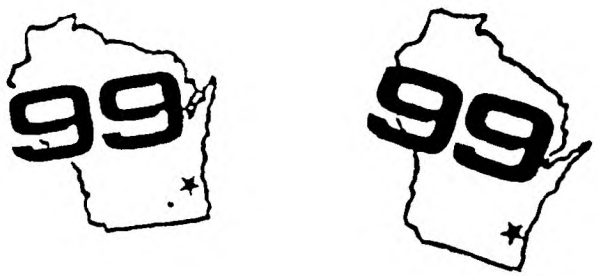
# HOCUS

Home Computer
Users Spotlight

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




## FEBRUARY-1989

MILWAUKEE AREA [illegible] GROUP
4122 GLENWAY WAUWATOSA WI 53222

President...D.Walden 5292173
Vice-Pres...J.Schroeder 2644735
Treasurer...P.Norton 4628954
Secretary...B.Kling [illegible]
Librarian...T.Moe 13923279
Librarian...F.Pabian [illegible]
Newsletter..G.Hitz [illegible]
S.I.G.....Schroeder/Walden/Hitz

Next Group Meeting - 2rd Saturday
March 11, 1989 - 12 noon t 4 PM
Wauwatosa S & L - 7500 West State

North Sub-Meeting - 1st Tuesday
April 11, 1989 - 7 PM til 1 PM
Security S & L - 5555 N Pt Washington

### NOTICE - No North Sub-meeting on Tuesday March 14, 1989

South Sub-Meeting - 3rd Tuesday
February 21, 1989 - 7 PM til 10 PM
Franklin State Bank - 7000 So 76th

Membership Dues $10 - Family $15

---

<<<<< HOCUS NEWSLETTER INDEX >>>>>

---

TI Runner CHEAT!
Chick De Marti ... 01

Relational Expressions
Jim Peterson ... 02

Disk Fix (recover blown disks)
Wesley R. Richardson ... 04

Glitched Program Recover
K.C.S. ... 06

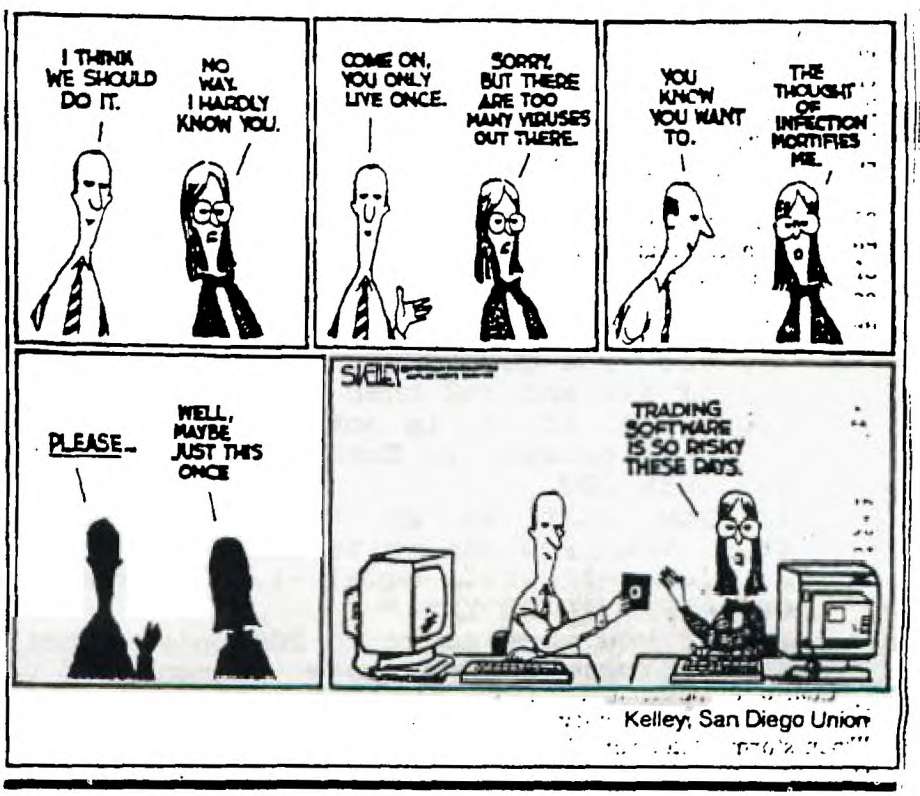
Tips From Tigercub # 52
Jim Peterson ... 06

VARNING Alles Dummkopfen!
Der Komputeroperator ... 08

## CHEAT FOR TI-RUNNER 'TRULY SICK SOFTWARE'

Assemble this program and LOAD & RUN it just before you LOAD TI-RUNNER. Although it has no visible effect right away you will notice some <heh heh!> 'changes' as you play...

```
        AORG >FF00
INC     DATA 1
BAS     DATA 1200
V01     ORI  R0,>4000
V02     SWPB R0
        MOVB R0,@>8C02
        SWPB R0
        MOVB R0,@>8C02
        ANDI R0,>3FFF
        RT
V03     DEC  @INC
        JEQ  12
        RT
V04     MOV  R11,R3
        CLR  R0
        CLR  R1
        LI   R2,>6000
        BL   @V02
V05     CI   R0,767
        JGT  V07
        MOVB @8800,R1
        CI   R1,>7800
        JUE  V06
        BL   @V01
        MOVB R2,@8C00
        INC  R0
        BL   @V02
        JMP  V05
V06     INC  R0
        JMP  V05
V07     MOV  @BAS,@INC
        B    *R3
        AORG >83C4
        DATA V03
        END
```



# THE POWER OF RELATIONAL EXPRESSIONS

by Jim Peterson

What the h... are those, you say? You may well ask. The "blue book" that came with your computer says nothing about them, and most of the programming tutorial books on the subject are equally silent. If you waded through the computerese and mathematese text of the User's Reference Guide, you found them discussed on page II-14 under Relational Expressions and on page II-51 under IF-THEN-ELSE, but you probably didn't realize their potential. Then, you graduated to Extended Basic and found those easy-to-use, in-the-clear logical expressions AND, OR, NOT and XOR, and you looked no farther.

So, what can a relational expression do? Nothing that can't be done without it. But it can often do the job so much more compactly, so much more efficiently, and therefore so much faster!

So, let's learn to use them. And let's learn in plain English, not computerese. The following may not be technically correct, but it's the way it all works out.

First, every expression has a true/false value, which is entirely different and separate from the value of the variables or numbers or strings it contains. On the TI-99/4A, a false statement has a value of Ø, which is easy to remember - A FALSEHOOD IS WORTH NOTHING. Unfortunately, a true statement has a value of -1, which doesn't fit in too well! On some other computer you may have learned that a true expression has a value of +1, but on the TI it's -1.

So, in ...F=7 :: IF F=8 THEN...., F=7 has a value of -1 because obviously F does equal 7, and F=8 has a value of Ø because it is not true.

Second, when an IF refers to a variable without an "=" sign, it means "<>Ø". For instance, IF X THEN 1ØØØ means "if X is more or less than Ø, if it is not Ø, if it is anything other than Ø, then go to 1ØØØ".

Third, the computer will try to use the expression mathematically before it tries to interpret its true/false value. Remember that everything within parentheses is worked first. For instance...X=1 :: Y=2 :: IF (X=1)+(Y=2) THEN 1ØØØ...Since both are true, this works out to IF (-1)+(-1)<>Ø THEN 1ØØØ, and since -1 plus -1 is not Ø, we go to 1ØØØ. On the other hand, X=1 :: Y=2 :: IF X=1+Y=2 THEN 1ØØØ will first be calculated as X=1+Y, which comes out as X=3, and then as X=3=2, which has a true/false value of Ø (false) because X=3 has a true/false value of Ø (false), not 2!

Finally, always remember that a variable keeps its previous value until the calculation of an entire equation is completed. X=3 :: X=X+(X+3)*X-X/X X+(X=Ø) is worked as X=3+(3+3)*3-3/3 3+(3=Ø).

Now that you have assimilated this vast knowledge, how can it be used? The most common way is in the expression IF (X=1)+(Y=2) THEN 2ØØ. In this case, if it is true that X=1 but Y does not equal 2, then -1+Ø is <>Ø so you go to 2ØØ. If X is not 1 but Y=2, then Ø+-1 is still <>Ø, and if X=1 and Y=2 then -1 plus -1 is still <>Ø, so you still go to 2ØØ, but if X is not 1 and Y is not 2 then Ø+Ø is not <>Ø so you do not. Of course, in Extended Basic, you could simply write IF X=1 OR Y=2 THEN 2ØØ.

If you want to go to 2ØØ only if X=1 or if Y=2 but not if both are true, then you can write IF (X=1)+(Y=2)=-1 because either -1 plus Ø or Ø plus -1 will equal -1. In Extended Basic, this is the "exclusive OR", IF X=1 XOR Y=2.

And if you want to go to 2ØØ only if both are true, you can write IF (X=1)+(Y=2)=-2, or more commonly IF (X=1)*(Y=2) because if either or both are not true the multiplication by Ø will give Ø. In Extended

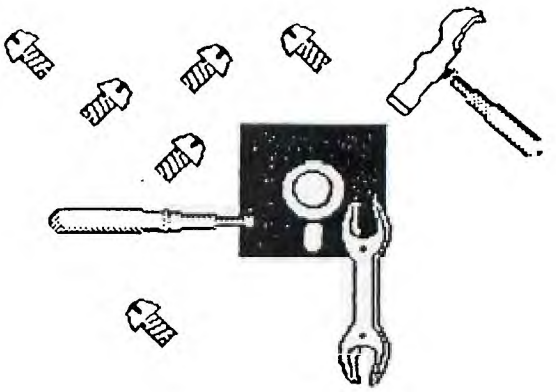
Basic, this is IF X=1 AND Y=2 .
And you can write more complicated versions, carefully watching your parentheses, such as IF (X=1)+((Y=2)*(Z=3)) which translates to IF X=1 OR Y=2 AND Z=3.
So, if you're programming in Extended Basic, why bother with all those parentheses? Why not just use OR and AND? In the above cases, that is true. But you have not yet begun to see the power of relational expressions!
Since the true/false value is a numeric value, it can be used in calculations, and it does not have to be used with an IF statement.
For instance, this is a statement that I have used within a loop to alternate control of the two joysticks between two players....X=X+1+(X=2)*2 :: CALL JOYSTICK(X,Y,Z) . In this, the first time around, X has not been given a value, so the equation is read X=Ø+1+(Ø=2)*2 and, since Ø does not equal 2, Ø+1+(Ø*2)=1 and joystick #1 is activated. Next time around, X=1 and X=1+1+(1=2)*2 gives X a value of 2, since 1=2 has a true/false value of Ø. The 3rd time around, X now has a value of 2, and X=2+1+(X=2)*2 which is worked as X=2+1+(-1)*2 and then X=2+1+(-2) which is X=2+1-2 and X=1 again!
If you think that's neat, look at this one from the Airport Area UG newsletter, credited to Robert Cooley - X=X=Ø :: CALL JOYST(X+2,Y,Z). Here, the first time around, X does equal Ø so the statement X=Ø has a true/false value of -1 so X=-1 and X+2 activates joystick #1. Then X=-1 so X=Ø has a true/false value of Ø so X=Ø so X+2 activates joystick #2...and so on! Of course, you could also write IF X=1 THEN X=2 ELSE X=1 if you prefer.
Another example: A=INT(1Ø*RND):: B=INT(1Ø*RND):: FOR J=A TO B ...Now, if the random B happens to be smaller than the random A, the loop falls through with nothing happening. You could add a line IF A>B THEN T=1 ELSE T=-1 and FOR J=A TO B STEP T . But why not just FOR A TO B STEP (B<=A)+ABS(A<=B) . If B<A then -1+ABS(Ø) gives a STEP -1 to count backwards, but if A<B then Ø+ABS(-1) gives STEP 1, and if A+B then Ø+ABS(Ø) equals STEP Ø! Here's another example - 1ØØ INPUT "SCREEN COLOR? ":S :: FOR SET=1 TO 14 :: X=SET+1-(SET>=S):: CALL COLOR(SET,X,X):: NEXT SET . That changes the character sets to colors 2 to 16 in sequence, skipping over whatever color has been selected for the screen.
Strings can also be manipulated. 1ØØ P$(1)="S" 11Ø INPUT "HOW MANY? ":N :: PRINT "THE PRICE IS "&STR$(n)&" DOLLAR"&P$(ABS(N>1)):: GOTO 11Ø . Or, more efficiently 1ØØ INPUT "HOW MANY? ":N :: PRINT "THE PRICE IS "&STR$(N)&SEG$(" DOLLARS",1,7-(N>1)):: GOTO 1ØØ - or, how about using STR$(N)&"DOLLAR"&CHR$((N<>1)*- 83)? If N<>1 then (-1)*-83 gives CHR$(83), which is "S", otherwise Ø*-83 gives CHR$(Ø) which is a blank.
However, it is also possible to overdo it. The following routine will read key input to move the cursor around the screen in all 8 directions, stopping at the borders or travelling along them if struck diagonally. However, it requires so many calculations for each key input that it is not the fastest method for accomplishing this.

```
1ØØ CALL CLEAR :: R=1 :: C=3
11Ø CALL KEY(3,K,ST):: IF ST=Ø THEN 11Ø
12Ø C=C+((K=82)+(K=68)+(K=67))*(C<32)-((K=87)+(K=83)+(K=9Ø))*(C>2)
13Ø R=R+((K=9Ø)+(K=88)+(K=67))*(R<24)-((K=87)+(K=69)+(K=82))*(R>1)
14Ø CALL HCHAR(R,C,42):: GOTO 11Ø
```

So - for compact, efficient programming, learn to use the relational expressions! But also learn when not to use them!

# Disk Fix

by WESLEY R. RICHARDSON
BLUEGRASS 99 COMPUTER SOCIETY, INC.



When you have a disk with several files that you have been working on and you do a catalog and it comes up DISKETTE IS BLANK, or DISK NOT INITIALIZED, it can be very frustrating. There are times when the sectors used and available get changed to values like 2389 free and 7887 used, but you know you have a single sided, single density (SSSD) disk drive, with a maximum of 360 sectors. It is also possible to have a disk which will not catalog, yet when Extended BASIC is selected, the disk will run the LOAD program and continue without a problem. These have happened to me and I am sure it has happened to others, so I thought I would document a way which may recover your disk for you.

The items which you will need are your blown disk, two blank disks, Disk Manager 1000 v3.5, Disko or Disk Patch, and a sector or track copier program, or the equivalent of any of the above. I will use the Funnelweb v4.10 DISK-PATCH for the sector editor.

1) The first step is to initialize a disk in the format which you believe the blown disk was, for example SSSD. For the disk name, use the name that you want on the blown disk after it is restored.

2) Using the sector copier or track copier, make a copy of the blown disk. If you get a read error in sector 0, just tell the program to ignore the error. If you are unable to copy the disk with the copier programs which you have available, you may still continue the following steps with the original disk, but be advised that you may lose everything on the disk.

3) Load DISK-PATCH or DISKO and then insert the back-up copy of the blown disk in drive 1. Select option 1 for disk sector editor. Then disk 1, and sector 0. The screen should come up with the data from sector 0. Pressing FCTN 2 will change the screen to ASCII and pressing FCTN 1 will change it to HEX. In ASCII, the first ten characters will be the disk name. In HEX, at byte 12h (h=HEXADECIMAL) will be 01 for single sided and 02 for double sided. At byte 13h, will be 01 for single density and 02 for double density.

4) Press FCTN 4 to go to sector 001h. You should find groups of four digits of HEX numbers such as 0002 0003 0009 0015 and so on. These indicate where the file names and file maps may be found. Write down each of these numbers in the order which they are found when read from left to right and top to bottom on the screen. Note also if the first number is 0000, then the disk will catalog as being blank and no file names will appear.

5) Press FCTN 4 to go to sector 002h. In the first ten ASCII characters you will find a file name. Write this down next to the appropriate four digit number you had in step 4). Do this for each of the numbers from step 4). If there were several files on the disk, you may need to press FCTN 9 and then option 1 again to go directly to the location. While in sector edit mode, pressing FCTN 6 will take you to the next lower numbered sector.

6) You now should have a table similar to the one below with the file name and location of each file on the disk.

| | | | |
|---|---|---|---|
| 0000 | A-SECTOR2 | 000D | PACMAO |
| 0003 | CENTIPEDE | 0005 | PINBALL |
| 0009 | DEFENDER | 0006 | PINBALM |
| 000A | KONG | 0007 | POLE/POS |
| 000B | KONH | 0008 | POLE/POT |
| 0004 | LOAD | 000E | TI/INVADER |
| 000C | PACMAN | 000F | TI/INVADES |

7) Note in the case that we did find a 0000 but a file was there, as in this case file A-SECTOR2 directory was located at sector 002h, then use the sector editor to view sector 001h. Move the cursor to the first 0000 in HEX and change it to read 0002. Then press CTRL W to write the sector back to the disk, and answer Y to the question RE-WRITE SECTOR?

...DISK FIX

8) Remove the copy of the blown disk and insert the formatted blank disk in drive 1. Select the sector editor, giving drive 1 and sector 0. After the sector comes up, remove the blank disk and insert the blown disk copy in drive 1. Press CTRL W to rewrite the sector.

9) Load Disk Manager 1000 version 3.5 (DM1000), and then put the blown copy disk back in drive 1. Select option 1, File Utilities. Then select option 2 for Recover file. Give the drive as 1. Enter the first file name on you list and press enter. The program will say SEARCHING DISK, then RE-BUILDING LOST FILE, then FILE RECOVERED. Press enter and then 2 for Recover file. Repeat these steps until all of the files are recovered.

10) Press 1 for Copy/Move/Delete... and give the disk number as 1. Your disk files should now be restored. If the disk free and used does not match up with the sum of the file sizes plus 2 sectors, then go to step 11), otherwise you are done.

11) Do this step only if the disk free is not correct. Place a D in the left column to delete all of the files and a U in the right column to unprotect all of the files. DM100D will unprotect and then delete all of the files. At this point a catalog should show free 358, used 2 for a SSSD disk. Go back to the recover file section of step 9) and recover each file again.

One other piece of advise, if you have a disk with a bad directory, do not write any files to the disk until you have a chance to fix the directory. If you write a new file, then you are taking the chance that part of another file will be over-written. This can happen because sector D may show that a location is free, when in fact it has part of a file in it.

The other advise is to always keep a back-up copy of anything which you do not want to lose. It is a good idea to keep a write protect tab on your master disk and keep it away from your work disk. On documents or programs, save your work to disk every 15 minutes so if the power goes off or your computer locks up, you only lose 15 minutes worth of work. Alternate saving to two disks when you have a large and important program or file.

If you always keep back-ups, I hope you will not need to use DISK-FIX, but if that time comes when the disk is blown, now you have something to try.

## FIXING GLITCHED XBASIC PROGRAMS AND DV/80 FILES:

Got an adventure graphics game on disk at our last club meeting. After playing through several screens the next one to load stopped with a syntax error. Listing the program showed several lines of code to be glitched. Trying to edit out the glitched code caused the screen to change from blue to red and then lock up the computer. Not wanting to wait a month for the next club meeting to exchange the disk I decided to experiment. First copied the disk with Jim Schroeder's REDISKIT. The program on the original disk would not even load because of a bad sector. Next saved the program to disk with the command LIST "DSKx.Filename". This DV/80 file must next be printed to disk with the TI-Formatter. It will not load into the Editor after listing because the file still has the glitches in it. Next load the formatter file into the Editor and delete the glitched lines and print back to disk with the command "C DSKx.Filename" to remove linefeed symbols put in by the formatter. If you are lucky to have a printout of the program before it got glitched it will be easy to add the missing code and the convert it back to program format with a DV/80 to program conversion utility. In my case the next screen to load after this one had identical code except for a few lines that were different, so I added the lines and thus reconstructed the glitched program. If neither of the above options are available you could try guessing at the missing code. Of course if you knew of someone else that bought the same disk and had a modem, he could send a replacement for the bad file to you, but that is not much of a challenge. The above proceedure will also work for glitched DV/80 message files from BBS. This is a lot easier as most of the above steps can be eliminated. Sometimes just printing the glitched file from the formatter to printer is all that is neccessery if you do not wish to save the file for later use. Have fun....KCS

## STAR MICRONICS NX-1000 TI Compatibility

Norman Goldberg

Star Micronics printer NX-1000 model V1.3 is apparently compatible with our TI99/4A, however the current models stamped V1.4 or V1.5 are not. Contact the manufacturer for the EPROM necessary. Latest EPROM labeled "V1.5 LFT I" should replace V1.4 or V1.5

TIPS FROM THE TIGERCUB

#52



TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

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

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

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

NUTS & BOLTS DISKS

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

TIPS FROM THE TIGERCUB

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

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

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

This one should come in handy for bowling league captains and Little League coaches.

```
100 DIM M(29,29),T$(30)
110 GOTO 130
120 N;O$;J;I;X;P$;S$;K
130 !@P-
140 DISPLAY AT(3,7)ERASE ALL
:"LEAGUE SCHEDULER";;;"by th
e Burwells              adapt
ed by Tigercub"
150 DISPLAY AT(8,1):" This p
rogram sets up a":"schedule
for up to 30 teams":"so that
 each plays each":"other onc
e and only once."
160 DISPLAY AT(12,1):" If an
 odd number of teams":"are s
cheduled, each gets one":"by
e."
170 DISPLAY AT(16,1):"Number
 of teams?" :: ACCEPT AT(16,
18)VALIDATE(DIGIT):N :: IF N
>30 THEN DISPLAY AT(18,1):"L
IMIT OF 30!" :: GOTO 170
180 DISPLAY AT(18,1)ERASE AL
L:"Schedule teams by name? Y
" :: ACCEPT AT(18,25)SIZE(-1
)VALIDATE("YN"):O$ :: IF O$=
"N" THEN 200
190 FOR J=1 TO N :: DISPLAY
AT(20,1):"Team no.";J;"name?
" :: ACCEPT AT(22,1):T$(J)::
 NEXT J :: GOTO 210
200 FOR J=1 TO N :: T$(J)="T
eam No. "&STR$(J):: NEXT J
210 IF N/2<>INT(N/2)THEN N=N
+1 :: T$(N)="bye"
220 DISPLAY AT(23,1):"Schedu
le by day, week, month":"or
what?" :: ACCEPT AT(24,10):S
$ :: FOR J=1 TO N-1 :: M(1,J
)=J+1
230 NEXT J :: FOR J=1 TO N-1
 STEP 2 :: GOSUB 260
240 NEXT J :: FOR J=2 TO N-2
 STEP 2 :: GOSUB 330
250 NEXT J :: GOSUB 390 :: S
TOP
260 FOR I=1 TO N-2 :: IF M(I
,J)=N THEN 280
270 M(I+1,J)=M(I,J)+1 :: GOT
O 290
280 M(I+1,J)=M(I,J):: GOTO 3
00
290 NEXT I
300 X=I+1 :: FOR I=X TO N-2
:: M(I+1,J)=M(I,J)-1
310 NEXT I
320 RETURN
330 FOR I=1 TO N-2 :: IF M(I
,J)=2 THEN 350
340 M(I+1,J)=M(I,J)-1 :: GOT
O 360
350 M(I+1,J)=M(I,J):: GOTO 3
70
360 NEXT I
370 X=I+1 :: FOR I=X TO N-2
:: M(I+1,J)=M(I,J)+1
380 NEXT I :: RETURN
390 DISPLAY AT(12,1)ERASE AL
L:"Output to - 2";;;" (1) Sc
reen":" (2) Printer" :: ACCE
PT AT(12,13)SIZE(-1)VALIDATE
("12"):K :: IF K=1 THEN 440
400 DISPLAY AT(18,1):"Printe
r? PIO" :: ACCEPT AT(18,10)S
IZE(-18):P$ :: OPEN #1:P$ ::
 PRINT #1:"LEAGUE SCHEDULE":
: :: FOR I=1 TO N-1 :: PRIN
T #1:S$;" #";I :: PRINT #1:T
$(1);" vs ";T$(M(I,1))
410 FOR J=2 TO N-2 STEP 2 ::
 PRINT #1:T$(M(I,J));" vs ";
T$(M(I,J+1))
420 NEXT J :: PRINT #1:"": :
430 NEXT I :: RETURN
440 FOR I=1 TO N-1 :: PRINT
TAB(7);"LEAGUE SCHEDULE": :
:: PRINT "WEEK #";I: : :: PR
INT T$(1);" vs ";T$(M(I,1)):
: FOR J=2 TO N-2 STEP 2 :: P
RINT T$(M(I,J));" vs ";T$(M(
I,J+1))
450 NEXT J :: PRINT "": : ::
 PRINT "PRESS ANY KEY FOR NE
XT WEEK"
460 CALL KEY(0,K,S):: IF S=0
 THEN 460
470 CALL CLEAR
480 NEXT I :: RETURN :: END
```

Some folks seem to think that the subprograms on my Nuts & Bolts disks are just flashy screen displays. Not so! This one will be on the next diskfull, if I ever get it full, which is most unlikely.
ACCEPT AT with a negative

size is useful to accept a default string from the screen, but the length of the string is limited to 28 characters; and if you want something other than the default, you must be sure to delete any extra characters.
CALL DEFAULT(R,C,M$,R$),
where R and C are the row and column to accept at, M$ is the default string which can be up to 254 characters long, and R$ is the string accepted, will display the default string, accept it if Enter is pressed, or accept any other string without having to blank out the extra characters. Just don't type too fast!

```
100 M$="TESTING" :: CALL CLE
AR
110 CALL DEFAULT(12,1,M$,R$)
:: DISPLAY AT(24,1):R$ :: GO
TO 110
10000 SUB DEFAULT(R,C,M$,R$)
:: R$="" :: X=ASC(M$)
10001 DISPLAY AT(R,C):M$
10002 CALL HCHAR(R,C+2,ASC(S
EG$(M$,1,1))):: CALL HCHAR(R
,C+2,30)
10003 CALL KEY(0,K,S):: IF S
=0 THEN 10002 ELSE IF K=13 T
HEN R$=M$ :: SUBEXIT ELSE DI
SPLAY AT(R,C):CHR$(K):: ACCE
PT AT(R,C+1):R$ :: R$=CHR$(K
)&R$
10004 SUBEND
```

CALL DEFAULT(R,C,N,RN), with N as the default value and RN as the value accepted, will do the same for numeric input, and will reject any non-numeric input. Errors due to fast typing can be prevented by omitting the DISPLAY AT(R,C):CHR$(K) in line 1002.

```
100 N=176453.897 :: CALL CLE
AR
110 CALL DEFAULTN(12,1,N,RN)
:: DISPLAY AT(24,1):RN :: GO
TO 9999
10000 SUB DEFAULTN(R,C,N,RN)
:: DISPLAY AT(R,C):N :: N$=S
EG$(STR$(N),1,1)
10001 CALL HCHAR(R,C+2,ASC(N
$)):: CALL HCHAR(R,C+2,30)
10002 CALL KEY(0,K,S):: IF S
=0 THEN 10001 ELSE IF K=13 T
HEN RN=N :: SUBEXIT ELSE DIS
PLAY AT(R,C):CHR$(K):: ACCEP
T AT(R,C+1):R$ :: R$=CHR$(K)
&R$
10003 ON ERROR 10004 :: RN=V
AL(R$):: GOTO 10005
10004 CALL SOUND(200,110,5,-
4,5):: DISPLAY AT(R,C):N ::
ON ERROR STOP :: RETURN 1000
2
10005 SUBEND
```

Ed Machonis discovered an easy way to count the words in a TI-Writer file, using TI-Writer itself. Just put in a line before line 0001, with .LM 0;RM 1;FI;PL nnn with nnn being the sector length of the file multiplied by 40. Save it, go into the Formatter and print it to disk under a different filename. Return to Editor, load the resulting file, page through it with FCTN 4 counting any blank lines, subtract the number of blanks from the last line number, and that's it! The Formatter takes about one minute to count 1000 words. If the resulting file is very large, you may have to load it in two sections.

```
100 M$="POS WILL FIND THE FI
RST OCCURRENCE OF A SUBSTRIN
G WITHIN A STRING BUT I OFTE
N NEED TO FIND THE LAST OCCU
RRENCE SO I WROTE THIS SUBPR
OGRAM"
105 INPUT "SUBSTRING?":L$
110 CALL LAST(M$,L$,P):: IF
P=0 THEN PRINT "NOT FOUND" :
: GOTO 105 ELSE PRINT SEG$(M
$,P,255):: GOTO 105
120 SUB LAST(M$,L$,P):: X=1
130 Y=POS(M$,L$,X):: IF Y=0
THEN P=0 :: SUBEXIT ELSE Z=Y
140 X=Y+1 :: Y=POS(M$,L$,X):
: IF Y=0 THEN P=Z :: SUBEXIT
 ELSE Z=Y :: GOTO 140
150 SUBEND
```

Here's a new way to make music. The algorithm in 110 sets up a 3-octave chromatic scale - note the N(1)=F, I have erroneously omitted it when I previously published that algorithm.
To change the key of the music you have programmed, just change the value of F. Lines 190-220 contain the part of the music that is repeated within the melody. A is the subscript of the melody note, B is the subscript number of the chord. These must be above 13, as the frequency is divided by 2 in the subroutine.
Each beat of the music has a GOSUB, to 230 to play a bass accompaniment with the first note of each bar, to 260 for the other notes of the bar. The chord note is divided by different values to play the three notes of the chord in succession, and multiplied by 3.75 in the 3rd voice to produce a bass note two octaves lower in the -4 noise. The melody note is multiplied by 1.01 in the second voice to give a richer tone.

```
100 DISPLAY AT(12,3)ERASE AL
L:"THE MAORI FAREWELL SONG"
! programmed by
  Jim Peterson
110 F=110 :: DIM N(36):: FOR
 J=1 TO 36 :: N(J)=INT(F*1.0
59463094^(J-1)):: NEXT J ::
N(1)=F :: T=-999
120 GOSUB 190 :: A=30 :: B=2
3 :: GOSUB 230 :: GOSUB 260
:: GOSUB 260 :: A=32 :: B=28
 :: GOSUB 230 :: GOSUB 260 :
: GOSUB 260 :: A=28
130 GOSUB 230 :: GOSUB 260 :
: GOSUB 260 :: A=30 :: B=23
:: GOSUB 230 :: GOSUB 260 ::
 A=28 :: GOSUB 260 :: A=27 :
: GOSUB 230 :: GOSUB 260
140 A=28 :: GOSUB 260 :: A=3
0 :: GOSUB 230 :: GOSUB 260
:: GOSUB 260 :: GOSUB 230 ::
 GOSUB 260 :: GOSUB 260 :: G
OSUB 190
150 A=30 :: B=23 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
:: A=32 :: B=16 :: GOSUB 230
 :: GOSUB 260 :: A=28 :: GOS
UB 260
160 A=33 :: B=23 :: GOSUB 23
0 :: GOSUB 260 :: A=32 :: GO
SUB 260 :: A=25 :: B=13 :: G
OSUB 230 :: GOSUB 260 :: GOS
UB 260
170 A=27 :: B=23 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
:: A=28 :: B=16 :: GOSUB 230
 :: GOSUB 260 :: GOSUB 260
180 B=28 :: GOSUB 230 :: GOS
UB 260 :: GOSUB 260 :: B=16
:: GOSUB 230 :: GOSUB 260 ::
 GOSUB 260 :: GOTO 120
190 A=32 :: B=28 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
:: A=28 :: B=16 :: GOSUB 230
 :: GOSUB 260 :: A=30 :: GOS
UB 260
200 A=32 :: B=28 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
:: B=16 :: GOSUB 230 :: GOSU
B 260 :: GOSUB 260 :: B=28 :
: GOSUB 230 :: GOSUB 260
210 A=30 :: GOSUB 260 :: A=3
3 :: B=23 :: GOSUB 230 :: GO
SUB 260 :: A=27 :: GOSUB 260
 :: A=28 :: B=16 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
220 B=28 :: GOSUB 230 :: GOS
UB 260 :: GOSUB 260 :: B=16
:: GOSUB 230 :: GOSUB 260 ::
 GOSUB 260 :: RETURN
230 CALL SOUND(T,N(A),5,N(B)
/1.585,9,N(B)*3.75,30,-4,9):
: GOSUB 290
240 CALL SOUND(T,N(A),5,N(B)
/1.334,9,N(B)*3.75,30,-4,9):
: GOSUB 290
250 CALL SOUND(T,N(A),5,N(B)
/2,9,N(B)*3.75,30,-4,9):: GO
SUB 290 :: RETURN
260 CALL SOUND(T,N(A),5,N(A)
*1.01,5,N(B)/1.585,9):: GOSU
B 290
270 CALL SOUND(T,N(A),5,N(A)
*1.01,5,N(B)/1.334,9):: GOSU
B 290
280 CALL SOUND(T,N(A),5,N(A)
*1.01,5,N(B)/2,9)
290 FOR D=1 TO 20 :: NEXT D
:: RETURN
```

MEMORY FULL.....

Jim Peterson

**********
**********

ACHTUNG VARNING!!

Achtung, Alles Touristen und die Non-Technischen Lookenpeepers! Das Machine Control ist Nicht fur Gerfingerpoken und Mittengrabben. Odervise ist Easy Geschlippen der Springenverken, Fusengeblowen und Corkenpoppen mit Sfpitzensparken. Der Machine ist nur fur Diggen by Experten. Ist Nicht fur Geverken by das ander Dummkopfen. So das Rubbernecken Sightsgeseeners must in das pockets, der Cottonpicken Hands Gekeepen. So Bitte Relaxen und Yust Gevatchen alles von die Lightsgeblinken.

Dankashoen Bitte !!

**********
**********