99 HOCUS

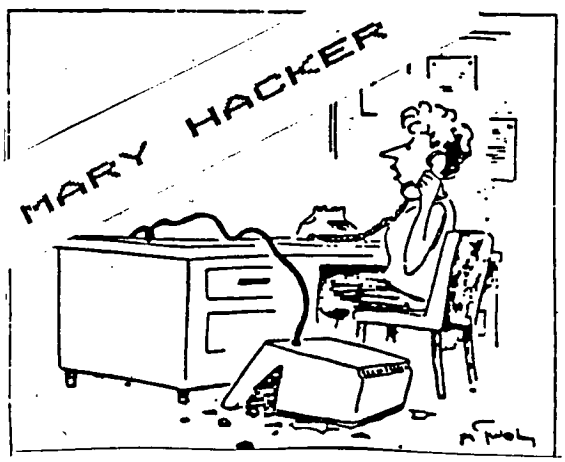
Home Computer
Users Spotlight
a monthly publication of the
Milwaukee Area 99/4 Users Group

M A Y - 1 9 8 6

MILWAUKEE AREA USER GROUP
4122 GLENWAY WAUWATOSA WI 53222

President...	J.Schroeder	2644735
Vice-Pres...	D.Walden	5292173
Treasurer...	P.Norton	4628954
Secretary...	G.Kasica	3217558
Librarian...	E.VonDerEhe	5490593
Librarian...	F.Pabian	3273618
Newsletter..	G.Hitz	5350133
SIG.....	Schroeder/Walden/Hitz	

Group Meeting
Second Saturday
Wauwatosa S&L 7500 W.State
1:00-4:00
SIG Meeting
First Monday
National S&L 3670 S.Moorland
7:00-10:00
Membership Dues
Individual - $10
Family - $15



I'm sorry, but our computer is down.

MULTI-PROGRAMMED BASIC

Colorado Springs Newsletter 'FRONT RANGER'

In this article we will talk about the TI-BASIC Operating System's use of the Free Memory pointer, how it decides where to load a program and how it knows if a program is in memory. After we learn these few tidbits, we'll learn how to trick the TI99 into holding onto three programs at once. We'll also get the computer to run each of the programs on our demand & save all three to one single disk file ready to run again!

So, where do we start? If we were a computer, we'd start at the first free space in memory! And that's where we will begin our study of the TI. In TI-BASIC the First Free address in VDP RAM is 14295. The space from 14296-16383 is taken up by the disk operating system. This FreeMemory marker is stored in the CPU RAM pad at address -31952 & at -31950. Why in both places? It has to do with how the TI can tell if a program is already in memory. As you may know, TI stores the entire line number set in one place. It then keeps track of the start of the Line Number Table (-31952) and the end (-31950). Whenever the user types LIST, RUN or SAVE, the operating system checks those two addresses. If they are equal, the Line Number Table is empty, there is no program present. If necessary the TI will honk at you and display the appropriate error message (CAN'T DO THAT).

Conversely, when the user types NEW, the operating system simply, and only, zeroes out the Line Number Table pointer. It checks for the highest available memory, usually 14295 & places that value at the beginning marker (-31952) and ending marker (-31950). The system does NOT erase your program from memory!

As an illustration of how the Line Number Table pointers work, let's type in the following lines in TI-BASIC (you will need the ED/ASM Module in place for this.):

First, in command mode type:

```
NEW
CALL INIT
CALL PEEK(-31952,A,B,C,D)
PRINT A;B;C;D
```

You should get the values 55, 215, 55, 215

This shows the Line Number Table is empty. If you type in LIST, TI will say 'CAN'T DO THAT' with a honk.

Now enter the following short program:

```
100 CALL CLEAR
110 CALL SOUND(150,1400,0)
120 PRINT "I'M HERE!"
130 PRINT
140 GOTO 110
```

Now let's check out the Line Number Table markers again. In command mode type

```
CALL PEEK(-31952,A,B,C,D)
PRINT A;B;C;D
```

You should get the values 55, 138, 55, 157

Now the computer knows that there is a program present and that its Line Number Table starts at 14218 and runs to 14237. If you now type RUN the program will beep and announce itself. Now type NEW. If you try to RUN or LIST your program you are told CAN'T DO THAT and chided with TI's infamous honk tone.

Now type the following:

```
CALL LOAD(-31952,55,138,55,157)
```

and then type LIST.

See...now TI thinks there's a program again!

Now put it anywhere. There's nothing sacred about putting the program HighUp in memory. In fact you can put it anywhere you wish as long as you tell the TI where to find it. Let's try another short example:

```
NEW
CALL LOAD(-31952,27,215,27,215)
100 CALL CLEAR
110 PRINT "I'M SHORT"
120 STOP
```

Now try SAVEing this short program. What happens? The TI SAVE command saves program in "memory image" format. In other words, it simply saves a copy of the program, byte for byte, from the VDP to the disk. Also, the TI SAVE command always starts at the top of available memory (14295) no matter where the line number table starts! This means that in our example above, the TI SAVEd every byte from 16383 to around 7100! Quite a few bytes for such a small program! This may seem an annoying flaw, but it is in fact, a great opportunity for some TI-hacking!

Since the TI doesn't care where you start the line number table, and since the system never erases memory, just changes the table markers, and since the TI can only keep its binary 'eyes' on one program at a time, we can use a few tricks to stick more than one program into VDP RAM at the same time. In fact, this is what we shall now do:

1) Enter two independent programs into the VDP RAM area in different locations.
2) Enter a third 'Master' program in another location.
3) Use the 'Master' program to move to and from each of the other two programs on demand.
4) Save the entire mess to disk under one file name.

NOTE: Be sure to enter all programs and commands EXACTLY as they are here. This is a case where every byte counts!

1) COLD start the computer (turn power off/on)
2) Select BASIC with ED/ASM module
3) Type NEW and CALL INIT
4) Enter the following program

```
100 REM #PROGRAM ONE#
110 REM
120 CALL CLEAR
130 CALL SCREEN(15)
140 CALL SOUND(150,1400,0)
150 PRINT TAB(9);"PROGRAM ONE"
160 FOR L=1 TO 10
170 PRINT
180 NEXT L
190 CALL LOAD(-31952,44,55,44,130)
200 STOP
```

5) Now let's find the start and end of the line number table for this program. In command mode type:

```
CALL PEEK(-31952,A,B,C,D)
PRINT A;B;C;D
```

You should get these values: 55 9 55 52

If you didn't, check your listing carefully, there's probably a typo, an extra character (or space), etc. Just go back and edit the offending line, you don't have to re-type the entire program.

6) Now let's trick the TI into thinking this program does not exist and also set up a new area in VDP memory for our second program. In command mode type:

```
CALL LOAD(-31952,50,215,50,215)
```

Just for kicks try LISTing your program. See...

Be sure you type these values and not some other ones! If you're not sure, use CALL PEEK at the same address to check those four important values. If all is OK, we are ready to type in our second program.

7) Now entire the following program, start with line 100 !

```
100 REM #PROGRAM TWO#
110 REM
120 CALL CLEAR
130 CALL SOUND(150,700,0)
140 PRINT TAB(9);"PROGRAM TWO"
150 FOR L=1 TO 10
160 PRINT
170 NEXT L
180 CALL LOAD(-31952,44,55,44,130)
190 STOP
```

8) After you have entered program #2, check for the current location of the line number table:

```
CALL PEEK(-31952,A,B,C,D)
PRINT A;B;C;D
```

If there are no errors you should get 50 31 50 70

9) We are now ready to write the 'MASTER' program that will allow us to move from one program to the other but first let's set up a new line number table area:

```
CALL LOAD(-31952,45,215,45,215)
```

10) Now enter this short program:

```
100 REM #MASTER PROGRAM#
110 REM
120 CALL CLEAR
130 CALL SOUND(150,110,0)
140 CALL SOUND 150,440,0)
150 CALL SOUND 150,220,0)
160 CALL SOUND(150,110,0)
170 REM
180 PRINT "MASTER PROGRAM"
190 PRINT "==============="
200 PRINT
210 PRINT
220 PRINT
230 INPUT "WHICH PROGRAM (1 OR 2)
    ? ";P
240 IF (P<>1)&(P<>2) THEN 230
250 IF P=2 THEN 270
260 CALL LOAD(-31952,55,9,55,52)
270 CALL LOAD(-31952,50,31,50,70)
280 STOP
```

11) Check the location of the line number table:

```
CALL PEEK(-31952,A,B,C,D)
PRINT A;B;C;D
```

You should see 44 55 44 130

12) To save all this together as one file just type:

```
SAVE DSK1.MASTER (or whatever)
```

All three programs will then be saved to disk and will be available when next you load this file.

Now let's see if your programs run. Make sure your line number table is set to the MASTER program

```
CALL LOAD(-31952,44,55,44,130)
```

and now RUN.

You will be asked to select program one or two. Select program one. When the program signals it is #DONE#, type LIST. You now should be looking at a listing of program one, not the MASTER! Now type RUN again. After program one is finished, immediately type RUN. Yes, you just ran the MASTER program again!

If you are getting screen lockup, wierd error messages, like SYNTAX ERROR IN LINE 0, etc then there's a mistake in a LOAD. The computer will blindly accept any values placed at address -31952 as valid line number table markers. When you type LIST, your TI will do its best to list whatever that table points to.

LOCKUPS REDUCED

If you seem to find that you need to take out and put in your Extended Basic module many times to get it to work right, it may be you have worn off the silver on your GROM extension assembly in your computer. If your computer has been running well for you and this is your only problem, then a change of your GROM ext. assy is probably in order. You will probably find it cheaper to simply order another one and replace it in your computer.

REPAIR PARTS DEPT. % TEXAS INSTRUMENTS INC. P.O. BOX 53
LUBBOCK TX. 79408 PART # 1049583-0001
COST $5.84 PLUS HANDLING ($1.30)

...did you know???

I'm sure everyone at one time has started typing in....
OLD DSK1. and suddenly noticed on the screen old dsk1.
Oh fudge, I'm in lower case lettering, the computer won't accept it, I'll just have to erase it and start again...

But you don't really have to, your computer will accept it! Just be sure that the entire old dsk1. is lower case and the filename is upper case and your'e okay. Don't switch to upper case in midstream, old dSK1. that tends to confuse him.

*** I thought everyone knew that ***

Everyone knows that if your extended basic program is named LOAD it will automatically load and run when you turn on the extended basic module, but surprizingly enough, everyone does not seem to know that Option 5 Program Image assembly programs will also automatically load and run by simply pressing ENTER if they are named UTIL1.

TI Writer Alternate Character Set

by Erwin von der Ehe

This article explains how to make your own character fonts for display on TI Writer. My interest developed because my daisy-wheel printer produces non-standard characters in place of some lesser used keyboard characters, while TI Writer displayed the character on the keyboard. At printing time I usually had surprises to edit out of my text. The procedure presented here allows redefinition of any character normally displayed by TI Writer. This technique can be used for the problem described above or for properly displaying other characters printed through the Transliterate command of the Formatter.

Requirements:

1) TI-99/4A, Disk System, Memory Expansion, TI Writer.
2)Editor/Assembler
3)DISK0 disk editor, on MATUIG Disk #32
4) Updated TI-Writer Files, MATIUG Disk #71
5) Grid paper to help define characters - reference CHAR subprogram in your BASIC Reference Manual or a program such as Sprite Maker, MATIUG Disk #94

Procedure:

1) Make a back-up copy of your TI-Writer program diskette. Use this back-up for all further activities in this article. This is IMPORTANT!
2) Determine which character (ASCII 0 through 127) you want to redefine
3) Determine the character definition code for the new character. Note: in 40-column display mode, characters are 8 pixels high by 6 pixels wide. Use the left 6 columns of your 8x8 pixel character definition grid for each character, but define the character code based on the full 8x8 grid.
4) Boot DISK0, using Editor/Assembler option #3, Load and Run. File name = DSK1.DISK0. Program name = START.
5) Before we go too much farther, here's how to navigate in DISK0:

FCTN	Description
1	Display sector in HEX
2	Display sector in ASCII
3	Return to E/A from menu
4	Display previous sector
6	Display following sector
8	Rewrite disk sector to match screen
9	Return to DISK0 menu
=	Quit - Go to Title Screen

6) After DISK0 is running, put your back-up TI-Writer program diskette into DSK1. Use Option #2, Search for Existing File, to find the start sector of the file CHARA1, the character definition file. Note the starting sector number. Press <FCTN>9 for menu.
7) Select Option #1, Disk Sector Editor and select the starting sector of your CHARA1 file.
8) You will see the display labelled File Sector 1, shown at the top of this page. Note in the screens included with this article, I've inserted dots (•) between the character definition codes for your convenience in locating the character code you want to redefine. Also, I've added the ASCII number next to selected character codes. If you have the 9-sector version of CHARA1, don't worry that only 5 sectors are shown here: the others only contain 0's and serve no useful purpose that I can see.
9) Use the navigation functions to display the sector you wish to change. Use the arrow keys to put the cursor under the starting character of the code you want to redefine. Be sure to view sectors in HEX mode.
10) Type in the new code. Now is the time to review this screen to see if this is really what you want. If it isn't, type over incorrect code until it's right or return to the menu. If the code is correct, use <FCTN>8 to indicate your intention. Answer Y to rewrite the sector.
11) Boot up TI-Writer using the disk with your revised file and check your results.

Notes:

1) This procedure also works on the CHAR1 file of QS-WRITER®.
2) This procedure also works on the CHARA1 file of FUNLWRITER, MATIUUG Disk #112.
3) TI-Writer can only call CHARA1 character file, and it must be on DSK1. You may wish to keep several TI-Writer Program Diskettes, each with a different customized CHARA1.
4) Disk Fixer® or another disk sector editor can be used in stead of DISK0.

EJv
21 Apr 86

File Sector1: ASCII #:

```
0000080007FA•0020000018242418• 0
•0020000818080 81C•002000182408
103C•0020001824082418•00200014
141C0404•0020001C10180418•0020
000810382418•0020001C04081010• 7
•0020001824182418•00200018241C
0408•202038001C101C10•00400020
20382438•0070507048541C14•0070
4070001C1010•0020001824 3C2018• 14
•0040081410 1C1010•004040401824
2418•0020202028080808•00404058
2408103C•0040405824082418•0040
4054141C0404•0040405C10180418• 21
•0040404810382418•0040405C0408
1010•0040405824182418•00404058
241C0408•0040404018243C24•0040
4050101C141C•004040401C10101C• 28
•00404444041C141C•007070707070
7070•0040
```

File Sector2:

```
4C50101C1010•0000000000000000• 32
•0010101010001000•002828280000
0000•00287C28287C2800•00385430
18543800•00444C1830644400•0020
502054483400•0008102000000000• 39
•0008101010100800•0020101010 10
2000•0044287C28440000•0010107C
10100000•0000000000301020•0000
007C00000000•0000000000303000• 46
•0004081020400000•003C4C546444
3800•0010301010103800•00384408
10207C00•0038441804443800•0008
1828487C0800•0078407804443800• 53
•0038407844443800•007C04081020
2000•0038443844443800•00384444
3C047800•0000303000303000•0000
303000301020•0000102040201000• 60
•0000007C007C0000•000010080408
1000•0038
```

File Sector3:

```
440810001000•0038445458403C00• 64
•00384444 7C444400•007844784444
7800•0038444040443800•00784444
44447800•007C407840407C00•007C
407840404000•0038444044C443800• 71
•0044447C44444400•0038101010 10
3800•0004040404443800•00444850
70484400•0040404040407C00•0044
6C5444444400•0044645454 4C4400• 78
•007C444444447C00•007844447840
4000•0038444454 4C3C00•00784444
78484400•0038443008443800•007C
101010101000•0044444444443800• 85
•0044444444281000•004444445454
2800•0044281010284400•00444428
10101000•007C081020407C00•0038
202020203800•0000402010080400• 92
•0038080808083800•001028440000
0000•0000
```

File Sector4:

```
000000007C00•0020100800000000• 96
•0000003848483C00•002020382424
3800•0000001C20201C00•0040 41C
24241C00•0000001C28301C00•000C
103810101000•0000001C241C0438• 103
•0020203824242400•001000301010
3800•0008000808084830•00202024
38282400•0030101010103800•0000
007854545400•0000003824242400• 110
•0000001824241800•000000382438
2020•0000001C241C0404•00000028
34202000•0000001C300C3800•0010
103810100C00•0000002424241C00• 117
•0000004428281000•000000445454
2800•0000002418182400•00000024
241C0438•0000003C08103C00•000C
10102010100C•0010101000101010• 124
•0060101008101060•000020540800
0000•0000 127
```

File Sector5:(Partial)

```
000000000000•0000000000000000
00000000000000000000000000000000
```

etc.

By john willforth

This is one of my early attempts to take a program written in Microsoft Basic and convert it to TI Basic. The trick is really not a trick of course, but the effect appears as if the computer can really read your mind. If the program is a little too long for you, just start at line 400, skipping the title and instructions. Good luck and have fun. OH BY THE WAY, there is a bug that I should warn you about, and that is occasionally the computer will randomly select TWO JACK OF DIAMONDS, in that event dont select a JACK OF DIAMONDS, as neither you or the computer will know which is correct. Now I never went back to fix that bug. I beleive that I saved it for YOU! I will offer a reward for the one who gives me the first FIX.

```
100 REM  THIS PROGRAM WAS            ALTERED IN ORDER          TO RUN ON THE
         TI-99/4A COMPUTER
110 REM  BY JOHN WILLFORTH
120 REM  HOPE YOU ENJOY IT.
130 REM      6/02/83
140 CALL CLEAR
150 CALL SCREEN(2)
160 FOR Q=1 TO 10
170 PRINT TAB(11);"COLUMN"
180 NEXT Q
190 PRINT
200 PRINT TAB(5);"CREATIVE COMPUTING":
210 PRINT TAB(3);"MORRISTOWN, NEW JERSEY"
220 PRINT :TAB(5);"JOHN F. WILLFORTH"
230 FOR Q=1 TO 8
240 PRINT TAB(11);"COLUMN"
250 NEXT Q
260 CALL SCREEN(12)
270 FOR DELAY=1 TO 1000
280 NEXT DELAY
290 FOR H=1 TO 30
300 PRINT TAB(11);"COLUMN"
310 NEXT H
320 CALL SCREEN(2)
330 CALL CLEAR
340 PRINT "THIS PROGRAM WILL SHOW YOU ACARD TRICK. AFTER THE FIRST DEAL, PICK A
CARD AND TYPE  THE NUMBER OF THE THE COLUMNCONTAINING IT."
350 REM
360 PRINT "THE DEALER WILL THEN PICK UPTHE CARDS, A COLUMN AT A    TIME, AND WIL
L DEAL THEM OUTAGAIN HORIZONTALLY. WHEN HE"
370 REM
380 PRINT "FINISHES EACH TIME, TYPE THENUMBER OF THE NEW COLUMN    CONTAINING YO
UR CARD.FOLLOW-ING THE LAST DEAL THE DEALER"
390 PRINT "WILL TURN OVER THE CARDS ONEAT A TIME, UNTIL HE REACHES THE  ONE  CAR
D  THAT.....  Y O U  P I C K E D ! ! ! !"
400 CALL SCREEN(10)
410 FOR DELAY=1 TO 4500
420 NEXT DELAY
430 CALL COLOR(9,2,10)
440 CALL COLOR(10,7,10)
450 CALL CLEAR
460 RANDOMIZE
470 PRINT TAB(2);"ONE";TAB(10);"TWO";TAB(19);"THREE":::
480 DIM A(21),B(21)
490 FOR I=1 TO 21
500 J=0
510 T=INT(52*RND)+1
520 REM
530 FOR Y=1 TO I-1
540 IF A(Y)=T THEN 510
550 NEXT Y
560 A(I)=T
570 NEXT I
580 N=0
590 FOR I=1 TO 3
600 FOR Z=1 TO 21
610 IF A(Z)=4*(INT(A(Z)/4))THEN 630
620 IF A(Z)-2=4*(INT(A(Z)/4))THEN 600
630 IF A(Z)-3=4*(INT(A(Z)/4))THEN 670
640 CALL CHAR(100,"1818DBFF7E7E3C18")
650 P=100
660 GOTO 750
670 CALL CHAR(104,"66FFFF7E7E3C1818")
680 P=104
690 GOTO 750
700 CALL CHAR(101,"1818DBFFFF997E7E")
710 P=101
720 GOTO 750
730 CALL CHAR(105,"10387CFE7C3810000")
740 P=105
750 N=N+1
760 IF N<>4 THEN 790
770 PRINT
780 N=1
790 IF A(Z)>35 THEN 920
800 F=INT(A(Z)/4)+2
810 IF F=10 THEN 840
820 PRINT TAB((N-1)*9);F;" ";CHR$(P);
830 GOTO 850
840 PRINT TAB((N-1)*9);F;CHR$(P);
850 IF N=3 THEN 870
860 GOTO 890
870 PRINT
880 GOTO 890
890 IF J=5 THEN 1370
900 IF J=10 THEN 1510
910 GOTO 1100
920 IF INT(A(Z)/4)=9 THEN 1010
930 IF INT(A(Z)/4)=10 THEN 990
940 IF INT(A(Z)/4)=11 THEN 970
950 A$=" JK"
960 GOTO 1020
970 A$=" QK"
980 GOTO 1020
990 A$=" KG"
1000 GOTO 1020
1010 A$=" AC"
1020 PRINT TAB((N-1)*9);A$;" ";CHR$(P);
1030 CALL SCREEN(10)
1040 IF N=3 THEN 1060
1050 GOTO 1080
1060 PRINT
1070 GOTO 1080
1080 IF J=5 THEN 1370
1090 IF J=10 THEN 1510
1100 NEXT Z
1110 PRINT ::
1120 PRINT "WHICH COLUMN CONTAINS YOUR  CARD";
1130 INPUT K
1140 CALL CLEAR
1150 CALL SCREEN(10)
1160 IF K<1 THEN 1190
1170 IF K>3 THEN 1190
1180 GOTO 1210
1190 PRINT " (1-3)"
1200 GOTO 1120
1210 PRINT ::
1220 T=1
1230 S=K+2-3*INT((K+1)/3)
1240 GOSUB 1460
1250 S=K
1260 GOSUB 1460
1270 S=K+1-3*INT(K/3)
1280 GOSUB 1460
1290 FOR C=1 TO 21
1300 A(C)=B(C)
1310 NEXT C
1320 NEXT I
1330 J=5
1340 FOR Z=1 TO 11+INT(10*RND)+1
1350 N=0
1360 GOTO 610
1370 PRINT :
1380 NEXT Z
1390 PRINT
1400 PRINT "  OOPS !!! YOUR CARD IS THE
     ":
1410 PRINT
1420 N=1
1430 J=10
1440 Z=11
1450 GOTO 610
1460 FOR R=S TO S+18 STEP 3
1470 B(T)=A(R)
1480 T=T+1
1490 NEXT R
1500 RETURN
1510 PRINT
1520 PRINT :"DO YOU WANT TO SEE IT AGAIN? TYPE (YES) AND PRESS ENTER";
1530 INPUT T$
1540 IF T$="YES" THEN 450
1550 END
```

TIPS FROM THE TIGERCUB

#3#

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 13# original programs in Basic and Extended Basic, available on casette or disk, only $3.## each plus $1.5# per order for PPM. Entertainment, education, programmer's utilities.
Descriptive catalog $1.##, deductable from your first order.
Tips from The Tigercub, a full disk containing the complete contents of this newsletter Nos. 1 through 14, 5# original programs and files, just $15 postpaid.
Tips from the Tigercub Vol. 2, another diskfull, complete contents of Nos. 15 through 24, over 6# files and programs, also just $15 postpaid. Or, both for $27 postpaid.
Nuts & Bolts (No. 1), a full disk of 1## Extended Basic utility subprograms in merge format, ready to merge into your own programs. Plus the Tigercub Menuloader, a tutorial on using subprograms, and 5 pages of documentation with an example of the use of each subprogram. All for just $19.95 postpaid.
Nuts & Bolts No. 2, another full disk of 1## utility subprograms in merge format, all new and fully compatible with the last, and with 1# pages of documentation and examples. Also $19.95 postpaid, or both Nuts Bolts disks for $37 postpaid.
Tigercub Full Disk Collections, just $12 postpaid! Each of these contains either 5 or 6 of my regular $3 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 - my own programs on these disks are greatly discounted from their usual price, and the public domain is 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'S GAMES
MORE GAMES
WORD GAMES
ELEMENTARY MATH
MIDDLE/HIGH SCHOOL MATH
VOCABULARY AND READING
MUSICAL EDUCATION
KALEIDOSCOPES AND DISPLAYS
For descriptions of these send a dollar for my catalog!

I goofed again! if you tried the Quickloader in Tips #29 with a disk containing more than 2# programs, you may have already noticed that line 14# should go to 16#, not 155.

Here's another Tigercub Challenge - can you run this and get these results?

```
>LIST
1## PRINT PI
11# PRINT MAX
12# PRINT PI
13# PRINT MAX
>RUN
 #
 #
 3.141592654

* SYNTAX ERROR IN 13#
```

Some of you sharp-eyed newsletter editors may have noticed that this text is being hyphenated to avoid some of those gaping blanks that occur when only a few long words will fit on a right-justified line. The only way that I have found to accomplish this is to set the TI-Writer right tab for the actual column width to be printed and then, whenever a word is hyphenated, backspace and replace the blanks on that line with carets, adding enough extra carets to justify the line - like this -

whenever^a^word^^is^^hyphen-

It helps to go into fixed mode with CTRL # when you are inserting extra carets.
When using this method, it is also necessary to set the paragraph indentation with IN # on the command line; if indentations are desired, they can be filled with caret signs, like this:
^^When using this method,

I am told that my old 3D Sprite Routine made it to the Golden Quickies section of CompuServe, so here is an updated version. I have found that sprites can be controlled much more easily (although not moved as rapidly) with CALL LOCATE, rather than turning them loose with CALL MOTION and then trying to catch up with them!

```
1## CALL CLEAR :: CALL SCREEN(5):: FOR SET=2 TO 8 :: CALL COLOR(SET,8,5):: NEXT SET :: DISPLAY AT(3,12):"3-D SPRITE DEMO"
11# DISPLAY AT(22,1):"BY TIGERCUB" :: CALL CHAR(4#,"FF818181818181FF818181818181FFFF#1#1#1#1#1#1FF#1#1#1#1#1#1#1FF")
12# CALL CHAR(36,RPT$("F",64)):: CALL MAGNIFY(4):: FOR X=2 TO 22 STEP 2 :: CALL SPRITE(#X,36,X/2+1-(X>7)-(X>13),32+X*6,4#+X*6):: NEXT X
13# S=1 :: CALL SPRITE(#S,4#,16,46,7):: FOR C=6 TO 42 STEP 2 :: CALL LOCATE(#S,46,C) :: NEXT C :: FC=44 :: FR=46 :: Y=#
14# FOR C=FC TO FC+44 STEP 2 :: CALL LOCATE(#S,FR,C):: NEXT C :: FC=FC+44 :: CALL SPRITE(#S+2,4#,16,FR,FC):: CALL DELSPRITE(#S):: TC=FC-32
15# FOR C=FC TO TC STEP -2 :: CALL LOCATE(#S+2,FR,C):: NEXT C :: TR=FR+34 :: FOR R=FR TO TR STEP 2 :: CALL LOCATE(#S+2,R,TC):: NEXT R
16# CALL SPRITE(#S,4#,16,TR,TC):: CALL DELSPRITE(#S+2):: FR=TR :: TR=FR-72 :: FOR R=FR TO TR STEP -2 :: CALL LOCATE(#S,R,TC):: NEXT R
17# CALL SPRITE(#S+2,4#,16,TR,TC):: CALL DELSPRITE(#S):: FR=TR :: TR=FR+5# :: FOR R=FR TO TR STEP 2 :: CALL LOCATE(#S+2,R,TC):: NEXT R
18# Y=Y+1 :: IF Y=11 THEN CALL DELSPRITE(#S+2):: GOTO 13# ELSE S=S+2 :: FC=TC :: FR=TR :: GOTO 14#
```

Ian Swales in Belgium can write some of the most intricate routines, and pull them into the tightest knot. I had searched everywhere for a sorting routine for 2-dimensional arrays, and invented some ridiculous ones, before Ian sent me this jewel.

```
1## !DEMO of two-dimensional sorting routine
11# !Set up array to be sorted
12# CALL CLEAR :: DIM A$(2#,4):: RANDOMIZE :: DEF X$=CHR$(26*RND+65)
13# FOR J=1 TO 2# :: A$(J,1)=X$&X$&X$ :: A$(J,2)=STR$(INT(1##*RND+1)):: A$(J,3)=X$&STR$(INT(I#*RND))):: A$(J,4)=INT(1#*RND))&X$ :: NEXT J
14# INPUT "SORT BY?(1-4)":K
15# J=2# !2-dimensional array sorting routine by Ian Swales
```

```
160 DIM Q(200):: FOR X=1 TO 2
0 :: Q(X)=X :: NEXT X
170 M=0
180 FOR X=1 TO J-1 :: IF A$(
Q(X),K)<=A$(Q(X+1),K)THEN 21
0
190 M=-1
200 T=Q(X):: Q(X)=Q(X+1):: Q
(X+1)=T
210 NEXT X
220 IF M THEN 170
230 FOR X=1 TO 20 :: FOR L=1
TO 4 :: PRINT A$(Q(X),L);"
";:: NEXT L :: PRINT :: NEXT
X :: GOTO 140
```

Did you ever need a routine that would accept either a string or a numeric value? Try this -

```
100 N=0 :: ON ERROR 110 :: A
CCEPT M$ :: N=VAL(M$):: GOTO
 120
110 ON ERROR STOP :: RETURN
120
120 ON (N=0)+2 GOTO 130,140
130 PRINT M$ :: GOTO 100
140 PRINT N :: GOTO 100
```

A useful tip from Stephen Shaw in England - if you have a long program which wil run only in Basic, and which will load from disk with CALL FILES(1) but runs out of memory when you try to run it; and if you have the MiniMemory module -
Insert MiniMemory module, select Basic, enter CALL FILES(1), Enter NEW, enter OLD DSK1.(filename). When loaded, enter SAVE EXPMEM2. When SAVEd, enter CALL LOAD(-31888,63,255), enter NEW, enter OLD EXPMEM2, and enter RUN. That is still a lot faster than loading a long program from tape!

Another reason for never using the default mode of so-called UPDATE when opening a file (without specifying INPUT or OUTPUT) is that you will get an I/O ERROR 81 if the file is write-protected.

Has anyone found a way to go from Extended Basic to Basic without losing the program in memory, or at least fouling it up?
CALL LOAD(-32116,4) has been published in many newsletters as a way to do this, but has anyone actually made it work?

If you are printing out of TI-Writer Editor, finish your letter with CTRL U, SHIFT L, CTRL U and when it is printed the paper will automatically feed to the top of the next sheet.

To make a note to yourself while programming, just type 1! and whatever you want to make note of, then LIST "PIO":1, and then type 1 and enter to delete the line.

TI-Writer puts an extra space after every period that is followed by a space. If you don't want this extra space after abbreviations such as "Mr." or St.", use a caret sign ^ instead of a space after the period, Mr.^Jones. But TI-Writer puts only one space after ? or ! so if you want two, put a caret after the symbol !^

One of the very best tips for this month comes from Paul A. Meadows, in the September 85 newsletter of T.I.N.S. (Nova Scotia, Canada) -
How to print up to 132 characters in a line (condensed print, of course) out of TI-Writer! Just prepare your file as usual but in line 9991 put formatter commands such as .LM 10;RM 132; IN +5;FI;AD . The Fill and Adjust are necessary, the Indent is up to you, as are the left and right margins - but notice that right margin set way over at 132?
Now, instead of saving the file with SF, type PF and then C DSK1.(filename) to print to the disk. This not only strips out the control C characters, it also erases the TI-Writer tab line that was applied to the last line of the file.
So now, with your printer opened and initialized for condensed print, go into the TI-Writer formatter mode and print your file!

I have made the following changes to my working copy of the Tigercub Menuloader. This sets up my Gemini printer to skip over the perforations and print full page width in elite print with a wide left margin for ring-binder punching. Other printers may need changes in these codes.

```
620 DISPLAY AT(12,1)ERASE AL
L:"PRINTER? PIO" :: ACCEPT A
T(12,10)SIZE(-10):P$ :: GOSU
B 895 :: PP=3
840 DISPLAY AT(24,1):"PRINTE
R NAME? PIO" :: ACCEPT AT(24
,15)SIZE(-14):PP$ :: GOSUB 8
95 :: PRINT #2:SEG$(D$,1,4)&
" - Diskname= "&N$
895 OPEN #3:P$,VARIABLE 132
:: PRINT #3:CHR$(27);"B";CHR
$(2);CHR$(27);"M";CHR$(10);C
HR$(27);"N";CHR$(6):: RETURN
```

I always keep a backup of everything, on the flipped side of another disk, and I often want to verify that the backup has everything that is on the master, and vice versa.

```
100 DISPLAY AT(3,6)ERASE ALL
:"TIGERCUB DOUBLECAT": :" To
 compare the contents of": :
"a disk with a backup." !by
Jim Peterson
110 DISPLAY AT(12,1):"INSERT
 MASTER DISK": :"PRESS ENTER
"
120 CALL KEY(0,K,S):: IF S=0
 THEN 120
130 DATA DF,DV,IF,IV,P
140 RESTORE :: FOR I=1 TO 5
:: READ T$(I):: NEXT I
150 DIM F$(127):: OPEN #1:"D
SK1.",INPUT ,RELATIVE,INTERN
AL :: INPUT #1:A$,J,J,K :: F
$(0)=A$&" "&STR$(K)
160 X=X+1 :: INPUT #1:F$(X),
I,J,K :: IF F$(X)="" THEN 17
0 :: F$(X)=F$(X)&" "&T$(ABS(
I)):: GOTO 160
170 X=X-1 :: CLOSE #1 :: DIS
PLAY AT(12,1)ERASE ALL:"REMO
VE MASTER DISK": :"INSERT BA
CKUP DISK": :"PRESS ENTER"
180 CALL KEY(0,K,S):: IF S=0
 THEN 180
190 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:A$,J,J,K :: DISPLAY AT(1,1)
ERASE ALL:F$(0);:: DISPLAY A
T(1,15):A$&" "&STR$(K);
200 Y=Y+1 :: R=R+1 :: GOSUB
290 :: INPUT #1:A$,I,J,K ::
IF A$="" THEN 260 :: K$=A$&"
 "&T$(ABS(I))
210 IF K$=F$(Y)THEN DISPLAY
AT(R+1,1):F$(Y);:: DISPLAY A
T(R+1,15):K$;:: GOTO 250
220 IF K$<F$(Y)THEN DISPLAY
AT(R+1,15):K$;:: Y=Y-1 :: GO
TO 250
230 DISPLAY AT(R+1,1):F$(Y);
:: R=R+1 :: GOSUB 290 :: Y=Y
+1
240 IF K$=F$(Y)THEN 210 ELSE
 IF K$<F$(Y)THEN 220 ELSE IF
 Y<X THEN 230 ELSE DISPLAY A
T(R,15):K$;
250 GOTO 200
260 IF Y>X THEN 280
270 R=R+1 :: GOSUB 290 :: FO
R J=Y TO X :: DISPLAY AT(R,1
):F$(J):: R=R+1 :: GOSUB 290
 :: NEXT J
280 DISPLAY AT(24,1):"     P
RESS ANY KEY" :: CALL KEY(0,
K,S):: IF S=0 THEN 280 ELSE
CLOSE #1 :: END
290 IF R<23 THEN RETURN
300 DISPLAY AT(24,1):"PRESS
ANY KEY" :: DISPLAY AT(24,1)
:" " :: CALL KEY(0,K,S):: IF
 S=0 THEN 300
310 CALL CLEAR :: R=1 :: RET
URN
```

And that is just about

MEMORY FULL!

Jim Peterson

TI PUBLIC DOMAIN PROGRAMS - Summary 86/1

-This is an abreviated list of the software donated to the PUBLIC DOMAIN LIBRARY in the last 4-5 months.

A_APOLLO -LOGO-lunar lander-alesson in structured problem solving
A_AMCUP -LOGO- logo sailing game
ABACUS -XB- caslculator program
ARCHHEO -B/XB- archeodroid -direct the search of 3 robots
BIRD -XB- bird brained flying fish catcher
BUGOUT -XB- HCM program load checker
CANNIBAL -B-dr livingston vs the cannibals game
CELLMATE -XB- a living cell simulation
COMPOSE -XB- a sound recorder
CTRIX -XB- cardtrix filecard system
DEFENDER -B/XB- orbital defender game
DIVISION -XB- division tutor
ELECSEC -B-home secretary-personal record and address directory
FIDDLE -B- uncle larry's 10 fidddle tunes
FLACK -B-flack attack-ground to air missile game
FROGO -LOGO- a logical logo learning lesson
GAMMON -XB- backgammon game
KORSELF -XB- an arcade typing adventure
LOANCALC-XB- personal loan calculator (also in BASIC)
MELTDOWM -XB-debug a reactor and save the world
MINE -XB- mine over matter - mining simulation
MUSIC MAGIC -XB-program lets you play, display, and save music.
MUSICASMBL -B/MM-assembly language composition
NANOASSM -B/XB- an assembler for nano processor
NANOEDITOR -B/XB- an editor for nano processor
NANOPROCESSOR -a 4 bit computer simulation
ORGANIZER -XB+32k+disk- 4 progs to organize your work
PEG -B/XB- learn basic game programming
PLAINS -B/XB- plains of salsbury battle simulation
QUIZ CONSTRUCTION SET -B/XB- 7programs to quiz
QUIZ PRINT -B/XB- a print routine for quiz consrtuction set
RUNDAY -B/XB- run day view -daily appointment calendar
SEASTATE -XB-sea of states-state capitals and dive for treasure
SERF -B/XB- an updated version of hammurabi(rule a small country)
SLITHER B/XB- an egg eating snake
SNAPFIX -XB- (merge file) update/fix for snapcalc (+2 sample files)
SLOTS -B-a las vegas simulation
TABLUT -XB-a 14th century strategy game
TOWER -B-tower of hanoi-a manual routine you can play
TRIGTRIX -B/XB- trigonometry -sines/cosines
WORMWOOD -B/XB- character graphics
VITAL -B/XB- vital signs -health simulation

WINNIPEG
MANITOBA
1986

POSTAL COL
CODE POS

EDMONTON TIERS
PO BOX 11983
EDMONTON ALBERTA
T5J 3L1