Topics

LA 99ers COMPUTER GROUP Newsletter

VOL. 5 NO. 7 Los Angeles, CA JULY 1986

TERRIES'S CORNER

Teresa Masters President

SUMMERTIME, and the living is...
ON THE MOVE

George Steffen stomping around the Atlantic Seaboard, flew low getting there. Non-stop to Denver, cat-nap and Chicago the second day, yikes. Tom Freeman heading out on a triple- header, Expo, Hawaii, and Philadelphia. Me, well to avoid computer DT'S, I made a few calls, as a result there will be a Mini-Meet of the Minds. Howie Rosenberg , Chris Faherty, Paul Charlton, George Steffen, all at Barry Travers abode in Philadelphia. You can take the girl away from the computer, but you cannot take the computer away from the girl. I look forward to this meeting, such positive ideas germinate when people of this ilk simply get together to rap. Bonnie Snyder, President of Colorado Springs Front Ranger Group will also be in the New York area and we too plan to meet. So getting away is OK.

Australia, Melbournes Fest of June 14th was a resounding success. In a recent telephone call it was very obvious the high was very high. There was sincere appreciation of the U.S. companies who supported this Fest with both merchandise and information. Craig Miller, and Richard Mitchell in the forefront. The most impressive was Myarc, Lou Phillips sent a representative who flew 38 hours, then went directly to the fest. Our friends down under feel great with that kind of attention. They tell me many stateside products were sold in good numbers. What great news for a society that has fractured a bit lately. Now if the Dove with the Olive Branch could visit a few of the hard heads among us, perhaps the egos could be buried without loss of face. "WHAT FOOLS THESE MORTALS BE"

Helping hands across the water. Stephen Shaw, a long distance (England) contributing member sent along a disk full of articles, Rambles as he calls them, see inside for samples, and thank you very much Stephen.

RLE, we are receiving many files for this fine program originally uploaded by its Authors (there are 3) to Compuserve. There will be examples in each Newsletter and Fred is preparing a "Disk of the Month" which will be available through the Library. Each one will contain several pictures for you to print out. This may be the fund raiser we are looking for. We may yet be able to get a BBS online. I hope so.

Walt Howe, do you know what you are doing to me? A couple of weeks ago I received a copy of your "Solitaire" program. Well it took me 4 days to beat it, TWICE. That was several days and nights ago. I have said over and over again MANY times, "just one more game and I'll quit for the day/night. Several hours later I force my hand over to the Quit button, trembling all the way, and limp on to bed. My plants suffer, my Fish suffer, my Butt suffers, I suffer. Thanks alot Walt. I do not see your address in the program so I have only this avenue to reach you. How does anyone else who wants to follow me down this path of destruction get their soon to regret hands on your diabolical disk????

FAIRWARE George Huttons commentary on this fine value to us all has received several answers. Most agree something must be done to protect the programmer. Perhaps in the end it will be the responsible Librarian of a responsible User Group who collects an additional sum over the nominal media and copying costs, and forwards it to the programmer. There also were valid comments that too many are jumping on the "send me money" bandwagon with programs whose value is a bit questionable. Public Domain is becoming a thing of the past. Yes, I know this program is your baby, the reward of your hard labor, but be honest, does it have wrinkles. Don't overload a good system with early efforts.

MINI TI-WRITER MANUAL by Stephen Shaw

TI WRITER FOR NOVICES

Now that we have FUNLWRITER (in the Disk Library) [Ed. Note: and BA-Writer] there are a few more copies of TI Writer around than there are TI Writer manuals... and the manuals are not merely Copyright, but also costly to copy...

So, this article is for you if you have TI WRITER and no manual...

The menu choice is:
1. EDITOR
2. FORMATTER
3. UTILITY

Option 3 is a LOADER for MACHINE CODE PROGRAMS in memory image format - shown as PROGRAM on disk directories. The TI Writer loader creates a unique environment, and is intended to be used for utility programs specially written for the module (none were written) but in practice you may find you can load many machine code programs with this option : if the console locks up or does strange things, it is probably because the program requires a specific part of the Editor/Assembler environment which is missing.

Got that option out of the way quickly! The other two will take longer...

First, OPTION 1. This creates a full screen editor, on which you create your text. The screen "paper" is 80 columns wide, and is shown to you 40 columns at a time.
You do not have a full single character horizontal scroll - the screen is split into three columns of 40 characters: the leftmost screen display is the first 40 columns. Then if you move the cursor to the right, you will trigger a switch to display columns 21 to 60, and finally 41 to 80. [Ed. Note: If line numbers appear at the left - toggled with FCTN 0 - then the leftmost screen contains only 34 columns.

When you select EDITOR you will note the cursor appears at the top of the screen, on what is called the COMMAND LINE. The use of this line is described in the section below on TEXT EDITOR COMMANDS.

First, lets create some words! See at the top of the screen some words, with some letters in CAPITALS? For instance Edit... the capital E means that if you ENTER an E on this line you go into EDIT mode... so ENTER a letter E.

Did you hold shift down or have ALPHA LOCK on? No need to when in this area: even with ALPHA LOCK off, capital letters will be entered.

Entering E causes the COMMAND line to leave the screen and you are presented with the start of your paper, on which you can type your letter.

To return to COMMAND line, you press the keys FCTN and 9 ("BACK").

First though, lets look at all the instructions you can give to the computer while staying in the Edit mode....

SHIFT and ALPHA LOCK have their usual uses! And you have an auto-repeat on the keys. If you need to auto repeat a character using the SHIFT key, you can release the SHIFT key when auto repeat has started and just hold the main key down - SHIFT will be assumed to continue until you release the key.

ENTER will place an odd character on the screen, which looks like a small C over a small R - this is the Carriage Return symbol, and is NOT printed.
It is important when REFORMATTING - more later!

When you come to the end of a line and keep typing, WORD WRAP will move you to a new line automatically, and also ensure that you do not have a word cut in half in the process.

Unfortunately, word wrap takes a finite time, and many even moderate typists will find that it pays to check the first word at the start of the wrapped line for missing letters - our console lacks a keyboard buffer, and any keys pressed while word wrap is in progress are ignored.

To end a paragraph, press ENTER and you will move to a new line, and a CR will be inserted at the end of the previous text.

Before we move on... TI WRITER is key-compatible with Wordstar, should you use that program on another computer! However, in this article I shall not deal with the Wordstar keys, but rather with the more convenient use of the TI Function keys.

As space is limited, each Edit mode command can only be described briefly here, but the following should help you make progress:

The Arrow keys: FCTN E S D and X move the cursor one space in the appropriate direction.
CTRL L moves the cursor to the top Left of the screen, but keeps the screen display the same.
CTRL 6 moves the cursor to the first word in the paragraph it is in the middle of, AND moves it to the top left of the screen - therefore moving the text on screen, usually upwards! [Ed. Note: Actually it moves to the top of the PREVIOUS paragraph.]
PARAGRAPHS are collections of words between CR symbols. That is, each CR marks the end of a paragraph.
CTRL 8 is New Paragraph- it has the same effect as ENTER, it adds a CR to the end of the current line and moves the cursor to the next line. [Ed. Note: It adds the CR where the cursor is, NOT at the end of the line, and inserts a blank line below, then puts the cursor there.]
CTRL V moves the cursor to the start of its current line.
CTRL 9 is New Page- it inserts not only a CR but

also a PA, which is also not printed- the PA symbol will cause your printer to move to a New Page.

CTRL 4 is a tricky one- NEXT PARAGRAPH. When you type CTRL and 4, e text moves up off the screen and the cursor moves to top left. However, the line of text that your cursor was on does NOT have a CR added to it! FCTN 5 is Next Window and enables you to quickly flick through the three columns of page. It is cyclic - from far right you go back to far left.

FCTN 4 is Roll Down - the cursor moves down 24 lines (having the appearance of moving the text up 24 lines- the cursor keeps its position on screen!). If there are not 24 lines below the cursor, it moves to the end.

FCTN 6 is Roll Up and moves the cursor up 24 lines.

FCTN 7 is TAB (more later) and moves the cursor to the next tab setting on the right, while

CTRL T moves the cursor to the next tab position to the LEFT.

CTRL 7 is interesting - it is the Word Tab. If there is no text after the cursor, the cursor will move one space right, otherwise it will move to the start of the next word.

All those commands move the cursor around - and for speed, remember that you have an auto-repeat function on the keys!

Other keys you may use in Edit mode are:

FCTN 9 (or FCTN +) to go back to the COMMAND LINE.

CTRL 1 is your OOPS key... in the commands below, if you press the keys in error you can recover by immediately pressing CTRL 1. NOTE that word IMMEDIATELY - I dont mean quickly! but rather that pressing any key between the commands listed below and Oops, will stop Oops working!

FCTN 1 - deletes character cursor is sat on

CTRL K - deletes all text to the right of the cursor

FCTN 3 - Not only deletes text but deletes the actual line!

CTRL 5 - really useful this one, it duplicates the line above! - HOWEVER it will also delete the line the cursor is on, so dont use it if the cursor is sitting on text you wish to keep!

Thats the end of the commands Oops can reverse. Now for some more... FCTN Ø is a toggle which enables you to display or not display the line numbers on the left side of the page - they are not printed anyway.

FCTN 2 is INSERT CHARACTER. Under normal circumstances, it opens up a line for text to be typed in. When done, remembering to end with a space! - press CTRL 2, which is REFORMAT.

FCTN 8 is INSERT LINE, and works by moving the line the cursor is on DOWN, leaving the cursor on a blank line.

CTRL 3 changes the screen colour combinations - not very many choices but better than none!

CTRL Ø (zero) toggles WORD WRAP.... when you switch on, the cursor is a solid block, and control keys work as above. If you toggle word wrap, the cursor becomes an open box... and...

With WORD WRAP off, we are in FIXED MODE and the following key commands alter:

INSERT CHAR (Fctn 2) will merely push the text to the right as you enter the inserted material - very like using INS when entering a Basic program. And when the text is pushed to the right hand side of the screen, it starts getting deleted, so careful!

REFORMAT (Ctrl 2) is used to terminate insert mode, also terminated by use of the other cursor movement keys.

New Paragraph, Last Paragraph, and Next Paragraph do not function in fixed mode.

Those are the directly active keys. You can also insert commands to your printer into the text, using CONTROL MODE.

CONTROL MODE makes available from the keyboard, ASCII characters Ø to 31 , so that you can send those codes to the printer : they are NOT printed, unless that is a part of your printer instruction set : see your printer manual for details.

You enter control mode by pressing CTRL U, which causes the cursor to become an UNDERLINE (Notice the cursor shape always tells you which mode you are in: Word Wrap, Fixed, or Control).

With the UNDERLINE cursor, you have access to the lower ASCCI codes by pressing the following key combinations:

ASCII 1 to ASCII 26 are simple SHIFTED A to Z - thats easy to remember!

ASCII Ø (zero) is a SHIFTED ZERO - thats easy to remember!

Then you'll need to write these down:

ASCII 27 is FCTN R
ASCII 28 is FCTN Z
ASCII 29 is FCTN T
ASCII 3Ø is SHIFT 6
ASCII 31 is FCTN U

As you enter these low codes, odd characters will appear on the screen - they will not be printed! - you will get used to their appearance in time. They are based on the HEXADECIMAL equivalent of the codes.

Remember to switch OUT of control code to use ordinary keys- toggle with CTRL U.

Your printer may for instance require a character 15 to switch to condensed print mode. To insert a character 15 in your text, you need to key:

CTRL U then SHIFT O then CTRL U again.

ESC is short for ESCAPE and is the ASCII value 27, or FCTN R

Consult your printer manual for details of the codes your printer needs.

Note that TI Writer and your printer may have similar codes: it is easy to be confused with the TAB settings on TI Writer and those of your printer: but they are different things! It is usually easier to use TI Writer TABs but for some difficult jobs it may be better to ignore TIW TABS and set and use TABS on your printer - see your printer manual!

One example of compatible but different commands is the UNDERLINE: the keyboard has an underline as FCTN U - but my printer has an underline function available by using ESC - 1 and ESC - Ø. If I use both, the printer prints a continuous underline, with a broken underline one pixel above it!

PAGE START: When you switch your printer on, wherever the platten is - and the paper held by it - is marked in printer memory as the start of the page. The printer then keeps count of the number of lines printed. If your printer has a default page length of 66 lines, and after 40 lines you send a PA symbol, or the standard page feed character, ASCII 12, then the printer will move the paper up 66-40=26 lines.

If you manually move the paper up or down, the printer does NOT count that movement! So if you are using either Form Feed command, take care to avoid all manual paper adjustments!

This article looks as though it could take over TI*MES! So a break here.

However, to print out your text, go back to the command line, enter PF (Print File) and then the printer name (eg PIO) and off it goes. To save text to disk, enter SF (Save File) and then DSK1.FILENAME or whatever. TI WRITER FOR NOVICES again. [Ed. Note: Part II.]

This time a close look at the COMMAND LINE commands...

When the cursor is flashing in the COMMAND section of screen, regardless of the prompt displayed you can go back to the initial command prompt by using FCTN 9, enabling you to quickly exit a function entered in error.

Printing can be halted by using FCTN 4.

E for Edit we have seen puts the cursor onto the text screen.

PF for PRINT FILE enables you to print your text, and is followed by the printer name: ENTER PF, and then when prompted, ENTER printer name.

PF can also be used:

To print PART of a text file by using line numbers in conjunction with the printer name:

1 16 PIO will print lines 1 to 16 to PIO

24 E PIO will print from 24 to the end to PIO

To print text to a disk:

Using PF instead of SF, you do NOT save the TAB settings to disk: important if you are using TI Writer to create a file which will be used as input for another program, such as Pilot 99, C-99 etc.

Using disk or printer, you can add not only line numbers but also control letters in front of the printer name, for instance:

L 1 16 PIO will print the first 16 lines of text to PIO WITH THE LINE NUMBERS... but the printed line will be shortened to 80 characters INCLUDING the space the line numbers use, so text is liable to be lost!

Adding a C will strip out control characters (eg 1 to 31) from the text, while F sends text as FIXED 80 instead of the usual VARIABLE 80.

NOTE: TI Writer can load and save both FIXED 80 AND VARIABLE 80 files! They must however be DISPLAY type.

SF (Save File) is used to save text to disk in the normal manner, and TAB settings will form the last data item in the file.

Use two line numbers (or E) to save PART of a file to disk. E stands for the end of the text presently in the console.

LF (Load File) is used to load a DISPLAY 80 file from disk, and the file may be variable or fixed. If you precede the disk filename with line numbers you may:

Load part of a file: FIRSTL LASTL FILENAME

Add disk file to existing text:

AFTERL FILENAME will load the disk file to commence after line AFTERL.

Add part of disk file to existing text:

AFTERL FIRSTL LASTL FILENAME

Note that if you add text, it is added after the line number specified, and any existing text after that line number is simply destroyed, irrevocably.

[Ed. Note: This is simply not true! All text before and after the split is preserved, with the new text inserted inbetween.]

If you wish to move chunks of text around, use MOVE - M. The computer prompts for FIRSTL LASTL TOL - the first and last line numbers of the chunk to be moved, and the line AFTER which it is to be INSERTED. MOVE does not write over existing text: it moves text down to make room.

C for COPY is similar to MOVE but does not delete the copied portion, merely replicates it. Copy also takes care not to delete text when in operation.

P for Purge clears out the text in the text buffer - you will be given an opportunity to reconsider! If you even then change your mind, if you immediately type RE (recover edit) in the command line, you MAY recover your text - all bar the top line!

Q for Quit exits the program, and you are given a choice of actions- E for Exit, S to save text and P to purge.

FS for FIND STRING will move the cursor to the first occurrence of the quoted string AFTER its present position - so move it to the start of your text to search the whole document! The text to be searched for is bracketed NOT with quotes but with diagonals like this /find me/

RS is Replace String. A word of warning: If you are in Word Wrap mode, using RS will reformat your ENTIRE text. If you do not wish the text reformatted, go to fixed mode before using RS! That is because RS uses INSERT and REFORMAT commands!

To use it you bracket the text as before, like this: /oldtext/newtext/

and then select the options Yes, No, All and Stop. If you select All then ALL instances are changed - and this can take a long time in a long document. Once ALL has been selected you cannot escape until it has finished!!!

S for Show is rather like Basic's GOTO - you input a text line number and the cursor will jump to it!

SD for Show Directory enables you to see what is on your disk. Remember to leave the directory by pressing ENTER.

and finally, T for TABS. After pressing T, the tab line will appear. The first character the computer looks for is a L for left margin. You may also have an I for indent - used for new paragraphs.

Indeed you can be very clever and OUTDENT by putting the I in front of the L on the Tab line!

Then each tab is marked by a T (you can put full stops OR spaces where you dont want a tab stop).

And fix the right margin with an R.

to edit text which is to the left of the left margin! (You can change tabs several times in a document). There is a left margin release to enable you to move the cursor past the left margin:

CTRL Y followed by good use of FCTN S will do the trick.

There is NO right margin release - you have to change the tab!

Note that use of REFORMAT (CTRL 2) or using RS (Replace String) when in word wrap mode, will reformat your text in accordance with the tab settings AT THAT TIME. Remember REFORMAT will work on all text in the current paragraph, FROM the current cursor position.

Well, thats another load of text.... Formatter next time....

This could be a long tutorial...

Stephen Shaw

UnRUNnable Basic Programs in XBasic

```
LISTED FROM A PROVEN RUNNING PROGRAM
  (Courtesy Stephen Shaw)
In case of difficulty, CHECK THOSE DIGITS!
  (Reformatted by Tom Freeman)
1 ! VDP UTILITY 2 by John Behnke, Chicago, USA
2 ! Enables you to RUN a TI BASIC program in XB
even if Char Sets 15 & 16 have been used.
3 ! To use:
  1. SELECT XB
  2. IF REQUIRED, use CALL FILES(1) and NEW
4 ! 3.LOAD TI BASIC PROGRAM
    4.MERGE THIS PROGRAM INTO IT.
    5.off you go... all the CALL LOADS take a little
5 ! while to work!
6 ! THIS NEXT LINE IS VITAL:
7 CALL VDPUTIL2
8 ! TI BASIC PROGRAM FITS IN    HERE:
9 ! ...........
10 ! .........

32714 SUB VDPUTIL2
32715 CALL CLEAR :: CALL INIT :: CALL LOAD(8196,63,232)
32716 CALL LOAD(16360,80,79,75,69,82,32,38,12,80,79,75,6
9,86,32,37,164,80,69,75,86,32,37,36)
32717 CALL LOAD(9491,100)
32718 CALL LOAD(9508,2,224,37,20,3,0,0,0,2,0,0,100,200,0
,37,18,4,192,2,1,0,1,4,3,2,32,12,4,32)
32719 CALL LOAD(9536,32,24,18,184,192,32,131,74,2,1,37,0
,208,160,131,18,9,130,2,34,255,255,4,32,32,44)
32720 CALL LOAD(9562,4,197,209,34,36,255,9,132,19,21,4,1
95,60,224,37,18,200,5,131,76,200,5,131,78,200,5)
32721 CALL LOAD(9588,131,80,2,5,64,0,161,68,2,131,0,1,17
,6,2,5,65,0,161,67,6,196,200,4,131,76)
32722 CALL LOAD(9614,200,5,131,74,4,192,192,66,5,129,4,3
7,254)
32723 CALL LOAD(9636,2,224,37,20,3,0,0,0,4,192,2,1,0,1,2
00,1,37,18,4,32,32,12,4,32,32,24,18,184)
32724 CALL LOAD(9664,200,32,131,74,37,0,184,32,131,18,37
,19,2,3,0,2)
32725 CALL LOAD(9680,4,192,192,67,4,32,32,12,4,32,32,24,
18,184,216,224,131,75,37,0,5,131,136,3)
32726 CALL LOAD(9704,37,18,22,242,192,32,37,0,2,1,37,2,1
92,131,2,34,255,254,4,32,32,36)
32727 CALL LOAD(9726,4,192,216,0,131,124,2,224,131,224,4
,96,0,112)
32728 CALL LOAD(9740,3,0,0,0,4,192,2,1,0,1,4,32,32,12,20
0,32,131,74,37,18,2,1,0,2,4,32,32,12,4,32)
32729 CALL LOAD(9770,32,24,18,184,192,32,131,74,208,32,3
7,19,4,32,32,48,4,91)
32730 CALL LOAD(8194,39,104)32731 SUBEND
32732 SUB CHAR(A,A$):: L=LEN(A$)
32733 A$=A$&RPT$("0",16-L)
32734 FOR I=1 TO 16 STEP 2
32735 A1$=SEG$(A$,I,1)
32736 A2$=SEG$(A$,I+1,1)
32737 IF A1$<":" THEN A1=VAL(A1$)*16 ELSE A1=(ASC(A1$)-5
5)*16
32738 IF A2$<":" THEN A1=A1+VAL(A2$)ELSE A1=A1+ASC(A2$)-
55
32739 CALL LINK("POKEV",767+8*A+(I+1)/2,A1)
32740 NEXT I
32741 SUBEND
32742 SUB COLOR(A,B,C)
32743 CALL LINK("POKEV",2063+A,(B-1)*16+C-1)
32744 SUBEND
32745 END
```

Listed with COLIST, A Tony McGovern Program

CORRECT NUMERATION . Jim Peterson.

```
100 CALL CLEAR
110 PRINT TAB(7);"NUMBER SPE
AKER": : :"by Jim Peterson":
"      of Tigercub Software"
: : :
120 PRINT " This program wil
l print any":" number of les
s than 67":"digits in number
s and in"
130 PRINT "words, and will s
peak the":"words.": : : :" R
equires Terminal Emulator":"
II and Speech Synthesizer.":
 : :
140 CALL CHAR(39,"0000000000
301020")
150 OPEN #1:"SPEECH",OUTPUT
160 DIM HIGH$(21),NN$(23)
170 DATA ONE,TWO,THREE,FOUR,
FIVE,SIX,SEVEN,EIGHT,NINE
180 DATA TEN,ELEVEN,TWELVE,T
HIRTEEN,FOURTEEN,FIFTEEN,SIX
TEEN,SEVENTEEN,EIGHTEEN,NINE
TEEN
190 DATA TWENTY,THIRTY,FORTY
,FIFTY,SIXTY,SEVENTY,EIGHTY,
NINETY
200 DATA THOUSAND,MILLION,BI
LLION,TRILLION,QUADRILLION,Q
UINTILLION,SEXTILLION,SEPTIL
LION,OCTILLION,NONILLION
210 DATA DECILLION,UNDECILLI
ON,DUODECILLION,TREDECILLION
,QUATTUORDECILLION,QUINDECIL
LION,SEXTEDECILLION
220 DATA SEPTENDECILLION,OCT
ODECILLION,NOVEMDECILLION,VI
GINTILLION
230 FOR J=1 TO 9
240 READ ONE$(J)
250 NEXT J
260 FOR J=1 TO 10
270 READ TEEN$(J)
280 NEXT J
290 FOR J=1 TO 8
300 READ TEN$(J)
310 NEXT J
320 FOR J=1 TO 21
330 READ HIGH$(J)
340 NEXT J
350 PRINT : : :
```

EXPLANATION OF PROGRAM LINES

Note how the ":" is used to get 3 lines of text into one PRINT statement.
In line 140, character ASCII 39, the apostrophe, is redefined as a comma because a true comma could not be used in the string P$ in line 610.
Line 150 activates the Speech Synthesizer.
The variable names HIGH$ and NN$ will contain moe than 11 subscripts, so they must be dimensioned in advance.
Lines 230-250 define ONE$(1) through ONES$(9) as being the words ONE through NINE from the DATA statement in line 170.
Similarly, The words in DATA statement 180 are read into TEEN$, those in DATA statement 190 into TEN$, and those in DATA statements 200-220 into HIGH$.
Line 360 speaks the word "NUMBER" and line 370 requests INPUT of the user's number in the form of a string rather than a numeric value so that a large number will not print out in exponential notation. However, if any non-numeric characters are mistakenly entered in N$, taking the VAL of it in line 420 would cause the program to crash. So, lines 380-410 check through N$ character by character. If a character is not found in the string "0123456789", the value of POS is 0 and the program requests another number.
Line 420 edits for an invalid negative number or a number containing a decimal, and line 430 edits for a number more than 66 digits long.
If N$ is input as 0, or a string of 0's, lines 470-500 print and speak "ZERO" and go back for another. Otherwise, line 510 checks whether N$ can be evenly divided into sets of 3 digits; if not, 520-530 add a 0 in front of it and go back to measure its length again - and again, if necessary. In 540, X is the number of sets of 3 digits in N$.
The loop 550-610 goes through the length of N$ in steps of 3, using JJ as a counter to assign each 3-digit segment to a subscript of NN$. At the same time, the string P$ is built up as a representation of the number having each 3-digit

```
360 PRINT #1:"NUMBER"
370 INPUT "NUMBER? ":N$
380 L=LEN(N$)
390 FOR J=1 TO L
400 IF POS("0123456789",SEG$
(N$,J,1),1)=0 THEN 360
410 NEXT J
420 IF (VAL(N$)<1)+(VAL(N$)<
>INT(VAL(N$)))THEN 360
430 IF L<67 THEN 470
440 PRINT "HEY! I CAN ONLY C
OUNT TO A":"VIGINTILLION!":
:
450 PRINT #1:"HAY I CAN ONLY
 COUNT TO A VIGINTILLION"
460 GOTO 360
470 IF VAL(N$)>0 THEN 510
480 PRINT : :"ZERO": :
490 PRINT #1:"ZERO"
500 GOTO 360
510 IF L/3=INT(L/3)THEN 540
520 N$="0"&N$
530 GOTO 380
540 X=L/3
550 FOR J=1 TO L STEP 3
560 JJ=JJ+1
570 NN$(JJ)=SEG$(N$,J,3)
580 IF J>1 THEN 610
590 P$=STR$(VAL(NN$(JJ)))
600 GOTO 620
610 P$=P$&"'"&NN$(JJ)
620 NEXT J
630 PRINT : : :P$: : :
640 FOR J=1 TO X
650 GOSUB 670
660 GOTO 1150
670 IF VAL(NN$(J))<>0 THEN 7
10
680 A$=""
690 FLAG=1
700 GOTO 1140
710 FLAG=0
720 H=VAL(SEG$(NN$(J),1,1))
730 T=VAL(SEG$(NN$(J),2,2))
740 TT=VAL(SEG$(NN$(J),2,1))
-1
750 VV=VAL(SEG$(NN$(J),3,1))
760 IF T=0 THEN 1000
770 IF T>9 THEN 810
780 A$=ONE$(T)
790 SP$=A$
800 GOTO 1000
810 IF T>19 THEN 880
```

set of numerals, except the first set, preceded by a comma (as redefined in line 140), as large numbers are usually written. It is then printed by line 630.

Now, the loop 640-1210 goes through the 3-digit sets in sequence, each time going to the subroutine 670-1140 to compute what should be printed and spoken, then jumping to 1150 to do so. A$ is the word to be printed, SP$ is the word to be spoken.

In line 670, if the set consists of all 0's the program drops through to 680, A$ will be a blank, a flag is set to 1 and we jump to 1140 which returns us to 660 and thence to 1150 where a blank is printed and the FLAG value of 1 causes us to jump over the speech routine and the printing/speaking of HIGH$.

But if the value of the set is more than 0, line 670 goes to 710 which makes sure that the FLAG is reset to 0, and then determines the values of the numerals in the set. H is the 1st numeral, T is the 2nd-3rd numerals, TT is the 2nd numeral and VV is the 3rd.

In line 760, if T (2nd-3rd numerals) is 00 the set must be an even hundred so we can skip over the checking of TEEN$ and TEN$ to line 1000 (which will drop us through to 1010 because we have already determined in 670 that all 3 digits are not 0).

In line 770, if T is more than 9 we can skip over the ONE$; otherwise, the value of T will pick out the correct subscript of ONE$ (from 170 and 230-250) to be printed and, in line 790, to be spoken. In 810, from 770, if T is more than 19 we can similarly skip over TEEN$; otherwise, the value of T picks out the proper subscript of TEEN$ (see 180 and 260-280).

In most cases SP$, the word to be spoken, can be defined as the same as A$, the word to be printed, but the words NINETEEN and NINETY would be mispronounced (that E in the middle confuses the computer!) so we must define them separately.

In line 880, from 810, if VV (the 3rd digit) is 0 we don't need ONE$, so in

```
820 A$=TEEN$(T-9)
830 IF T<>19 THEN 860
840 SP$="NINE TEEN"
850 GOTO 1000
860 SP$=A$
870 GOTO 1000
880 IF VV<>0 THEN 950
890 A$=TEN$(TT)
900 IF TT<>8 THEN 930
910 SP$="NINE TEE"
920 GOTO 1000
930 SP$=A$
940 GOTO 1000
950 A$=TEN$(TT)&"-"&ONE$(VV)
960 IF TT<>8 THEN 990
970 SP$="NINE TEE"&ONE$(VV)
980 GOTO 1000
990 SP$=A$
1000 IF H=0 THEN 1080
1010 IF T=0 THEN 1050
1020 A$=ONE$(H)&" HUNDRED &
"&A$
1030 SP$=ONE$(H)&" HUNDRED &
"&SP$
1040 GOTO 1140
1050 A$=ONE$(H)&" HUNDRED"
1060 SP$=A$
1070 GOTO 1140
1080 IF (J<X)+(T=0)+(VAL(N$)
<100)THEN 1140
1090 A$=" & "&A$
1100 IF (TT<>8)*(T<>19)THEN
1130
1110 SP$=" & "&SP$
1120 GOTO 1140
1130 SP$=A$
1140 RETURN
1150 PRINT A$
1160 IF FLAG=1 THEN 1200
1170 PRINT #1:SP$
1180 PRINT HIGH$(X-J)
1190 PRINT #1:HIGH$(X-J)
1200 GOSUB 670
1210 NEXT J
1230 A$=""
1240 JJ=0
1260 P$=""
1270 FOR D=1 TO 500
1280 NEXT D
1290 GOTO 350
```

890 the value of TT (the 2nd digit) gives us the correct subscript of TEN$ (lines 190 and 290-310); else, 880 takes us to 950 where TT again picks out the right word for TEN$, a dash (-) is added after it, and then the value of VV picks out the correct word for ONE$.

In all cases, the program jumps to line 1000. If H (the 1st digit of the set) is 0, we do not need the word HUNDRED so we skip to 1080. If T in 760 was 0 we need only the HUNDRED, so go to 1050. Else, H gives us the subscript of ONE$ to be placed in front of the word HUNDRED in lines 1020-1030 and both of these words are placed in front of whatever A$ and SP$ already consist of from 780, 820 or 950.

Line 1080, from 1000, checks to see if we are on the last set of 3. If so, and if T was more than 0 and N$ was more than 99, line 1090 places the symbol & before the printed number; SP$ in 1110 pronounces & as "AND".

In all cases, the routine goes to 1140 which returns it to 660 and thence to 1150, which prints out the words. We have already described what line 1160 does. Line 1170 speaks the words that were just printed.

Line 1180 finds the correct subscript for HIGH$ by subtracting the current value of J (the 550-1210 loop we are in) from the value of X (the total number of loops to be made), and then prints and speaks the correct sub-script of HIGH$ (from 200-220 and 320-340).

After the first pass through the loop we enter the 670-1140 subroutine from 1200 instead of from 650.

When the loop has been completed, and the entire number has been spoken and and printed, we must cancel out the values of A$, JJ and P$, which were formed by adding onto themselves, be-fore we pause briefly and then go back to ask for the next number.

Thanks to Stephan Shaw of England for his contributions, the "MINI TI-WRITER MANUAL" and the programs, "UnRUNable Basic Programs" and Jim Peterson's "NUMBER SPEAKER".

Did you know that...?

by Chick De Marti

PRINT USING or DISPLAY USING can read variables, arrays and strings?

```
Exam.  100 V=395
       110 F$="####"
       500 DISPLAY AT(12,1):USING F$
       :V
Other valid statements:

       700 PRINT USING A$:X,Y
       800 DISPLAY USING RPT$("#",5)
       &V$:A(12)
```

<*><*><*><*><*><*>

MONOPOLY

Computer analysis by Irvin Hetzel of Iowa State University has reveal ed the 10 spaces most likely to be hit during a game of Monopoly, are: Illinois Avenue, Go, B & O Railroad Free Parking, Tennessee Avenue, New York Avenue, Reading Railroad, St. James Place, Water Works, and Pennsylvania Railroad.

Per Murray Suid and Ron Harris...of "Computers and Data Processing"

<*><*><*><*><*><*>

SPEED:

There is a wide diversity in the speed of various computers. The execution of a command on a small computer may be measured in less than a millisecond, a one-thousanth of a second. Many computers execute an instruction in microseconds, one millionth of a second. The modern supercomputers are approaching the nanosecond range. How fast is a nanosecond?

If a nanosecond were one mile... then 1 second would be 2000 trips to the moon and back.

If a nanosecond were one minute.. then a second would be 1900 years!

<*><*><*><*><*><*>

If you find a line in your program with "GOTO 32767", this is an ERROR MESSAGE that is printed when you re-sequence a program that has a GOTO to a non-existing line number.

* * * * * * * * * * *

A computer's weakness? It only knows the truth... thus resulting in only literal translations. An example might be... "The Spiret Is Willing, But The Flesh Is Weak" might be interpreted as..."The Wine Is Agreeable, But The Meat Was Spoiled."

Some programs have been developed that DO seem to have the ability to think (or at least, carry on a conversation). "A UCLA psychologist invented PARRY"...a program taught to..."act like a 28-year-old horse player with paramoid ideas about being prusued by a vengeful bookie". In an article from the text-book COMPUTERS AND DATA PROCESSING by Capron and Williams, a demonstration is revealed that..."when interviewed via a terminal by six spychiatrists (it) was so successful...the psychiatrists as often as not guessed it was human."
In another form of 'thinking' in 1978, "David Levy, the International Chess Master"...played a game of chess with..."Northwestern University's computer chess champion, CHESS 4.7...The computer made some moves that Levy later said were such that he found it hard to believe he was not playing with an outstanding human opponent.

Several years ago,British mathematician , Alan Turing, proposed a test of thinking machines (where) a human being is seated before two hidden terminals, one operated by another person and one by a computer." By carrying out a conversation through the terminals,they were asked which was human operated and which was a computer. Both PARRY and CHESS 4.7 ..."passed and had to be concidered for all practical purposes a thinking machine."

More about A.I. (ARTIFICIAL INTELLINGENCE) in later issues...

CLOSE ENOUGH
by Chick De Marti

A computer can be taught to accept imperfection. When in the GRAPHICS mode you can dictate the percentage of coincidence. I thought it would be nice if the computer would accept some of my math when it was plus or minus, say 10% of the correct answer. THis little routine does just that. NOTE: You can change the percentage in lines 160 and 220.

```
100 CALL CLEAR
110 INPUT "A NUMBER ":TEST
120 PRINT : :
130 FOR I=20 TO 1 STEP -4
140 X=I/100 :: N=TEST-X
150 PRINT N,
160 IF (N<TEST-.10)THEN GOSUB 250 ELSE GOSUB 270
170 NEXT I
180 PRINT
190 FOR I=1 TO 20 STEP 4
200 X=I/100 :: N=TEST+X
210 PRINT N,
220 IF N>TEST+.10 THEN GOSUB 250 ELSE GOSUB 270
230 NEXT I
240 END
250 PRINT "SORRY!"
260 RETURN
270 PRINT "CLOSE ENOUGH"
280 RETURN
```

<*><*><*><*><*><*><*><*><*><*><*><*>

Check out line 110 ! "I DIDN'T KNOW THAT!"

```
100 CALL CLEAR :: PRINT "HERE I GO": : :
110 GOSUB 200 DELAY_ROUTINE :: PRINT "I'M BACK"
120 END
200 FOR I=1 TO 400 :: NEXT I
210 RETURN
```

In Harms's Way

by Bill Harms

A WINDOW-CALCULATOR program in Extd Basic (or rather 2 Progs and an article describing them) is now in your club library, written by Bill Harms. One prog. is a very short 10 key-like calculator and the other prog. is a bit longer, since it's a bit fancier. The prog. can be windowed onto a existing screen because the screen lines are saved to an array and then recalled when you exit the calculator. It's set-up as a subprogram with line #'s of 20000 so it can be merged into your existing program and accessed with CALL CALC. The CALL CALC would be added at a menu or screen prompt with your existing or a new CALL KEY or ACCEPT AT. The variable are only local to the subprogram so you don't have to mess around changing your program. The longer version even has a printout option that creates an "adding machine"-like tape. The 2 versions can just be RUN as standalone Calculator programs.

A lazy man's Name & Address program was added to the club library by Bill Harms that he wrote recently, when he (I) wanted labels to go with the form letters created with TI-WRITER. It's called TIWRITMAIL. It's really a set of (an article explaining the use of the prog., the prog itself, a demo file (D/V-80) of 15 records, and a file shownning the layout of a record (line) to be used. I use TI-WRITER as the file editor with it's capability to: Delete Line, Insert Line, Move Lines, Load (Merge) Blocks of Lines from other files, etc, etc., since one line is a record. A record is set-up to be 9 digits of status codes, Last Name, First Name, Street Address, City-State-Zip. Each is a fixed length and you'd set-up tabs in TI-WRITER to hop to the next field. The bad part is that only 80 char. makes for short names and addresses, but it's OK in most cases. The article and prog. are fully REMed to explain what is happening and how to change/ customize the fields to your taste (within limits). The Options in the program are: to print Labels or a straight Listing (double spaced) to paper or disk; to print Labels to paper or disk from a previously created Value File via TI-WRITER. The program is written in console BASIC, so if there are commas in the file being read, next field is printed on the next line. Don't put a comma after the City. Or change the INPUT to be LINPUT and run it in EXTD BASIC. The main thing is that it is really simple to use or change and saves lots of time.

If you use the TI MICROSOFT-MULTIPLAN, you might be interested in getting 2 articles I wrote and put in the club library. Also there is a program that demo's the process. What Process? Well, the way to create a file in BASIC or EXTD BASIC that can be loaded into Multiplan with the Transfer Option Symbolic. This is a way of getting info. into your spreadsheet besides typing it in.. R. M. Mitchell, the publisher of "The Smart Programmer" previously "SUPER 99 MONTHLY" by Bytemaster Computer Services, 171 Mustang Street, Sulphur, LA 70663 really published the techniques in 3 issues last Summer. I modified his program, which is to load text into Multiplay(sic.) in 5 clms of 16 char. each (which is the good old 80 char. type file created with TI-WRITER). The Multiplan manual briefly describes the Symbolic Link format, but it leaves out some critical things. The articles describe the SYLK format a little bit and a small program using it to create a file of values, ie. $$ or units in 2 columns, and then get a sum of the right-hand column and name it. The techniques were modified somewhat to provide a way for me to dump into Multiplan the total $$ I spent each month on about 60 categories of expense and 5 categories of Income. I was tired of hand entering them so I just had to write AS-Tran (a program to recap a checkbook). Anyway, even though

Symbolic file do load slowly into Multiplan, they go faster than typing in the info. And you have the capability of sending someone, even on a different computer, a copy of your spreadsheet, because Multiplan SYLK contains all the Symbols to create all the Multiplan info. ie, formulas, cell format. The one nasty thing not mentioned in the Multiplan manual about BASICly created files is that they must be created as Display Fixed 128. Then you have to sector edit the File Descriptior Record in byte 12 to be 0202. This changes the "description" of the file to Internal Fixed 128 so it can be loaded into Messyplan(sic.). Barry Traver another great programmer/person and publisher (DiskaZine) wrote a fine little sector editor you can run out of EXTD BASIC to do the Sector Editing. This is in the prog. in the library. In his DiskaZine he published several greatly enhanced versions of this sector editor call RAW (Read And Write). Also the secrets of the use of quote marks is explained. All in all it's rather like a simple method of punctuation/language and one can, with some experimenting get the data to really FLOW. There is one more secret to be read in the articles, the discovery of which, I'll leave to you!!!-- E X P L O R E -- In Harms' Way

(We hope to make this column a regular feature, with Bill 's continued cooperation. Ed.)

From MICROpendium March 1986 issue

XB TI-WRITER DISCOVERY

While experimenting with the Extended BASIC TI-Writer loader (in our library) that came from Austria I discovered that the utility option has a feature which will allow the loading of some Assembly Language programs that prevously required the Editor/Assembler cartridge! Here are the steps in loading an assembly language program with the TI-Writer loader:

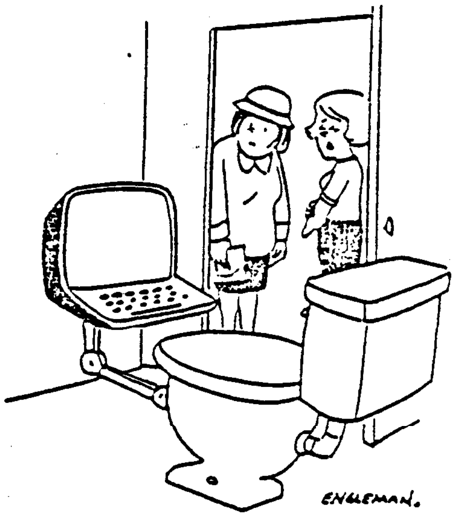
1. Select number 3 on the menu (Utility).

2. Type in the device and file name for the assembly program.

3. Select graphics load (number 2 on menu).

The program should now load and automatically run. However, there is one catch, this loader will only load certain assembly language programs. The only way to find out if it will load a particular assembly language program is to just try it. If an error occurs then the loader will not load it. One program I found that the utility loader would load was the new disk manager program we have in our library (not DM 1000 it loads from XB anyway.). This utility option on the XB TI-Writer loader is handy if you do not have the Editor/Assembler cartridge and you wish to run an assembly language program that will not load in Extended BASIC.

Gary Cox



"Most people read in the bathroom Chick is different."

Forth Floating Point Fixes

by Glenn Davis

Two tidbits for the Forth Forum this month. If you have ever tried using the -FLOAT words, you may have noticed that they are quite slow. How slow? I tried the *Creative Computing* benchmark by David Ahl in TI BASIC, Extended BASIC, and TI Forth. When TI BASIC beat them both, I knew something was very odd. Both BASICs beat Forth on this benchmark.

Since Forth interprets its code up to 40 times faster than BASIC does, I knew that the crawl was due to inefficiency. The -FLOAT routines are written in Forth, so they have a lot of overhead (from the "inner interpreter") for the amount of data moved. To help speed things up, I rewrote the -FLOAT routines as CODE words (that means they are in machine language, and are not transportable from one CPU to another) and fit them on the screens where the original words were. So, merely replace screens 45, 46, and 48 of a Forth system disk (not the original disk!) and you may load -FLOAT as usual. (Note: if you have a BSAVEd version that includes -FLOAT remember to re-compile, reload and re-BSAVE it).

Several of the words on SCR#46 have been modified and a few "unimportant" words have been removed, to save space in both the source and object codes. Note the sequence "12 SYSTEM" or "C SYSTEM" is used in several places in these words. I replaced them with XMLLNK.. They are synonyms. -FLOAT always must have the synonyms loaded (since FRND uses RND, which is a -SYNONYM word).

Another screen that needed to be modified was SCR#48, which has words for compiling floating-point numbers and output. The word >F was too slow for me, so I did some tinkering and improved speed more than 75%. (Look the word up in the Forth manual if you don't know what it does) >F originally compiled the actual digit-string into the dictionary and did a conversion at run-time (bad move). This version converts the digit-string at compile-time. If the code is being interpreted, the number is left on the stack, otherwise it is compiled into the dictionary with code to push the value onto the stack at run-time. This process is completely transparent to the programmer, so >F will act just like the old version of the word, only much faster.

My other tidbit has to do with "undocumented" Forth instructions for the -FLOAT words. "Huh? The source code is all there!" you say? True. But five single-precision words do not have -FLOAT counterparts: ABS, MINUS, ROT, MAX, and MIN. ABS leaves the absolute value of a number and MINUS serves as a unary minus to negate a number. In this article I will address the first two and leave the remaining three for you to work out. They are REALLY easy. (Hint: just use the double-length words in the TI Forth manual as examples, and replace words referring to double-length numbers with their floating-point counterparts).

Often we need to multiply a floating point number by -1 or take its absolute value. Since words are not provided EXPLICITLY for these operations as they are with the other word sets (MINUS and DMINUS), I found the need to write some. In my prototyping to find the optimal code I realized that, due to the nature of the Radix-100 ROM routines, the single-precision words would work on floating point numbers too! In floating point, the top word on the stack (the "most significant" part of the 4-word value) is in two's complement if the VALUE is negative. The original TI User's Manual has an appendix on it, if you don't believe me.

To illustrate, type (with -FLOAT loaded already):

```
PI FDUP F. MINUS F. <ENTER>
```

and the computer will print PI and then its opposite. ABS will allow the FP routines to return only non-negative numbers

```
PI MINUS FDUP F. ABS F. <ENTER>
```

Simple, quick, and elegant. That's what Forth is famous for.

see programs page 193

\$

Courtesy of C I M 99 U.G.
of Montreal, Canada

\$

TI-FORTH

Fourth Floating Point Fixes Programs

```
SCR#45:

( FLOATING POINT <4 WORD> CODE STACK ROUTINES  20APR85 GED )
0 CLOAD PI   BASE->R DECIMAL 33 R->BASE   CLOAD RANDOMIZE
BASE->R DECIMAL 74 R->BASE . CLOAD ;CODE   BASE->R HEX
CODE FDUP    C009 , 0229 ; FFF8 , C049 , CC70 , CC70 ,
             CC70 , C450 , 045F ,
CODE FDROP   0229 , 0008 , 045F ,
CODE FOVER   0229 , FFF8 , C049 , C001 , 0220 , 0010 ,
             CC70 , CC70 , CC70 , C450 , 045F ,
CODE FSWAP   C009 , C070 , C080 , C0F0 , C130 , C670 ,
             CA70 , 0002 , CA70 , 0004 , CA50 , 0006 ,
             0220 , FFFA , CC01 , CC02 , CC03 , C404 , 045F ,
CODE F!      C039 , CC39 , CC39 , CC39 , C439 , 045F ,
CODE F@      C019 , 0229 , FFFA , C049 , CC70 , CC70 ,
             CC70 , C450 , 045F ,
834A CONSTANT FAC  835C CONSTANT ARG
R->BASE -->
SCR#46:

( FLOATING POINT ARITHMETIC ROUTINES 12JUL82-LCT 26APR85-GED )
BASE->R HEX
: >FAC FAC F! ;
: >ARG ARG F! ;
: FAC   >FAC F@ ;
: SETFL  >FAC >ARG ;
: F+    SETFL 0600 XMLLNK FAC> ;
: F-    SETFL 0700 XMLLNK FAC> ;
: F*    SETFL 0800 XMLLNK FAC> ;
: F/    SETFL 0900 XMLLNK FAC> ;
: S->FAC   FAC ! 2300 XMLLNK ;
: FAC->S   1200 XMLLNK FAC @ ;
: FAC>ARG  FAC ARG 4 MOVE ;
: F->S >FAC FAC->S ;            : S->F S->FAC FAC> ; DECIMAL
: FRND 3 0 DO 100 RND 100 RND 256 * + LOOP
  100 RND 16128 + ;                            R->BASE -->
SCR#48:

( FLOATING POINT - COMPILE NO TO STACK 12JUL82 LCT) BASE->R HEX

: F$   PAD 1+ SWAP >R R CMOVE R> PAD C! VAL FAC> ;
( Compile No to stack at compile time  12MAR86 GED )
: >F  BL WORD HERE COUNT  F$
   STATE @   IF  >R >R [COMPILE] DLITERAL
                 R> R> [COMPILE] DLITERAL
             ENDIF ; IMMEDIATE

( FLOATING POINT OUTPUT ROUTINES )
: JST   PAD C@ - SPACES PAD COUNT TYPE ;
: F.R   >R >FAC STR R> JST ;
: F.    0 F.R ;
: FF.R  >R >R >R >FAC R> 0 R> STR. R> JST ;
: FF.   0 FF.R ;
R->BASE -->
```

DISK SWEEPER
by Adrian Robinson

In recent months a TI BASIC Disk Sweeper program has appeared in several of the users group newsletters and in the Feb 86 MICROPENDIUM. The program was authored by Steve Patterson of the New Horizons Users Group of Ohio. A disk sweeper is a program which "deletes" all files on the disk by erasing all pointers to the files from the Directory without reformatting the disk. The file data remains on the disk but cannot be accessed.

I will not repeat Steve's 44-line program here, but the core of the program may be represented by the following:

```
100 OPEN #1:"DSK1.",RELATIVE
,INTERNAL,INPUT
110 INPUT #1:A$
120 INPUT #1:A$
130 IF A$="" THEN 160
140 DELETE "DSK1."&A$
150 GOTO 120
160 CLOSE #1
```

This is the familiar Catalog routine with a DELETE instruction added. Line 110 inputs the disk name, record 0. Line 120 inputs file names, line 140 deletes them and line 150 forms a loop. This routine will not delete all the files on the disk, so Steve sets up an outer loop to repeat the routine 5 times.

Now, let us see if a little analysis will let us improve on a good idea. The first improvement can be seen if we RECognize (sic) exactly how the instructions are performed. The INPUT instruction is, of course, an auto-incrementing instruction. After line 120 is executed the first time, the INPUT pointer is moved to the second record position in the directory link map. Now, when line 140 is executed, it not only deletes the first file but also re-orders the link map by moving all succeeding entries up one position so that the original file #2 is in the first link map position. Line 150 returns to the INPUT instruction which now inputs the second entry in the link map, which is file #3 on the disk, leaving file #2. Proceeding in this way, the routine will delete every second file. Repeated passes through the routine will reduce the number of files by one-half (approx) in each pass.

If you will excuse a little arithmetic, it may be shown that, if there are N files on a disk, then the number remaining (R) after P passes through the above routine will be:

R=INT(N/2^P)

or, the number of passes required to sweep N files will be:

P=1+INT(log N/log 2)

so we can see that 5 passes will sweep a maximum of 31 files from a disk while the maximum allowable 127 files will be swept with 7 passes.

Perhaps by now you may have guessed why I spelled "recognize" as I did above. The directory file is a RELATIVE file and a quick "fix" can be made to the sweep routine by simply replacing line 120 with:

```
120 INPUT #1,REC 1:A$
```

Now the routine will sweep any number of files in one pass since line 120 always inputs the first entry and line 140 consecutively feeds each file to the first entry position. In addition, we may delete line 110 since we need never look at the disk name.

The routine still contains a small problem if there are any write protected files on the disk. The DELETE instruction will not delete write protected files and will abort with an error if one is encountered. This can be corrected by rewriting the routine as follows:

```
100 OPEN #1:"DSK1.",RELATIVE
,INTERNAL,INPUT
110 R=1
120 INPUT #1,REC R:A$,T
130 IF A$="" THEN 190
140 IF T<0 THEN 170
150 DELETE "DSK1."&A$
160 GOTO 120
170 R=R+1
180 GOTO 120
190 CLOSE #1
```

T is the file type and T<0 indicates write protection. The routine now bypasses each protected file and increments the record pointer to the next file. This may actually be a bonus since we can now write protect the files we want to keep, then run this routine to sweep the disk of everything else.

Add some bells and whistles and you have a working single-pass TI BASIC disk sweeper program. I have run out of space for a full program listing.

CAUTION: Running the above 10-line routine will sweep the disk in drive 1, so be very careful!! Be sure to include adequate safeguards in your program. Temporarily, including a line:

```
90 END
```

can prevent accidental running.

Thanks to the User's Group of Orange County and their ROM Newsletter.

L.A. 99ER'S JULY 23, 1986 PROGRAM

Steve Chalcraft will be master of ceremonies. Vice president George Steffen will be acting president for Terry Masters (she's on vacation.) Librarian Fred Moore will present his choices from the library.

Manuel and Teresita Fernandez will tell us how they started with Multiplan and what they are doing with it. Manuel and Teresita are relatively new members and this should be an excellent opportunity for everyone to meet this charming couple.

Dave Whitcombe will tell us what he does with his 99-4A. Dave is an engineer in the airospace industry and if you think your machine is just a toy I'm sure he will change your mind.

Jane Hartwig is serving the coffee and cookies, and if you don't know Jane take this opportunity to meet her.

Also, with whatever time we have remaining we will try a little different method of answering your questions. The plan is to have you write your question on a 3 x 5 card (one question per card, please) and put it in the box. Our M.C. will read the questions and solicit answers from the floor. The cards and box will be available at the door, or bring your own card and put it in the box. G.A.H.

* * * * * * COLOR BLEND * * * * * *

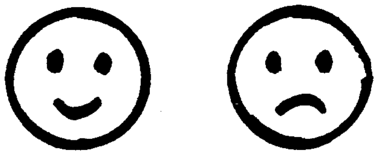
If you want pastel colors in your programs, make every other dot in your CHAR a one or a zero and then call the background color to be white (16). The program below will change the cyan color to a pastel shade

```
90 CALL SCREEN(16)
100 CALL COLOR(1,8,16)
110 CALL CHAR(32,"55AA55AA55AA55AA")
120 CALL CLEAR
130 GOTO 130
```

Try also 14, 12, 10, and 2 as the second number in line 100 for other colors. John Johnson, (Cedar Valey 99'ers Users Group)

COMPUTER BRIDGE (VOLUME 5, NUMBER 5, MAY 1986) PAGE 5

KIDS KORNER



```
   1 REM  *** HAPPY-FACE ADDITION
   2 REM     A KID'S MATH PROGRAM
   3 REM  USING 2 SUB-PROGRAMS BY
   4 REM  ***  FRED D'IGNACIO  **
   5 REM
   6 REM      By Chick De Marti
   7 REM
   8 REM  ************************
100 CALL CLEAR
110 R1=INT(RND*9)+1
120 R2=INT(RND*9)+1
130 PRINT "    ";R1;" AND";R2;" =":: 
140 INPUT ANS
150 IF ANS=R1+R2 THEN 160 ELSE 180
160 GOSUB 1000
170 GOTO 190
180 GOSUB 2000
190 PRINT :"WANT TO TRY AGAIN? (Y/N)";
200 INPUT AN$
210 IF SEG$(AN$,1,1)="Y" THEN 100
220 CALL CLEAR
230 END
1000 MSG$="ALL RIGHT!"
1005 T=10
1010 CALL CHAR(129,"071F3F7F73F3FFFF")
1020 CALL CHAR(130,"E0F8FCFECECFFFFF")
1030 CALL CHAR(131,"FFFFEF777B3C1F07")
1040 CALL CHAR(132,"FFFFF7EEDE3CF8E0")
1050 FOR ROLL=1 TO 6
1060 PRINT
1070 NEXT ROLL
1080 CALL HCHAR(20,15,129)
1090 CALL HCHAR(20,16,130)
1100 CALL HCHAR(21,15,131)
1110 CALL HCHAR(21,16,132)
1120 L=LEN(MSG$)
1130 REM
1140 FOR I=1 TO L
1150 CALL HCHAR(22,T+I,ASC(SEG$(MSG$,I,1)))
1160 NEXT I
1170 IF ANS<>R1+R2 THEN 2060
1180 FOR X=1 TO 3
1190 FOR Q=600 TO 1000 STEP 50
1200 CALL SOUND(100,Q,10)
1210 NEXT Q
1220 NEXT X
1230 PRINT
1240 RETURN
2000 MSG$="OH, OH ... SORRY!"
2005 T=8
2010 CALL CHAR(129,"071F3F7F73F3FFFF")
2020 CALL CHAR(130,"E0F8FCFECECFFFFF")
2030 CALL CHAR(131,"FFFFFC7B773F1F07")
2040 CALL CHAR(132,"FFFF3FDEEEFCF8E0")
2050 GOTO 1050
2060 CALL SOUND(500,311,8)
2070 CALL SOUND(1000,131,8)
2080 GOTO 1230
```

BE GOOFY
STAY AWAY FROM
TI USERS GROUP
MEETINGS

| This suggestion is from the |
| NORTHWEST OHIO User's Group |

LA99er LIBRARY CORNER

NEW ADDS FOR JULY

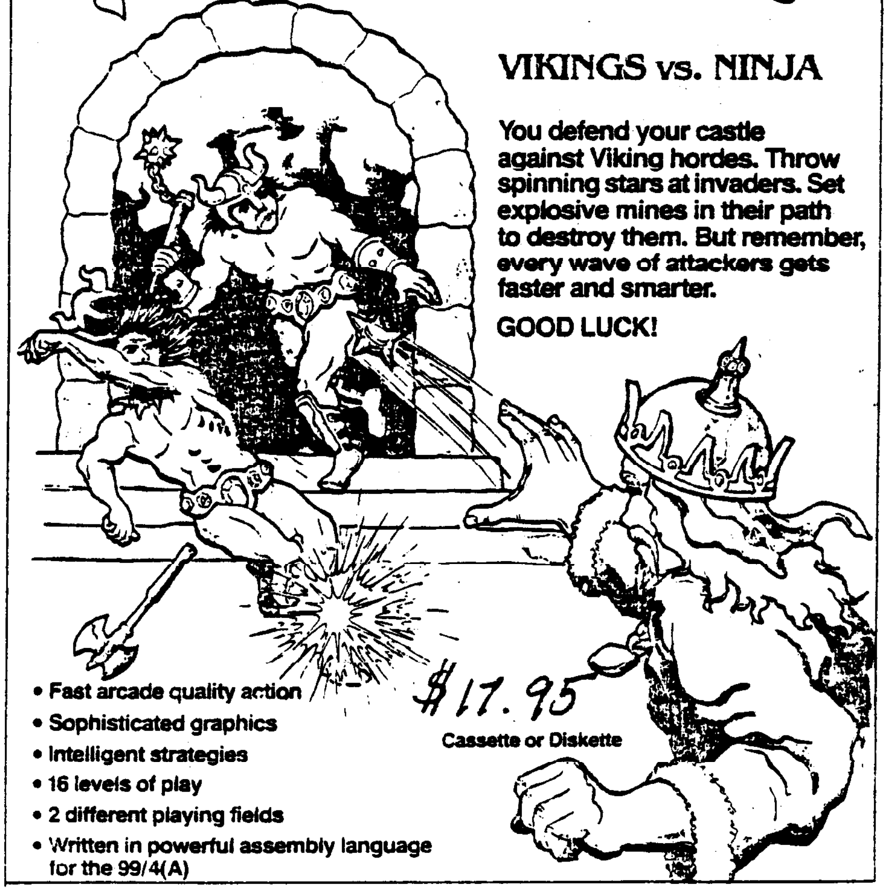
S/N	NAME	T	C	SEC	DON	CYAR	REMARKS
2006	4TH TUTORIAL	W	N	574	GLS	A8	$5.00 TEXT,USE TI WRITER,35 PAGES OF FORTH TUTORIAL. WELL WRITTEN
2091	PRINT ART RLE 3	A	N	346	RLE	CY97	$5.00 E/A-BILLCROBY,BLOO/MILO,BUGBUNNY,MAMMOTH,RUINS,SCROOGE,WCFIELD,WARGIRL
2092	PRINT ART RLE 4	A	N	225	RLE	CYA7	$5.00 E/A-BAMBI,BISSET,KAILOP,SGTMAJ AND OTHER-EXCELLENT DETAIL TO PRINTER
2093	MUSIC #10	E	N	358	KG	CYA7	$5.00,X/B,COLOR,GRAPHIC,SPEECH-10 NEW MUSICAL PROGRAMS BY KEN GILLILAND
2094	MUSIC #11	E	N	325	KG	CYA7	$5.00,X/B,COLOR,GRAPHIC,SPEECH-7 MORE NEW MUSICAL PROGRAMS BY KEN GILLIAND.
4083	2D GRAPHICS	F	F	712	JPP	YA8	$4.00 FREEWARE BY JEAN PIERRE MORIN-TI LOGO IMPLEMENTATION WRITTEN IN FORTH.
4084	MATH	E	F	42	JZ	A6	$2.00 FREEWARE BY JOE ZEFF-MATH LOGIC FUNTION FOR C99,MATH & LOG FUNTIONS
4085	TECHIE	E	F	353	OUG	A8	$2.00 FREEWARE BY MONTY SCHMIDT-A SBS PROGRAM-MODEM SETUP,ACCESS AND DOWNLOAD
4086	SUPER SHAPE	E	F	29	SS	A5	$2.00 FREEWARE STOCK SOFTWARE-EDIT SHAPES(CHARACHTERSS) TO PRINTER AND DISK
4087	ML UTILITY	E	F	185	OUG	A7	$2.00 FREEWARE BY ART GREEN-5 ASSEMBLY UTILITY,5 DISK UTILIT[illegible] & PRINT ART.

DISK OF THE MONTH FOR JULY

** KING of the CASTLE **

$2,50

KING of the CASTLE



VIKINGS vs. NINJA

You defend your castle against Viking hordes. Throw spinning stars at invaders. Set explosive mines in their path to destroy them. But remember, every wave of attackers gets faster and smarter.

GOOD LUCK!

$17.95

Cassette or Diskette

- Fast arcade quality action
- Sophisticated graphics
- Intelligent strategies
- 16 levels of play
- 2 different playing fields
- Written in powerful assembly language for the 99/4(A)

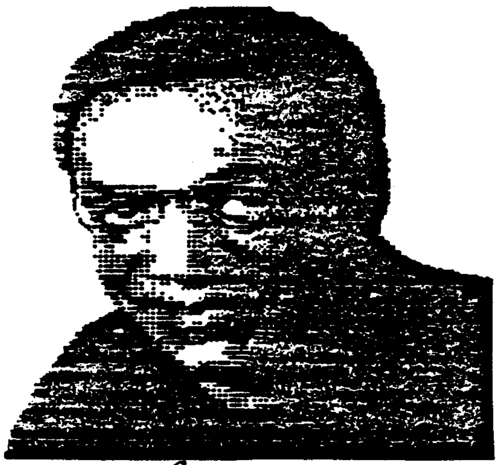
MARKETPLACE

(the marketplace is a fund raiser for the club, that is, the "profit" goes to maintain the quality of this News-letter. In general the price listed splits the difference between cost and retail. Please help your Club.)

MILLERS GRAPHICS	
DISKASSEMBLER	18.50
ORPHAN CHRONICLES (priceless)	9.95
ADVANCED DIAGNOSTICS	18.50
EXPLORER	22.50
NIGHT MISSION	18.50
GRAM KRACKER (80K EXPANDED)	185.00
SMART PROGRAMING FOR SPRITES	6.25
MYARC	
RS232	82.00
D/D DISK CONTROLLER	155.00
128K RAM DISK/SPOOLER	175.00
512K RAM DISK/SPOOLER	280.00
EXTENDED BASIC II LEVEL IV	80.00
128K RAM DISK W/XBASIC II	235.00
512K RAM DISK W/XBASIC II	340.00
MEGATRONICS	
EXTENDED BASIC II PLUS	72.50
INTERN (BOOK ON GPL)	16.50
128K GRAM CARD	227.50
HARDWARE & SUPPLIES	
TEAC 55BV DSDD DRIVES	110.00
DISKETTES DSDD	1.00
64K EPSON INT. PRINT BUFFER	45.00
COLOR RIBBONS (EPSON)	4.00
BACK ISSUES	
SUPER 99 MONTHLY	1.25
MICROPENDIUM	1.25
SMART PROGRAMMER JUNE 1986	1.50
BEST OF NEWSLETTERS W/DISK	5.00
FORTH NOTES VOL 1-5 (2.50 EA)	10.00
BEGINNER'S FORTH NOTEBOOK	2.50
ASSEMBLY NOTES VOL 1	2.50
TECHNICAL AND BUSINESS BOOKS	5.00
SAMS BOOKS (VARIOUS)	5.00
SAMS BOOKS WITH CASSETTES	7.50

HORIZON RAM DISK

We are now taking orders for this new Ram Disk. It is available in several formats, both assembled and kit form. Group purchase will enable us to get a discount. Please advise your intent



NEW IN OUR LIBRARY 2 SPECIAL DISKETTES OF GREAT RLE PICTURES CHECK WITH FRED

CLUB OFFICERS

President:		Membership Chairman:	
Terrie Masters	(213) 271-6930	Tom Freeman:	(213) 454-1943
Vice President:		Librarian:	
George F. Steffen	(213) 329-3527	Fred Moore	(213) 670-4293
Secretary		Library Assistant:	
Terry Wilson	(213) 563-3869	Chick De Marti	(213) 532-8499
Treasurer		Library Assistant:	
Margaret H. Hutton	(213) 541-5339	Allen S. Whiteman	(213) 379-8031
Editor:		Equipment Chairman:	
Terrie Masters	(213) 271-6930	Joe Fierstain:	(213) 377-9834

Topics solicits advertisers at the following rates:

Full page:	$20.00
Half page:	12.00
Quarter page:	8.00

All submitted advertisements must be camera ready.

JULY 1986

MEETING

Wednesday JULY 23rd

BRING A FRIEND

CLUB MEETINGS

Los Angeles 99er Computer Group: Fourth Wednesday of each month, 7:15 PM, at Torrance Library, 3031 Torrance Blvd., Torrance.

Pomona Valley 99ers Computer Group: Second Tuesday of each month, 7:00 PM, at Cable Airport Cafe, 13th & Benson, Upland. Call Joy Warner, 982-9971, nights.

San Fernando Valley 99er Computer Group: Second Tuesday of each month, 7:30 PM, Doctors' Conference Room, Sherman Oaks Community Hospital, 4929 Van Nuys Blvd., Sherman Oaks.

San Gabriel Valley 99/4 Users Group: First Wednesday of each month at West Covina Public Library, 1601 W. Covina Parkway, West Covina.

Users Group of Orange County: Third Thursday of each month, 7:30 PM at Westchester Community Service Center (1 block east of Beach Blvd.), Jackson and Westminster, Westminster, CA.

Executive Board meeting of the Los Angeles 99ers is held at 7:15 PM, the first Tuesday of each month at Merit Savings Bank, 18501 S. Western Ave., Torrance. ** ALL MEMBERS ARE INVITED **

The first line on the address label shows the last issue you will receive for members or the last issue received by us for exchanges.

Board Meetings for the LA 99ers is held on the 1st Tuesday of each month at MERIT SAVINGS at 18501 Western Ave. Gardena The meeting starts at 7:15. *** EVERYONE IS INVITED ***

LA 99ers Computer Group
P.O. Box 3547
Gardena, CA 90247-7247

99'Fest-wesT'86

MARINA D[illegible] CA 902[illegible]
PM
7 JUL
1986

FIRST CLASS

Exchng * May 86
Miami County Area 99/4A U.G.
P.O. Box 1194
Peru, IN 46970