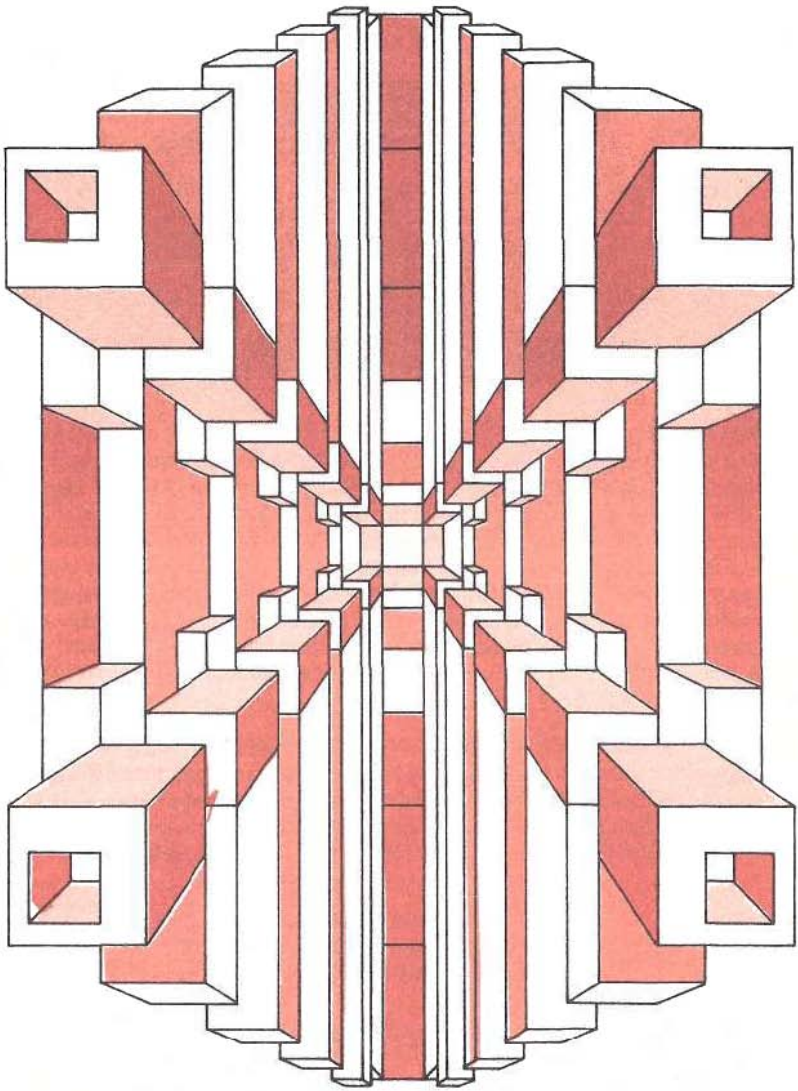
May 1979

PROG/80

Dedicated to the serious programmer

$3.00



Radio Shack gives you a TERM program with its RS-232-C board, but that's just to whet your appetite. **ST80** was written by the author of **Renumber 1.2**, so you know what kind of features to expect: CONTROL key, ESC key, REPEAT key, a RUN key, and a functioning BREAK key. Also lets you list incoming data on your lineprinter. You can reprogram the RS-232-C switches from the keyboard, making baud rate changes simple. Full upper/lower case keyboard and video driver are included, plus instructions on how to make a simple hardware modification to display upper/lower case letters (This change is optional. Unmodified TRS-80 will display capital (upper case) letters only.). Supplied on tape, loads with SYSTEM command. Disk users can also load and run the program under DOS.

Requires at least 16K Level II BASIC, a RS-232-C serial board, and a modem to work with a timesharing computer. The cursor control format includes clear screen, backspace, advance, down and up space, clear to end of line and home, using the most common control character format currently in use (similar to CDC terminals). **Level II, 16K** **Price, $49.95 on tape**

NEW ST80D FOR DISK
Yes! Data Spooling is here, and MORE . . .
For 32K Disk systems — $79.95

TSE TRS-80 Software Exchange

17 Briar Cliff Drive Milford, New Hampshire 03055

PROG/80

Dedicated to the serious programmer

May 1979

In This Issue ...

Clock Routines 6
Lance Micklus

Rescue 10
George Blank

Scattergram/Correlation 16
Gary S. Breschini

Piece of the ROM 27
Lance Micklus

Super Graphics Without Disk 29
James Garon

Boomer Box 36
Frank B. Rowlett, Jr. &
Garry W. Woodruff

Upper/lower Case Mod 47
Lance Micklus

Graphics Aid 51
Philip Brown

Squish/BAS 62
Bill Driscoll

INIT 66
George Meyer

Editor/Publisher
Roger W. Robitaille, Sr.

Contributing Editors
George Blank
Lance Micklus

Software Editor
Paul F. Johnson

Design/Production
Lee Hansen

Business Manager
Elizabeth Robitaille

Layout/Composition
Alice Scofield
Ellie Mae Erion

Subscriptions Manager
Diana Bishop

Customer Service
Bette Keenan

Correspondence
Kathleen Sullivan

Accounting
Rita Ellis

PROG/80 is published quarterly by SoftSide Publications, Milford, NH. Telephone (603) 673-5144 Subscription rates: USA regular bulk rate-$10 per year. USA first class, Canada, Mexico, APO/FPO, overseas surface mail-$14 per yr. Overseas airmail-$18 per yr. **Make all remittances payable in US funds** Mail Subscription inquiries to PROG/80 Subscriptions Manager, PO Box 68, Milford, NH 03055.

READING THE MAIL

Sometimes I wonder how anything at the main publishing office in New Hampshire ever gets done ... they do, too. But, despite its quickly-put-together appearance, PROG/80 received a warm welcome from many readers. (Remember the first issue of **SoftSide**?)

The first issue of PROG/80 contained several mistakes, including a few gross errors which we'll try to catch up on in this issue. Most of you caught the error in the upper-lower case modification article concerning the 2101 which should have read 2102. Then there was the INKEY article which mentioned a third program which never made it to print. (It was an example of how to use the routines which did appear in the feature. I'm still searching for part three ... somehow, it got misplaced.)

The clock routine we printed was the wrong routine, although the one we did print is a nice little routine you might want to put into a game program. There is more, but I'm sure you get the idea. Most of this happened because we collected the preliminiary material over a period of six months while trying to figure out how to present it. And in the interim, some of it got misplaced and some got shuffled. Also, our proofreaders had an attack of bleary eyes in the final rush to print.

This issue of PROG/80 is going to be closer to what future issues will be like. Understand that SoftSide is a source for coded, canned **software** ... load it and run it. PROG/80 is for people who want to write their own software. So, we're going to look at things in a different light.

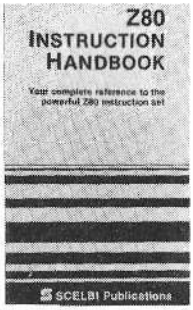
There will be games in PROG/80, but the main difference is that if a game is published, it will be selected for its value to to someone who wants to take it apart and use the ideas in his own program. In other words, the game will be the framework for presenting special subroutines — the real reason for listing the program.

Machine language will also be an acceptable area of PROG/80 interest, as will FORTRAN, and COBOL (when it becomes available).

Every reader is welcome to participate and submit articles for possible publication. We pay a reasonable rate for features and programs. Don't be hesitant to submit your ideas!

Lance

Reference



Z80 Instruction Handbook
Scelbi Publications
Convenient pocket-size manual describes Z80 capabilites in easy-to-understand terminology. Designed as a practical reference to mnemonics, machine codings and usage — for programmers of every level, from beginner to professional ... anyone working in Z80 machine or assembler language. **Price, $4.95**

The BASIC Handbook
Dr. David A. Lien
Definitive reference work explaining over 50 versions of the language in detail. All you need to know about the major statements, functions, operators, and commands pertaining to use in micro, mini and mainframe computers. **Price, $14.95**

Sargon: A Computer Chess Program
Dan & Kathe Spracklen
Documentation covering all algorithms in Sargon can be found in this comprehensive guide book. Contains table of contents, block diagram, 4 part introduction, Z80 listing and index to subroutines. **Price, $14.95**

TSE TRS-80 Software Exchange
17 Briar Cliff Drive Milford, New Hampshire 03055

CLOCK ROUTINES

by Lance Micklus

Okay disk spinners! Here's some stuff for you. If you're just a tape puller, skip this article and go on to the next one.

You can put these routines into your Disk BASIC programs to manipulate the clock. The first one will set both the time and date. It works by seeing if the month is zero, or if the clock has changed at all after waiting in a FOR-NEXT loop.

From DOS there is a command called CLOCK which causes the clock to appear in the upper right hand corner of the screen. Here are two routines in BASIC that duplicate CLOCK and CLOCK (OFF), so you can control this display from your BASIC programs.

Maybe you'd like to have the clock on but want it in the upper left hand corner. Just make X the screen position and call our next routine to move the display. Think of it as a PRINT@ statement. You can put the display on the bottom line if you want, but don't scroll. Try it and see why.

If for some reason, you'd like to see the date displayed rather than the time, we also have a routine to change that, and another to put it back to time again.

```
18000 REM *** ROUTINE TO SET TIMEDATE IF NOT ***
18020 REM *** PREVIOUSLY SET AT RUN TIME     ***
18040 A$=TIME$
18060 FOR J=0 TO 1000 : NEXT J
18080 IF A$=TIME$ CMD"R" : GOTO 18120
18100 IF PEEK(16453)<>0 THEN 18200
18120 PRINT"ENTER TIME (HH:MM:SS)?";:LINEINPUT A$
18140 POKE 16451,VAL(LEFT$(A$,2))
18160 POKE 16450,VAL(MID$(A$,4,2))
18180 POKE 16449,VAL(RIGHT$(A$,2))
18200 INPUT"ENTER DATE (MM/DD/YY)";A$
18220 POKE 16454,VAL(LEFT$(A$,2))
18240 POKE 16453,VAL(MID$(A$,4,2))
18260 POKE 16452,VAL(RIGHT$(A$,2))
18280 RETURN
18300 REM *** ROUTINE TO TURN CLOCK OFF ***
18320 CMD"T"
18340 POKE 17676,163 : POKE 17677,69
18360 CMD"R"
18380 RETURN
18400 REM *** ROUTINE TO TURN THE CLOCK ON ***
18420 CMD"T"
18440 POKE 17676,169 : POKE 17677,76
18460 CMD"R"
18480 RETURN
18500 REM *** ROUTINE TO MOVE CLOCK DISPLAY ***
18520 IF X<0 OR X>1015 RETURN
18540 X=15360+X
18560 POKE 16638,INT(X/256)
18580 POKE 16637,X-256*INT(X/256)
18600 RETURN
18620 REM *** ROUTINE TO SWITCH TO DATE ***
18640 POKE 19640,70
18660 POKE 19643,ASC("/")
18680 RETURN
18700 REM *** ROUTINE TO SWITCH TO TIME ***
18720 POKE 19640,67
18740 POKE 19643,ASC(":")
18760 RETURN
```

A UTILITY PROGRAM THAT GREATLY EXTENDS THE KEYBOARD, VIDEO, AND PRINTER SUBROUTINES IN YOUR LEVEL II ROM!

by Lance Micklus

KVP runs under DOS or Level II BASIC. It is relocatable under your control, and so may be used simultaneously with other machine language programs. At least 16K of memory is required.

Here are some of the things you'll be able to do:

- **USE AN EXTERNAL KEYBOARD** Or, use any other serial input device in place of the TRS-80 keyboard
- **SIMULATE A RADIO SHACK SCREEN PRINTER** using an ordinary printer
- **USE MOST ANY ASCII SERIAL PRINTER** such as Teletype 33 or Spinterm
- **ELIMINATE A COMMON SOURCE OF PROGRAM ERRORS** by running your keyboard in upper case only, or run in upper/lower case mode just like a typewriter
- **TELL THE TRS-80 YOU HAVE NO PRINTER AT ALL**
- **PRACTICALLY ELIMINATE KEYBOARD BOUNCE** The amount of debouncing is user-adjustable
- **EXCHANGE PROGRAMS WRITTEN IN BASIC WITH OTHER COMPUTERS** *From the Sorcerer to the* IBM 370 (and TRS-80's, too!)
- **DISPLAY UPPER AND LOWER CASE LETTERS** on your video monitor screen
- **THE LIST GOES ON AND ON!**

Self-relocating for 16K, 32K or 48K systems

$24.95 on tape $29.95 on disk

TSE TRS-80 Software Exchange

17 Briar Cliff Drive Milford, New Hampshire 03055

RESCUE

by George Blank

What is frustration? Here is my definition: frustration is spending hours entering a program from the keyboard and then losing it because your system crashes, or you type LLIST and have no printer. It has happened to me many times, most often from the LLIST problem, but sometimes because my washing machine shuts off and sends a voltage transient up the line, or even when I turn off my electric typewriter. Sometimes my system crashes to disk for no identifiable reason...I'm tired of retyping programs, so I wrote **RESCUE** instead.

Rescue is short machine language program that will restore a BASIC program in Disk BASIC after your system crashes to DOS, or locks up on LLIST when you have no printer, or even after you press RESET or type NEW.

The BASIC program is **NOT** RESCUE. Instead, it will create a machine language program for you and provide instructions on using TAPEDISK to save it. It will also give you instructions on the use of the program. Write them down for future reference.

Essentially, here is the method: with a BASIC program in memory, press RESET or type NEW to lose your program. If Rescue isn't in memory, return to DOS and type LOAD''RESCUE''. Now return to BASIC, protecting memory as required. On a 32K machine you would answer MEMORY with 49088. Now thpe SYSTEM, and answer ?* with /49088. The program should execute and return you to BASIC. Then type RUN and you've got your program back.

This is set up for Disk Basic only. The same program would work in Level II BASIC if the memory locations START, MSB1, MSB2, and the first pointer were changed to reflect the beginning of Level II BASIC at (Hex) 42E9 instead of 68BA.

All the computer does to your old program in these cases is simply reset four program pointers to their original values. It doesn't disturb your program until you load something else on top of it or turn off the computer. If you reset the pointers to their program values, you can have your program back. Rescue will do that for you in only a few seconds.

All it does is search your program for the begining of the second line, restore the second line pointer, search for the end of your program, set the end of program pointer, simple variable pointer, and the array variable pointer to the end of the program (in the process wiping out your variable values) and return you to BASIC. Then you type RUN and your program is operating again.

You may not use DIR or other commands that use memory after the start of BASIC programs without permanently losing your program. But you can return to DOS, load the Rescue machine language code into high memory (NOT the BASIC program here, which will destroy your program, but the short machine language program it will create), return to BASIC, protecting memory for the subroutine, type SYSTEM, then enter the starting address and you have your program back. You can LIST it or RUN it.

The program is set up for Disk BASIC, as the problem is not as severe to Level II users. However, it could be converted for Level II. All you have to do is change START to 42E9 , MSB1 to 42, MSB2 to 43, and the address PT1 to 42E9. (All these values are Hexadecimal, and must be in decimal form for the program.)

Here are the pointers:

Address of Second Program Line	68BA (Disk) 42E9 (Level II)
End of Program	40F9
End of Simple Variables	40FB
End of Array Variables	40FD

To use the program, you must create the machine language code and save it on Disk or system tape prior to the need for it. If you wish, you could just load it using the BASIC program here if you do it before you load the program you will be using. Write down the instructions the program gives you, as they will be specific for your computer. Then, when you need to RESCUE a program, load RESCUE, return to BASIC (protecting the high memory), type SYSTEM and the starting address.

You may find other uses for RESCUE. I use it to find out how much memory my programs use. I press BREAK in the middle of a program (when all the variables and the string space is in use), then type ? MEM, NEW, ?MEM, then I subtract the first MEM value from the second, type SYSTEM, /49088 (I have a 32K computer) and restart my program with RUN. •

```
       * TRS-80 RESCUE PROGRAM CREATOR *
       * GEORGE BLANK, LEECHBURG, PA   *
       * LD HL,START   INC HL   LD A,(HL)   CP 0 *
10 DATA  33,191,104,   35,      126,        254,0
19 REM
       * JR NZ,LP1  INC HL   INC HL   LD A,(HL)  CP 68H *
20 DATA  32,250,    35,      35,      126,       254,104
29 REM
       * JR Z,(+4)  CP 69H   JR NZ,LP1  DEC HL   LD PT1,HL *
30 DATA  40,4,      254,105,  32,239,    43,      34,186,104
39 REM
       * INC HL   INC HL   LD A,(HL)  CP 0   JR NZ,LP2 *
40 DATA  35,      35,      126,       254,0,   32,250
49 REM
       * INC HL   LD A,(HL)  CP 0   JR NZ,LP2 *
50 DATA  35,      126,       254,0,   32,244
59 REM
       * INC HL   LD A,(HL)  CP 0   JR NZ,LP2 *
60 DATA  35,      126,       254,0,   32,238
69 REM
       * INC HL   LD A,(HL)  CP 1CH   JR NZ,LP3 *
70 DATA  35,      126,       254,28,   32,246
79 REM
       * INC HL   LD PT2,HL   LD PT3,HL   LD PT4,HL *
80 DATA  35,      34,249,64,  34,251,64,  34,253,64
89 REM
       * JP BASIC *
90 DATA  195,25,26
92 REM
       * POINTER TABLE
       * PT1  POINTER TO SECOND LINE OF PROGRAM  68BA (HEX)
       * PT2  END OF PROGRAM POINTER  40F9
       * PT3  SIMPLE VARIABLES POINTER  40FB
       * PT4  ARRAY VARIABLES POINTER  40FD
99 REM
       * MEMORY PROTECT FEATURE *
100 CLS:PRINT"RESCUE SUB-ROUTINE"
110 PRINT"THIS ROUTINE WILL ONLY WORK IF YOU ANSWERED"
120 PRINT"MEMORY?";CHR$(95);"    WITH ";
128     REM * MM=MSB OF SUB-ROUTINE LOCATION *
            * MA=POKE ADDRESS                *
```

```
129     REM * DETERMINE MEMORY IN COMPUTER   *
130 POKE(-3),83:IF PEEK(-3)=83 THEN MM=255:MA=-64:M$="65472":GOT
0200
140 POKE(-16387),83:IF PEEK(-16387)=83 THEN MM=191:MA=-16448:M$=
"49088":GOTO200
150 POKE(32765),83:IF PEEK(32765)=83 THEN MM=127:MA=32704:M$="32
704":GOTO200
160 PRINT"YOU DONT HAVE ENOUGH MEMORY":STOP
200 PRINTM$:PRINT:PRINT"PRESS ENTER TO LOAD, STARTING AT ";MA
210 INPUT A$
298     REM * M= LOCATION IN MEMORY    *
            * MV= MEMORY CONTENTS       *
299     REM * POKE PROGRAM INTO MEMORY *
300 FOR M=MATOMA+61
310  READ MV:IF MV=9 THEN LET MV=MM
330  POKE M,MV
340  PRINT MV;
350 NEXT M
400 PRINT:PRINT"THIS PROGRAM CANNOT BE USED IN BASIC, AS IT DESTR
OYS YOUR"
410 PRINT"PREVIOUS PROGRAM.  YOU MUST FIRST SAVE IT ON DISKETTE A
S"
420 PRINT"A MACHINE LANGUAGE PROGRAM."
430 INPUT"PRESS ENTER FOR INSTRUCTIONS";A$
500 CLS
510 PRINT"TO USE THE PROGRAM:"
520 PRINT:PRINT"    IF IT IS NOT IN MEMORY, YOU MUST LOAD IT FRO
M DOS:":PRINT"(IF NOT IN DOS, PRESS RESET)":PRINT
530 PRINT"TYPE:","  ","LOAD";CHR$(34);"RESCUE";CHR$(34)
540 PRINT"  ","  ","BASIC"
550 PRINT"ANSWER MEMORY?";CHR$(95);"  WITH",M$:PRINT
560 PRINT"ONCE PROGRAM IS IN MEMORY:":PRINT"TYPE:","  ","SYSTEM"
570 PRINT"ANSWER  *?";CHR$(95);"  WITH","/";M$
580 PRINT:PRINT"    IT IS SUGGESTED THAT YOU WRITE DOWN THESE IN
STRUCTIONS"
590 PRINT"AND SAVE THEM FOR WHEN YOU USE THE PROGRAM.":INPUT"  (
PRESS ENTER)";A$
600 CLS:PRINT
610 PRINT"SAVE THIS PROGRAM SO THAT YOU DON'T HAVE TO RETYPE IT,
"
```

```
620 PRINT"IF YOU LOSE IT, THEN SAVE THE MACHINE CODE AS FOLLOWS:
"
630 PRINT"(WRITE INSTRUCTIONS DOWN)"
660 PRINT" "," ","TAPEDISK"
670 PRINT"ANSWER  *?";CHR$(95);"  WITH",
680 IF MM=255 PRINT"F RESCUE:0 FFC0 FFFD FFC0"
690 IF MM=191 PRINT"F RESCUE:0 BFC0 BFFD BFC0"
700 IF MM=127 PRINT"F RESCUE:0 7FC0 7FFD 7FC0"
710 PRINT" "," ","E"
720 CMD"S"
```

BASIC STATISTICS

This powerful set of procedures is of use to students, instuctors, behavioral and research scientists, statisticians — anyone using these statistical formulas for practical or research applications:

RANK-ORDER DATA A simple program utilizing a Shell-Metzner sorting routine to rank data in an ascending manner.

CENTRAL TENDENCY Given a set of raw data, this program ranks and displays raw data (optional), N, X, X , variance, standard deviation, the Median, and the Mean.

PEARSON PRODUCT-MOMENT CORRELATION COEFFICIENT Given N pair (X,Y) of data, the program computes mean, standard deviation for S and Y, and R. An option is available to utilize a regression equation to predict Y given any value of X.

CHI-SQUARE Given raw data for any number of rows and column, the program will optionally display a raw data printout with observed and expected values; row, column, and grand totals; and gives the used CHI and DF.

FISHER T-TEST Given 2 sets of raw data for either equal or unequal N, the program computes and displays N, mean, standard deviation and standard error of the mean for both data samples as well as T and DF.

SIMPLE ANALYSIS OF VARIANCE Given raw data for any number of conditions, the program computes and displays N, Mean and Standard Deviation for each condition as well as SSbg, SSwg, SStot, DFbg, DFwg, DFtot, MSbg, MSwg, and the F.

Z-SCORES AND STANDARD SCORES Given N scores, the program computes a Z-score for each N. The user has an available option to compute a standard score for each N given the desired Population Mean and S.D.

RANDOM NUMBER GENERATOR Given the upper and lower limits, this program produces a list of N random numbers useful in research and experimental design.

NOTE: The basic formulas for these major statistical procedures were derived from the textbook, "Elementary Statistics", by Janet T. Spencer, Benton J. Underwood, Carl P. Duncan, and John W. Cotton. Appleton - Century - Crofts Psychology Series, New York, 1968.

Available on Digital Audio Cassette for the Level II TRS-80 Microcomputer - $20.00

TSE TRS-80 Software Exchange

17 BRIAR CLIFF DRIVE MILFORD, NEW HAMPSHIRE 03055

SCATTERGRAM/ CORRELATION PROGRAM

by Gary S. Breschini

This program is written in Level II BASIC for the TRS-80 and occupies approximately 3K. It provides a scatter diagram (scattergram) of pairs of data on a coordinate-axis system. In addition, it displays the correlation coefficient (coefficient of linear correlation), r. This is a measure of the strength of relationship between two variables ... determined by the effect a change in one has on the other. The upper and lower values of both variables can be specified. The data is entered using the INKEY$ function, eliminating the necessity for pressing the **ENTER key**, and allowing the display to be constructed at the beginning of the data run and points to be added one by one.

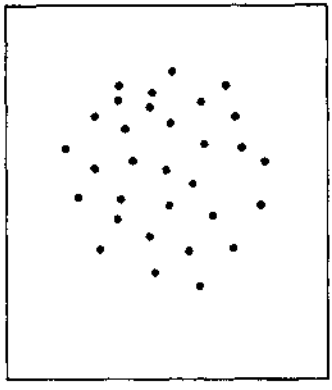
The scattergram is designed to display bivariate data, that is, data consisting of two items of data which are variable but which are in some way related. The two items of data are usually called X and Y and are generally paired, such as the height and weight of an individual, or the length and width of an object, etc. The scatter diagram is a convenient method for pictorial display of the pairs of data X and Y on a coordinate-axis system (graph).

Once the points are displayed on the scattergram, they can be checked visually and mathematically for interrelationships. The visual examination can use the sample plots below for comparison, while the mathematical test for linear correlation (r) is performed. As was pointed out by Johnson (1976):

The primary purpose of correlation analysis is to measure the strength of linear relationship between variables. The coefficient of linear correlation, r, is the measure of strength of linear relationship between two variables. This strength of relationship is determined by the amount of

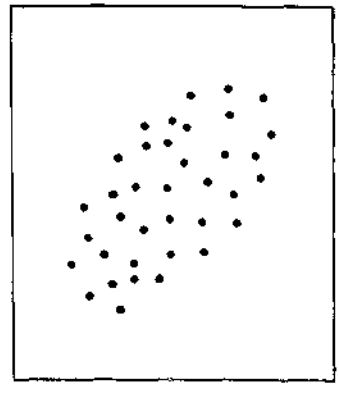
EXAMPLES OF LINEAR RELATIONSHIPS

Fig. 1



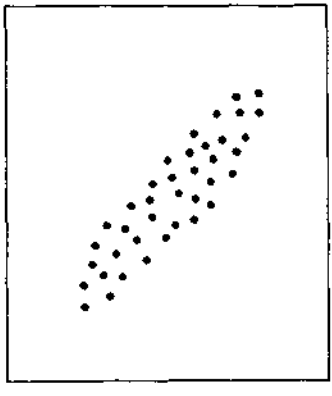
NO CORRELATION

Fig. 2



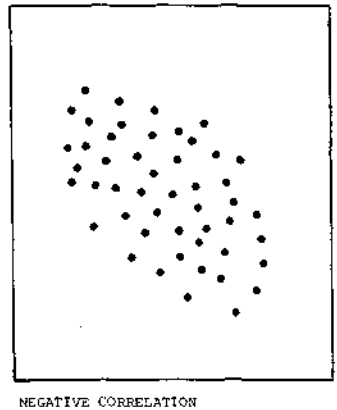
POSITIVE CORRELATION

Fig. 3



HIGH POSITIVE CORRELATION

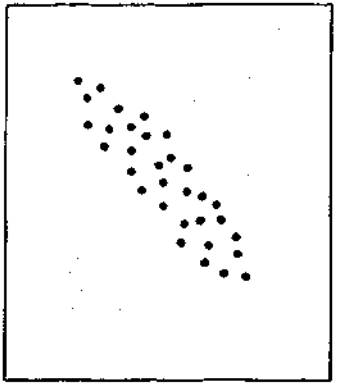
Fig. 4



NEGATIVE CORRELATION

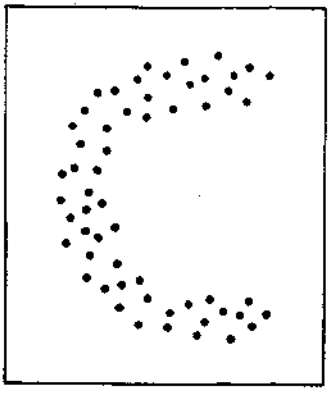
EXAMPLES continued

Fig. 5



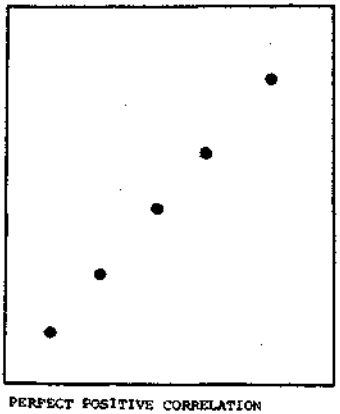
HIGH NEGATIVE CORRELATION

Fig. 6



NO LINEAR CORRELATION (a definite pattern, but one which is not linear.)

Fig. 7



PERFECT POSITIVE CORRELATION

effect any change in one variable has on the other. Let's use X and Y as our variables. If as X increases there is no definite shift in the values of Y, we will say there is a lack of correlation. If as X increases there is a definite shift in the values of Y, we will say that the correlation is [a] positive when Y tends to increase, [b] zero whey Y tends to stay at about the same value, or [c] negative when Y tends to decrease. The preciseness of this shift will determine the strength of the linear correlation [1976:100-101].

The formula used for the calculation of r was:

$$r_{xy} = \frac{S_{xy}}{S_x S_y} \quad \text{where:} \quad S_{xy} = \frac{1}{n-1}\left[\sum xy - \frac{1}{n}\sum x \sum y\right] \text{ and:}$$

$$S_x = \sqrt{\frac{\sum x^2 - \left[\sum x\right]^2 / n}{n-1}} \qquad S_y = \sqrt{\frac{\sum y^2 - \left[\sum y\right]^2 / n}{n-1}}$$

The n-1 method was used for calculating the standard deviations (S and S) so that the formulas are suitable for use with small sample sizes.

Once the values of X and Y have been entered, and r calculated, it is necessary to interpret the results. The seven examples given of linear correlation can be compared to the scattergrams created with actual data and an estimate of the correlation can be made. The correlation coefficient, r, expresses the relationship between the variables, and ranges from a-1 to a+1. The -1 expresses a perfect negative correlation, while the +1 expresses a perfect positive correlation (**Fig.7**). All other plots will occur between these two extremes.

When the value of r is close to either -1 or +1 we suspect a linear correlation, and when the value of r is near 0 we suspect that there is no linear correlation. In order to make a decision as to whether or not given values of r signify a linear correlation, we use Table 1. The use of this is demonstrated by **Fig. 8**.

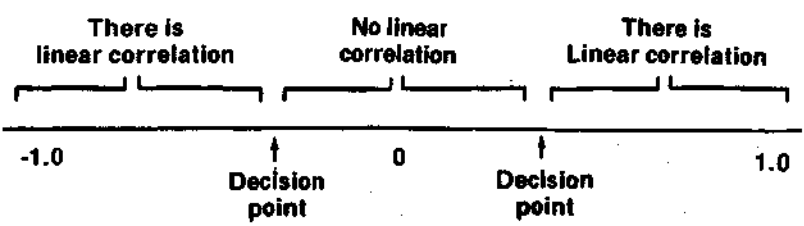


Fig. 8 DECISION POINTS

If the calculated value of r lies between the decision point and either -1 or +1 there is assumed to be a linear correlation. If the value of r is between the two decision points, no linear relationship can be assumed. The decision points are found through the use of Table 1. Using the sample size (the number of pairs entered) locate or estimate the decision point in Table 1 for that sample size. If the value of r lies between the decision point and either -1 or +1, the relationship appears to be linear. If the value of r lies between the decision point and 0 no linear relationship can be assumed.

TABLE 1

n	Decision point	n	Decision point	n	Decision point	n	Decision point
5	0.878	12	0.576	19	0.456	30	0.361
6	0.811	13	0.553	20	0.444	40	0.312
7	0.754	14	0.532	22	0.423	50	0.279
8	0.707	15	0.514	24	0.404	60	0.254
9	0.666	16	0.497	26	0.388	80	0.220
10	0.632	17	0.482	28	0.374	100	0.196
11	0.602	18	0.468				

Caution must be utilized in the interpretation of the results of this statistical procedure. Because there appears to be a linear relationship between two variables, it is tempting to assume a cause and effect relationship. Knowledge of the particular data is necessary to decide whether or not a true cause and effect relationship is likely. For example: if the height and weight of several individuals is plotted, and a linear relationship is suggested, knowledge of the situation would lead to the assumption that the two are related in a cause and manner. If, on the other hand, the program suggested that there was a linear relationship between increasing population in the United States over the past fifty years and increase in deaths from heart attacks during that same period, it would be rash to assume that there is a cause and effect relationship between the two factors without an intimate knowledge of the situation. Many factors can be found to be related in a linear fashion which have no cause and effect relationship.

The program is initiated by a RUN statement. It then asks, "DO YOU WANT INSTRUCTIONS (YES=1)?". At this point, an instruction set can be accessed or the program can be entered by pressing **ENTER**. The user is then instructed "INSERT MAXIMUM VALUE OF X?". At this point the user can insert the maximum X value desired on the scattergram. Following this, the minimum value of X and the same values for Y can be entered. This determines the coordinates of the scattergram and makes sure that (1) all points fall within the scattergram, and (2) that the scale is large enough to give a good visual separation between points.

When the program begins, a coordinate-axis system is drawn, complete with values of X and Y at points along the axes. At the bottom left of the grid is displayed "X(#####) Y(#####)". When values of X and Y are entered, the #'s will change one by one to the input values. Values greater than 9999 cannot be input, nor can values larger than that be used for establishing the size of the grid. Leading or trailing blanks must be input, but the position within the input rows is not critical. Decimal points can also be inserted. When all ten of the #'s have been replaced by either data, zeros or blanks, the data will automatically be entered into the calculations. At this point, the "X(#####) Y(#####)" will be replaced, signifying that new data may be entered. After the fifth entry of data, the correlation coeffiecient will be displayed and will be updated after each new entry of data. After each pair of numbers is entered, the point on the coordinate-axis which best corresponds to the point will be lighted, creating the scattergram. If a mistake is made in the entry of data, place an X in the last position of the Y row. This causes the entry to be ignored. Values of X and Y larger or smaller than the stated limites will also be ignored, and an error message will be displayed along the bottom right edge of the display.

The entry of numbers pairs which have one of the two numbers the same, such as **11,44 13,44 23,44 19,44 32,44** will cause a divide by 0 error, as the standard deviation of X and Y is used as a divisor in calculating r, and with one number always the same, this results in an attempt to divide by zero. To avoid this, the numbers entered must not have either the X or Y values all the same. (In some cases, values which are very near to the above condition will cause the divide by zero error as they will be rounded to zero during calculations. In any case, the correlation between such numbers is either 0 or so close to it that it is rounded to 0, as for any change of one variable there is no change in the other (i.e. no correlation.)

REFERENCES

Doran, J. E. andF. R. Hodson

1975 **Mathematics and Computers in Archaeology.** Harvard University Press, Cambridge, Mass.

Johnson, R.

1976 **Elementary Statistics, 2nd Edition.** Duxbury Press, North Scituate, Mass.

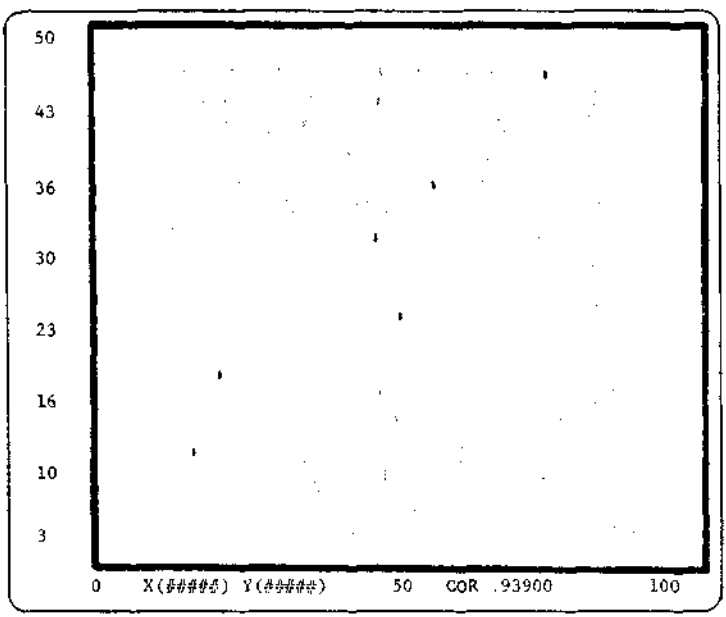
Steel, R.G.D. and Torrie, J.H.

1960 **Principles and Procedures of Statistics.** McGraw-Hill Book Co., New York.

After the initial display, the values of X and Y are entered.

```
   Numbers entered              Displayed

54      36                 X(00054) Y(  36 )   COR
50      23                 X(   50) Y(23   )   COR
43      30                 X(43    ) Y( 30  )  COR
22.4    18.6               X(22.4  ) Y( 18.6)  COR
75.18   44.44              X(75.18) Y(44.44)   COR   .90531
18      10                 X(18    ) Y(10   )  COR   .93900
```

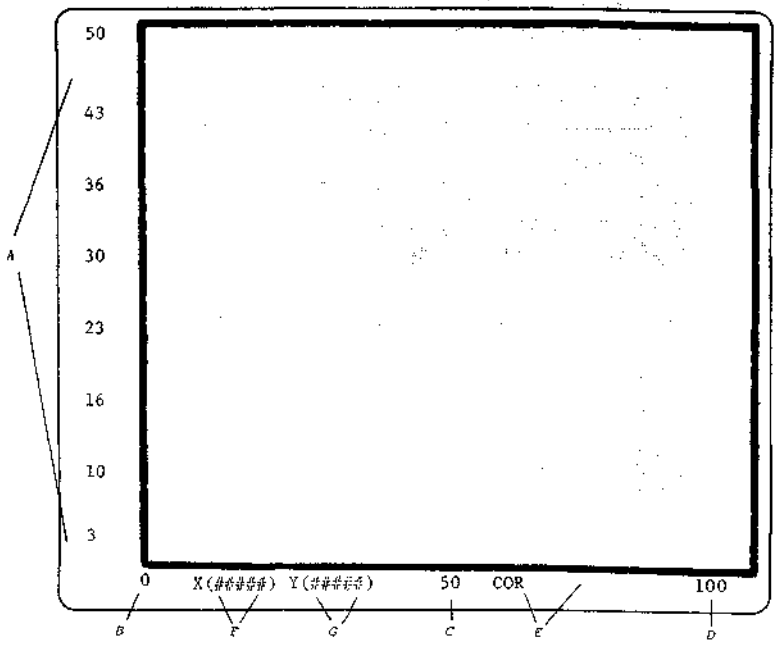


As can be seen from the data which was entered, leading blanks or zeros, and any needed trailing blanks or zeros must be entered, but the position of the data within the input rows is noncritical.

After entering the needed data, a BREAK and RUN will restart the program for a second run.

SAMPLE RUN: The following is an example of a run made with this program.

```
DO YOU WANT INSTRUCTIONS (YES=1)?  press enter to continue
INSERT MAXIMUM VALUE OF X?   100
INSERT MINIMUM VALUE OF X?   0
INSERT MAXIMUM VALUE OF Y?   50
INSERT MINIMUM VALUE OF Y?   0
```



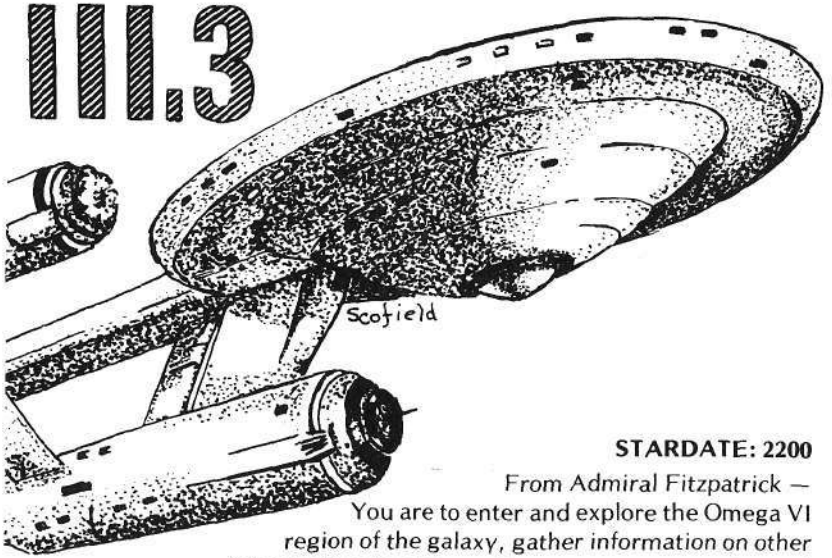
EXPLANATION OF DISPLAY FEATURES

A = Values of Y from highest to near lowest
B = Lowest value of X
C = Intermediate value of X
D = Highest value of X
E = Location of correlation coefficient
F = Input row for X
G = Input row for Y

```
1 REM SCATTERGRAM/CORRELATION PROGRAM COPYRIGHT 1979 BY GARY S.
BRESCHINI, 379 CORRAL DE TIERRA RD, SALINAS, CA 93908 1/79
3 CLS:PRINT@81,"SCATTERGRAM & CORRELATION":PRINT:PRINT:INPUT"DO
YOU WANT DIRECTIONS (YES=1)";A:IFA=1GOSUB1400
5 CLS:INPUT"INSERT MAXIMUM VALUE OF X";XH
6 INPUT"INSERT MINIMUM VALUE OF X";XL:XX=XH-XL
8 IFXL>XHPRINT"LOWER LIMIT EXCEEDS UPPER LIMIT.":GOSUB5000:GOTO5
12 IFXH>9999PRINT"X UPPER LIMIT TOO LARGE.":GOSUB5000:GOTO5
15 INPUT"INSERT MAXIMUM VALUE OF Y";YH
20 INPUT"INSERT MINIMUM VALUE OF Y";YL:YY=YH-YL
25 IFYL>YHPRINT"LOWER LIMIT EXCEEDS UPPER LIMIT":GOSUB5000:GOTO1
5
30 IFYH>9999PRINT"Y UPPER LIMIT TOO LARGE.":GOSUB5000:GOTO15
40 ZX=XX/114:ZY=YY/43
50 CLS:FORX=12TO127:SET(X,0):SET(X,44):NEXT
60 FORY=0TO43:SET(12,Y):SET(127,Y):NEXT
70 PRINT@965,XL;:PRINT@971,"X(#####) Y(#####)";:PRINT@1011,"
      ";
75 A$="0":B$="0":C$="0":D$="0":E$="0":F$="0":H$="0":I$="0":J$="0
0":X$="0":Y$="0"
80 PRINT@1017,XH;:PRINT@993,INT(XL+(XH-XL)/2);
90 PRINT@1000,"COR";
95 I=(YH-YL)/15
100 FORX=0TO14STEP2:PRINT@(0+(64*X)),INT(YH-(X*I));:NEXT
1000 A$=INKEY$:IFLEN(A$)=0THEN1000
1020 PRINT@973,A$;
1030 B$=INKEY$:IFLEN(B$)=0THEN1030
1040 PRINT@974,B$;
1050 C$=INKEY$:IFLEN(C$)=0THEN1050
1060 PRINT@975,C$;
1063 D$=INKEY$:IFLEN(D$)=0THEN1063
1066 PRINT@976,D$;
1080 E$=INKEY$:IFLEN(E$)=0THEN1080
1090 PRINT@977,E$;
1095 X$=A$+B$+C$+D$+E$
1100 F$=INKEY$:IFLEN(F$)=0THEN1100
1110 PRINT@982,F$;
1120 G$=INKEY$:IFLEN(G$)=0THEN1120
1130 PRINT@983,G$;
1133 H$=INKEY$:IFLEN(H$)=0THEN1133
```

```
1136 PRINT@984,H$;
1150 I$=INKEY$:IFLEN(I$)=0THEN1150
1160 PRINT@985,I$;
1170 J$=INKEY$:IFLEN(J$)=0THEN1170
1180 PRINT@986,J$;
1190 Y$=F$+G$+H$+I$+J$
1200 IFJ$="X"GOTO70
1300 Q=Q+1:C=VAL(X$):D=VAL(Y$)
1302 K=C:L=D:C=C-(XL):D=D-YL:C=(C/ZX)+13:D=D/ZY
1304 IF K<XL THEN PRINT@1011,"X TOO SMALL";:GOSUB5000:GOTO70
1305 IFL<YLTHENPRINT@1011,"Y TOO SMALL";:GOSUB5000:GOTO70
1307 IFC>127THENPRINT@1013,"X TOO BIG";:GOSUB5000:GOTO70
1308 IFD>43THENPRINT@1013,"Y TOO BIG";:GOSUB5000:GOTO70
1320 D=44-D:SET(C,D):PRINT@971,"X(#####) Y(#####)";
1325 GOSUB1350:PRINT@1005,"      ";:PRINT@1004,CC;:GOTO70
1350 N=N+1:SX=SX+K:SY=SY+L:AA=AA+(K*K):BB=BB+(L*L):XY=XY+(K*L)
1352 IFQ=<4GOTO70
1355 XD=SQR((AA-((SX*SX)/N))/(N-1))
1357 YD=SQR((BB-((SY*SY)/N))/(N-1))
1360 CV=1/(N-1)*((XY)-((1/N)*(SX*SY)))
1365 CC=CV/(XD*YD):RETURN
1400 PRINT"THIS PROGRAM WILL DISPLAY A SCATTERGRAM USING VALUES
OF X AND Y. IT WILL ALSO PROVIDE THE CORRELATION COEFFICIENT BETW
EEN THE    VALUES OF X AND Y.  SET THE UPPER AND LOWER LIMITS DE
SIRED FOR  BOTH X AND Y AND INSERT VALUES ";
1410 PRINT"OF X AND Y.  THE DISPLAY WILL    PLOT THE POINT AND A
FTER THE FIFTH POINT WILL BEGIN DISPLAYING  THE CORRELATION COEF
FICIENT.  VALUES LARGER THAN 9999 CANNOT BE ENTERED.  IF YOU HAV
E VALUES LARGER, DELETE THE LAST DIGIT(S).";
1420 PRINT"  STANDARD DEVIATIONS (IN THE CORRELATION FORMULAS) A
RE CALCULATEDUSING THE N-1 METHOD."
1430 PRINT:INPUT"PRESS ENTER TO CONTINUE";A
1440 PRINT"THERE IS NO NEED TO PRESS ENTER WHEN INSERTING VALUES
 OF X & Y.  MERELY INSERT 2 NUMBERS (THEY WILL BE DISPLAYED AS TH
EY ARE ENTERED).  LEADING (OR TRAILING) ZEROS OR BLANKS MUST BE
INSERTED.  ";
1445 PRINT"DECIMAL POINTS CAN BE ENTERED IN PLACE OF ONE OF THE
5 DIGITS.  TYPING AN X IN PLACE OF THE LAST DIGIT WILL CAUSE THE
 ENTRY TO  BE IGNORED (IN CASE OF ERRORS)."
1450 INPUT"PRESS ENTER TO CONTINUE";A:RETURN
5000 FORX=1TO2000:NEXT:RETURN
```

STAR TREK III.3



STARDATE: 2200

From Admiral Fitzpatrick —

You are to enter and explore the Omega VI region of the galaxy, gather information on other inhabitable planetary systems you may encounter and defend yourself against hostiles in case of attack. You are in command of the Starship ENTERPRISE and her ship's complement of 371 officers and crew. Omega VI is composed of 192 quadrants containing star systems and planets (a few habitable). Information on Omega VI is sketchy, but astronomical hazards such as pulsars, Class 0 stars and black holes are known to be present in the region.
It is also patrolled by Klingon battle cruisers,
so look before you leap.

Specs: Star Trek III.3

Play Board: 8 by 8 by 3 quadrants
Weapons Systems: Phasers and Photon Torpedoes
Power Systems: Warp and Impulse
Computer Systems: Science and Ship's computer
Sensors: Long and Short Range
Reports: Damage Control and Status
Play Elements: 20 Klingon battle cruisers, 100 stars and planets, black holes, pulsars

Available on Digital Cassette for Level II, 16K — $14.95

TSE TRS-80 Software Exchange

A PIECE OF THE ROM

by Lance Micklus

KEYBOARD ROUTINES

One of the first problems encountered by a machine language programmer is how to get the data from the keyboard. There are three very nice routines in the BASIC ROM which can do this. When using any of the routines, be sure you save DE and IY registers before the call.

The first call is the standard one in the Editor/Assembler instructions. The address to call is X'2B'. This is an inkey routine. If none of the keys are pressed, the A register will contain a zero. If a key is pressed, the A register contains the byte value of that key.

By ORing the A register on the return from the call, we can test the Z flag to see if any key was pressed. If we must wait until a key is pressed, then we just jump back to the call to address X'2B',, or else fall through and do something with the key that was pressed.

Here's what the routine looks like in assembler language:

```
LOOP1   CALL    2BH
        OR      A
        JR      Z,LOOP1
```

A second way to do it is to call address X'49'. This call gets a character from the keyboard also, but until a key is pressed, the wait loop is built in.

You just call it and when you return, the A register will contain the byte.

The third method is really nice. You must create a buffer in your program for a line of text from the keyboard to go. Load the start address of the buffer in register pair HL. Load register B with the size of the buffer *minus one byte. If you forget to subtract out the one byte, then the* RETURN character might overflow the buffer since it isn't included in the byte count. Now you make a call to address X'40'.

The ROM routine will get a line of text, display it on the screen while it's being entered and take care of RUB OUTS. It returns when either the **BREAK** or **ENTER** key is pressed. It's up to you to figure out what to do if the **BREAK** key was pressed.

In your buffer will be the line that was typed in. Register B will tell you how many characters are in the buffer, excluding the RETURN character but including the **BREAK** character X'01'.

SUPER GRAPHICS WITHOUT DISK

by James Garon

In the first issue fo PROG/80 there was an excellent article by George Blank called **Super Graphics**. Of course, it only works if you have a disk, but it makes SUPER graphics MUCH easier. Since I don't have a disk, I decided to write this BASIC program to act as a "graphics editor".

Save the program on tape and load it in **before** you write a program using **Super Graphics**.

Graphics Editor allows you to step through your Level II program (which may use line numbers up through 59999) and to alter the program in ways not possible by use of the EDIT command. In particular, it allows you to POKE tokens (decimal 128 through 255) into your program, where they'll replace characters in a string. This means there must be a "dummy" string in your program to receive these tokens.

To use Graphics Editor, type RUN 60000. The editor will ask, "WHICH LINE NUMBER?". Respond with the line number containing the "dummy" string. A heading will appear:

LOC. DEC. CHR$ NEW

LOC. is the location of the first byte in the line
DEC. is the decimal value of that byte
CHR$ shows **a.** how that byte would appear in a string (if DEC. is between 32 and 191) OR **b.** if the byte is a CONTROL character (0-31) OR **c.** if the byte is a SP. COMP. (space compression) code 192-255. (These codes cause 0 through 63 blanks to be printed)

A question mark appears under NEW to prompt you to enter one of the following:
ENTER to leave that byte unchanged and display the next

↑ to display the previous byte (unless the byte currently displayed is the first byte of the line)

1 to 255 This value will replace the old value at the location, and if it is between 129 and 191, the corresponding graphic character will appear to the right. Note that consecutive graphic characters are staggered to make them easier to identify (This applies to the CHR$ column also)

0 to exit the Editor before the end of the line is reached

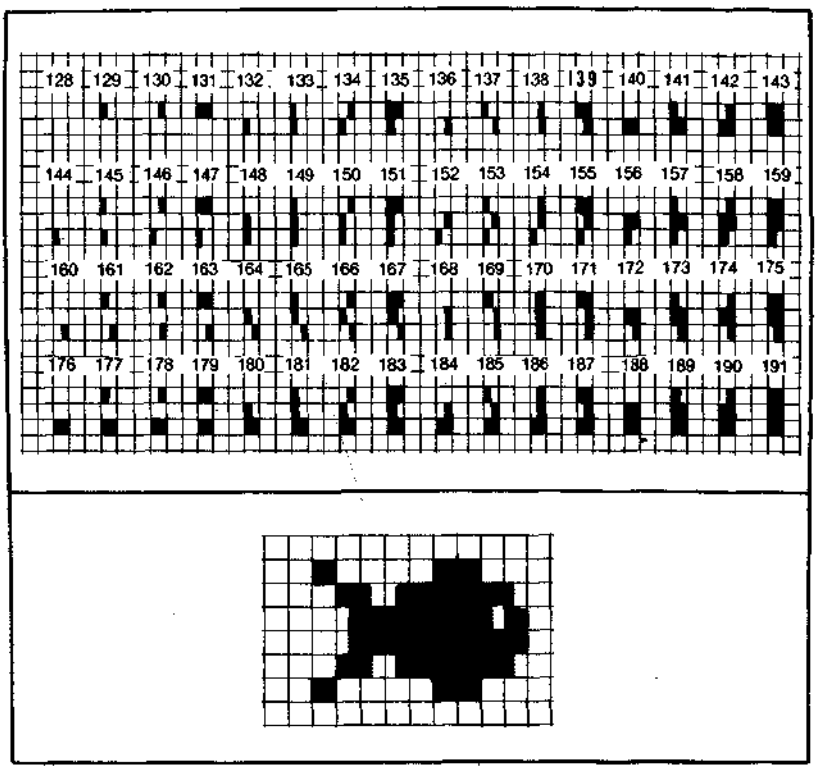
```
60000 CLS:REM SUPER-GRAPHICS
      FOR THOSE WITH NO DISK
60010 INPUT"WHICH LINE NUMBER";N:
 E=17129:IF N>59999 THEN 60030
60020 S=PEEK(E)+256*PEEK(E+1):
    L=PEEK(E+2)+256*PEEK(E+3):
    IF L<N THEN E=S:GOTO 60020
    ELSE IF L=N THEN 60040
60030 PRINT"LINE NOT IN PROGRAM":
    GOTO 60010
60040 PRINT
    " LOC.    DEC.      CHR$","NEW
60050 FOR I=E+4 TO S-2
60060 X=PEEK(I):PRINT I;X,;
    IF X<192 AND X>128 PRINT
    TAB(POS(0)+4*(I/2-INT(I/2)))
    CHR$(X),:GOTO 60080
60070 IF X=32 PRINT"BLANK",ELSE
    IF X<32 PRINT"CONTROL",ELSE
    IF X>191 PRINT"SP. COMP."X-192,
    ELSE PRINT CHR$(X),
60080 X$="":INPUT X$:IF X$=""THEN
    60120 ELSE IF ASC(X$)=91
    AND I>E+4 I=I-1 ELSE 60100
60090 GOTO 60060
60100 X=VAL(X$):IF X=0 END ELSE
    IF X<0 OR X>255 PRINT"ERROR":
    GOTO 60060 ELSE POKE I,X
60110 IF X<192 AND X>128 PRINT
    TAB(40+4*(I/2-INT(I/2)))
    CHR$(27)CHR$(X)
60120 NEXT
```

Let's use Super Graphics to create a fish and have him swim across the screen. First load Graphics Editor. (This is always done prior to typing any lines in the main program). Next, enter the program, including the "dummy" string at line 120:

```
100 DEFSTR  A
120 A="THIS IS WHERE THE GRAPHICS GO!"
130 CLS:FOR P=640 TO 692
140 PRINT @  P,A:  IF RND(2)=1 PRINT  @  P-54, "0"
150 FOR I=1  TO 30:NEXT:PRINT  @  P, CHR$(31)
160 PRINT @  960,  :NEXT
170 FOR J=1 TO 8:FOR I=1 TO 50:NEXT
180 PRINT:NEXT
190 GOTO 130
```

Next we use Graphics Editor to magically alter line 120. From the illustrations, you can see that the fish consists of thirty-one characters:

32(blank), 130, 173, 180, 184, 188, 191, 156, 180, 26(cursor down)
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, (cursor left 10 times)
32(blank, 160, 158, 135, 139, 143, 191, 191, 143, 135



We run 60000 and press **ENTER** until we see the first byte following the quote (a''T''). User input is shown underlined.

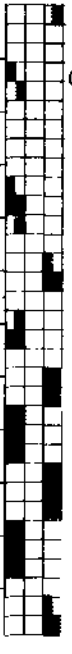
RUN 60000

WHICH LINE NUMBER? 110

LINE NOT IN PROGRAM

WHICH LINE NUMBER? 120

LOC.	DEC.	CHR$	NEW	COMMENTS
17141	65	A	? **ENTER**	
17142	213	SP.COMP.21	? **ENTER**	213 is the token for ''=''
17143	34	SP.COMP.21	? **ENTER**	
17144	84	T	? 32	
17145	72	H	? 130	
17146	73	I	? 137	Oops! Should have been 173 at location 17146
17147	83	S	?	
17146	137		? 173	
17147	83	S	? 180	
17148	32	BLANK	? 184	
17149	73	I	? 188	
17150	83	S	? 191	
17151	32	BLANK	? 191	
17152	87	W	? 156	
17153	72	H	? 180	
17154	69	E	? 26	
17155	82	R	? 24	
17156	69	E	? 24	



```
17157  32     BLANK      ? 24
17158  84     T          ? 24
17159  72     H          ? 24
17160  69     E          ? 24
17161  32     BLANK      ? 24
17162  71     G          ? 24
17163  82     R          ? 24
17164  65     A          ? 24
17165  80     P          ? 32
17166  72     H          ? 160
17167  73     I          ? 158
17168  67     C          ? 135
17169  83     S          ? 139
17170  32     BLANK      ? 143
17171  71     G          ? 191
17172  79     O          ? 191
17173  32     BLANK      ? 143
17174  33     !          ? 135
17175  34     "          ? ENTER
```

Graphics Editor stops automatically after displaying last byte in line 120

```
READY
>—
```

Now type RUN to see the fish swim. If there's anything wrong with the fish, **BREAK**. RUN 60000 and correct line 120 as necessary. (You may wish to examine line 120 with Graphics Editor in any case to see how the graphics are displayed)

The final step is to remove Graphics Editor by hitting **BREAK** and typing:

DELETE 60000 - 60120

If you now type:

? 50 - FRE(A$)

you will get 0. This shows that Super Graphics use **none** of the available string space! Note that Super Strings may also be placed in PRINT statements and in DATA statements.

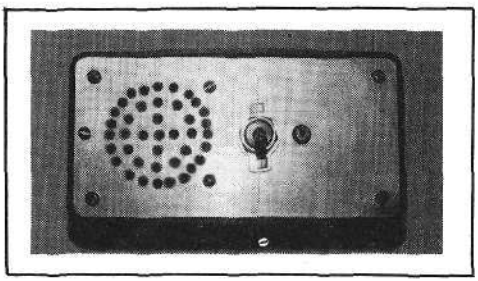
Warning: Never EDIT Super Graphcs or they instantly change to Super Garbage!

Basic Tokens as Stored in Memory

Decimal	Token	Hexadecimal	Decimal	Token	Hexadecimal	Decimal	Token	Hexadecimal
128	END	8Ø	171	LSET	AB	214	<	D6
129	FOR	81	172	RSET	AC	215	SGN	D7
130	RESET	82	173	SAVE	AD	216	INT	D8
131	SET	83	174	SYSTEM	AE	217	ABS	D9
132	CLS	84	175	LPRINT	AF	218	FRE	DA
133	CMD	85	176	DEF	BØ	219	INP	DB
134	RANDOM	86	177	POKE	B1	220	POS	DC
135	NEXT	87	178	PRINT	B2	221	SQR	DD
136	DATA	88	179	CONT	B3	222	RND	DE
137	INPUT	89	180	LIST	B4	223	LOG	DF
138	DIM	8A	181	LLIST	B5	224	EXP	EØ
139	READ	8B	812	DELETE	B6	225	COS	E1
140	LET	8C	183	AUTO	B7	226	SIN	E2
141	GOTO	8D	184	CLEAR	B8	227	TAN	E3
142	RUN	8E	185	CLOAD	B9	228	ATAN	E4
143	IF	8F	186	CSAVE	BA	229	PEEK	E5
144	RESTORE	9Ø	187	NEW	BB	230	CVI	E6
145	GOSUB	91	188	TAB	BC	231	CVS	E7
146	RETURN	92	189	TO	BD	232	CVD	E8
147	REM	93	190	FN	BE	233	EOF	E9
148	STOP	94	191	USING	BF	234	LOC	EA
149	ELSE	95	192	VARPTR	CØ	235	LOF	EB
150	TRON	96	193	USR	C1	236	MKI$	EC
151	TROFF	97	194	ERL	C2	237	MKS$	ED
152	DEFSTR	98	195	ERR	C3	238	MKD$	EE
153	DEFINT	99	196	STRING$	C4	239	CINT	EF
154	DEFSNG	9A	197	INSTR	C5	240	CSNG	FØ
155	DEFDBL	9B	198	POINT	C6	241	CDBL	F1
156	LINE	9C	199	TIME$	C7	242	FIX	F2
157	EDIT	9D	200	MEM	C8	243	LEN	F3
158	ERROR	9E	201	INKEY$	C9	244	STR$	F4
159	RESUME	9F	202	THEN	CA	245	VAL	F5
160	OUT	AØ	203	NOT	CB	246	ASC	F6
161	ON	A1	204	STEP	CC	247	CHR$	F7
162	OPEN	A2	205	+	CD	248	LEFT$	F8
163	FIELD	A3	206	-	CE	249	RIGHT$	F9
164	GET	A4	207	*	CF	250	MID$	FA
165	PUT	A5	208	/	DØ	251	'	FB
166	CLOSE	A6	209	↑	D1	252		FC
167	LOAD	A7	210	AND	D2	253	.	FD
168	MERGE	A8	211	OR	D3	254	!	FE
169	NAME	A9	212	>	D4	255	ISA	FF
170	KILL	AA	213	=	D5			

Note: Commands at FB and FC not fully described

BOOMER BOX



by Frank B. Rowlett, Jr. and Garry W. Woodruff

Is playing games on your TRS-80 sometimes dull? Do you ever wish it would produce sounds under program control and have animation on the screen like the big arcade games? If so, and you have Level II BASIC, you can do it inexpensively.

For sound, you'll have to construct a simple device called the "Boomer Box". It isn't complicated or difficult to assemble, and the parts are available at your local Radio Shack store. The "Boomer Box" connects via the tape recorder REMOTE control plug ... you don't have to modify your TRS-80. And, while we designed the "Boomer Box" specifically for the TRS-80, it should work with any other computer system having a relay or switch to turn a tape recorder on and off under program control.

The "Boomer Box" is designed around Radio Shack's Complex Sound Generator (Texas Instruments SN76477N) IC (Integrated Circuit) chip. We adapted a circuit from the data sheet that comes with it to allow the device to produce a sound similar to a long gunshot or short explosion. A variety of sounds can be produced by controlling the duration of the sounds with timing loops or multiple activations of the box, or both. It's possible to get a shot, machine gun fire, explosions and multiple detonations. Its only limitation is your imagination and whatever level of sound simulation you consider acceptable for your purposes.

The tools you'll need are listed in Figure 1. Note that they consist of only items that most do-it-yourself types have on hand. The parts are listed in Figure 2 along with the Radio Shack catalog numbers. Of course you can obtain them elsewhere, but Radio Shack has them all ... and for under twenty dollars.

Fig. 1

TOOLS

Soldering Iron	Drill [Electric or Hand]
Wire Strippers	1/ inch Drill Bit
Wire Cutters	5/32 inch Drill Bit
Small Phillips Screwdriver	1/2 inch Drill Bit
Sharp Knife	

PART NUMBER	DESCRIPTION	RADIO SHACK CATALOG NUMBER	quantity needed	QUANTITY PER PACKAGE	COST PER PACKAGE
C1	220 pF capacitor	272-124	1	2	.29
C3	.1μF capacitor	272-135	1	2	.39
C2	1μF capacitor	272-1406	1	1	.39
C4	10μF capacitor	272-1411	1	1	.49
R4	5.6K resistor	271-031	1	2	.19
R8	2.2K resistor	271-027	1	2	.19
R1, R7	56K resistor	271-043	2	2	.19
R5	150K resistor	271-047	1	2	.19
R2, R6	220K resistor	271-049	2	2	.19
R3	1M resistor	271-059	1	2	.19
Q1	RS-2014 transistor	276-2014	1	1	.79
Q2	RS-2024 transistor	276-2024	1	1	.79
IC1	Complex Sound Generator IC chip	276-1765	1	1	2.99
SPKR	2 inch 8-ohm speaker	40-245	1	1	1.89
J1	3/32 inch subminiature phone jack	274-292	1	2	.99
S1	switch, ON/OFF plate	275-602	1	1	.79
B1	9V battery clips	270-325	1	5	.99
	transistor sockets	276-548	2	6	1.59
	28-pin IC socket	276-1997	1	1	.89
	perfboard, .100 x .100 2¾ inches x 6 inches	276-1395	1	1	.99
	experimenter box, 5 1/16 inches x 2 5/8 inches x1 5/8 inches				
	aluminum cover	270-233	1	1	1.49
	round head machine screws, 4-40 x ¼ inch	64-3011	3	42 asst.	.89
	hex nuts, 4-40	64-3018	3	30	.89
	9-volt battery	23-464	1	1	.59
					$19.26

Fig. 2

Radio Shack does package several identical items together so that you may be required to purchase several more than you need. This will give you spare parts for future projects, or several of you could get together and build a bunch of "Boomer Boxes" at one time.

Begin by lining up all your components and making sure you can identify them easily. While you're at it, you can bend all the resistors into square U-shapes with a flat bottom of ½ inch containing the resistor. This will make for easier going later on. Now be sure you have all your tools, and that they're handy and ready to use.

The Complex Sound Generator chip comes with a diagram showing you all the pin numbers. We will refer to the pin numbers as they will be mounted in the IC socket. **Do not put the chip in the IC socket, the transistors in the transistor sockets or connect the battery until you reach the appropriate place in the instructions.** If you do, you risk ruining your chips or transistors. **Be very careful when handling the IC chip**...those little pins are sharp and you can wind up with it embedded in your fingers.

Be careful when soldering. Don't use an excessive amount of solder in making a connection, and don't overheat any of the components. Use just enough heat and solder to get a good joint. Mistakes in soldering can ruin the entire project. We have included sockets for the IC chip and the transistors to make assembly less critical for the inexperienced.

There are two phases in assembling the "Boomer Box"; the first is to assemble the components on the perfboard, and the second is to mount the perfboard and other components in the box. We'll go through the assembly of the perfboard step-by-step. Read these instructions thoroughly before starting ... it is the most complex part of building the "Boomer Box", and requires careful attention and checking as you put it together.

Be sure you have completed the current step in the assembly before undertaking the next step. Do not jump to future steps; complete each step as you come to it.

As you go along, refer to the schematic in Figure 3 and the component layout in Figure 4. If you vary from these or substitute other components, we can't predict what will happen ... and you're on your own.

Before beginning assembly of the components on the perfboard, cut it to its final size of 2 7/16 inches by 4 1/16 inches. The easiest way to cut the board is to use a sharp knife and score deeply along the holes where you want to cut. Then carefully break the board along the scored line.

ASSEMBLY INSTRUCTIONS FOR PERFBOARD

Step 1 Insert the 28-pin IC socket into the perfboard as shown in Figure 4. Note the orientation of pin 1 of the IC socket. Be careful not to bend any pins of the socket when inserting it into the board.

Step 2 Insert resistor R1 (56K, green-blue-orange) at a right angle to pin 4 of the IC socket. Bend the lead nearest the IC socket over to pin 4 and solder. Clip off the excess lead.

Step 3 Insert resistor R2 (220K, red-red-yellow) at a right angle to pin 5 of the IC socket. Bend the lead nearest the IC socket over to pin 5 and solder. Clip off the excess lead.

Step 4 Insert resistor R3 (1M, brown-black-green) at a right angle to pin 7 of the IC socket. Bend the lead nearest the IC socket over to pin 7 and solder. Clip off the excess lead.

Step 5 Insert resistor R4 (5.6K, green-blue-red) at a right angle to pin 10 of the IC socket. Bend the lead nearest the IC socket over to pin 10 and solder. Clip off the excess lead.

Step 6 Insert resistor R5 (150K, brown-green-yellow) at a right angle to pin 11 of the IC socket. Bend the lead nearest the IC socket over to pin 11 and solder. Clip off the excess lead.

Step 7 Insert resistor R6 (220K, red-red-yellow) at a right angle to pin 12 of the IC socket. Bend the lead nearest the IC socket over to pin 12 and solder. Clip off the excess lead.

Step 8 Insert resistor R7 (56K, green-blue-orange) at a right angle to pin 24 of the IC socket. Bend the lead nearest the IC socket over to pin 24 and solder. Clip off the excess lead.

Step 9 Insert capacitor C1 (220pF) at a right angle to pin 6 of the IC socket. Bend the lead nearest the IC socket over to pin 6 and solder. Clip off the excess lead.

Step 10 Insert capacitor C2 (1.0 μF) at a right angle to pin 8 of the IC socket with the positive lead nearest pin 8. Bend the lead nearest the IC socket over to pin 8 and solder. Clip off the excess lead.

Step 11 Insert capacitor C3 (.1 μF) at a right angle to pin 23 of the IC socket. Bend the lead nearest the IC socket over to pin 23 and solder. Clip off the excess lead.

Step 12 Cut a 4½ inch piece of 22-gauge wire. Strip 1 inch of insulation from one end and ½ inch from the other end. Lay the 1 inch stripped end next to the unconnected ends of the resistors and capacitors connected to pins 4, 5, 6, 7, 8, 10 and 11 of the IC socket. Bend the leads from the resistors and capacitors over the wire and solder each one to it. Clip off the excess leads.

Step 13 Bend the wire from Step 12 around the IC socket and put the ½ inch stripped end near the unsoldered leads of R7 and C3 (from pins 24 and 23 of the IC socket). Bend the leads over the wire and solder them to it. Clip off the excess leads.

Step 14 Cut a 1 inch piece of wire and strip all the insulation from it. Solder one end to pin 2 of the IC socket. Solder the other end to the lead of R1 (from pin 4 of the IC socket) where it connects to the wire soldered there in Step 12.

Step 15 Cut a 1 inch piece of wire and strip all the insulation from it. Solder one end to pin 1 of the IC socket and the other end to pin 25 of the IC socket.

Step 16 Cut a 1½ inch piece of wire and strip ½ inch of insulation from each end. Solder one end to pin 25 of the IC socket and the other end to pin 15 of the IC socket.

Step 17 If the transistor sockets have four pins, leave the three pins that are in line and pull out the fourth pin using a pair of pliers. Insert one of the transistors sockets (the one for Q2) near the edge of the perfboard with the flat part towards the IC socket (see Figure 4). Insert the other transistor socket (the one for Q1) with the flat away from the IC socket (see Figure 4).

Step 18 Bend the two pins from each transistor socket that are closest to each other together, and bend the unsoldered lead from R6 (from pin 12 of the IC socket) to these leads. **Do not solder yet.**

Step 19 Cut a 6 inch piece of wire and strip ¼ inch of insulation from one end and ½ inch of insulation from the other end. Feed the ½ inch piece of bare wire from the top of the perfboard to the unsoldered junction from Step 18. Now solder all four leads together. The other end of the wire will be connected to the speaker in Step 31.

Step 20 Inser R8 (2.2K, red-red-red) near the edge of the perfboard near the transistor socket for Q2 and pin 14 of the IC socket (see Figure 4). Bend over one lead of the resistor to the junction of R5 and the wire from Step 12 and solder them all together. Clip off the excess lead. Now bend the other end of R8 over to the middle pin of the transistor socket for Q2. **Do not solder yet.**

Step 21 Cut a 1 inch piece of wire and strip ¼ inch of insulation from each end. Bend the wire to the middle pin of the transistor socket for Q2. Bend the unsoldered lead of R8 from Step 20 to this junction. Solder all the leads together and clip off the excess lead.

Step 22 Cut a ½ inch piece of wire and strip all the insulation from it. Bend the unsoldered end of the 1 inch wire from Step 21 to the middle pin of the transistor socket for Q1. Place one end of the ½ inch piece of wire to the junction and solder all of them together. Solder the other end of the ½ inch piece of wire to pin 13 of the IC socket. Be careful to not let the ½ inch piece of wire touch pin 14 of the IC socket.

Step 23 Cut a 5 inch piece of wire and strip ½ of insulation one end and ¼ inch of insulation from the other end. Feed the ¼ inch piece of bare wire end through a hole in the perfboard near the unsoldered pin of Q1's transistor socket. **Do not solder yet.**

Step 24 Cut a 1 inch piece of wire and strip ¼ inch of insulation from each end. Connect one end to the unsoldered pin of Q1's transistor socket and solder it and the piece of wire from Step 23 to the transistor socket's pin. Solder the other end of the 1 inch wire to pin 14 of the IC socket.

Step 25 Solder the other end of the 5 inch piece of wire from Step 23 to one terminal of switch S1.

Step 26 Solder the RED lead from the battery clip (B1) to the other terminal of switch S1.

Step 27 Feed the black lead from the battery clip (B1) through the perfboard near R7 and C3. Solder the lead to where R7 and C3 are soldered together from Step 13.

Step 28 Insert C4 (10 μF) near the transistor socket for Q2 with the negative lead nearest the socket. Bend the negative lead to the unsoldered lead of the transistor socket for Q2. **Do not solder yet.**

Step 29 Cut a ½ inch piece of wire and strip all insulation from it. Connect one end to the unsoldered lead of the transisotr socket for Q2 and solder all leads together. Solder the other end to the junction of R8 and R5 from Step 20.

Step 30 Cut a 6 inch piece of wire and strip ½ inch of insulation from one end and ¼ inch of insulation from the other end. Feed the ¼ inch stripped end through the perfboard near the positive lead of C4. Bend the wire to the positive lead of C4 and solder. Solder the other end to the positive terminal on the speaker.

Step 31 Now connect the unsoldered end of the 6 inch piece of wire from Step 19 to the unsoldered terminal of the speaker and solder.

Step 32 Cut a 5 inch piece of wire and strip ¼ inch of insulation from each end. Feed one end of the wire through the perfboard near pin 9 of the IC socket and solder. Solder the other end of the wire to the center terminal of J1 (the subminiature jack).

Step 33 Cut a 5 inch piece of wire and strip ¼ inch of insulation from each end. Feed one end through the perfboard near pin 1 of the IC socket and solder it to pin 1. Solder the other end of the wire to the left terminal of J1 as viewed with the threaded end farthest from you.

Step 34 Now all soldering is completed. Check your work very carefully for shorts or bad solder joints.

Step 35 Insert IC1 into the IC socket. Be sure to orient it correctly ... pin 1 should be nearest to R1. Check carefully that all pins are inserted into the socket. Watch out for the pins ... they're sharp!!

Step 36 Insert Q1 and Q2 into their respective sockets. Line up the flat sides of the transistors with the flat sides of the sockets.

Step 37 Turn S1 to the ON position

Step 38 Touch B1 (the battery clip) to the appropriate terminals of the 9-volt battery. You should hear a grating sound from the speaker. If you don't hear any sound, recheck all your solder joints. Also, check the orientation of the IC chip and the transistors in their sockets.

Step 39 Connect the battery to the battery clip (B1). Now short the two wires soldered to the jack (J1). If a long "boom" isn't heard, then recheck everything.

Now that all the components are assembled on the perfboard and tested, it's time to put them into the box. First remove the aluminum cover from the box and use the diagram in Figure 5 to locate the holes to be drilled in it. Where the speaker goes, drill a pattern of 1/8 inch holes to let

the sound out. Drill all holes carefully and as precisely as possible ... and let the drill do the work. Any excessive pressure can bend the soft aluminum and you'll get unsightly bends and bows in the cover. After all holes are drilled, remove the burrs with a sharp knife.

Mount the subminiature jack in the 5/32 inch hole and tighten the retaining nut. Now mount the ON/OFF switch in the ½ inch hole and tighten its retaining nut (make sure the ON/OFF plate is on the outside of the cover). Finally, mount the speaker with the three 4-40 by ¼ inch screws and nuts.

If you have trouble with the nuts holding the speaker to the cover, you can make some clips to hold it: take a paper clip and bend part of it into a U-shape that will fit around the 4-40 screw. Trim it so that it will bear on the rim of the speaker and hold it to the cover. You can use one clip on each screw if necessary (we got away with only one).

Now, carefully push the perfboard into the bottom of the box. You can use a knife or small file to notch the perfboard where the vertical ribs in the center of the box are. The perfboard should fit slightly snug ... if it doesn't, you can use some cardboard wedges to help hold it in. It shouldn't rattle around inside the box (but don't get it so wedged in you can't easily get it back out).

Now connect the battery to the battery clip and put the battery inside the box. Put the lid on the box and screw down the four screws that hold it on. The assembly of your "Boomer Box" is now complete.

To connect the "Boomer Box" to your TRS-80, put the plug that goes to the REMOTE jack of the tape recorder into the jack on the "Boomer Box". You now use the OUT statement of Level II BASIC to turn the sound on and off. An OUT 255, 0 turns the sound on, and an OUT 255,4 turns it off. Use timing loops to control the duration of the sound (remember, you must turn the sound off before you can turn it back on again; and you must give the relay time to latch and unlatch when you turn the sound on and off).

To cause animation on the screen, use the PRINT @ statement. You can print a character string representing a complex graphic image on the screen wherever you want it. The character string will probably be made up of graphic and cursor control characters so that one print statement will cause the whole image to be printed at one time.

It's usually best to pre-load the strings at the start of the program so that no time is wasted during the animation cycle in loading strings. Also, be sure to include a method of blanking out residue of the previous graphic image on the screen.

To have an airplane move from right to left across the screen, use the short program in Figure 6. Note the loop from line numbers 80 to 110 that keeps the airplane moving across the screen. You'll have to use the **BREAK** key to stop this program.

```
10 REM * DEMONSTRATION PROGRAM *
20 CLEAR200:DEFINTX:DEFSTRP
30 P1=CHR$(176):P2=CHR$(143):P3=CHR$(131):P4=CHR$(26)+STRING$(14
,CHR$(24)):P=CHR$(168)+"   "+CHR$(160)+P1+P1+"    "+CHR$(168)+CH
R$(188)+" "+P4+CHR$(171)+CHR$(139)+STRING$(5,P2)+STRING$(6,P3)+"
 "
40 REM NOTE: BEFORE EACH P4 (LINE FEED PLUS 24 BACKSPACES)
50 REM       A BLANK APPEARS TO CLEAR OUT RESIDUE IMAGE FROM
60 REM       THE PREVIOUS PRINT.
70 CLS
80 FORX=433TO384STEP-1:  ' LOCATE THE PLANE ON THE SCREEN
90 PRINT@X,P;:  ' PRINT THE IMAGE OF THE PLANE ON THE SCREEN
100 FORX0=1TO10:NEXTX0:  ' TIMING LOOP TO SLOW THE PLANE DOWN
110 NEXTX:  ' MOVE THE PLANE OVER ONE CHARACTER
120 GOTO70:  ' START A NEW PLANE ACROSS
130 '
140 '
150                 ' FIGURE 6 '
```

By combining the "Boomer Box" with an animation program and the INKEY$ function, you can create an arcade-like program that is complete with animation, sound effects and keyboard control. The sample program accompanying this article, "Anti-Aircraft Gunner", combines all these elements. Studying it should give you the necessary insight into how to write your own arcade-type animated programs with sound effects and player control.

Have fun ... and we hope you get a BANG out of this article!

```
10 'ANTI-AIRCRAFT GUNNER' BY FRANK B. ROWLETT, JR. (2-3-79)
20 'DEMONSTRATION PROGRAM FOR THE "BOOMER BOX"'
40 CLEAR500:DEFINTX-Z:DEFSTRE,P
50 P1=CHR$(176):P2=CHR$(143):P3=CHR$(131):P4=CHR$(26)+STRING$(14
,CHR$(24)):P=CHR$(168)+"   "+CHR$(160)+P1+P1+"    "+CHR$(168)+CH
R$(188)+" "+P4+CHR$(171)+CHR$(139)+STRING$(5,P2)+STRING$(6,P3)+"
 "
60 E1=CHR$(133)+"* "+CHR$(160)+"*"+CHR$(161)+CHR$(156)+" *"+CHR$
(144)+" "+CHR$(161)+CHR$(182)+" "+P4+CHR$(147)+CHR$(139)+CHR$(16
7)+CHR$(155)+"**"+CHR$(148)+CHR$(135)+CHR$(131)+CHR$(130)+STRING
$(3,CHR$(131))+" "+P4+CHR$(129)+"  *"+CHR$(130)+CHR$(134)+"  *"
70 E2="   **  ** *   "+P4+" * * ** ** *  "+P4+"** *** *  *** "+P
4+" *  **  *** * "+P4+"   * **  *"
```

```
80 E="N":CLS:INPUT"DO YOU WANT INSTRUCTIONS";E:IFASC(E)<>89THEN1
00ELSEPRINT:PRINT"THE SPACE BAR FIRES YOUR ANTI-AIRCRAFT GUN.  T
HE SHELL":PRINT"MUST HIT THE PLANE IN ITS BELLY TO DESTROY IT.
DON'T":PRINT"FORGET TO LEAD THE PLANE WHEN SHOOTING AT IT."
90 PRINT:PRINT:INPUT"PRESS 'ENTER' TO CONTINUE";E
99 REM * BEGIN GAME *
100 OUT255,4:Z1=25:Z2=0:CLS:GOSUB500:OUT255,0:FORY=0TO5:NEXTY:OU
T255,4
110 FORX=433TO384STEP-1:E="":PRINT@X,P;:PRINT@479,"+";
120 E=INKEY$:Z=0:GOSUB300:IFZ=1THEN150
130 IFE=""THEN140ELSEIFZ1>0THENSET(62,39):OUT255,0:FORYY=0TO100:
NEXTYY:OUT255,4:Z1=Z1-1:PRINT@978,Z1;
140 NEXTX
150 GOSUB400
160 IFZ1<1THEN190ELSEPRINT@479,"+";:Y=RND(50)+15:FORZ=0TOY
170 GOSUB300:E=INKEY$:IFE=""THEN180ELSEIFZ1>0THENSET(62,39):OUT2
55,0:FORYY=0TO100:NEXTYY:OUT255,4:E="":Z1=Z1-1:PRINT@978,Z1;
180 NEXTZ:GOTO110
190 GOSUB400:PRINT@222,"GAME OVER":PRINT@512,"HIT 'ENTER' TO PLA
Y AGAIN:";:INPUTE:GOTO100
299 REM * TRACK SHELL TO TARGET *
300 IFPOINT(62,39)THENRESET(62,39):SET(62,36):RETURN
310 IFPOINT(62,36)THENRESET(62,36):SET(62,33):RETURN
320 IFPOINT(62,33)THENRESET(62,33):SET(62,30):RETURN
330 IFPOINT(62,30)THENRESET(62,30):SET(62,27):RETURN
340 IFPOINT(62,27)THENRESET(62,27):SET(62,24):RETURN
350 IFPOINT(62,24)ANDPOINT(64,22)ANDPOINT(61,22)THEN360ELSERESET
(62,24):RETURN
360 OUT255,0:PRINT@X,E1;:FORY=0TO50:NEXTY:OUT255,4:FORY=0TO3:NEX
TY
370 OUT255,0:PRINT@X-64,E2;:FORY=0TO100:NEXTY:OUT255,4:FORY=0TO3
:NEXTY
380 OUT255,0:FORY=0TO150:NEXTY:Z=1:Z2=1+Z2:PRINT@1015,Z2;:OUT255
,4:RETURN
399 REM * CLEAR WREAKAGE AND PLANES *
400 PRINT@320,CHR$(30);CHR$(26);CHR$(30);CHR$(26);CHR$(30);CHR$(
26);CHR$(30);CHR$(26);CHR$(30);:RETURN
499 REM * DRAW FRAME *
500 PF=STRING$(63,CHR$(191))
510 PRINT@0,PF;CHR$(191);PF;CHR$(191);
520 PRINT@21," ANTI-AIRCRAFT GUNNER ";
```

```
530 PRINT@896,PF;CHR$(191);PF;:POKE16383,191
540 PRINT@966," SHOTS LEFT: 25  ";:PRINT@1008," HITS: 0  ";
550 RETURN
```

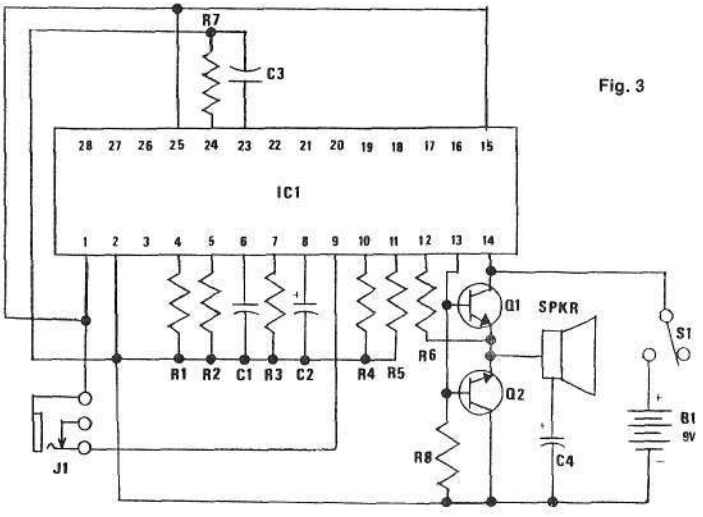
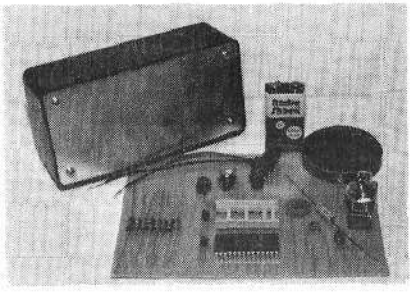
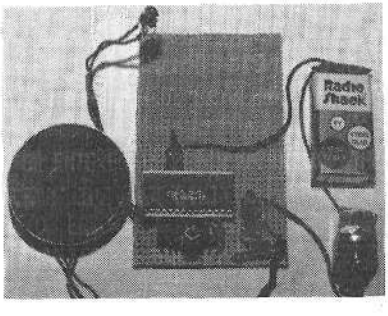


Fig. 3





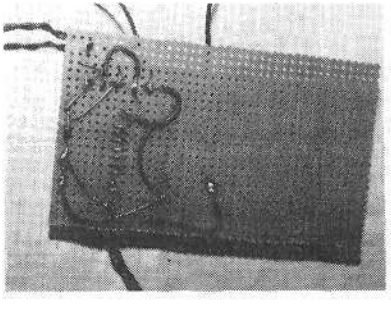


Fig. 4

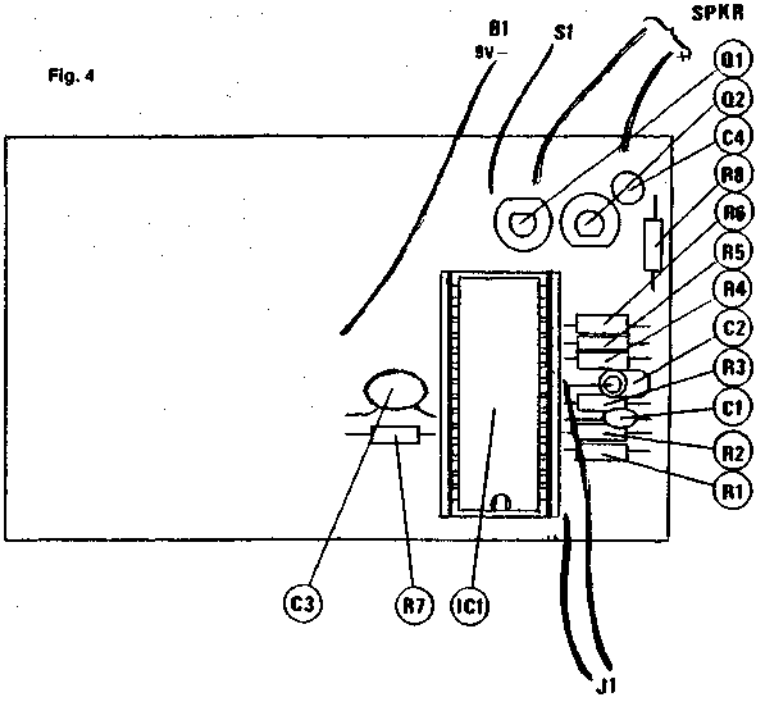
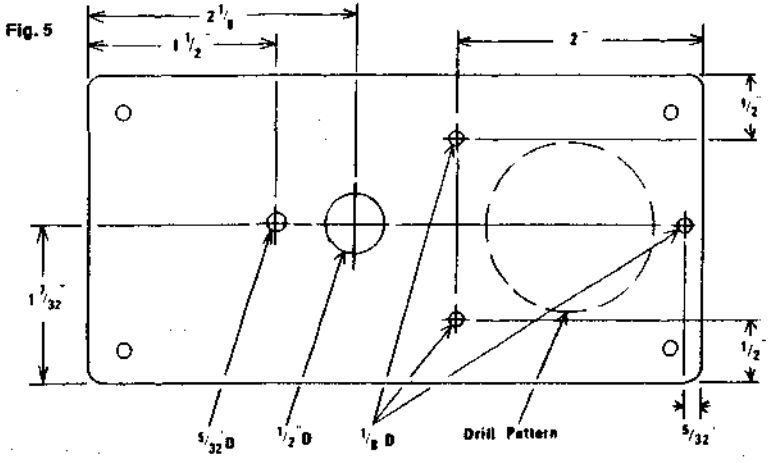

B1
9V –
S1
SPKR
Q1
Q2
C4
R8
R6
R5
R4
C2
R3
C1
R2
R1
C3
R7
IC1
J1

Fig. 5


2 1/8
1 1/2
2
1/2
1 7/32
1/2
5/32 D
1/2 D
1/8 D
Drill Pattern
5/32

UPPER & lower Case Mod

by Lance Micklus

Our first issue of PROG/80 certainly generated a large amount of interest in modification of the TRS-80 for upper/lower case. It also caused the telephone to ring off the hook. There was a typographical error which most people caught immediately: you should use 2102's not 2101's. The second question was that it seemed as if the modification wouldn't work ... at least not on paper.

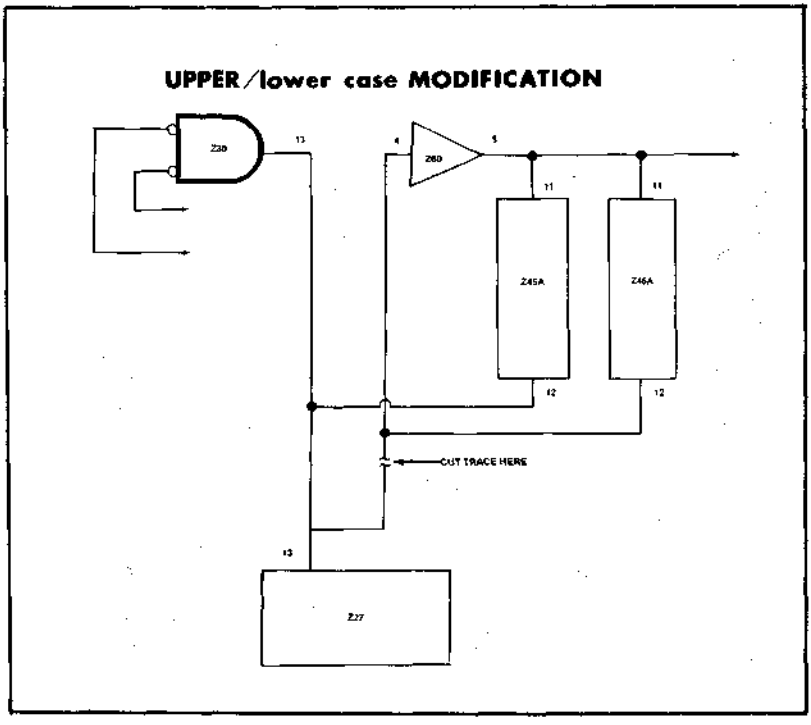
The "at least not on paper" is the catch. Many readers didn't have sufficient experience working with printed circuit boards to know that schematics only show you what is connected to what. They don't show you HOW what is connected to what. And that may very well be why no one ever thought of using this modification.

Sometimes it helps to draw the schematic in a different way. The illustration following is drawn the way I see it. If you forget about the modifications shown on the drawing and compare it with the schematic in the TRS-80 technical manual, you'll see that they are identical. The key is the way Z30-13, Z27-13, and Z60-4 are connected together. In my schematic and in the Radio Shack schematic, these three points are all connected together. So, circuit-wise, the schematics are the same.

Hopefully, this clears up most of your questions. I might add that most of those who called got straightened out and didn't need to call back. I assume that means that they got it all working.

What about the software? Making this modification isn't going to give you upper/lower case letters on the screen yet. The only thing you can do is POKE lower case characters into the video RAM. To display them requires a machine language subroutine. If you have KVP, the subroutine is built into it already. But, if you don't want to buy KVP just to display lower case letters on your screen, then the BASIC program below will create a subroutine for you in high memory.

```
0 ' * * *  UPPER / LOWER CASE DRIVER  * * *
100 CLS:PRINT CHR$(23)
120 PRINT@320,"ENTER MEMORY SIZE  "
140 INPUT "1=4K 2=16K 3=32K 4=48K ";M
160 IF M<0 OR M>4 THEN 100
180 FOR I=1 TO M
200 READ I1,I2,I3,I4
220 NEXT I
240 READ I6 : IF I6 <> -999 THEN 240
260 FOR I=I1 TO I2
280 READ B
300 POKE I,B
320 NEXT I
340 POKE 16414,216
360 POKE 16415,I3
380 PRINT
400 PRINT"BE SURE TO ANSWER MEMORY"
```



```
420 PRINT"SIZE QUESTION WITH"; I4
440 PRINT
460 END
480 DATA 20440, 20649, 79, 20440
500 DATA 32728, 32757, 127, 32728
520 DATA -16424, -16395, 191, 49112
540 DATA -40, -11, 255, 65496
560 DATA -999
580 REM
600 REM -- MACHINE LANGUAGE PROGRAM --
620 REM
640 DATA 221, 110, I 3, 221, 102
660 DATA 4, 218, 154, 4, 221, 126
680 DATA 55, 183, 400, 1, 119, 121
700 DATA 254, 32, 218, 6, 5, 254
720 DATA 128, 210, 166, 4
740 DATA 195, 125, 4
```

GRAPHICS AID

by Philip Brown

To create a truly fantastic game or special purpose program, it helps to have interesting, fast graphics. This is most easily and conveniently accomplished by developing strings from data in the beginning of the program. **Graphics Aid** is a programmer's tool to translate graphics constructed on the screen into data that can be output through use of the video, cassette, line printer or disk. Output disk or cassette files can be input, allowing the user to work on the same graphics at different times.

DATA INITIALIZATION

When the program begins, it asks if there are any strings already. If the program is being run for the first time, no strings would exist. If it has been previously run, data that was stored by the program on disk or cassette can be input at this point.

Next, the program will print out the data it has and will ask for the line number to be worked on. The program will construct up to eight strings — a half screen — with lengths of up to sixty-three characters each. Next, the tab for that string must be set. To help with the decision, there is a line above the graphics showing where a tab would begin.

```
          10        25        42
0---5---0---5---0---5---0---5---0---5---0---5---0---
```

GRAPHIC CHARACTER CONSTRUCTION

With a line and tab chosen, the program will go into its **graphic character construction sequence**.

```
SPECIFY WHICH BLOCKS ARE TO BE TURNED ⟨ON⟩ AB

Q = CONSTRUCT          K = CANCEL          CD
R = RECALL MENU                            EF
```

By hitting the letters **A B C D E F** graphic characters are constructed in the upper right-hand corner of the screen. If you accidentally construct the wrong character, typing a **K** for cancel will clear the present character and start the construction of a new graphic character. If you wish to get out of the construction sequence, typing a **R** will return you to the menu.

STRING CONSTRUCTION

With the correct graphic character in the corner of the screen, typing a **Q** will begin the **string construction**. The graphic character will appear as the last character on the string you are presently constructing. At that point, you are given three choices: **Reject, Include,** or **Repeat.**

Pressing a **1** for **Include** will leave the graphic character as the last character of the string and you will be returned to the graphic character construction sequence.

Pressing a **2** for **Repeat** will add another of the same character to the end of the string. This can be done until a **0** or a **1** is pressed, or until there is no more room left in the string.

Pressing a **0** for **Reject** will remove the character from the end of the string and you will be returned to the graphic character construction sequence.

MENU

The menu will first ask if you wish to quit. If you answer YES to this, the program will begin its **data output sequence**. Otherwise, the program will ask you if you wish to edit or not. An affirmative response will begin the **edit sequence**. A negative answer will start the **graphic character construction sequence**.

EDIT SEQUENCE

After inputting the chosen line number and tab, editing will begin with a number of choices:

RIGHT ARROW to advance
LEFT ARROW to move left
X discontinues editing
Anything else starts editing

and the character presently in question will be blinking.

•Hitting a RIGHT ARROW will move the indicator to the next character to the right.

•Hitting a LEFT ARROW will move the indicator to the next character to the left.

•Hitting an "X" will discontinue editing and return you to the menu.

•Hitting a "D" will delete the presently indicated character.

•Hitting an "I" will go to the graphic character construction and insert the constructed character in the string at the chosen place, moving the remaining right side of the string to the right side of the character.

•Hitting any other key will begin editing of the character in question. The program will proceed to the graphic character construction, following which the present character will be replaced with the constructed character.

In either editing or inserting, you are given the choice to reject the new characters.

DATA OUTPUT

Indicating you wish to quit in the menu will bring you to this point. If you wish to output data, this can be done here on the screen, on the printer, or on disk or cassette. The choice is yours. **Good luck!**

```
10 '     GRAPHICS AID
20 '     BY PHILIP BROWN
40 '     ********************************
50 CLEAR 1000
60 REM** 70-110 ** INITIALIZE DATA **
70 DIM RS$(8),T(8)
80 FOR I=1 TO 8
90 RS$(I)=""
100 T(I)=0
110 NEXT
120 CLS
130 GOTO 2090
140 PRINT@256,"";
150 FOR I=1 TO 6
160 PRINT "0----5-----";
170 NEXT I
180 PRINT "0--";
190 GOTO 620
200 QQ=LEN(RS$(L))+T
210 PP=0
220 GG=0
230 REM** 240-440 ** GRAPHIC CHARACTER CONSTRUCTION **
240 PRINT@0,"";:PRINT STRING$(255," ");
250 PRINT@0," ";
260 PRINTTAB(62);"AB";:PRINTTAB(62);"CD";:PRINTTAB(62);"EF";
270 Y=128:S1=0:S2=0:S3=0:S4=0:S5=0:S6=0
280 PRINT@5,"SPECIFY WHICH BLOCKS ARE TO BE TURNED <ON>";
```

```
290 PRINT@75,"Q = CONSTRUCT";
300 IF GG=1 THEN 310 ELSE PRINT@139,"R = RECALL MENU";
310 PRINT@96,"K = CANCEL";
320 A$=INKEY$:IF A$="" THEN 320
330 C$=CHR$(191)
340 IFS1<>1 IFA$="A"PRINT@62,C$;:S1=1:Y=Y+1
350 IFS2<>1 IFA$="B"PRINT@63,C$;:S2=1:Y=Y+2
360 IFS3<>1 IFA$="C" PRINT@126,C$;:S3=1:Y=Y+4
370 IFS4<>1 IFA$="D" PRINT@127,C$;:S4=1:Y=Y+8
380 IFS5<>1 IFA$="E" PRINT@190,C$;:S5=1:Y=Y+16
390 IFS6<>1 IFA$="F" PRINT@191,C$;:S6=1:Y=Y+32
400 IFA$="Q" GOTO460
410 IF GG=1 THEN 430
420 IF A$="R" THEN 570
430 IF A$="K" THEN 240
440 GOTO280
450 REM** 460-560 ** STRING CONSTRUCTION **
460 IF PP=1 THEN 1080
470 PRINT@0,"";:PRINT STRING$(255," ");
480 PRINT@(320+L*64),"";:PRINTTAB(T);RS$(L);CHR$(Y);
490 PRINT@200,"PRESS 0 TO REJECT, 1 TO INCLUDE, 2 TO REPEAT";
500 X$=INKEY$:IF X$="" THEN 500
510 IF (X$<"0")+(X$>"3") THEN 490
520 IF X$="0" PRINT@(320+L*64+T+LEN(RS$(L))),"";:GOTO 240
530 QQ=QQ+1
540 IF QQ=63 THEN RS$(L)=RS$(L)+CHR$(Y):GOTO570
550 IFX$="1" THEN RS$(L)=RS$(L)+CHR$(Y):GOTO240
560 RS$(L)=RS$(L)+CHR$(Y):GOTO480
565 REM** 570-690 ** MENU **
570 PRINT@0,"";:PRINT STRING$(255," ");
580 PRINT@10," ";:INPUT"DO YOU WISH TO QUIT";M$
590 IF M$="YES" OR M$="Y" THEN 1370
600 PRINT@74," ";:INPUT"WILL YOU BE EDITING";M$
610 IF M$="YES" OR M$="Y" THEN 710
620 PRINT@74," ";:INPUT"WHAT LINE WILL YOU WORK ON";L
630 IF (L<1)+(L>8) THEN 570
640 PRINT@138," ";:INPUT"HOW MUCH WILL YOU TAB IN  ";T
650 IF (T>62-LEN(RS$(L)))+(T<0) THEN 570
660 T(L)=T
670 PRINT@(320+L*64),"";:PRINTTAB(T);RS$(L);
680 PRINT STRING$(63-T-LEN(RS$(L)),"");
```

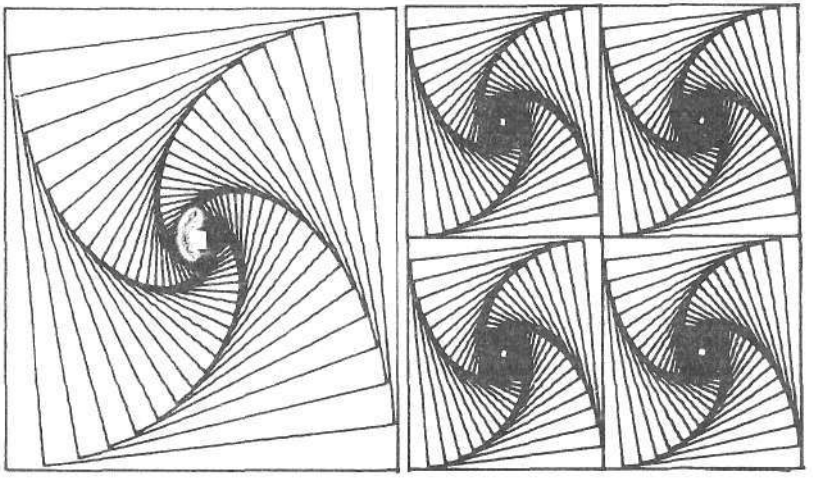
```
690 GOTO 200
700 REM** 710-1350 ** EDIT SEQUENCE **
710 PRINT@0,"";:PRINT STRING$(160," ");
720 PRINT@10," ";:INPUT"WHAT LINE WILL YOU WORK ON";L
730 IF (L<1)+(L>8) THEN 570
740 IF LEN(RS$(L))=0 THEN 570
750 PRINT@74," ";:INPUT"HOW MUCH WILL YOU TAB IN";T
760 IF (T>63-LEN(RS$(L)))+(T<0) THEN 710
770 T(L)=T
780 GG=1
790 PRINT@(320+L*64),"";:PRINTTAB(T);RS$(L);
800 PRINT STRING$(63-T-LEN(RS$(L))," ");
810 PRINT@(320+T),"+";:PRINT@(896+T),"+";:SS=T+1
820 PRINT@0,"";:PRINT STRING$(255," ");
830 PRINT@10,"HIT SPACE BAR TO ADVANCE, Z TO MOVE LEFT"
840 PRINT@74,"X DISCONTINUES EDITING"
850 PRINT@138,"D DELETES THE CHARACTER, I INSERTS A CHARACTER"
860 PRINT@202,"ANYTHING ELSE STARTS EDITING";
870 IF LEN(RS$(L))=0 THEN PRINT@(319+SS)," ";:PRINT@(895+SS)," "
;:GOTO 570
880 Z$=INKEY$:IF Z$="" THEN 880
890 MM=0
900 IF (SS>LEN(RS$(L))+T)*((Z$<>"Z")*(Z$<>"X")) THEN 880
910 IF Z$<>" " THEN 930 ELSE IF SS>=LEN(RS$(L))+T THEN 870
920 SS=SS+1:PRINT@(318+SS)," +";:PRINT@(894+SS)," +";:GOTO 870
930 IF Z$="X" THEN PRINT@(319+SS)," ";:PRINT@(895+SS)," ";:GOTO
570
940 IF Z$<>"D" THEN 1030
950 EE=SS-T:IF SS<>1 THEN 970
960 RS$(L)=RIGHT$(RS$(L),LEN(RS$(L))-1):GOTO 1000
970 IF SS<>LEN(RS$(L)) THEN 990
980 RS$(L)=LEFT$(RS$(L),LEN(RS$(L))-1):GOTO 1000
990 RS$(L)=LEFT$(RS$(L),EE-1)+RIGHT$(RS$(L),LEN(RS$(L))-EE)
1000 PRINT@(320+L*64),"";:PRINTTAB(T);RS$(L);
1010 PRINT STRING$(63-T-LEN(RS$(L))," ");
1020 GOTO 870
1030 IF Z$<>"Z" THEN 1050 ELSE IF SS<=T+1 THEN 870
1040 SS=SS-1:PRINT@(319+SS),"+ ";:PRINT@(895+SS),"+ ";:GOTO 870
1050 IF Z$<>"I" THEN 1070
1060 IF LEN(RS$(L))+T<63 THEN MM=1 ELSE 870
1070 PP=1:GOTO 240
```

```
1080 IF MM=1 THEN 1250
1090 REM** 1100-1230 ** STRING CHARACTER EDIT **
1100 PRINT@(319+L*64+55),CHR$(Y);
1110 PRINT@0,"";:PRINT STRING$(255," ");
1120 PRINT@205,"PRESS 0 TO REJECT, 1 TO INCLUDE";
1130 X$=INKEY$:IF X$="" THEN 1130
1140 IF X$<>"0" THEN 1160
1150 PRINT@(320+L*64),"";:PRINTTAB(T);RS$(L);:GOTO 820
1160 IF X$<>"1" THEN 1130
1170 EE=55-T
1180 IF EE<>1 THEN 1200
1190 RS$(L)=CHR$(Y)+RIGHT$(RS$(L),LEN(RS$(L))-1):GOTO 820
1200 IF EE<>LEN(RS$(L)) THEN 1220
1210 RS$(L)=LEFT$(RS$(L),LEN(RS$(L))-1)+CHR$(Y):GOTO 820
1220 RS$(L)=LEFT$(RS$(L),EE-1)+CHR$(Y)+RIGHT$(RS$(L),LEN(RS$(L))
-EE)
1230 GOTO 820
1240 REM** 1250-1350 ** STRING CHARACTER INSERTION **
1250 EE=55-T
1260 PRINT@(320+L*64),"";
1270 PRINTTAB(T);LEFT$(RS$(L),EE-1);CHR$(Y);RIGHT$(RS$(L),LEN(RS
$(L))-EE+1);
1280 PRINT@0,"";:PRINT STRING$(255," ");
1290 PRINT@205,"PRESS 0 TO REJECT, 1 TO INCLUDE";
1300 X$=INKEY$:IF X$="" THEN 1300
1310 IF X$<>"0" THEN 1330
1320 PRINT@(320+L*64+T),RS$(L);" ";:GOTO 820
1330 IF X$<>"1" THEN 1300
1340 RS$(L)=LEFT$(RS$(L),EE-1)+CHR$(Y)+RIGHT$(RS$(L),LEN(RS$(L))
+1-EE)
1350 GOTO 820
1360 REM** 1370-2060 ** DATA OUTPUT **
1370 PRINT@0,"";:PRINT STRING$(255," ");
1380 PRINT@10,"";:INPUT"DO YOU WANT DATA";F$
1390 IF F$="NO" OR F$="N" THEN END
1400 PRINT@10,"LINE PRINTER, SCREEN, CASSETTE, OR DISK";
1410 X$=INKEY$:IF X$="" THEN 1410
1420 IF (X$="L")+(X$="P") THEN 1630
1430 IF X$="S" THEN 1470
1440 IF X$="C" THEN 1850
1450 IF X$="D" THEN 1950
```

```
1460 GOTO 1410
1465 REM** 1470-1610 ** SCREEN OUTPUT **
1470 PRINT@0,"";:PRINT STRING$(60," ");
1480 PRINT@10,"";:INPUT"WHICH STRING DO YOU WANT";F
1490 IF (F<1)+(F>8) THEN 570
1500 IF LEN(RS$(F))=0 THEN 1400
1510 PRINT@0,"";:PRINT STRING$(255," ");
1520 PRINT@217,"TAB =";T(F);
1530 FOR I=1 TO LEN(RS$(F)) STEP 3
1540 PRINT@64,"",I,I+1,I+2
1550 PRINT"",ASC(MID$(RS$(F),I,1)),
1560 IF I>LEN(RS$(F))-1 THEN PRINT" ---", ELSE PRINT ASC(MID$(RS
$(F),I+1,1)),
1570 IF I>LEN(RS$(F))-2 THEN PRINT" ---"; ELSE PRINT ASC(MID$(RS
$(F),I+2,1));
1580 PRINT@10,"HIT ANYTHING TO CONTINUE";
1590 X$=INKEY$:IF X$="" THEN 1590
1600 NEXT I
1610 GOTO 1370
1620 REM** 1630-1630 ** LINE PRINTER OUTPUT **
1630 D=PEEK(14312):IF D=63 THEN 1670
1640 PRINT@0,"";:PRINT STRING$(60," ");
1650 PRINT@20,"PRINTER NOT READY";:FOR I=1TO1500:NEXT
1660 GOTO 1370
1670 FOR I=1 TO 8
1680 IF RS$(I)="" THEN LPRINT"NO STRING #";I:GOTO 1810
1690 LPRINT"","","","STRING #";I
1700 LPRINT"","","","  TAB =";T(I)
1710 FOR J=1 TO LEN(RS$(I)) STEP 6
1720 LPRINT "",J;"=",J+1;"=",J+2;"=",J+3;"=",J+4;"=",J+5;"="
1730 LPRINT "",ASC(MID$(RS$(I),J,1)),
1740 FOR O=1 TO 4
1750 IF J>LEN(RS$(I))-O THEN LPRINT" ---", ELSE LPRINT ASC(MID$(
RS$(I),J+O,1)),
1760 NEXT O
1770 IF J>LEN(RS$(I))-5 THEN LPRINT" ---", ELSE LPRINT ASC(MID$(
RS$(I),J+5,1))
1780 LPRINT" "
1790 NEXT J
1800 LPRINT" "
1810 NEXT I
```

```
1820 FOR I=1TO3:LPRINT" ":NEXT
1830 GOTO 1370
1840 REM** 1850-1930 ** CASSETTE OUTPUT **
1850 PRINT@0,"";:PRINT STRING$(60," ");
1860 PRINT@10,"PREPARE THE RECORDER AND ENTER WHEN READY";
1870 X$=INKEY$:IF X$="" THEN 1870
1880 CMD"T
1890 FOR I=1 TO 8
1900 PRINT#-1,RS$(I)
1910 PRINT#-1,T(I)
1920 NEXT I
1930 GOTO 1370
1940 REM** 1950-2060 ** DISK FILE OUTPUT **
1950 PRINT@0,"";:PRINT STRING$(60," ");
1960 D=PEEK(14305):IF D<>255 THEN 1980
1970 PRINT@10,"THERE ARE NO DISKS";:FOR I=1TO1500:NEXT:GOTO1370
1980 PRINT@10,"";:INPUT"WHAT WILL THE FILENAME BE";P$
1990 PRINT@74,"";:INPUT"WHAT WILL THE FILENUMBER BE(1-3)";P
2000 OPEN "O",P,P$
2010 FOR I=1 TO 8
2020 PRINT#P,RS$(I)
2030 PRINT#P,T(I)
2040 NEXT I
2050 CLOSE P
2060 GOTO 1370
2070 REM** 2080-2440 ** DATA INPUT **
2080 PRINT@0,"";:PRINT STRING$(255," ");
2090 PRINT@10,"";:INPUT"DO YOU ALREADY HAVE SOME STRINGS";U$
2100 IF U$="YES" OR U$="Y" THEN 2120
2110 PRINT@0,"";:PRINT STRING$(60," ");:GOTO 140
2120 PRINT@82,"DISK OR CASSETTE";
2130 X$=INKEY$:IF X$="" THEN 2130
2140 IF X$="D" THEN 2180
2150 IF X$="C" THEN 2320
2160 GOTO 2080
2170 REM** 2180-2300 ** DISK FILE INPUT **
2180 PRINT@0,"";:PRINT STRING$(255," ");
2190 D=PEEK(14305):IF D<>255 THEN 2210
2200 PRINT@10,"THERE ARE NO DISKS";:FOR I=1TO1500:NEXT:GOTO2080
2210 PRINT@10,"";:INPUT"WHAT IS THE FILENAME";P$
2220 PRINT@74,"";:INPUT"WHAT IS THE FILENUMBER";P
```

```
2230 IF (P<1)+(P>3) THEN 2080
2240 OPEN "I",P,P$
2250 FOR I=1 TO 8
2260 INPUT#P,RS$(I)
2270 INPUT#P,T(I)
2280 NEXT I
2290 CLOSE P
2300 GOTO 2400
2310 REM** 2320-2390 ** CASSETTE INPUT **
2320 PRINT@0,"";:PRINT STRING$(255," ");
2330 PRINT@10,"PREPARE THE RECORDER AND ENTER WHEN READY";
2340 X$=INKEY$:IF X$="" THEN 2340
2350 CMD"T
2360 FOR I=1 TO 8
2370 INPUT#-1,RS$(I)
2380 INPUT#-1,T(I)
2390 NEXT I
2400 CLS
2410 FOR I=1 TO 8
2420 PRINT@(320+I*64+T(I)),RS$(I);
2430 NEXT I
2440 GOTO 140
```



Logarithmic spirals

Squish/bas

by Bill Driscoll

Squish is a program for the TRS-80 DOS user which compresses BASIC programs by removing all unnecessary blanks and line feeds. After processing, your overall disk and memory requirements will be reduced. The amount of compression is determined by your coding techniques ... the more spaces and line feeds used, the higher the space reduction will be. The biggest advantage of Squish is that you'll be able to store more programs on disk or reduce overall program disk space, thereby creating additional space for data.

RUN PROCEDURE

1. Load your BASIC program into memory
2. Save "PROGRAM NAME", A This saves the program on disk in ASCII format
3. Run "SQUISH/BAS" When Squish is loaded, it will ask you for the program name: Enter the program name just saved in ASCII

Once Squish is loaded, you won't be able to **BREAK** out with the **BREAK KEY**. Squish compressess the program in its own space on disk, therefore, once started, it must complete the entire compression operation or the program will be lost. When compression is completed, Squish will stop and ask you to hit **ENTER KEY** to continue. When **ENTER** is hit, the "squished" program will be automatically loaded to insure proper operation. Look the program over and then save "PROGRAM NAME".

I recommend you copy your working programs to a squish disk. If anything happens to one of the programs, you can always go back and copy it from your original disk.

```
10 'PGM:SQUISH/BASVER1.0DATE:02/24/79AUTHOR:BILLDRISCOLL
20 DEFINTA-Z
30 CLEARMEM/2
40 CLS
50 PRINTCHR$(23);TAB(5);"BASIC SQUISH PROGRAM"
60 PRINT
70 PRINT"PROGRAM NAME TO BE SQUISHED"
80 POKE &H5C8C,0
90 INPUTFI$
100 OPEN"I",1,FI$
110 OPEN"O",2,FI$
120 IFEOF(1)THEN290
130 LINEINPUT#1,A$
140 RC=RC+1:AA=AA+LEN(A$)
150 PRINT@320,"RECORDS READ    ";RC;
160 FORC=1TOLEN(A$)
170 PRINT@384,"SCANNING POS.          ";C;" ";
180 PRINT@448,"COMPRESS SWITCH <0=ON>";SW;
190 IFMID$(A$,C,1)=CHR$(34)ANDSW=0THENSW=1ELSEIFMID$(A$,C,1)=CHR
$(34)THENSW=0
200 PRINT@512,"CHARACTERS ELIMINATED ";TSE;
210 IFSW=0ANDMID$(A$,C,1)=" "ORMID$(A$,C,1)=CHR$(10)THENTSE=TSE+
1:NEXTELSEB$=B$+MID$(A$,C,1):NEXT
220 PRINT#2,B$
230 WR=WR+1
240 PRINT@576,"RECORDS WRITTEN ";WR;
250 PRINT@640,"BYTES SCANNED   ";AA;
260 B$=""
270 SW=0
280 GOTO120
290 CLOSE
300 PRINT
310 INPUT"HIT ENTER TO CONTINUE";ZZ$
320 POKE &H5C8C,1
330 LOADFI$
```

RENUMBER

No, it's not a game, but it CAN make renumbering your programs seem like child's play!

LOADS FROM AND OPERATES ON EITHER DISK OR TAPE BASED PROGRAMS!

If you find yourself renumbering to provide room for additional lines, or just to make things neater, this program has got what it takes to make life easier ... it can renumber a 12K program in just 32 seconds.

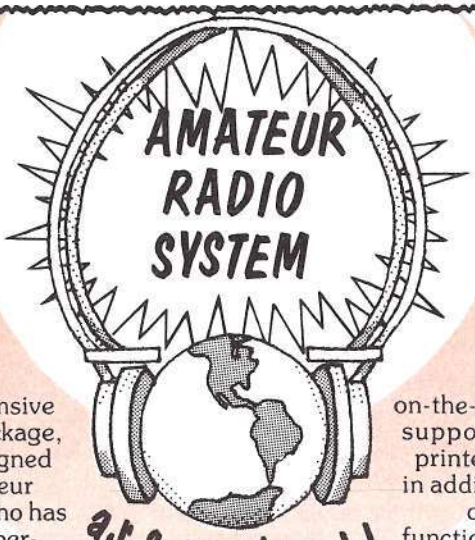
User has complete control over which lines are renumbered — and how — including all GOTO's and GOSUB's. You can even renumber the middle and leave the beginning and end alone! If an undefined line is found, the program will display both the line which caused the error and the unfound line number, thereby making corrections much simpler.

You may have seen other renumbering programs, but NONE with this many features: no external tables are used, runs in 1300 bytes of high memory regardless of program size, loads from and operates on **either** disk- **or** tape- based programs.

Versions available for 4K, 16K, 32K and 48K machines. (Unless specified otherwise, 16K tape automatically supplied) Also available on disk or as source listing

Level II tape (specify version) — $15.00
Diskette (3 versions on one disk) — $25.00
Source Listing — $20.00

TSE TRS-80 Software Exchange
17 BRIAR CLIFF DRIVE MILFORD, NEW HAMPSHIRE 03055



A comprehensive software package, custom-designed for the amateur radio buff who has a TRS-80. Operates in a real-time mode in conjuction with on-the-air activites, support for line printer reports in addition to disk data storage functions. Minimum 32K disk system with one drive.

PROGRAM HIGHLIGHTS INCLUDE:

- **Complete amateur radio routine** Output/input for callsign, time/date contact, frequency, mode, location, name, signal report, QSO end time, QSL sent/received confirmation.
- **Comprehensive amateur DX prefix file** Information on DX prefixes, zone, country, great circle bearing, access anytime
- **Q-signal file** All international Q-signals and ARRL net
- **Special net log routine** Review and print contact stations, check in/out times, net control name and callsign, net start/end, net operating frequency
- **Operating frequency schedule** Allowable modes and requirements for 80-, 40-, 20-, 15-, 10-, 6-, 2-meter bands
- **Propagation forecast** Based on solar flux and K-index
- **Memo/message pad** CW contacts; video and print notation of QSO information or copied message

Available for single disk, 32K TRS-80 system

Two drives will greatly increase storage capabilities

$24.95

TSE TRS-80 Software Exchange

17 BRIAR CLIFF DRIVE MILFORD, NEW HAMPSHIRE 03055

INIT

by George Meyer

This program, called **INIT**, because I'm too lazy to type INITIALIZE, is written for a TRS-80 with 32K RAM.

My serial printer driver is loaded at 'BEDO' hex (-16688 DEC) to 'BF3F' hex (-16577 DEC). Remember, you have to use minus (-) values to poke addresses over 32767. My keyboard debounce routine is at 'BFC9' hex (-16439 DEC) to 'BFFF' hex (-16385 DEC), and the address of the serial driver must be poked into the TRS-80 line printer driver routine at '4025' hex to '4027' hex (16421 to 16423 DEC).

If you don't need these routines, you can use the program to bring your system up and load 'debounce'. Just eliminate lines 10 through 50 and data lines 100 through 230. If you're not using 32K, then you'll have to change line 60 to the address you want for your debounce routine.

To set up your system you must create a directory file of your disk. Save it on disk with the command SAVE"DIRECTORY/TXT", A enter the 'INIT' program and save it. SAVE"INIT", return to DOS. Enter the command, auto BASIC and hit **ENTER**. From now on, every time you start your system, it will display DOS then BASIC (it will load BASIC automatically. Also, BASIC loads faster this way). Answer the memory size to protect any routines you will load, then run "INIT". Your routines will be POKED into memory and the file directory will be listed on the screen.

The directory will be erased when you load a program, but anytime you want to look at it, make sure you use the 'MERGE' command to reload it. This will load the directory from line 60000. It will not affect any lines below that number.

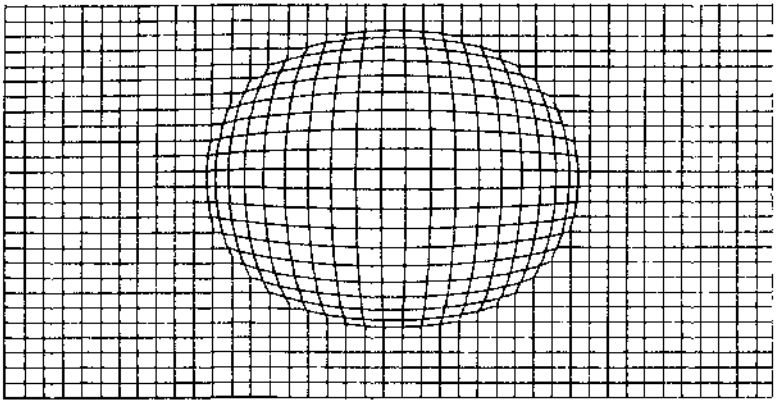
Next, 'RUN 60000' and you have your file list and can load any program you like without returning to DOS.

NOTE: My printer driver looks for an 'S' on port 'EB' as a busy signal which is output from the printer. It may not work for you.

HAVE FUN AND SAVE THE FINGERS AND DISK DRIVES!!

```
0 ' 'INIT' PROGRAM USED TO LOAD ALL I/O DRIVERS & DIRECTORY
10 POKE 16421,2:POKE 16422,208:POKE 16423,190
12 CLS
15 PRINT@512,"RUNNING PRINTER ROUTINE."
20 FOR A%=-16688 TO -16577
30 READ B
40 POKE A%,B
50 NEXT A%
52 CLS
55 PRINT@512,"RUNNING KEYBOARD DEBOUNCE ROUTINE."
60 FOR A%=-16439 TO -16385
70 READ B
80 POKE A%,B
90 NEXT A%
92 CLS
94 PRINT@512,"LOADING DIRECTORY."
95 RUN "DIRECTRY"
100 DATA 229,197,245,58,62,191,254,1
110 DATA 202,0,191,62,1,50,62,191
120 DATA 211,232,219,233,230,248,246,4
```

```
130 DATA 50,61,191,211,234,219,233,230
140 DATA 7,33,53,191,6,0,79,9
150 DATA 126,211,233,62,50,50,63,191
160 DATA 241,193,225,219,234,203,119,202
170 DATA 3,191,219,235,254,19,202,10
180 DATA 191,121,211,235,254,13,194,42
190 DATA 191,58,63,191,61,254,0,202
200 DATA 43,191,50,63,191,14,10,195
210 DATA 3,191,201,62,50,50,63,191
220 DATA 14,12,195,3,191,34,68,85
230 DATA 102,119,170,204,238,0,0,0
250 DATA 175,17,10,0,205,11,0,25,34,22,64,205,97,27,195
260 DATA 25,26,33,54,64,1,1,56,22,0,10,95,174,115,163,32
270 DATA 8,20,44,203,1,242,226,191,201,95,197,1,220,5,205,96
280 DATA 0,193,10,163,200,195,251,3
DISK FILE DIRECTORY PROGRAM
60000 CLEAR 100
60005 A$=CHR$(34)
60010 CLS
60020 PRINT"DIRECTORY DRIVE ";A$;" 0 ";A$
60025 PRINT STRING$(63,"-")
60030 PRINT"TAPEDISK/CMD","DISKDUMP/BAS","HEATHPR/CIM"
60040 PRINT"INIT","DIRECTRY","PARTS"
60050 PRINT"PARTSP","TAROT","TICTAC"
60060 PRINT"BILLS","FILTERS","JIGSAW"
```



Spherical plotting

TSE Order Form

Special prices in effect 60 days from mailing

DESCRIPTION	MEMORY LEVEL		PRICE
ADD HANDLING CHARGE			$1.00
ADDITIONAL CHARGES			
TOTAL ENCLOSED WITH ORDER			

☐Check ☐VISA ☐Master Charge

☐Money Order

ALL SOFTWARE GUARANTEED TO LOAD AND RUN. If you experience difficulties, simply return the tape or disk for free replacement. Send to the attention of Bette Keenan, Customer Service Representative; please enclose a brief note and your name and mailing address with the software.

Not responsible for typographical errors

TRS-80 Software Exchange

17 Briar Cliff Drive Milford, New Hampshire 03055

Telephone (603) 673-5144

Level II software available on disk for a $5.00 (per order) medium charge. This extra fee is for any number of programs transferred to disk from tape when you order. If the order exceeds the capacity of a single disk, we absorb the extra cost.

Please state level and memory size on order form ... otherwise, we automatically ship Level II cassettes.

Be sure to include handling charge and any additional charges when figuring your total. All orders shipped within 48 hours.

Charge card account number

Signature.....

Exp. Date.....Inter. #.....

Charge customers: Please fill in account information above and below

Name.....

Address.....

City.....State.....ZIP.....

ALL SOFTWARE SOLD ON AN AS-IS BASIS WITHOUT WARRANTY TSE assumes no liability for loss or damage caused or alleged to be caused directly or indirectly by equipment or products sold or exchanged by them or their distributors, including but not limited to any interruption in service, loss of business or anticipatory profits or consequential damages resulting from use or operation of such equipment or software.

PROG/80

PO Box 68 Milford, NH 03055

U.S. POSTAGE
PAID
—BULK RATE—
PERMIT NO. 21
MILFORD, NH 03055