HOCUS

Home Computer Users Spotlight

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

JANUARY—1988

MILWAUKEE AREA USER GROUP
4122 GLENWAY WAUWATOSA WI 53222

President...D.Walden	5292173
Vice-Pres...J.Schroeder	2644735
Treasurer...P.Norton	4628954
Secretary...B.Kling	5295161
Librarian...E.VonDerEhe	5490590
Librarian...F.Fabian	3272618
Newsletter..G.Hitz	5350133
S.I.G.....Schroeder/Walden/Hitz	

Next Group Meeting
February 13,1988 2nd Saturday
Wauwatosa Savings & Loan
7500 West State Street
12:00 Noon - 4:00 PM

Northside Sub-Meeting
February 2, 1988 1st Tuesday
Security Savings & Loan
5555 North Port Washington
7:00PM - 10:00PM

Southside Sub-Meeting
February 16, 1988 3rd Tuesday
Franklin State Bank
7000 South 76st Street
7:00PM - 10:00PM

Annual Membership Dues
Individual - $10
Family - $15

<<<< HOCUS NEWSLETTER INDEX >>>>

Guess who?

IS HE AT THAT COMPUTER AGAIN?

```
100 DISPLAY AT(3,10)ERASE AL
L:"GO-SEARCH": :TAB(7);"by J
im Peterson"
110 DISPLAY AT(7,1):" For us
e before modifying a":"progr
am.":" Searches a program sa
ved in":"MERGE format, finds
 all":"lines containing a ju
mp,"
120 DISPLAY AT(12,1):"sorts
into 'to' line number":"sequ
ence, outputs to screen":"or
 printer."
130 DATA GOTO,GOSUB,THEN,ELS
E,RUN,RETURN,RESTORE,USING,E
RROR
140 DIM C(200):: FOR J=1 TO
9 :: READ GO$(J):: NEXT J ::
 A=1
150 DISPLAY AT(16,1):"FILENA
ME? DSK" :: ACCEPT AT(16,14)
:F$
160 ON ERROR 170 :: OPEN #1:
"DSK"&F$,INPUT ,VARIABLE 163
 :: GOTO 180
170 DISPLAY AT(18,1):"I/O ER
ROR" :: ON ERROR STOP :: RET
URN 150
180 DISPLAY AT(18,1):"OUTPUT
 TO P":"  (S)creen":"  (P)ri
nter" :: ACCEPT AT(18,11)SIZ
E(-1)VALIDATE("PS"):Q$
190 IF Q$="P" THEN DISPLAY A
T(22,1):"PRINTER? PIO" :: AC
CEPT AT(22,10)SIZE(-18):P$ :
: OPEN #2:P$ :: D=2
200 LINPUT #1:A$
210 IF POS(A$,CHR$(201),3)=0
 THEN 260
220 LN=ASC(SEG$(A$,1,1))*256
+ASC(SEG$(A$,2,1)):: P=3
230 X=POS(A$,CHR$(201),P)::
IF X=0 THEN 260 :: LREF=ASC(
SEG$(A$,X+1,1))*256+ASC(SEG$
(A$,X+2,1)):: S=X-1
240 Z=ASC(SEG$(A$,S,1)):: G=
(POS("13413517612916913614821
37165",STR$(Z),1)+2)/3 :: IF
 Z<100 OR G<1 THEN S=S-1 ::
IF S<3 THEN G=1 ELSE 240
250 C$=STR$(LREF)&"."&STR$(L
N)&STR$(G):: C(A)=VAL(C$)::
A=A+1 :: P=X+3 :: C$="" :: G
OTO 230
260 P=3 :: C$="" :: IF EOF(1
)THEN CLOSE #1 ELSE 200
270 DISPLAY AT(24,1):"SORTIN
G" :: A=A-1 :: CALL LONGSHEL
LN(A,C())
```

IMAGINATIVE PROGRAMMING

by Jim Peterson

Most of the fun (and frustration) of programming is in figuring out a way to do something you haven't done before. Self-taught programmers often devise some highly unconventional methods of solving problems, and this example contains some doozies!

This is a greatly improved version of a program which appeared in Tips from the Tigercub. It is used to make a list of all program lines to which another line jumps, in order to avoid deleting those lines when modifying a program. Since it reads a program saved into a D/V 163 file by SAVE DSK (filename),MERGE it requires a disk drive.

Line 130 lists most of the Extended Basic statements which can direct a jump. In order to keep the array at less than 10 subscripts, for reasons to be explained later, I omitted the rarely used GO, BREAK and UNBREAK.

Line 140 dimensions an array to be used to store the records, and reads the DATA into another array. Line 150 asks for the filename of the MERGE format program to be examined. Note that the ACCEPT places the cursor after DSK, to make it plain that drive number and filename are to be entered. Line 160 opens the file and jumps over the error routine in 170. If the filename is not on the disk, etc., line 170 prints the error message and goes back to give you another chance. ON ERROR STOP prevents an erroneous I/O ERROR message if some later error should occur.

Line 180 obtains your choice of output, with a default of P, and rejects anything other than a single character P or S.

If printer output is selected, line 190 asks for printer designation, opens printer.

Line 200 reads in a program line, using LINPUT rather than INPUT so as to accept any and all characters. ASCII 201 is the token code that always precedes a line number reference, so line 210 searches for ASCII 201, starting at the 3rd character so as not to look for it in the program line number itself. If POS=0 there are no jumps from this line, so to 260.

Else, line 220 converts the tokenized 2-byte line number, of the line being examined, to its decimal form in LN, and sets P to start search for ASCII 201 at the 3rd byte. In line 230, X will be the position of the first ASCII 201, and the 2 bytes following it will be the line number jumped to, which is converted to its decimal equivelant LREF. The search for the statement directing the jump will begin at S, one byte to left of ASCII 201.

```
280 IF D=0 THEN DISPLAY AT(1
2,1)ERASE ALL:"ANY KEY TO PA
USE"
290 FOR J=1 TO A :: A$=STR$(
C(J)):: X=POS(A$,".",1):: Y=
VAL(SEG$(A$,LEN(A$),1)):: A$
=SEG$(A$,1,LEN(A$)-1)
300 PRINT #D:SEG$(A$,1,X-1);
TAB(7);GO$(Y);" FROM ";TAB(2
1);SEG$(A$,X+1,LEN(A$))
310 CALL KEY(0,K,ST):: IF ST
=0 THEN 330
320 CALL KEY(0,K2,ST2):: IF
ST<1 THEN 320
330 NEXT J :: END
340 SUB LONGSHELLN(N,NN())
350 D=N
360 D=INT(D/3)+1 :: FOR I=1
TO N-D :: IF NN(I)<=NN(I+D)T
HEN 390 :: T=NN(I+D):: J=I
370 NN(J+D)=NN(J):: J=J-D ::
 IF J<1 THEN 380 :: IF T<NN(
J)THEN 370
380 NN(J+D)=T
390 NEXT I
400 IF D>1 THEN 360
410 SUBEND
```

In line 240, Z is the ASCII code found at position S. The POS statement lists the 3-digit ASCII codes of the words listed in line 130, in the same sequence. If Z matches one of them, (G+2)/3 gives the subscript number of GO$ for that word. Otherwise, if Z<100 the ASCII is not a token, or if G=0 then (G+2)/3 is less than 1, so the next search starts one byte to the left. If S becomes less than 3, the token being searched for must be that for GO, BREAK, or UNBREAK, which are not in the POS statement, so we will use the subscript 1 for GOTO instead. Otherwise, go back and continue searching.

When the token is found, line 250 converts the "to" line number into a string followed by a period, followed by the string representation of the program line number followed by the string representation of the subscript number of GO$. For instance, if line 100 was 100 GOTO 1000, C$ would be 1000.1001, which can be converted into a valid decimal 1000.1001 .

Then, the search will continue starting at the 3rd byte after the last ASCII 201 we found, the add-on variable C$ is cancelled before being reused, and we go back to see if the program line contains another jump. If not, IF X=0 THEN 260, where we check EOF to see if the end of the file is reached, otherwise go back to read in the next program line, until finished.

Now, the data in C() is in program line number sequence, but we want to list it in "to" line number sequence. That's the reason for the weird method of decimal storage. Note that tacking the subscript number, from 1 to 9, onto the end of the decimal prevented the trailing zeros from being dropped. If C$="1000.100" then VAL(C$) would equal 1000.1, but VAL("1000.1001") equals 1000.1001.

So now, a simple sorting routine arranges the data into "to" line number sequence and, within that, into program line number sequence and even within that into subscript number sequence!

Finally, lines 290-330 convert the C() array back into a string so it can be taken apart into its three components and printed. Note that if printer output was not selected in line 180-190, D was not given a value and still equals 0 (its value in subroutine LONGSHELLN is not passed back to the main program) and PRINT #0 causes output to the screen.

Guess who?

HE'S AT THAT
COMPUTER AGAIN

BOX-DOTS

By Wesley R. Richardson

```
100 REM BOX-DOTS
110 REM TI-99/4A EXTENDED BASIC WITH J
    OYSTICKS, TWO PLAYER GAME
120 REM WESLEY R. RICHARDOSON, NOV 198
    7
130 REM BLUEGRASS 99 COMPUTER SOCIETY,
     INC.
140 REM VARIABLES A,A$,B,B$,C,D,E,H$,I
    ,J,M,N,P,Q,R,S,T,W$,Z
150 CALL CLEAR :: CALL SCREEN(6):: FOR
     I=0 TO 12 :: CALL COLOR(I,16,1)::
     NEXT I
160 CALL CHAR(100,"0100000000000000FF0
    0000000000000010101010101010101FF010
    10101010101")! BORDERS
170 T=1 :: CALL CHAR(98,RPT$("F",16),9
    9,"3030FCFC303000")! CROSS
180 REM RESTART POINT
190 GOSUB 1030
200 DISPLAY AT(6,3):"PRESS ALPHA LOCK
    DOWN"
210 DISPLAY AT(9,3)SIZE(-18):"PLAYER #
    1 INITIAL?" :: ACCEPT AT(9,22)SIZE
    (1)VALIDATE(UALPHA):A$
220 IF LEN(A$)=0 THEN 210
230 DISPLAY AT(12,3)SIZE(-18):"PLAYER
    #2 INITIAL?" :: ACCEPT AT(12,22)SI
    ZE(1)VALIDATE(UALPHA):B$
240 IF LEN(B$)=0 THEN 230
250 A=ASC(A$):: B=ASC(B$):: IF B<>A TH
    EN 270
260 DISPLAY AT(13,3):"LETTER OTHER THA
    N "&A$ :: GOTO 230
270 DISPLAY AT(6,1):RPT$("d",26):: DIS
    PLAY AT(15,3):"PUT ALPHA LOCK UP"
    :: DISPLAY AT(16,3):"THEN PRESS EN
    TER"
280 CALL KEY(0,I,J):: IF J=0 THEN 280
290 CALL CHARPAT(A,A$,8,B$):: CALL CHA
    R(105,A$,113,B$)
300 RESTORE :: FOR I=1 TO 15 :: READ W
    $ :: DISPLAY AT(I+4,1):W$ :: NEXT
    I
310 DATA "BOX-DOTS IS PLAYED BY TWO","
    PLAYERS.  EACH TRIES TO GET","THE
    MOST NUMBER OF BOXES"
320 DATA "WITH HIS LETTER IN IT.","","
    PLAYERS TAKE TURNS USING","JOYSTIC
    KS TO MOVE THEIR"
330 DATA "CURSOR, AND THE FIRE BUTTON"
    ,"TO DRAW IN A LINE.","","YOU MUST
     DRAW AT LEAST ONE"
340 DATA "LINE ON EACH TURN, AND AFTER
    ","COMPLETING A BOX.","","HIGH SCO
    RE WINS."
350 CALL BOXES(A$):: CALL BOXES(B$)
360 DISPLAY AT(21,8):"PRESS ANY KEY" :
    : CALL SOUND(200,1760,0)
370 CALL CHAR(104,A$,112,B$)
380 CALL KEY(0,I,J):: IF J=0 THEN 380
390 GOSUB 1030 :: A=0 :: B=0 :: M=20 :
    : N=25
400 CALL SPRITE(#1,99,1,80,80,#2,99,1,
    80,170,#3,98,1,185,25,#4,98,1,185,
    81)
410 DISPLAY AT(10,1):"VERTICAL  SIZE
    (1-20)?";M :: DISPLAY AT(12,1):"HO
    RIZONTAL SIZE (1-25)?";N
420 ACCEPT AT(10,25)SIZE(-2)VALIDATE(N
    UMERIC):M :: M=INT(M):: IF (M<1)+(
    M>20)THEN 410
430 ACCEPT AT(12,25)SIZE(-2)VALIDATE(N
    UMERIC):N :: N=INT(N):: IF (N<1)+(
    N>25)THEN 410
440 P=13+8*INT((21-M)/2):: Q=174-8*INT
    ((20-M)/2)
450 R=21+8*INT((26-N)/2):: S=222-8*INT
    ((25-N)/2)
460 Z=INT(M*N):: GOSUB 1030
470 FOR I=3 TO INT((25-M)/2):: CALL HC
    HAR(I,3,32,26):: NEXT I
480 FOR I=24-INT((20-M)/2)TO 23 :: CAL
    L HCHAR(I,3,32,26):: NEXT I
490 FOR J=3 TO INT((30-N)/2):: CALL VC
    HAR(3,J,32,21):: NEXT J
500 FOR J=29-INT((25-N)/2)TO 28 :: CAL
    L VCHAR(3,J,32,21):: NEXT J
510 GOSUB 990
520 REM MAIN LOOP
530 CALL COLOR(#T,9+T,#(T+2),9+T)
540 CALL JOYST(T,I,J):: CALL MOTION(#T
    ,-J-J,I+I)
550 CALL KEY(T,I,J):: IF J=0 THEN 540
560 IF I<>18 THEN 540
570 CALL MOTION(#T,0,0):: CALL POSITIO
    N(#T,I,J):: IF (I<P)+(I>Q)+(J<R)+(
    J>S)THEN 540
580 I=I+3 :: J=J+3
590 O=8*INT((I+3)/8)+1 :: C=8*INT((J+3
    )/8):: E=ABS(I-O)-ABS(J-C):: IF E=
    0 THEN 540
600 IF E>0 THEN 710
610 REM HORIZONTAL
620 I=(O+7)/8 :: J=INT((J+7)/8):: CALL
     GCHAR(I,J,C)
630 IF C<>100 THEN 670
640 CALL HCHAR(I,J,101)
650 I=I-1 :: GOSUB 910 ! CHECK FOR BOX
     AT I-1,J
660 GOTO 810
670 IF C<>102 THEN 540
```

MORE BOX-DOTS

```
680 CALL HCHAR(I,J,103)
690 GOSUB 910 :: D=E :: I=I-1 :: GOSUB
    910 :: E=D+E ! CHECK FOR BOX AT I
    ,J AND I-1,J
700 GOTO 810
710 REM VERTICAL
720 I=INT((I+7)/8):: J=C/8 :: CALL GCH
    AR(I,J,C)
730 IF C<>100 THEN 770
740 CALL HCHAR(I,J,102)
750 J=J+1 :: GOSUB 910 ! CHECK FOR BOX
    AT I,J+1
760 GOTO 810
770 IF C<>101 THEN 540
780 CALL HCHAR(I,J,103)
790 GOSUB 910 :: D=E :: J=J+1 :: GOSUB
    910 :: E=D+E ! CHECK FOR BOX AT I
    ,J AND I,J+1
800 GOTO 810
810 REM TURN FINISHED ?
820 IF E=0 THEN 890
830 IF (A+B)<2 THEN 540
840 DISPLAY AT(24,17):"AGAIN? Y/N" ::
    FOR I=1 TO 6 :: CALL SOUND(200,220
    *2^I,0):: NEXT I
850 ACCEPT AT(24,24)SIZE(-1)VALIDATE("
    YNyn"):W$
860 CALL DELSPRITE(ALL):: T=T-(T=1)+(T
    =2)
870 IF (W$="Y")+(W$="y")THEN 180
880 CALL CLEAR :: CALL CHARSET :: DISP
    LAY AT(12,3):"THANKS FOR PLAYING.
     WR" :: STOP
890 CALL COLOR(#T,1,#(T+2),1):: T=T-(T
    =1)+(T=2):: GOTO 520
900 STOP
910 REM CHECK FOR BOX
920 CALL GCHAR(I,J,E):: E=(E=103):: IF
     E=0 THEN 980
930 CALL GCHAR(I,J-1,E):: E=(E=102)+(E
    =103)+(E=104)+(E=112):: IF E=0 THE
    N 980
940 CALL GCHAR(I+1,J,E):: E=(E=101)+(E
    =103)+(E=104)+(E=112):: IF E=0 THE
    N 980
950 CALL HCHAR(I,J,96+T*8)
960 A=A-1*(T=1):: B=B-1*(T=2):: E=1
970 GOSUB 990
980 RETURN
990 REM SCORE
1000 A$=STR$(A):: B$=STR$(B):: W$=STR$(
     Z-A-B)
1010 DISPLAY AT(24,3):"1="&SEG$(" "&A$
     ,LEN(A$),3)&"  2="&SEG$(" "&B$,LE
     N(B$),3)&"  AVAIL="&SEG$(" "&W$,L
     EN(W$),3)
1020 RETURN
1030 REM SET-UP DOTS
1040 DISPLAY AT(1,11):"BOX-DOTS"
1050 FOR I=3 TO 23 :: DISPLAY AT(I,1):R
     PT$("d",26):: NEXT I
1060 RETURN
1070 SUB BOXES(A$)
1080 B$="FF" :: H$="0123456789ABCDEFF"
1090 FOR I=4 TO 16 STEP 2
1100 B$=B$&SEG$(A$,I-1,1)
1110 FOR J=1 TO 16 STEP 2
1120 IF SEG$(A$,I,1)=SEG$(H$,J,1)THEN B
     $=B$&SEG$(H$,J+1,1):: GOTO 1140
1130 NEXT J
1140 NEXT I
1150 A$=B$
1160 SUBEND
1170 END
```

Guess who?

HE'S STILL AT
THAT COMPUTER

DEBUGGING

by Jim Peterson

When you have finished writing a program, the next thing you should do is to run it. And, very probably, it will crash!

Don't be discouraged. It happens to the very best of programmers, very often.

So, the next thing to do is to debug it. And you are lucky that you are using a computer that helps you to debug better than some that cost ten times as much.

There are really three types of bugs. The first type will prevent the program from running at all - it will crash with an error message. The second type will allow the program to run, but will give the wrong results.

And the third type, which is not really a bug but might be mistaken for one, results from trying to run a perfectly good program with the wrong hardware, or with faulty hardware. As for instance, trying to run a Basic program, which uses character sets 15 and 16, in Extended Basic.

First, let's consider the first type. The smart little TI computer makes three separate checks to be sure your program is correct. First, when you key in a program line and hit the Enter key, it looks to see if there is anything it can't understand - such as a misspelled command or an unmatched quotation mark. If so, it will tell you so, most likely by SYNTAX ERROR, and refuse to accept the line.

Next, when you tell it to RUN the program, it first takes a quick look through the entire program, to find any combination of commands that it will not be able to perform. This is when it may crash with an error message telling you, for instance, that you have a NEXT without a matching FOR, or vice versa.

And finally, while it is actually running and comes to something that it just can't do, it will crash and give you an error message - probably because a variable has been given a value that cannot be used, such as a CALL HCHAR(R,C,32) when R happens to equal 0.

The TI has a wide variety of error messages to tell you when you did something wrong, what you did wrong, and where you did it wrong. But, it can be fooled! For instance, try to enter this program line (note the missing quotation mark).
100 PRINT "Program must be saved in"merge format."

And, sometimes you may be told that you have a STRING-NUMBER MISMATCH when there is no string involved, because the computer has tried to read a garbled statement as a string.

Also, the line number given in the error message is the line where the computer found it impossible to run the program; that line may actually be correct but the variables at that point may contain bad values due to an error in some previous line.

If the error occurs in a program line which consists of several statements, and you cannot spot the error, you may have to break the line into individual single-statement lines. This is the easiest way to do that - Be sure the line numbers are sequenced far enough apart. Bring the problem line to the screen, put a ! just before the first ::, and enter it. Bring it back to the screen with FCTN 8, retype the line number 1 higher, use FCTN 1 to delete the first statement and the ! and ::, put a ! before the first :: and continue. Then, when you have solved the bug, just delete the ! from the original line and delete all the temporary lines.

Pages 212-215 of your Extended Basic manual list almost all the error codes, and almost all the causes of each one - it will pay you to consult these pages rather than guessing what is wrong.

You may create some really bad bugs when you try to modify a program that was written by someone else - especially if you add any new variable names or CALLs to the program. Your new variable might be one that is already being used in the program for something else, perhaps in a subscripted array. I have noticed that programmers rarely use 0 in a variable name, so I always tack it onto the end of any variable that I add to a program.

Also, the program that you are modifying may have ON ERROR routines, or a prescan, already built in. The ON ERROR routine was intended to take care of a different problem than the one you create, so it could lead you far astray - you had better delete that ON ERROR statement until you are through modifying.

The prescan had better be the subject of another lesson, but if the program has an odd-looking command !@P- up near the front somewhere, it has a prescan built in. And if so, if you add a new variable name or use a CALL that isn't in the program, you will get a SYNTAX ERROR even though there is no error. One way to solve this is to insert a line with !@P+ just before the problem line, and another with !@P- right after it.

When a program runs, even though it crashes or is stopped by FCTN 4 or a BREAK, the values assigned by the program to variables up to that point will remain in memory until you RUN again, or make a change to the program, or clear the memory with NEW. This can be very useful. For instance, if the program crashes with BAD VALUE IN 680, and you bring line 680 to the screen and find it reads
CALL HCHAR(R,C,CH)
just type PRINT R;C;CH and you will get the values of R, C and CH at the time of the crash. You will find that R is less than 1 or more than 24, or C is less than 1 or more than 32, or CH is out of range.

In Extended Basic, you can even enter and run a multi-statement line in immediate mode (that is, without a line number), if no reference is made to a line number. So, you can dump the current contents of an array to the screen by
FOR J=1 TO 100::PRINT A(J);:: NEXT J - or you can even open a disk file or a printer to dump it to.

You can also test a program by assigning a value to a variable from the immediate mode. If you BREAK a program, enter A=100 and then enter CON, the program will continue from where it stopped but A will have a value of 100.

You can temporarily stop a

program at any time with FCTN 4, of course (the manual says SHIFT C, but it was written for the old 99/4), and restart it from that point with CON. Or you can insert a temporary line at any point, such as 971 BREAK if you want a break after line 970. Or, you can put a line at the beginning of the program listing the line numbers before which you want breaks to occur, such as 1 BREAK 960,970,980 Note that in this case the program breaks just BEFORE those listed line numbers. You can also use BREAK followed by one or more line numbers as a command in the immediate mode.

The problem with using BREAK and CON is that BREAK upsets your screen display format, resets redefined characters and colors to the default, and deletes sprites. So, it is sometimes better to trace the assignment of values to your variables by adding a temporary line to DISPLAY AT their values on some unused part of the screen. If you want to trace them through several statements, it will be better to GOSUB to a DISPLAY AT. And if you need to slow up the resulting display, just add a CALL KEY routine to the subroutine.

Sometimes, your program will appear to be not flowing through the sequence of lines you intended (perhaps because it dropped out of an IF statement to the next line!) and you will want to trace the line number flow. This can be done with TRACE, either as a command from the immediate mode or as a program statement, which will cause each line number to print to the screen as it is executed. If used as a command, it will trace everything from the beginning of the program, so it is usually better to insert a temporary line with TRACE at the point where you really want to start. Once you have implemented TRACE, the only way to get rid of it is with UNTRACE.

TRACE has its limitations because it can't tell you what is going on within a multi-statement line, and it will certainly mess up any screen display. Sometimes it is better to insert temporary program lines to display line numbers. I use CALL TRACE( ) with the line number between the parentheses, and a subprogram after everything else

```
30000 SUB TRACE(I)::DISPLAY
 AT(24,1):I :: SUBEND
```

Some programmers use ON ERROR combined with CALL ERR as a debugging tool, but I can't tell you much about that because I have never used it. ON ERROR can give more trouble than help if not used very carefully, and I cannot see that CALL ERR gives any information not available by other means.

Sometimes you can debug a line by simply retyping it. It is only very rarely that the computer is actually interpreting a line differently than it appears on the screen, but retyping may result in correcting a typo error that you just could not see. In fact, most bugs turn out to be very simple errors.

When you are debugging a string-handling routine, don't take it for granted that a string is really as it appears on the screen - it may have invisible characters at one or both ends. Try PRINT LEN(M$) to see if it contains more characters than are showing; or PRINT "I"&M$&"I" to see if any blanks appear between the asterisks and the string.

There is no standard way to debug a program. Each problem presents a challenge to figure out what is going wrong, to devise a test to find out what is really happening.

Don't debug by experimenting, by changing variable values just to see what will happen, etc. Even if you succeed, you will not have learned what was wrong so you will not have learned anything - and if your program contains lines that you didn't understand when you wrote them, you will have real problems if you ever try to modify the program. (Believe me, I speak from experience!)

You know who!

This is a reprint of an article from the C.C.99'ER, Feb. 1986, by JIM BURES.

TIC-TAC-TOE with LIGHT-PEN

I've seen several schematics for LIGHT-Pens that work with the T.I., but I haven't seen any programs that utilize the pen. I decided to build one for the novelty of it, but I had to write a small program for it. I found all the parts at a RADIO SHACK, and the part numbers are listed in this article. Because of several differences in the schematics I reviewed, and only a basic understanding of electronics, I constructed the project on an experimenters breadboard, so that changes could be made.

After several tries, the pen worked, but was so sensitive to the slightest ray of light from anywhere set the pen off. A black hood over the photo-transistor with only a small hole in the end and some delay loops in the program solved the problem. Now about the program. It is in Extended Basic only because I wanted to use sprites.

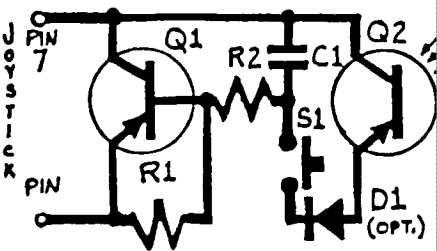
Lines 130 to 350 set up the game board and defines characters. Lines 360 to 550 are the meat of the program. Line 430 checks to see if the light-pen has been set off by a light source (in this program it is a white colored sprite). The rest of the program lines will determine which marker (X or Y) goes where. The program is not really complete because no checks are done to see who wins. I'll leave that to you. JIM

(Ed. note: There is a basic version of this program, that does more checking, but for the sake of providing a program, and a light-pen for you who havent built one, I'm including this one. There is also another program that uses a double set of light-pens for gaming between two players on one screen.)

```
100 REM *  TIC-TAC-TOE   *
110 REM * LIGHT-PEN DEMO *
120 REM *  BY JIM BURES  *
130 F=1 :: Z=1
140 DIM A(9,2)
150 CALL COLOR(3,12,1):: CALL CO
LOR(4,12,1)
160 A(1,1)=16 :: A(1,2)=16 :: A(
2,1)=16 :: A(2,2)=104 :: A(3,1)=
16 :: A(3,2)=184
170 A(4,1)=80 :: A(4,2)=16 :: A(
5,1)=80 :: A(5,2)=104 :: A(6,1)=
80 :: A(6,2)=184
180 A(7,1)=144 :: A(7,2)=16 :: A
(8,1)=144 :: A(8,2)=104 :: A(9,1
)=144 :: A(9,2)=
184
190 CALL SCREEN(2)
200 CALL COLOR(10,13,1)
210 CALL CLEAR
220 S1$="8040201008040201010204
8102040800102040810204080804020
1008040201"
230 CALL CHAR(96,S1$)
240 CALL MAGNIFY(4)
250 S2$="FF8080808080808080808
0808080FFFF0101010101010101010
1010101FF"
260 CALL CHAR(100,S2$)
270 CALL DELSPRITE(ALL)
```

SCHEMATIC and PARTS LIST

R1 - 470 ohm R2 - 47K ohm
Q1 - 2N... Tran. R.S. # 276-2023
Q2 - TIL14 Phototran R.S.# 276-145
C1 - 0.01 mF capacitor R.S.# 272-1472
S1 - Any small momentary contact switch
D1 - OPTIONAL Diode 1N914 or 1N4A



Acquire a large felt tip pen to use a housing for the above aparatus. An old joystick cable will make a perfect cable with the female 9-PIN D TYPE connector, which will make assembly easier, faster and definitly more economical.

(** WP99 UG assumes no responsibility for any damage you may do to your console attempting to install this feature

```
280 CALL CHAR(104,"10101010101
010")
290 CALL VCHAR(1,20,104,24)
300 CALL VCHAR(1,10,104,24)
310 CALL CHAR(105,"FF00")
320 CALL HCHAR(8,1,105,32)
330 CALL HCHAR(16,1,105,32)
340 S3$="FFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFF"
350 CALL CHAR(108,S3$)
360 Z=Z+1
370 IF F<=0 THEN 8=96 ELSE S=10
380 FOR I=1 TO 9
390 R=A(I,1):: C=A(I,2)
400 IF A(I,1)=0 THEN 470
410 CALL SPRITE(#1,108,16,R,C)
420 FOR D=1 TO 6
430 CALL JOYST(1,X,Y)
440 IF X THEN 490
450 NEXT D
460 CALL DELSPRITE(#1)
470 NEXT I
480 GOTO 370
490 CALL DELSPRITE(#1):: CALL S
OUND(500,990,1):: CALL SOUND(50,
0000,30)
500 FOR D=1 TO 6 :: NEXT D
510 CALL SPRITE(#Z,S,S,R,C)
520 O=O+1 :: IF O=9 THEN 550
530 A(I,1)=0 :: A(I,2)=0
540 F=-F :: GOTO 360
550 GOTO 550
```

COMPETITION COMPUTER PRODUCTS

2629 W. NATIONAL AVE. MILWAUKEE, WIS. 53204

414-672-4010

BANKCARDS - CHECKS - DISCOVER CARDS - COD WELCOME!

* NOW — DISKS .49 EACH! *

GENUINE TI JOYSTICKS *6 PR/SEE GENE

WE WILL BUY ANY TI HARDWARE OR SOFTWARE YOU NO LONGER NEED - CALL!

STORE HOURS; MON THRU FRI 10-6 SAT 10-3

WE TAKE TI SYSTEMS IN TRADE ON IBM COMPATIBLES.

NEW-NEW NEW-NEW

* P.E.P S/WARE TO TRANSFER FILES TO MS/DOS COMPUTERS *

* DATA CASSETTE SALE 20% OFF - THIS MONTH ONLY *

128K/512K MEMORY EXPANSION CARD BY MYARC $200.00/$327.50

MINIWRITER III+ WORD PROCESSOR CARTRIDGE W/PRINTER INTERFACE $89

COMPLETE LINE OF DATABIOTICS INC. SOFTWARE

* * LATE STYLE KEYBOARD - FITS ALL 99/4A $19.95 * *

* * NIGHT MISSION BY MILLER GRAPHICS * *

* * LOTS OF NEW 3RD PARTY SOFTWARE * *

IF IT'S AVAILABLE - WE USUALLY STOCK IT!

* BETTER BANNER $19.95 *

NEW-NEW NEW-NEW

NEW AND USED TI99/4A COMPUTERS AVAILABLE!

EXPANSION SYSTEMS AVAILABLE - NEW AND USED!

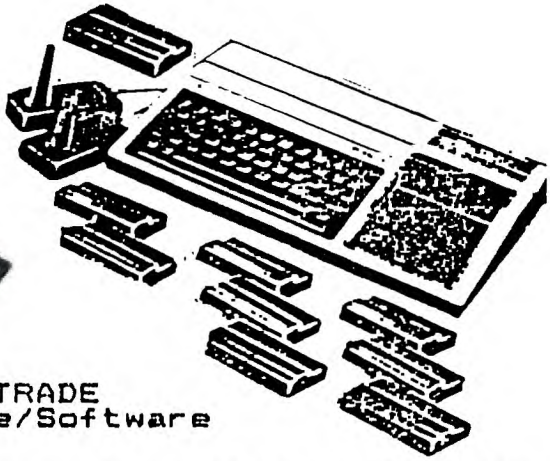
* HUGE SOFTWARE INVENTORY - MORE IN STOCK THAN EVER BEFORE! *

BEFORE YOU MAIL ORDER OR BUY ELSEWHERE - GIVE US A CALL - WE WILL TRY TO MEET OR BEAT ANYBODY'S PRICES. REMEMBER THAT WE ARE HERE TO HELP YOU HAVE A QUESTION OR PROBLEM. WE DO NOT CHARGE EXTRA FOR BANKCARDS. WE WANT YOUR BUSINESS AND WE'LL PROVE IT!

TED, GENE, JIM & RON

2nd BYTES

THE USED / NEW COMPUTER STORE



We BUY, SELL and TRADE
TI Equipment/Hardware/Software

9721 W. Greenfield Ave. West Allis 774-1155

TIPS FROM THE TIGERCUB

#44

Copyright 1987

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 130 original programs in Basic and Extended Basic, available on cassette or disk, now reduced to just $2.00 each, plus $1.50 per order for cassette or disk and PP&H. Cassette programs will not be available after my present stock of blanks is exhausted.
Descriptive catalogs, while they last, $1.00 which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $10 postpaid. Each of these contains either 5 or 6 of my regular $2 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 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

NUTS & BOLTS (No. 1), a full disk of 100 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. Reduced to $15.00 postpaid.
NUTS & BOLTS NO. 2, another full disk of 108 utility subprograms in merge format, all new and fully compatible with the last, and with 10 pages of documentation and examples. Also $15 postpaid.

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
x NUTS & BOLTS #3 is now    x
x ready, another full disk  x
x of 140 new merge-format   x
x utility subprograms, all  x
x compatible with the pre-  x
x vious. With 11 pages of   x
x documentation, $15 ppd.   x
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

TIPS FROM THE TIGERCUB, a full disk containing the complete contents of this newsletter Nos. 1 through 14, 30 original programs and files, reduced to $10 ppd.
TIPS FROM THE TIGERCUB VOL. 2, another diskfull, complete contents of Nos. 15 through 24, over 60 files and programs, also just $10
TIPS FROM THE TIGERCUB VOL. 3, another 62 programs, tips and routines from Nos. 25 through 32, $10 postpaid.
TIPS FROM THE TIGERCUB VOL. 4, another 48 programs and files from issues 33 through 41, also $10 postpaid.

Thanks to Steve Chapman and Bill Wallbank of Stone & Webster Engineering Corp. TIUG for this one. If V=2I you are in Extended Basic, otherwise you are in Basic. I am not sure it will work with all consoles and modules. -

```
100 RANDOMIZE @0
110 V=INT(RND#100)
```

How can you input a blank (CHR$ 32) with ACCEPT AT? As far as I know, you can't. With LINPUT, just hit the space bar, and with INPUT, type " ". But with ACCEPT AT the space bar gives a null string and " " gives " " ! However, you can code around it -

```
I$=CHR$(34)&CHR$(32)&CHR$(32
)!! ACCEPT AT(1,1):T$ :: IF
T$=I$ THEN T$=CHR$(32)
```

And, to clear up the puzzling behavior of the 'quote marks' -

```
100 CALL CHARPAT(34,CH$)!! C
ALL CHAR(35,CH$)!written by
Jim Peterson
110 DISPLAY AT(1,7)ERASE ALL
:"THE # PUZZLE":" You can't
enter PRINT # or PRINT ### -
 the computer  demands an
even number of #."
120 DISPLAY AT(5,1):"1 PRINT
 ## 'prints a null string (n
othing)":"2 PRINT ### !print
s #"
130 DISPLAY AT(8,1):"3 PRINT
 ####  !prints #":"4 PRINT ##
### !crashes as  STRING-NUM
BER MISMATCH"
140 DISPLAY AT(11,1):"5 PRIN
T ###### !crashes as  SYNTAX
 ERROR"
150 DISPLAY AT(13,1):"6 PRIN
T ######  !prints ##":"7 PRIN
T ########  !prints ###":"8 PR
INT ######## 'print ####"
160 DISPLAY AT(16,1):"9 PRIN
T #########  'prints ###":"10
 PRINT ##########  !crashes as
 STRING-NUMBER MISMATCH"
170 DISPLAY AT(19,1):"11 PRI
NT ##########  !crashes as SY
NTAX ERROR":"12 PRINT ######
#### '####"
180 DISPLAY AT(22,1):"13 PRI
NT ##########  '####":"14 P
RINT ############'######"
190 DISPLAY AT(24,1):"TRY IT
! LINE NO.(1-14)?" :: ACCEPT
 AT(24,25)VALIDATE(DIGIT)SIZ
E(2)BEEP:LN :: IF LN<1 OR LN
>14 THEN 190
200 CALL CLEAR :: ON LN GOSU
B 230,240,250,260,280,290,30
0,310,320,330,340,350,360,37
0
210 PRINT :::::"Press any ke
y"
220 CALL KEY(0,K,S):: IF S=0
 THEN 220 ELSE 110
230 PRINT "" :: RETURN
240 PRINT "#" :: RETURN
250 PRINT "## :: RETURN
260 PRINT "#2" !crashes as
STRING-NUMBER MISMATCH - the
 # is misinterpreted as a mu
ltiplier'Same with +,-,/
270 !with anything else, inc
luding numerals, crashes as
SYNTAX ERROR - but inserts a
 space before the character'
280 PRINT "#3" :: !crashes
290 PRINT "### :: RETURN
300 PRINT "#5" :: RETURN
310 PRINT "#8" :: RETURN
320 PRINT "##### :: RETURN
330 PRINT "##9" !crash
340 PRINT "##11" !crash
350 PRINT "########## :: RETU
RN
360 PRINT "#####5##### :: RET
URN
370 PRINT "#####11##### :: RE
TURN
```

The method of closing an "ajar" file, described in Tips #28, doesn't always work, but this one seems to be reliable -

```
100 ON ERROR 500 :: OPEN #1:
"DSK1.TEST" :: INPUT #1:A$ :
: PRINT A$ :: STOP
500 ON ERROR 510 :: CLOSE #1
510 INPUT "CHECK DISK AND DR
IVE, PRESS ANY KEY":DUMMY$ :
: RETURN 100
```

This one is just for the fun of it - it uses the contents of computer memory to create designs -

```
100 DISPLAY AT(3,10)ERASE AL
L:"COLORPEEK": :TAB(7);"by J
im Peterson": : :" Watch the
 computer's memory": :"displ
ayed in color."
110 DISPLAY AT(12,1):"Choose
 1 :"(1) plain colors": :"(2
) bars & checks": :"(3) patt
erns" :: ACCEPT AT(12,8)VALI
```

```
BATE("123")SIZE(1):@ :: CALL
 CLEAR :: IF @=1 THEN 170
120 DISPLAY AT(12,5):"wait,
please" :: IF @=3 THEN 140
130 FOR CH=32 TO 143 :: CALL
 CHAR(CH,RPT$("FF",8)):: NEX
T CH :: GOTO 160
140 RANDOMIZE :: FOR CH=32 T
O 88 :: FOR J=1 TO 4 :: X$=S
EG$("0018243C425A667E8199A58
DC3DBE7FF",INT(16*RND+1)*2-1
,2):: B$=B$&X$ :: C$=X$&C$ :
: NEXT J :: CALL CHAR(CH,B$&
C$)
150 CALL CHAR(CH+55,B$&C$):::
 B$,C$="" :: NEXT CH
160 FOR SET=0 TO 14 :: CALL
COLOR(SET,SET+1,16-SET):: NE
XT SET :: CALL SCREEN(2):: G
OTO 180
170 FOR SET=0 TO 14 :: CALL
COLOR(SET,SET+2,SET+2):: NEX
T SET :: CALL SCREEN(16)
180 FOR J=-1 TO -2000 STEP -
1 :: CALL PEEK(J,A):: A=A-(A
<33)*(A+32):: A=A+(A)143)*(A
/2):: R=R+1+(R=24)*24 :: CAL
L HCHAR(R,1,A,32)
190 C=C+1+(C=32)*32 :: CALL
VCHAR(1,C,A,24):: NEXT J ::
GOTO 100
```

Unlike most of the number games played against the computer, you can win this one -

```
100 CALL CLEAR :: CALL SCREE
N(16):: DISPLAY AT(3,8):"THE
 '37' GAME" !by Jim Peterson
110 DISPLAY AT(5,1):" We wil
l take turns picking";"a num
ber from 1 to 5, but";"not t
he number that was just":"pi
cked."
120 DISPLAY AT(10,1):" The n
umbers we pick will be";"add
ed to the total count."
130 DISPLAY AT(13,1):" Whoev
er reaches 37 is the";"winne
r, but if you go over";"37 y
ou lose."
140 CALL SHOW(20,1,"Press an
y key to start")
150 CALL KEY(0,K,S):: IF S=0
 THEN 150
160 DATA 4,11,17,24,30,37
170 DATA 262,330,392,523,523
180 DATA 1047,784,659,523,52
3
190 C,P=0 :: CALL CLEAR :: C
ALL MAGNIFY(2):: R=10 :: FOR
 J=1 TO 5 :: CALL SPRITE(#J,
48+J,5,R,10):: R=R+30 :: NEX
T J
200 CALL SHOW(24,1,"(Y)ou or
 (C)omputer first?"):: ACCEP
T AT(24,28)VALIDATE("YC")SIZ
E(1):Q$ :: DISPLAY AT(24,1):
""
210 IF Q$="C" THEN CALL SHOW
(22,8,"I pick 4"):: CALL COL
OR(#4,1):: P=4 :: C=4 :: CAL
L SHOW(3,10,"COUNT=4")
220 CALL SHOW(20,8,"Pick you
r number"):: ACCEPT AT(20,26
)VALIDATE("12345"):N :: IF N
=P THEN 220
230 IF P>0 THEN CALL COLOR(#
P,5)
240 CALL COLOR(#N,1):: P=N :
: C=C+N :: CALL SHOW(3,10,"C
OUNT= "&STR$(C)):: IF C=37 T
HEN 320 ELSE IF C>37 THEN 34
0
250 RESTORE 160
260 READ I :: IF C<I THEN B=
I-C ELSE IF I<37 THEN 260
270 CALL SHOW(22,8,"I'm thin
king..."):: FOR Y=1 TO 700 :
: NEXT Y
280 IF B>5 AND B/2=INT(B/2)T
HEN B=B/2
290 IF B>5 OR B=P THEN B=1-(
P=1)
300 CALL SHOW(22,8,"I pick "
&STR$(B)):: CALL COLOR(#P,5)
:: CALL COLOR(#B,1):: P=B ::
 C=C+B :: CALL SHOW(3,10,"CO
UNT= "&STR$(C))
310 IF C=37 THEN 340 ELSE IF
 C>37 THEN 320 ELSE 220
320 RESTORE 170 :: FOR J=1 T
O 5 :: READ F :: CALL SOUND(
100,F,5,F#1.03,5):: NEXT J :
: CALL SHOW(12,8,"YOU WIN!")
330 CALL SHOW(15,8,"Play aga
in? (Y/N)"):: ACCEPT AT(15,2
6)VALIDATE("YN"):Q$ :: IF Q$
="N" THEN STOP ELSE 190
340 RESTORE 180 :: FOR J=1 T
O 5 :: READ F :: CALL SOUND(
300,30000,30,30000,30,F,30,-
4,5):: NEXT J :: CALL SHOW(1
2,8,"YOU LOSE!"):: GOTO 330
350 SUB SHOW(R,C,T$):: FOR J
=1 TO 10 :: DISPLAY AT(R,C):
" " :: DISPLAY AT(R,C):T$ ::
 NEXT J :: SUBEND
```

A couple more peculiarities of the computer -

```
100 DISPLAY AT(3,8)ERASE ALL
:"POS PUZZLE #1":" f
rom Tigercub"
110 DISPLAY AT(9,1):"Why doe
s the computer say":"that I=
1 if you answer the":"prompt
 with the Enter key":"(null-
string) ?"
120 DISPLAY AT(14,1):"110 IN
PUT M$"
130 DISPLAY AT(15,1):"120 I=
POS(""TESTING"",M$,1)::":"PR
INT I :: GOTO 100"
140 !POS PUZZLE #1 - why doe
s the computer say that I=1
if you answer the prompt wit
h Enter (null-string) ?
 - Jim Peterson
150 INPUT M$
160 I=POS("TESTING",M$,1)::
PRINT I :: GOTO 140
```

And -

```
100 DISPLAY AT(3,8)ERASE ALL
:"POS PUZZLE #2":" f
rom Tigercub"
110 DISPLAY AT(7,1):"Why doe
s the computer say":"that th
e first position of":"null-s
tring is at whatever":"posit
ion it is told to start";"se
arch at?"
120 DISPLAY AT(13,1):"100 M$
="""""
130 DISPLAY AT(14,1):"110 DI
SPLAY AT(20,1):""POS?"" :: A
CCEPT AT(20,6):P"
140 DISPLAY AT(16,1):"120 I=
POS(""TESTING"",M$,P):: DISP
LAY AT(22,1):""I="";I :: GOT
O 110"
150 M$=""
160 DISPLAY AT(21,1):"POS?"
:: ACCEPT AT(21,6):P
170 I=POS("TESTING",M$,P)::
DISPLAY AT(23,1):"I=";I :: G
OTO 160
```

Here is an improvement to the PRINTSPEAKER in Tips #40 - in lines 130 and 160, change the CHR$(I)&"1" to CHR$(I)&"255" . This will avoid problems if the program being converted opens FILE #1.

Irvin Hott informs me th
assembly routines which ha
been imbedded into IB.s
programs, using ALSAVE
SYSTEM, can be saved
cassette and reloaded. Th
could be very useful fo
those who have a stand-alon
or "matchbox" 32k.

And, a mini-game for yo
to have fun with or improv
on -

```
1 !       2-LINE GAME
        by Jim Peterson
- use SHD keys to paint th
white line on the highway
2 !if it is too easy, chan
the 6 in A$=RPT$(CHR$(143),
) to 5 and the 5 in C)T+5
 4
100 CALL CLEAR :: A$=RPT$((
R$(143),6):: CALL COLOR(14,
,2,2,16,16):: CALL SCREEN(
:: T=11 :: C=14 :: CALL HCH
R(22,C+2,42):: RANDOMIZE
110 T=T+INT(3*RND-1)+(T=21
(T=1):: PRINT TAB(T);A$ ::
ALL KEY(3,K,S):: C=C+(K=83
(K=68):: CALL HCHAR(22,C+2
2):: IF C<T OR C>T+5 THEN
OP ELSE 110
```

And finally, one of t
best examples of compa
programming I have ev
seen -

```
1 !JOHN WITTE'S 3-LINE VER
ON OF JOHN WILLFORTH'S WAV
POWER - PUBLISHED IN GREAT
 OMAHA UG NEWSLETTER
100 CALL CLEAR :: A$(1)="A
DEF8FEDC8A" :: FOR I=1 TO
:: CALL CHAR(72-I,RPT$("0"
$I-2)&"FFFF",47,"30303EFF7
E1E04"):: A$(I+1)=SEG$(A$(
,2,12)&SEG$(A$(I),2,1)):: N
T I
110 CALL SPRITE(#5,47,2,18
180,-23,0,#6,47,2,80,100,-
,0):: CALL MAGNIFY(2)
120 FOR I=1 TO 12 :: PRINT
$(I+(I)7)*2&(I-7))&A$(1+I+
)6)*2&(1-6)):: NEXT I :: G
O 120
```

Memory full
Jim Peterson