN, MA.
LIMA UG CONFERENCE	MAY 20	LIMA CAMPUS/OHIO STATE UNIV. LIMA, OHIO

I saw this little tip published in a couple of newsletters for the Horizon. Well, it also works on the Grand Ram.

A 32K TESTER

If you are having problems with your computer and have it narrowed down to the 32k card in the P-Box, here's a little Basic program that you can use to test the memory chips. I use it to find exactly which memory chip is bad (assuming this is the problem). This program was developed by Joe Nuvolini at Front Range and John Willforth at the Pittsburgh User Group.

You will need a Mini Memory to run it since you don't want to use the 32k because it's supposed to be defective, remember?

John Willforth says it checks the memory chips for the most common defects and should pinpoint the defective chip. He also says that if the program shows all chips in a row to be defective except for one chip, then suspect that chip to be the one that is defective.

The program that I typed in follows, if you have a question, the original articles are in our library.

```
100 ! 32k tester by Nuvolini/Willforth
110 N=0
120 CALL CLEAR
130 CALL SCREEN(13)
140 PRINT " MEMORY EXPANSION CHECKER
145 PRINT
150 PRINT " PLACE YOUR MEMORY EXPANSION

T THE TOP"
180 PRINT :" ENTER :":"       1 TO CHECK

      3 TO END"
190 CALL KEY(0,K,S)
200 IF S=0 THEN 190
210 IF K<49 THEN 190
220 IF K>51 THEN 190
230 R=K-48
240 IF R=1 THEN 700
250 IF R=2 THEN 720
260 IF R=3 THEN 680
270 IF R=1 THEN 300
280 N=27
290 GOTO 310
300 N=35
310 V=N
320 CALL CLEAR
330 IF R=1 THEN 350
340 GOTO 370
350 PRINT "TEST OF TOP ROW OF 4116'S"
360 GOTO 380
370 PRINT :"TEST OF BOTTOM ROW OF 4116'S
380 PRINT "READING FROM RIGHT TO LEFT.."
390 FOR T=1 TO 2
400 FOR I=0 TO 7
410 IF T=1 THEN 440
420 IN=2 I
430 GOTO 450
440 IN=0
450 CALL LOAD(A,IN)
460 CALL PEEK(A,D)
470 IF IN=D THEN 510
480 PRINT "    CHIP U ";STR$(N);" IS BAD
490 CALL SCREEN(10)
500 GOTO 520
510 PRINT "    CHIP U ";STR$(N);" IS OK"
520 N=N-1
530 PRINT "WRITTEN =";IN;" READ =";D
540 NEXT I
550 GOSUB 740
560 CALL CLEAR
570 IF T=1 THEN 610
580 PRINT " END OF SECOND PASS"::::::::
590 GOSUB 740
600 GOTO 640
610 PRINT " END OF FIRST PASS"::::::::
620 GOSUB 740
630 N=V
640 NEXT T
650 PRINT
660 INPUT "PRESS ENTER TO CONTINUE ":X$
670 GOTO 110
680 CALL CLEAR
690 END
700 A=-12288
710 GOTO 270
720 A=12287
730 GOTO 270
740 FOR DELAY=1 TO 600
750 NEXT DELAY
760 RETURN
```

Exploring BASIC Programs
by Tim MacEachern

The program listed below demonstrates how BASIC programs are stored in the 99/4A. The program as listed will work in Extended BASIC with the Memory Expansion card or peripheral attached. A similar program can be run in normal BASIC with the Editor/Assembler or Mini Memory module inserted. To convert this program to normal BASIC simply change the calls to subroutine 'PEEK' in lines 200, 240 and 260 into calls to subroutine 'PEEKV'. That is, add a 'V' between the 'PEEK' and the '(' in each line. This program will not work properly in Extended BASIC unless you have the memory expansion.

The techniques used in this program are intended to make it as easy to understand as possible, while still showing how the DEF statement in BASIC can be used to do all the hard work for you. For instance, lines 100 to 130 of th program create a function HEX which will convert a string of hexadecimal (base 16) digits into a decimal number. As can be seen in lines 150 and 170, this allows us to write the actual hexadecimal addresses as used by assembler language programmers.

Line 130 takes the string of hexadecimal digits given to it and pads it with leading zeroes to make sure that there are four hex digits. Then function HEX4 is called to evaluate this four-digit hex number. In line 120, HEX4 splits the number into two two-digit hex numbers and combines ~~them to get~~ the proper decimal result. Similarly, line 110 splits a two-digit hex number into two one-digit numbers. Line 100 then is used to figure out the value of each separate hexadecimal digit.

Using nested DEF statements as in this program can simplify development of a working program, but be warned that DEF statements take considerably longer to run than the exact same code put directly into your lines wherever needed. Still, you may find it convenient to write some programs that consist solely o DEF statements! After such a program is RUN in normal BASIC (or in Extended BASIC without the memory expansion), the defined functions will be available t use in BASIC's calculator mode. For instance, if your program consisted of lines 100 to 130 only, it would provide a conversion function from hex to decimal that you could use while in calculator or direct command mode.

Let's get back to the program. Line 140 defines a function that is used to convert a 16-bit unsigned number (from 0 to 65535) into a 16-bit signed number (from -32768 to 32767). For some strange reason BASIC insists on signed number for addresses passed to PEEK, PEEKV, LOAD and POKEV. So whenever an unsigned address is calculated function MA is used to convert it to a signed number. This function works by comparing its argument to the largest positive value allowed. If the number is too big the comparison yields a value of -1. The res of the expression then causes 65536 to be subtracted from the argument value, giving the correct result. If the original number is okay (from 0 to 32767) th comparison yields a result of 0 and the value of the function is the same as the value of its parameter. It seems complicated to write functions like this, but try to figure them out - you may find them fascinating.

BASIC stores your program in two sections. In the top of memory it stores each line of the program, not necessarily in the correct order. As a matter of fact, each time you edit a line, it becomes the last line in this area, with all other lines packed together above it. Each statement is made up of three parts. The first byte is the length of the rest of the statement in memory. The last byte is zero, and in between are bytes that represent the particular BASIC statement you have written. BASIC keywords are translated into a single byte each (known as a token) while strings and numeric constants are represented as a leading token (199 or 200) followed by a length byte, followed by the ASCII character values of the string. By running this program you can determine how other elements of a BASIC program are stored.

Underneath the statements (that is, lower in memory) is a list of statement numbers and pointers to the first token in each statement. Each statement in your program has a four-byte entry in this list. The bottom two bytes store the statement number. The top two bytes are a pointer to the first token in the statement (the byte following the length byte). This program goes through this list and prints out each token in the statements of your program.

Pointers to the top byte in the statement pointer list and the bottom byte in the list are stored in the scratchpad RAM and read by lines 150 to 180. The loop that starts in line 190 examines each statement in the program. If you have gotten this far in the article, you will understand how the rest of the lines in the program print out each token of each line.

This is one of a series of articles I will be writing for 99/4A newsletters. I hope you found it interesting. Permission to copy this article for publication is given on two conditions: first, that these conditions on publication be published with the article and second, that copies of the next three issues of the publishing newsletter be sent to me at the following address:

Tim MacEachern
P.O. Box 1105
Dartmouth, N.S.
Canada B2Y-4B8

```
100 DEF HEX1(X$)=POS("123456789ABCDEF",X$,1)
110 DEF HEX2(X$)=HEX1(SEG$(X$,1,1))+HEX1(SEG$(X$,2,1))
120 DEF HEX4(X$)=HEX2(SEG$(X$,1,2))6+HEX2(SEG$(X$,3,2))
130 DEF HEX(X$)=HEX4(SEG$("0000"X$,LEN(X$)+1,4))
140 DEF MA(X)=X+65536*(X>32767)
150 CALL PEEK(MA(HEX("8332")),A,B)
160 TOSL=MA(A6+B)
170 CALL PEEK(MA(HEX("8330")),A,B)
180 BOSL=MA(A6+B)
190 FOR PTR=TOSL-3 TO BOSL STEP -4
200 CALL PEEK(PTR,A,B,C,D)
210 PRINT "STATEMENT #";A6+B
220 PRINT "TOKENS:"
230 SPTR=MA(C6+D)
240 CALL PEEK(SPTR-1,L)
250 FOR I=0 TO L-1
260 CALL PEEK(SPTR+I,X)
270 PRINT X;
280 NEXT I
290 PRINT : :
300 NEXT PTR
```

FEBRUARY 14, 1989 HAPPY VALLENTINE'S DAY

MUNCH OFFICERS AND NUMBERS (all in 508 area unless noted)

President/Mail	W.C. Wyman	839-4134
Vice^President	Bruce Willard	852/3250
Secretary	Jim Cox	
Treasurer	Jim Cox	869-2704
Acting Editor	Jim Cox	
Adv.Prog. Chair	Dan Rogers	248-5502
Library	Al/Lisa Cecchini	
Disk Librarian	Lou Holmes 617	322/1562
Tape Librarian	Walter Nowak 413	436/7675
+++++++	Jack Sughrue	476/7630

FEBRUARY MEETING. This meeting will be a very important one to all of our members. The TI New England Fayah is scheduled for Saturday April 1, 1989 at the Ramada Inn in Woburn, off Route 128. We have to decide what fund raising activities we will do for the Fayah and who will be doing what at our booth. These decisions have to be made now as the Fayah is only seven weeks away.

JANUARY MEETING. President Corson Wyman called a very cold meeting to order with 16 members present. Jack Sughrue demonstraated Tigercub Jim Peterson's Nuts and Bolts Disks for the membership. Jim haad supplied Jack wjith a special demo disk that was very interesting. It was really something to see the demo run on a speeded up Geneve. Unfortunately speed does not always generate heat! Dan Ross conducted the Assembley S.I.G. under very difficult conditions. Our members should be proud that almost everyone stayed for the entire evening.

RAFFLE. Each month we have a raffle and the dollar donation per ticket helps to cover the monthly fee to rent the hall. This month's raffle will have a TI programming book, two educational game carts and at least one other item.

MONTHLY SALES. At each meeting you have the opportunity to buy and/or sell new or used hardware, software, books and original programs. Please have prices marked on any items you have to sell.

LIBRARY NOTICE. Please return any items borrowed from our library. If you can not come to a meeting or give these items to someone who will be at the meting, please mail any library items to the group address which is listed on the cover of this newsletter. There are no late fees, we don't care how long they have been out, please return these items.

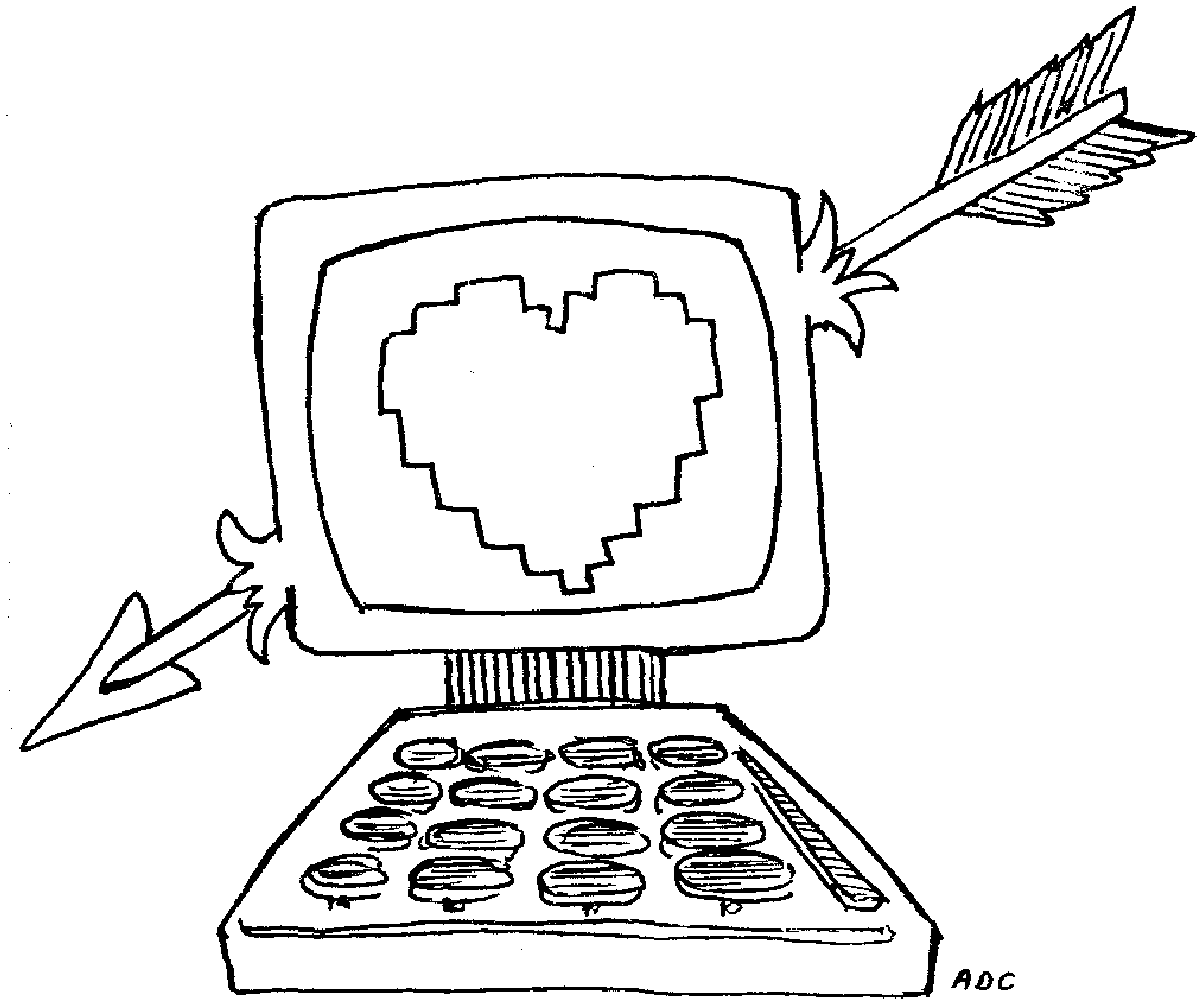
REPRINTS. Reprints of any items in this newsletter is permitted as long as credit is given to M.U.N.C.H.

ARTICLES. I am always looking for articles for this newsletter, anything which interests you will probably interest other members of the TI community, so please share your ideas and opinions with all of us.

WELCOME NEW MEMBER: Allan Kresock of Johnson City, New York.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    *   *  #    #  +    +   %%%   $   $
    ** **  #    #  ++   +  %   %  $   $
    * * *  #    #  + +  +  %      $   $
    *   *  #    #  +  + +  %      $$$$$
    *   *  #    #  +   ++  %   %  $   $
    *   *   ###    +    +   %%%   $   $
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
----------------------------------------------------
Mass Users of the Ninety-nine and Computer Hobbyists
----------------------------------------------------
FEBRUARY 1989 Monthly Newsletter Version 8.02
```

HAPPY VALENTINE'S DAY



JOIN THE CROWD AT OUR FEB. 14 MEETING

U.N.C.H.
O. Box 7193
INCOLN ET
ESTER, MA. 01605
xt Meeting: FEB. 14

FIRST CLASS

MASTER: Forwarding and Address Correction Requested