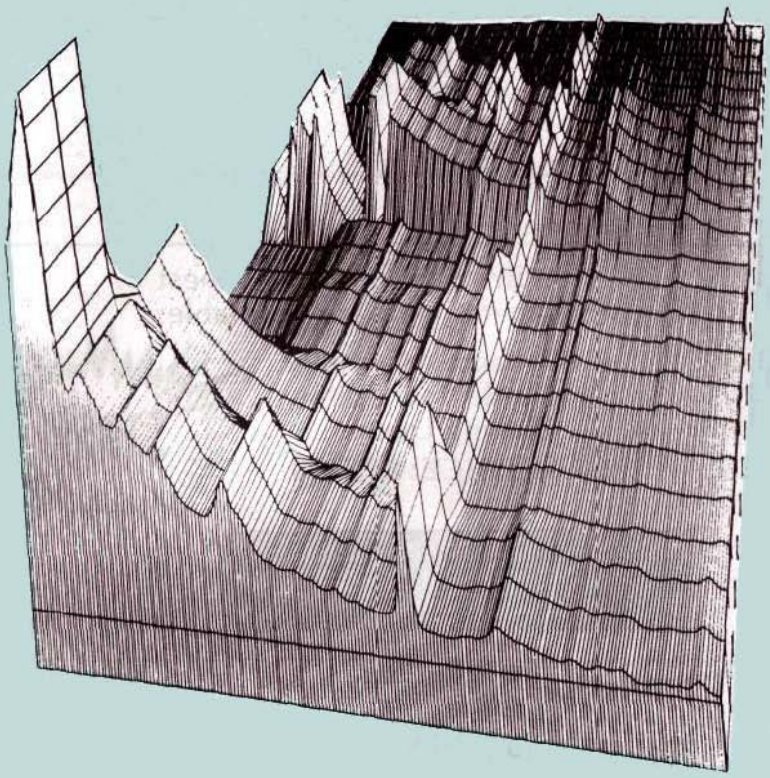
JULY 1979

PROG 80

Dedicated to the serious programmer

$3.00



PROG/80

Dedicated to the serious programmer

July 1979

In This Issue ...

Publisher
Roger W. Robitaille, Sr.

Editor
Paul F. Johnson

Software Editor
Jack Moore

Design/Production
Sharon Demmerle

Business Manager
Elizabeth Robitaille

Layout/Composition
Alice Scofield
Ellie Mae Erion

Subscriptions Manager
Diana Bishop

Customer Service
Bette Keenan

Correspondence
Freida Day

Accounting
Rita Ellis

Contributing Editors
George Blank
Lance Micklus

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.

A SOUND SUBROUTINE

by James Garon

While "messing around" one day, I stumbled upon a short but versatile machine-language subroutine. It allows you to add an impressive array of sound effects to any **BASIC** program. To hear the sounds, just plug the "AUX" wire - which normally goes to the cassette - into any amplifier.

To create the USR routine, follow these steps carefully:

STEP 1: Type in the following lines:

```
10 M$="HERE'S WHERE THE SOUND GOES"
20 I=VARPTR(M$)
25 J=PEEK(I+1)+256*PEEK(I+2)
26 FOR K=J TO J+26
27 READ X :POKE K,X :NEXT
28 DATA 205,127,10,77,68,62,1,105,
        211,255,45,32,253,60,105,
        211,255,45,32,253,13,16,
        238,175,211,255,201
```

STEP 2: If you do **not** have a Disk, add this line:

```
30 POKE 16526,PEEK(I+1):
    POKE 16527,PEEK(I+2)
```

If you **do** have a Disk, use this line instead:

```
30 DEFUSR0 = PEEK(I+1)+256*PEEK(I+2):
    POKE 14308,0
```

STEP 3: Run the program. Be sure the amplifier is on.

STEP 4: Type in the following Test line (without a line number)

```
For I = 0 TO 1 STEP 0: L=USR(0):NEXT
```

If you hear a repeated "whoop", proceed to step 5. If not, press **BREAK** and begin again at step 1.

STEP 5: Press **BREAK** and then **DELETE** lines 35 thru 28.

STEP 6: SAVE OR CSAVE the program for future use. The following loops may be used in a program wherever appropriate. (For now, just type them without a line number to hear how they sound).

```
FOR I=1 TO 50: L=USR(1111): NEXT
FOR I=1 TO 50: L=USR(11111-I): NEXT
FOR I=2060 TO 4096: L=USR(I): NEXT
FOR I=1 TO 100: L=USR(3E3+RND(200)): NEXT
FOR I=1 TO 100 STEP 1/4: L=USR(3E3-ABS(150-3*I)): NEXT
```

Experiment with your own loops. The only restriction (other than good taste) is that the X in USR (X) must be between -32768 and +32767.

You may be wondering why the USR routine is stored in M$ instead of high memory. There are several reasons:

1) You do not need to set memory size, since the routine is embedded in line 10.

2) You do not need to calculate the "entry point" of the USR routine, since the information is available in VARPTR(M$) (see lines 20 and 30)

3) The time needed for **BASIC** to "learn" the routine is greatly reduced. As soon as line 10 is executed, the computer knows the entire routine.

4) Finally, this method saves program memory; each byte of the subroutine wastes up to 4 bytes (3 digits and a comma) and if it is stored in a DATA statement. On the other hand, each byte of M$ is one byte of the USR routine.

For those who are into machine language, here is the subroutine:

```
DEC   HEX   MNEMONIC      COMMENTS
205   CD    CALL 0A7F     ;PASS ARGUMENT TO SUBROUTINE
127   7F
10    0A
77    4D    LD C,L        ;SAVE FREQUENCY
68    44    LD B,H        ;SAVE DURATION
```

```
62      3E      LD A,01         ;FOR RISING SQUARE WAVE
1       01
105     69      LD L,C          ;GET FREQUENCY
211     D3      OUT FF          ;BEGIN SQ.-WAVE
255     FF
45      2D      DEC L           ;COUNT TO MIDDLE
32      20      JR NZ           ; OF WAVE
253     FD
60      3C      INC A           ;FOR FALLING WAVE
105     69      LD L,C          ;GET FREQUENCY
211     D3      OUT FF          ;MIDDLE OF SQ.-WAVE
255     FF
45      2D      DEC L           ;COUNT TO END
32      20      JR NZ           ; OF SQ.-WAVE
253     FD
13      0D      DEC C           ;INCREASE PITCH
16      10      DJNZ            ;IF DURATION > 0
238     EE                      ; MORE SQ.-WAVES
175     AF      XOR A           ;FOR END OF SQ.-WAVES
211     D3      OUT FF
255     FF
201     C9      RET             ;BACK TO BASIC
```

NOTE: Whenever we do an **OUT** to port 255, we also affect the size of the characters displayed. If the USR routine is to be used when the screen is in 32-character format, make the following changes in line 28 (step 1):

7th byte: was 1 — change to 10
14th byte: was 60 — change to 61
24th byte: was 175 — change to 61

BUS-80 IS HERE

It costs $100 for the documentation and/or an additional $100 for the software on diskette. We highly recommend subscribing to the complete program - documentation PLUS software.

In adopting this approach, we're going straight for the jugular (so to speak).

We anticipate a very competitive market for business software relating to the TRS-80 within the next year, and wish to establish ourselves immediately in a dominant position. In doing so, we are presuming over 1,000 participants in the BUS-80 project. Pricing accordingly, we truly feel BUS-80 will become such a fantastic bargain that few serious businessmen who intend to use a TRS-80 within their enterprise could possibly pass up the value offered. Really, how could you pass this up?

We're sure you must be interested in just what you'll be getting .

Well, just about everything you need! Within the year, (and probably within six months you'll receive a disk-based Inventory system — Accounts Receivable system — Accounts Payable system — General Ledger system — Sales — and Payroll.

We're not talking about stripped-down systems

Elements of BUS-80 are already prepared and have been sold individually to satisfied customers for as much as $150. The Name/Address system requires an entire diskette itself (over 50,000 bytes) with some optional subroutines relegated to another supplementary disk.

BUS-80 is not only competitive, it will set the standard by which value is compared. And that standard will be hard to meet.

We would like it understood from the outset that while BUS-80 will deliver a core system for an extremely reasonable price, we'll also be offering other pieces of software for general sale. Usually, BUS-80 participants will be given a discount - in any event, by today's standards, tremendous value will be realized. **The first element of BUS-80 is currently being offered**

THAT'S THE PITCH - BELIEVE IT - IT'S TRUE!

BUS-80

The Business Software People®

17 Briarcliff Drive Milford, NH 03055 **603-673-5144**

INTRODUCING LEVEL III BASIC

Now do more than ever before With the most powerful Basic you can buy For the TRS-80.

Open the manual and load the cassette. Then get ready to work with the most powerful Basic interpreter you've ever had your hands on . . . Level III Basic for Radio Shack Computers. It loads right on top of the Level II ROM, and in just 5K of space, opens up your capability to new dimensions. For starters, this new cassette-based interpreter gives you the whole catalog of disk programming power. Plus graphics commands. Plus Powerful editing commands. Plus long error messages, hex and octal constants and conversions, user defined functions and a number of commands never before available on either cassette or disk interpreters!

EASIER LOADING, FEWER KEYBOARD ERRORS. G2 Level III Basic eliminates aggravations you've had, including keyboard "bounce" and those super-sensitive tape deck settings. Programs will load easier, and you'll have far less trouble with input errors.

BASIC ACCESS TO RS-232. Until now, if you wanted to access your RS-232 interface, you had to work in assembly language. G2 Level III Basic does the work for you, letting you use your interface with Basic statements.

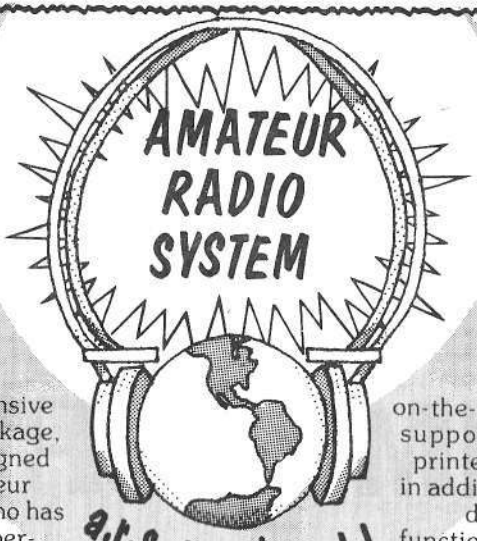
HAVE YOU WISHED FOR MORE POWER? This new interpreter gives you 10 machine language user calls for subroutines, long errors messages, a new TIME$ call for your real time accessory, plus measure or limit input timing that lets you put a time limit on responses when you're playing games or giving exams. And the list doesn't stop here.

EASIER AND MORE POWERFUL GRAPHICS. This new Basic includes three simple commands that can eliminate dozens of program steps. PUT transfers information from a designated array to your screen; GET reverses the process. LINE makes your computer do the work when you input beginning and end points. Give it two diagonally opposite corner locations, and it'll outline the rectangle you're looking for.

ONLY MICROSOFT COULD DO IT. G2 Level III Basic was created by Microsoft, the same company that wrote Level II Basic for Radio Shack. And it actually uses Level II as a foundation for this enhanced add-on. By the time you've mastered all it can do, calling up the flexibility of the graphics commands, and even enjoying the convenience of renumbering, you'll wonder how it was all possible. It's like getting a whole new computer for your computer.

AVAILABLE NOW FOR ONLY $49.95. You get the power that might otherwise cost you hundreds of dollars in additional equipment for only $49.95. Price inlucdes the Users Manual, a Quick Reference Card, and a preprogrammed cassette tape. Load the tape, open the manual, and get ready to work with the most powerful Basic Interpreter you've ever had your hands on. Level III Basic for the TRS-80.

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

MMSFORTH

INTRODUCTORY OFFER

The **MMSFORTH** system diskette or cassette tape provides for the expansion of **FORTH** commands by the user. There are many programs and routines provided as examples of **FORTH** programming, such as:

Routines For:
- String Handling
- Graphics
- File Sorting
- Screen Printing

Programs For:
- Game of Life
- Checkbook Balancing
- String Sort
- Number Guessing Game

The **TRS-80 Software Exchange** intends to fully support the introduction of **MMSFORTH** with the development of supporting application modules. Early **MMSFORTH** projects are:

- floating-point package •
- assembler/cross compiler to provide • standard TRS-80 load modules
- large flexible mailing list system •
- generalized data base management system •
- word-processing package (**FORTHWRITE**) •

MMSFORTH, by **Miller Microcomputer Services,** includes introductory documentation with further references to the ***MicroFORTH*** *primer of* **FORTH, *Inc.*** *This manual is an* invaluable reference for the **FORTH** programmer, and can be purchased separately by anyone desiring more information on the **FORTH** language structure.

30-DAY INTRODUCTORY PRICE

MMSFORTH cassette version, Level II, 16K	$34.95
MMSFORTH disk version, Level II, 16K	44.95
MicroFORTH primer	15.00

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

MACHINE LANGUAGE VIDEO OUTPUT

by Paul Johnson

One of the first problems a machine language programmer faces is how to provide input and output for his program. Usually, input is from the keyboard (see "A Piece of the ROM", **PROG-80**, May 1979). This article will discuss output to the CRT (video display).

First, we'll explain a few basics. The video display is "memory mapped"; there are 1024 locations on the screen, corresponding to 1024 (1K) bytes of "video RAM" (address 3C00H to 3FFFH, or 15360 to 16383). The computer constantly scans video RAM, and displays at each point on the screen the alphanumeric or graphic character associated with the ASCII code stored in the corresponding byte. When the screen is clear, video RAM contains only 20H (32 Decimal), the ASCII code for "space". In order to print an E in the lower right corner of the screen, we must load the ASCII code for E (45H) into address 3FFFH.

Suppose we want to print the word HELLO on the screen. In BASIC we could say 'PRINT"H";:PRINT"E";:PRINT"L";: etc. Our machine language equivalent would be:

```
5500                    ORG     5500H
5500 21003C             LD      HL,3C00H        ;UPPER LEFT CORNER
5503 3648               LD      (HL),48H        ;ASCII 'H'
5505 23                 INC     HL              ;POSITION FOR NEXT CHAR.
5506 3645               LD      (HL),45H        ;ASCII 'E'
5508 23                 INC     HL
5509 364C               LD      (HL),4CH        ;ASCII 'L'
550B 23                 INC     HL
550C 364C               LD      (HL),4CH        ;ASCII 'L'
550E 23                 INC     HL
550F 364F               LD      (HL),4FH        ;ASCII 'O'
5511 C31155   LP1       JP      LP1             ;ENDLESS LOOP...
```

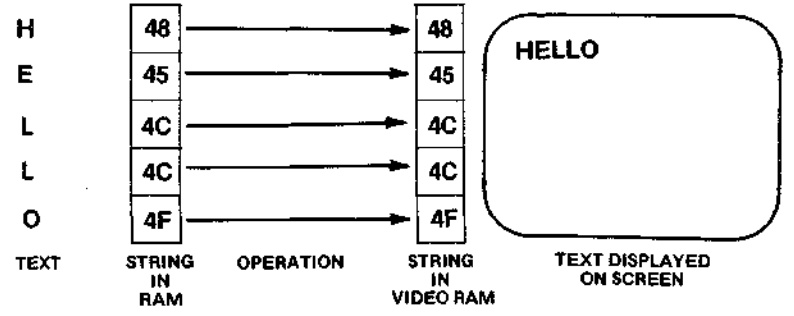
By changing the address in the first line from 3C00 to some other location in video RAM, we can print anywhere on the screen.

In BASIC, a much more powerful PRINT format than the one shown above is the instruction, 'PRINT @ X,B$. This allows the printing of string information at a specified location on the screen. The same instruction is easily achieved in machine language. In fact, the method we will describe is the one used by the BASIC interpreter when it needs to PRINT. First, we need to have a string (text and/or graphics) to print. A string can be considered as a set of ASCII codes stored in memory. The initial source of the string data could be the keyboard, cassette tape, load instructions as above, or even ROM. The important thing is that it is stored somewhere in memory as a contiguous group of data bytes.

To print our string, we first must know three things:

1) The address of the first byte of the string
2) The length of the string
3) The address in video RAM where 'printing' is to begin

This block diagram shows what we will do:



The operation consists of copying the string, byte-for-byte, into video RAM. To perform this task we will use one of the most powerful Z-80 instructions: LDIR (load, Increment, Repeat), or "block move". LDIR is a very useful, yet simple, instruction. It uses the HL and DE register pairs as 16-bit pointers (i.e., HL and DE contain the addresses where transfer will take place). The HL register points to the source address; the DE register points to the DEstination address. The BC ("Byte Count") register contains the number of bytes to be transferred; it serves as a sort of FOR-NEXT loop.

To use LDIR, it is necessary to initialize the above-mentioned registers. Then, when LDIR is executed, it automatically performs the following steps:

1) copy a byte of data from the source (HL) address to the destination (DE) address
2) increment both HL and DE to the next higher address
3) decrement (count down) BC
4) if BC = 0 stop, otherwise start again at 1

Here is our "PRINT @ X; B$" implemented in machine language (assuming our string is already stored in memory):

```
LD  HL,  X        Where ; X  is starting address of string,
LD  DE,  Y              ; Y  is starting location for print,
LD  BC,  Z              ; Z  is length of string
LDIR                    ; Block Move
```

Block move may also be used to white out the screen. Notice how HL "chases" DE around the screen!

```
LD  HL, 3C00H          ; Upper Left Corner
LD  (HL), 0BFH         ;Whiteout Upper Left Position
LD  DE, 3C01H          ;Upper Left + 1
LD  BC, 400H           ; 1K Positions on screen
LDIR
```

In yet another use, HL is set to 3C40H, DE is set to 3CØØ, and BC is set to 3CØH. The result? Everything moves up one line, scrolling the display. Why? Because each byte is moved "backward" in memory 64 decimal bytes, which corresponds to a shift upward to a position directly above the starting point on the screen.

In closing, if you wish to try any of these programs (which are presented here as instructions rather than finished routines), insert an ending routine similar to the one found on page 6, lines 170-210 of the Radio Shack Editor Assembler manual. This will freeze the display for 5 seconds, so that you can see the results of your work. Remember, the only way to learn a technique is to try it!

Memo To RESCUE Program Users*

As set up, the RESCUE program will only work if you use the default value of 3 files. If you answer files? with 0 or any number except 3 or ENTER, BASIC will begin at a different location. That would require several changes in the program, at the 1st, 9th, 11th, and 14th assembly language instructions found in lines 10, 20, and 30. There would be different values for each number of files. You could find the beginning of BASIC using DEBUG to substitute the new addresses.

*RESCUE was published in May **PROG-80**

Don't Miss Out!

If you've moved recently, or are planning on moving in the near future, please verify by filling out the form below. This way, you'll be sure not to miss any issues. [include your present label]

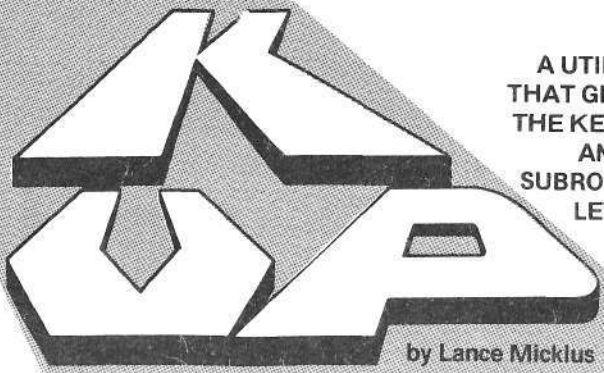
Name: _____

Address: _____

City: _____

State: _____ **Zip Code:** _____

When Effective: _____



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:

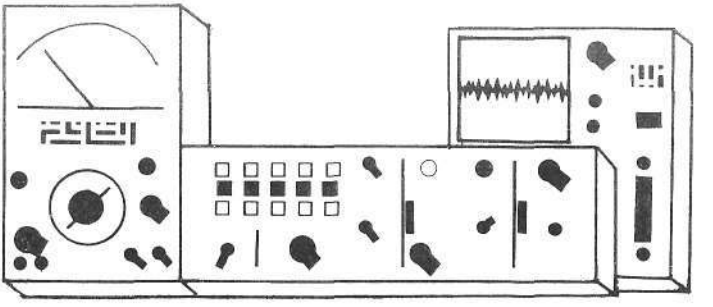
- **USE AN EXTERNAL KEYBOARD** Or, use any other serial input device in place of the TRS-80 keyboard
- **ELIMINATE A COMMON SOURCE OF PROGRAM ERRORS** by running your keyboard in upper case only, or run in upper/lower case mode just like a typewriter
- **PRACTICALLY ELIMINATE KEYBOARD BOUNCE** The amount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

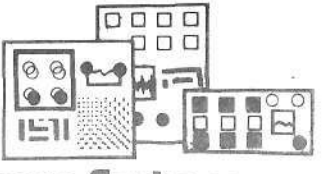


ELECTRONICS ASSISTANT

John Adamson

Electronics Assistant is a set of nine circuit design programs in one. Written by a professional for the serious electronics buff, Electronics Assistant will draw schematics and help you design active and passive low-, band-, and high-pass filters, coils, attenuator networks, and three types of impedance-matching networks. Features extensive graphics and a one key selection routine. Circuit designers and students will wonder how they lived without their Electronics Assistant!

For 16K Level II
Price, $9.95



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

HIGH DENSITY DATA STORAGE

George Blank

In programs that use a lot of data, you may run out of memory before you can effectively store all your information. The solution to this problem is a technique called bit packing, in which you use only one bit of memory to store each piece of information.

With bit packing, you can store up to 16 different pieces of information in one integer variable, 32 in a single precision variable, and 64 in a double precision variable. In order to store the maximum number of information sets, you need data with only two conditions. That is, you can store in one bit whether or not a certain condition is met, but need several bits to determine one of several conditions.

In adventure games it is often necessary to store a great deal of information including the possible exits from each room, objects and characters and special conditions in that room, and references to help statements. This article will use the storing of exit information in such a game as the example for a demonstration of a general purpose bit packing and bit unpacking routine.

Theory

An integer variable uses sixteen bits to store data. The leftmost bit tells whether the number is positive or negative, and the rest contain the binary code for the number, as follows;

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Ø
	Sign	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

The bits are numbered as they are because the number represents the power of 2 indicated by that bit. For example, $2^0 = 1$, $2^1 = 2$, $2^9 = 512$, and $2^{14} = 16,384$. If any bit is a 1 you add the value of that bit to the others to find the number indicated. If any bit is a zero, you skip over that bit in determining the value. The sign bit, number 15, could represent 32,768, but TRS-80 BASIC uses a 1 in that bit to represent a negative number and a 0 to represent a positive number and an integer variable.

Single and Double Precision Variables are more difficult to use for bit packing, not only because it takes longer and slower routines to pack and unpack the information, but also because they use some of their bits to indicate the position of the decimal point in scientific notation. For that reason, this article will only consider integer variables.

The Routines

General purpose bit packing and unpacking routines are given in the sample program for integer variables. The Unpacking routine begins at line 1000 and the Packing routine at line 2000. Lines 30 to 290 contain a sample program to illustrate the use of these subroutines. All values used in the subroutines begin with B, making it easy to reserve variables for this purpose. In both routines, the packed data is indicated by the variable BX, the unpacked data from each bit by the variables B(0) to B(15), and working variables used in the subroutines are B, the current bit being examined, and BV, the value of that bit.

Unpacking

To use the unpacking subroutine, store the variable to be unpacked in BX and call the subroutine at 1000 as is done by line 130 in the sample program. The subroutine first examines bit 15 by checking for a negative number, storing a one in B(15) if it is negative, a zero if it is not. Then it checks bits 14 down to 0 in turn by testing to see if the number given is less than the current power of 2. If the number is less, a zero is stored in the B array. If the number is equal or larger, a 1 is stored in the array and that power of two is subtracted from the number to prepare it for the next test. Then the routine divides BV, the bit value, by 2 and tests the next bit.

Packing

The packing routine works in opposite fashion to the unpacking routine. The data to be packed is stored in the array B(0) to B(15), as demonstrated partially in lines 220 to 270 of the sample program. Then the subroutine is called, as in line 280, and the storage variable is then set equal to BX. The packing routine begins with 1, equal to 2^0, and adds the current power of two to BX for each number in the B array set to 1. After each number is tested, the value BV is multiplied by 2 to give the next power of two. Line 2020 bypasses this for 214 to avoid an overflow error. Line 2040 sets the sign bit to correspond to B(15).

Going Beyond Single Bit Data

If we wish to store more information than can fit in a single bit, we can use several bits. For example, if we had 10 characters who might be in the

rooms, with only one in any single room, we could use bits, 6, 7, 8, and 9 to store the character number. This single line could give us a character number from 0 to 15: CH=B(6)+2*B(7)+4*B(8)+8*B(9)

To store a character in a room, once we determined that no other character was present, we could simply multiply the character number by the power of two represented by the first bit used for storage and add it to the value. Thus, to store character number 8 in bits 6-9 of C(Ø), which might represent a monster in the basement, we just use this line:

$$C(Ø)=C(Ø)+8*64$$

This is because 2^6, the power of bit 6, equals 64.

As an exercise, you might wish to add a list of characters to the sample program and store them in the array C(Ø) to C(12) as well as the room exits. If you can do this, you fully understand the method and should be able to use it in your own programs. Please note: these subroutines are not copyrighted. You may use them exactly as they are in your own programs, and you do not have to give credit for them.

```
10 REM * BIT PACKING DEMONSTRATOR *
20 REM * GEORGE BLANK - MAY 5 1979 *
30 DEFINT A-C:DEFSTR D-F:DIM B(15):DIM C(12):DIM D(12):DIM E(6)
40 FOR A=0 TO 12:READ C(A),D(A):NEXT
50 DATA 20,BASEMENT,3,PORCH,30,MUSIC ROOM,3,LIVING ROOM,9,DINING
 ROOM,45,KITCHEN,49,LAUNDRY
60 DATA 62,HALL,2,STUDY,2,GUEST ROOM,8,NURSERY,8,MASTER BEDROOM,
1,BATHROOM
70 FOR A=0 TO 5:READ E(A):NEXT
80 DATA NORTH,EAST,SOUTH,WEST,UP,DOWN
90 CLS: PRINT"ROOM"," ","EXITS"
99 REM
   * DISPLAY ROOMS AND EXITS *
100 FOR A=0 TO 12
110  PRINT A; TAB(12)D(A),
120  IF C(A)=0 PRINT"NONE":GOTO 190
130  BX=C(A):GOSUB 1000
140  FOR B=0 TO 5
150   IF B(B)=1 PRINT E(B);" ";
160  NEXT B
170  PRINT
190 NEXT A
199 REM
```

```
    * CHANGE EXITS *
200 INPUT"IN WHAT ROOM (NUMBER) WILL YOU CHANGE THE EXITS";A
210 PRINT@ 896,CHR$(31);PRINT@ 896,D(A);
220 FOR B=0TO5
230  PRINT@ 916,"CAN YOU EXIT ";E$(B);" (Y/N) ?";CHR$(95);"
240  B(B)=0
250  I$=INKEY$:IF I$=""THEN 250
260  IF I$="Y"THEN B(B)=1
270 NEXT B
280 GOSUB 2000:C(A)=BX
290 GOTO 90
899 GOTO 899
999  REM
    * BIT UNPACKING ROUTINE *
    BX=INPUT  B=CURRENT BIT  BV=BIT VALUE  B(B)=OUTPUTS
1000 IF BX>=0 THEN B(15)=0 ELSE BX=BX+-1:B(15)=1
1010 BV=16384:FOR B=14 TO 0 STEP -1
1020   IF BX<BV THEN B(B)=0 ELSE B(B)=1:BX=BX-BV
1030 BV=BV/2:NEXT B
1090 RETURN
1999 REM
    * BIT PACKING ROUTINE *
    B(B)=INPUTS  B=CURRENT BIT  BV=BIT VALUE  BX=OUTPUT
2000 BX=0:BV=1:FOR B=0 TO 14
2010  IF B(B)=1 THEN BX=BX+BV
2020 IF B<14 THEN BV=BV*2
2030 NEXT B
2040 IF B(15)=1 THEN BX=0-BX
```

PROGRAMMING HINT

One way to add interest to a game is with real time action. This routine will pause a few seconds for an input, then continue if none is given, setting an input flag (I$ = "F") to indicate that no input was provided. Then you can test for I$ equal to F to assess a penalty if you wish. The timing can be adjusted with the FOR loop.

```
10 FOR A = 1TO500:I$ = INKEY$:IF I$" " THEN NEXT:I$ = "F"
20 IF I$ = "F" PRINT"YOU WERE NOT FAST ENOUGH!"
```

HISTOGRAM AND BASIC STATISTICS PROGRAM

General: This program is written in Level II basic for the TRS-80, and can be run in less than 4K of memory. It provides a histogram (bar graph) with as many as 14 bars. In addition, it calculates and displays with each entry error of the mean, and the coefficient of variation. The upper and lower values for the bar graph can be specified, as well as the number of characters to be included in the data string. This program is particularly useful as it allows both a visual and mathematical look at the data as it is entered.

Statistical Background: The histogram is a method for displaying univariate data, that is, a series of numbers representing a single variable. For example, such variables as test scores for a class, golf or bowling scores, can be entered into the program and displayed visually. The basic statistics will be calculated, providing additional information.

One of the most important items of data that can be obtained visually from a histogram is an estimate of the frequency distribution of the data that has been entered. Figure 1 provides several examples of possible frequency distributions. These are normally displayed with the primary axis along the bottom, but for this program the primary axis has been shifted to the vertical to allow longer bars to be presented.

The formulas which have been utilized for each of the statistical procedures are listed below:

Number.-- n — Determined without calculations; the number of items of data which have been entered.

Number.-- n — Determined without calculations; the number of items of data which have been entered.

Mean.-- $\frac{\Sigma X}{n}$ — This represents the sum of the items of data (X) divided the number (n).

Variance.-- $s^2 = \frac{\Sigma X^2_i - (\Sigma X_i)^2/n}{n-1}$ — This formula represents the sum of squares minus the square of the sums divided by n, all of which is divided by n-1.

Standard deviation.-- $s = \sqrt{s^2}$ — The standard deviation is the positive square root of the variance.

Standard error of the mean.-- $s_{\bar{X}} = s/\sqrt{n}$ — The standard error of the mean is the sample standard deviation of the mean.

Coefficient of variation.-- $CV = \frac{100s}{\text{mean}}$ — The ratio of the sample standard deviation to the sample mean expressed as a percentage.

The n-1 method has been used for calculating the variance and standard deviation so that the formulas are suitable for use with small sample sizes.

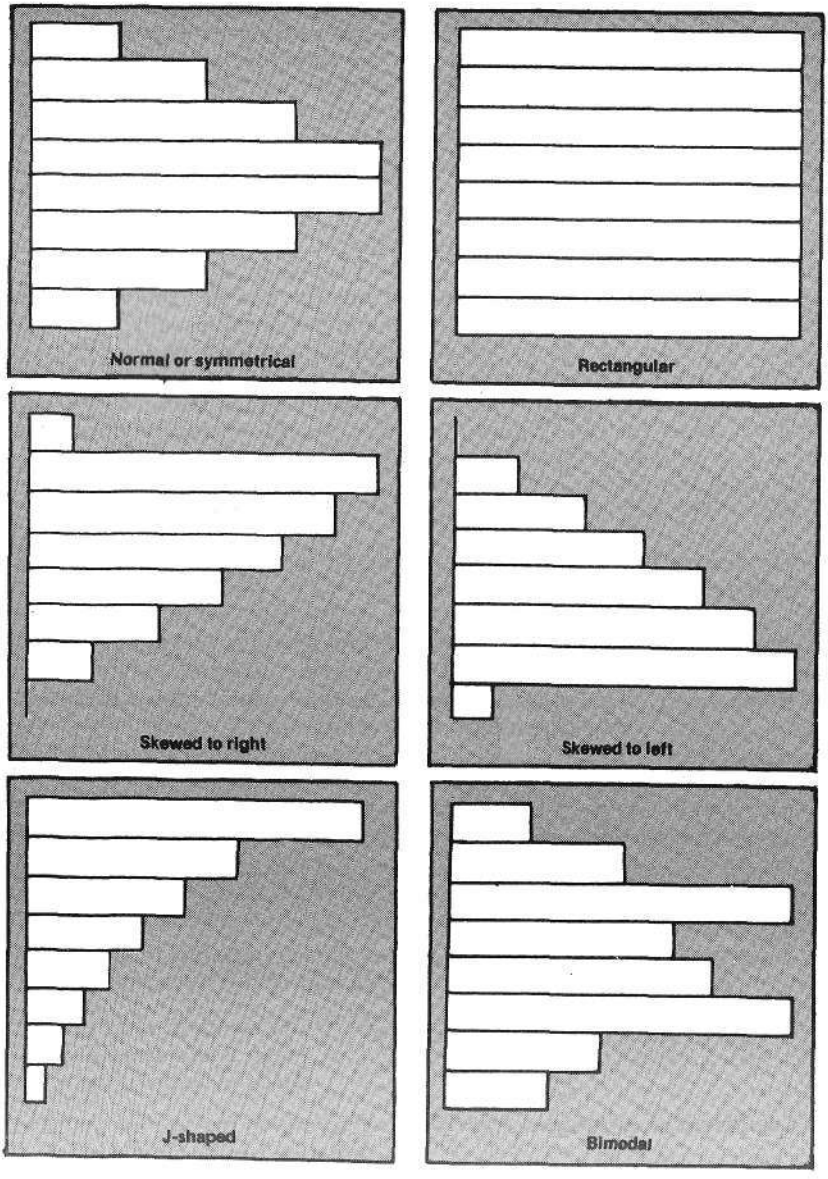


FIGURE 1: SAMPLE HISTOGRAM DISTRIBUTIONS

Figures are normally oriented along the main axis, i.e. so that the bars are oriented up and down. The terms left and right skewed are meant to apply to figures with the normal orientation.

Brief definitions of these distributions [from Johnson 1976] are as follows;

--Symmetrical;	This shape is a mirror image on the opposite sides of a line dividing distribution in the middle.
--Rectangular;	Every value appears with equal frequency.
--Skewed;	One tail is stretched out longer than the other. The direction of skewness is on the side of the longer tail.
--J-shaped;	There is no tail at all on one side of the most populous class.
--Bimodal;	The two "most" populous classes are separated by one or more classes. This situation often implies that two populations are being sampled. (Johnson 1976:61).

The six statistics that are provided are described below;

Number.--The number of items of data that have been entered.

Mean.--The average of all of the items of data that have been entered. This and the other statistics described below are only displayed after the fith item of data is entered.

Variance.--The variance of a sample is a measure of the distribution or spread of data around the mean. For example, if the numbers 66, 68, 70, 72 & 74 or the numbers 64, 67, 70, 73 & 76 are entered, the mean is still 70, but the variance (spread of the numbers) is greater, at 22.5. The variance, then, is large when numbers are far from the mean.

Standard deviation.--The standard deviation is the positive square root of the variance. It is often easier to conceptualize the standard deviation of a set of data than it is the variance because the standard deviation is expressed in the same units as the data. If the variable (x) is measured in inches, the variance 2^S is measured in square inches, while s is measured in inches. Like the variance, the standard deviation is a measure of the spread of a set of data. In the example given above, the standard deviation for the first data set (66, 68, 70, 72, & 74) is 3.16, while for the second data set (64, 67, 70, 73 & 76) it is 4.74, indicating a greater relative spread from the mean.

Standard error of the mean.--This statistic represents the sample standard deviation of the mean. Basically this is the standard deviation of a series of means drawn from the population being sampled. Those unsure of the meaning of this statistic should consult several elementary statistic books for the background data required for its proper use.

Coefficient of variation.--This is a very simple but useful statistic. It is calculated by dividing the standard deviation by the mean and multiplying by 100 (making it a percent). This statistic is also a measure of the spread of the data set. The CV should be relatively small when experimental data are being used. The following table gives some ballpark figures against which to compare sample data:

CV	TYPE OF DATA/EXPERIMENT
< 10%	a planned experiment should be
< 5%	a good experiment is
~ 20%	typical for biological data
> 30%	typically is not a normal distribution

The Program

The program is initiated by a RUN statement. It then asks "DO YOU DIRECTIONS (YES=1)?" Enter a 1 to view the instructions, or just press ENTER to proceed directly to the program.

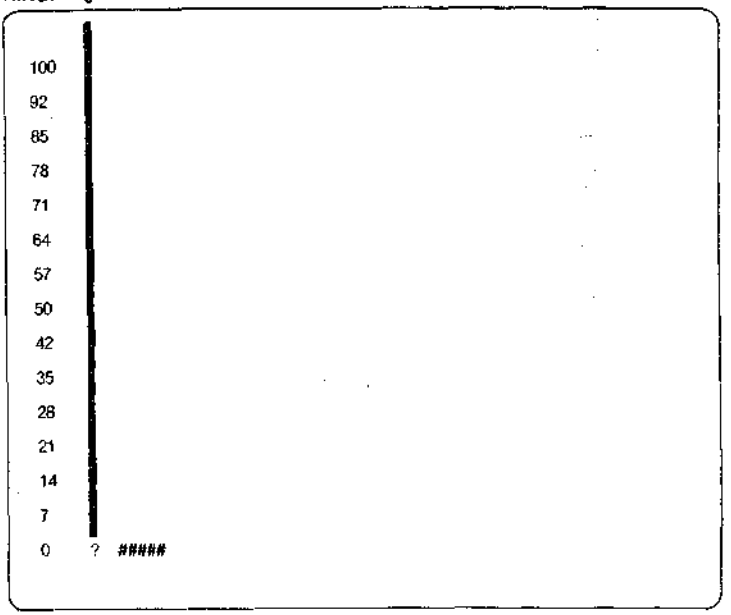
The first point that must be decided is the number of bars desired in the histogram. The program will ask "HOW MANY BARS?" Any number upt to a maximum of 14 bars can be displayed. The program will then ask "UPPER LIMIT?" and "LOWER LIMIT?" Enter numbers which will allow the entire data set to be displayed but which will not compact all of the data into only 1 or 2 bars. The maximum number that can be entered as the upper limit is 9999, and the lower limit must be between 0 and the upper limit. The program will reject numbers that do not meet these criteria, and restate the question. Additionally, when the data is being entered, numbers greather than the upper limit and less than the lower limit will not be entered into the histogram or into the statistical calculations. The final question the program will ask is "HOW MANY CHARACTERS (INCLUDING DECIMAL POINT) IN THE LONGEST DATA STRING?" Some examples of possible data strings and the number of characters are presented below:

12	2 characters	1.1	3 characters	234.340	6 characters
102	3 characters	1.05	4 characters	1212.321	8 characters
1239	4 characters	12.0	4 characters	43212.2323	10 characters

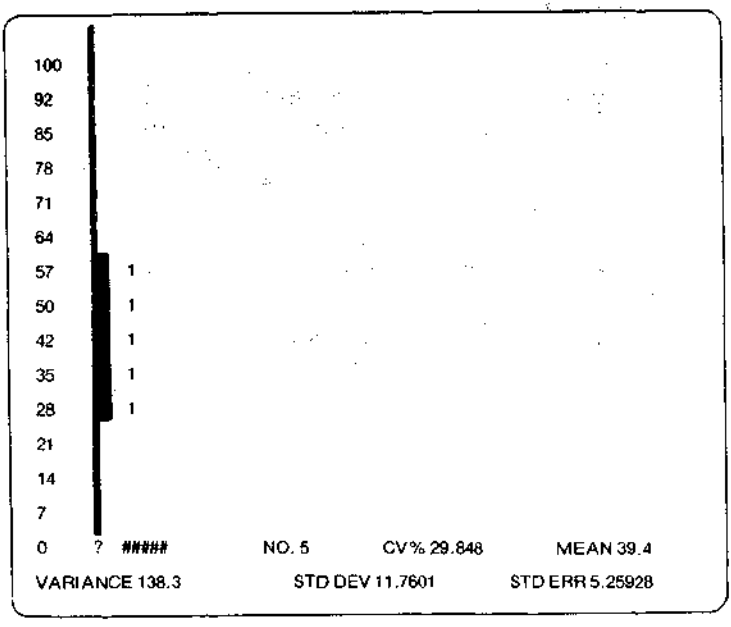
The maximum number of characters that can be entered is 10. The program will reject any larger response.

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

```
DO YOU WANT DIRECTIONS (YES = 1)?   enter 1 to see directions, press enter to continue
HOW MANY BARS?                      14
UPPER LIMIT?                        100
LOWER LIMIT?                        0
HOW MANY CHARACTERS (INCLUDING DECIMAL POINT) IN THE LONGEST DATA STRING? 5
```



This is the screen display before the entry of any data. The five symbols at the bottom left of the screen will accept the five (or fewer) data characters that are entered, and will display the character entered. If a mistake is made, the backspace key will remove it. When the data has been entered into the grid, press **ENTER**. The data will be input, and the grid cleared for the next data entry. After five items have been entered, the screen will begin to display the statistics. After entering the following numbers (44, 55, 23, 40 & 35) the display will appear as shown on the next page.



HINTS

•The maximum number of entries that can be handled by each bar is about 53, after which that appears to the right of each begins to run off the screen.

•Don't forget that the histogram is meant to be right side up, i.e., the bars should be vertical.

REFERENCE

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

```
10 CLS:PRINT:PRINT CHR$(23)"     HISTOGRAM/"
20 PRINT "          SCATTERGRAM"
30 PRINT:PRINT:PRINT:PRINT"   WHAT IS YOUR CHOICE";
40 X$=INKEY$:IF X$="" THEN 40
50 IF X$="H"ORX$="1" THEN 80
60 IF X$="S"ORX$="2" THEN 600
70 GOTO 40
80 REM HISTOGRAM/BASIC STAT COPYRIGHT 1979 BY GARY S. BRESCHINI
- FOR LESS THAN 48 VALUES/BAR
90 CLS:CLEAR100:PRINT@144,"HISTOGRAM & BASIC STATISTICS"
100 PRINT:INPUT"DO YOU WANT DIRECTIONS (YES=1)";O:IFO=1GOSUB540
110 PRINT:INPUT"HOW MANY BARS";O
120 IFO>14PRINT"NO MORE THAN 14 BARS":GOTO110
130 INPUT"UPPER LIMIT";T
140 IFT>9999PRINT"UPPER LIMIT TOO BIG":GOTO130
150 INPUT"LOWER LIMIT";W:IFW<0PRINT"LOWER LIMIT CANNOT BE NEGATI
VE":GOTO150
160 Q=(T-W)/O:IFW=>TPRINT"LOWER LIMIT LARGER THAN UPPER LIMIT":G
OTO150
170 INPUT"HOW MANY CHARACTERS (INCLUDING DECIMAL POINT) IN THE L
ONGEST    DATA STRING";QQ:IFQQ>10PRINT"TOO MANY CHARACTERS.":GOT
O170
180 CLS:FORY=0TO(O*3)-1:SET(13,Y):NEXT
190 FORU=0TOO:PRINT@0+INT(64*U),INT(T-(U*Q)+.1);:NEXT
200 X$="":A$="":PRINT@902,"";:PRINT"? ";STRING$(QQ,CHR$(35));:FO
RU=1TOQQ:PRINTCHR$(24);:NEXT
210 A$=INKEY$:IFLEN(A$)=0THEN210
220 IFA$=CHR$(13)GOTO280
230 IFA$=CHR$(8)ANDX$=""PRINTCHR$(24);CHR$(24);:GOTO200
240 IFA$=CHR$(8)ANDX$<>""PRINTCHR$(8);CHR$(35);CHR$(24);:X$=LEFT
$(X$,LEN(X$)-1):GOTO210
250 IFA$=<CHR$(45)ORA$=>CHR$(58)THEN210
260 IFLEN(X$)=QQTHEN210
270 PRINTA$;:X$=X$+A$:GOTO210
280 X=VAL(X$):IFX<WGOTO200
290 IFX>TGOTO200
300 IFX=WTHENX=X+.001
310 IFX<=T ANDX<(T-Q)A=A+1:PRINT@7,STRING$(A,191);:PRINT@7+A,A;
320 IFX<=T-QANDX>T-(2*Q)B=B+1:PRINT@71,STRING$(B,191);:PRINT@71+
```

NOTE: Lines **600** on are an updated version of "SCATTERGRAM" (See May PROG/80.)

```
B,B;
330 IFX<=T-(2*Q)ANDX>T-(3*Q)C=C+1:PRINT@135,STRING$(C,191);:PRIN
T@135+C,C;
340 IFX<=T-(3*Q)ANDX>T-(4*Q)D=D+1:PRINT@199,STRING$(D,191);:PRIN
T@199+D,D;
350 IFX<=T-(4*Q)ANDX>T-(5*Q)E=E+1:PRINT@263,STRING$(E,191);:PRIN
T@263+E,E;
360 IFX<=T-(5*Q)ANDX>T-(6*Q)F=F+1:PRINT@327,STRING$(F,191);:PRIN
T@327+F,F;
370 IFX<=T-(6*Q)ANDX>T-(7*Q)G=G+1:PRINT@391,STRING$(G,191);:PRIN
T@391+G,G;
380 IFX<=T-(7*Q)ANDX>T-(8*Q)H=H+1:PRINT@455,STRING$(H,191);:PRIN
T@455+H,H;
390 IFX<=T-(8*Q)ANDX>T-(9*Q)I=I+1:PRINT@519,STRING$(I,191);:PRIN
T@519+I,I;
400 IFX<=T-(9*Q)ANDX>T-(10*Q)J=J+1:PRINT@583,STRING$(J,191);:PRI
NT@583+J,J;
410 IFX<=T-(10*Q)ANDX>T-(11*Q)K=K+1:PRINT@647,STRING$(K,191);:PR
INT@647+K,K;
420 IFX<=T-(11*Q)ANDX>T-(12*Q)L=L+1:PRINT@711,STRING$(L,191);:PR
INT@711+L,L;
430 IFX<=T-(12*Q)ANDX>T-(13*Q)M=M+1:PRINT@775,STRING$(M,191);:PR
INT@775+M,M;
440 IFX<=T-(13*Q)ANDX>T-(14*Q)N=N+1:PRINT@839,STRING$(N,191);:PR
INT@839+N,N;
450 IFX=W+.001THENX=X-.001
460 Z=Z+1:S=S+X:R=R+(X*X):IFZ<=4GOTO200
470 V=(R-((S*S)/Z))/(Z-1):CV=100*SQR(V)/(S/Z):SE=SQR(V)/SQR(Z)
480 PRINT@921,"  ";:PRINT@916,"NO. "Z;
490 PRINT@932,"       ";:PRINT@925,"CV% "CV;
500 PRINT@970,"       ";:PRINT@960,"VARIANCE "V;
510 PRINT@997,"    ";:PRINT@985,"STD DEV "SQR(V);
520 PRINT@949,"         ";:PRINT@943,"MEAN "S/Z;
530 PRINT@1015,"     ";:PRINT@1004,"STD ERR "SE;:GOTO200
540 PRINT@16,"HISTOGRAM & BASIC STATISTICS"
550 PRINT:PRINT"TO INITIALIZE THE PROGRAM ENTER:":PRINT" - NUMBE
R OF BARS DESIRED":PRINT" - UPPER LIMIT OF DATA":PRINT" - LOWER
LIMIT OF DATA":PRINT" - MAXIMUM NUMBER OF DIGITS PER ENTRY":PRIN
T:PRINT"VALUES OUTSIDE OF LIMITS ARE IGNORED.";
560 PRINT"  THE UPPER LIMIT FOR DATA IS 9999.  A MAXIMUM OF 14 B
```

```
ARS ARE POSSIBLE.  INPUT THE MAXIMUM LENGTH OF DATA STRINGS ";
570 PRINT"(INCLUDING DECIMAL POINT).  IF AN ERROR IS MADE, THE B
ACKSPACE WILL ERASE IT.  WHEN THE DATA IS CORRECT, PRESS ENTER."

580 PRINT"  AFTER THE FIFTH ENTRY SUMMARY STATISTICS WILL BE  DI
SPLAYED.  (VARIANCE AND STANDARD DEVIATION ARE CALCULATED BY  TH
E N-1 METHOD.)"
590 INPUT"PRESS ENTER TO CONTINUE";U:CLS:RETURN
600 REM SCATTERGRAM/CORRELATION PROGRAM COPYRIGHT 1979 BY GARY S
  BRESCHINI, 379 CORRAL DE TIERRA RD, SALINAS, CA  93908  1/79
610 CLS:PRINT@81,"SCATTERGRAM & CORRELATION":PRINT:PRINT:INPUT"D
O YOU WANT DIRECTIONS (YES=1)";A:IFA=1GOSUB1160
620 CLS:INPUT"INSERT MAXIMUM VALUE OF X";XH
630 INPUT"INSERT MINIMUM VALUE OF X";XL:XX=XH-XL
640 IFXL>XHPRINT"LOWER LIMIT EXCEEDS UPPER LIMIT.":GOSUB1230:GOT
0620
650 IFXH>9999PRINT"X UPPER LIMIT TOO LARGE.":GOSUB1230:GOTO620
660 INPUT"INSERT MAXIMUM VALUE OF Y";YH
670 INPUT"INSERT MINIMUM VALUE OF Y";YL:YY=YH-YL
680 IFYL>YHPRINT"LOWER LIMIT EXCEEDS UPPER LIMIT":GOSUB1230:GOTO
660
690 IFYH>9999PRINT"Y UPPER LIMIT TOO LARGE.":GOSUB1230:GOTO660
700 ZX=XX/114:ZY=YY/43
710 CLS:FORX=12TO127:SET(X,0):SET(X,44):NEXT
720 FORY=0TO43:SET(12,Y):SET(127,Y):NEXT
730 PRINT@965,XL;:PRINT@971,"X(#####) Y(#####)";:PRINT@1011,"
      ";
740 A$="0":B$="0":C$="0":D$="0":E$="0":F$="0":H$="0":I$="0":J$="
00":X$="0":Y$="0"
750 PRINT@1017,XH;:PRINT@993,INT(XL+(XH-XL)/2);
760 PRINT@1000,"COR";
770 I=(YH-YL)/15
780 FORX=0TO14STEP2:PRINT@(0+(64*X)),INT(YH-(X*I));:NEXT
790 A$=INKEY$:IFLEN(A$)=0THEN790
800 PRINT@973,A$;
810 B$=INKEY$:IFLEN(B$)=0THEN810
820 PRINT@974,B$;
830 C$=INKEY$:IFLEN(C$)=0THEN830
840 PRINT@975,C$;
850 D$=INKEY$:IFLEN(D$)=0THEN850
```

```
860 PRINT@976,D$;
870 E$=INKEY$:IFLEN(E$)=0THEN870
880 PRINT@977,E$;
890 X$=A$+B$+C$+D$+E$
900 F$=INKEY$:IFLEN(F$)=0THEN900
910 PRINT@982,F$;
920 G$=INKEY$:IFLEN(G$)=0THEN920
930 PRINT@983,G$;
940 H$=INKEY$:IFLEN(H$)=0THEN940
950 PRINT@984,H$;
960 I$=INKEY$:IFLEN(I$)=0THEN960
970 PRINT@985,I$;
980 J$=INKEY$:IFLEN(J$)=0THEN980
990 PRINT@986,J$;
1000 Y$=F$+G$+H$+I$+J$
1010 IFJ$="X"GOTO730
1020 Q=Q+1:C=VAL(X$):D=VAL(Y$)
1030 K=C:L=D:C=C-(XL):D=D-YL:C=(C/ZX)+13:D=D/ZY
1040 IF K<XL THEN PRINT@1011,"X TOO SMALL";:GOSUB1230:GOTO730
1050 IFL<YLTHENPRINT@1011,"Y TOO SMALL";:GOSUB1230:GOTO730
1060 IFC>127THENPRINT@1013,"X TOO BIG";:GOSUB1230:GOTO730
1070 IFD>43THENPRINT@1013,"Y TOO BIG";:GOSUB1230:GOTO730
1080 D=44-D:SET(C,D):PRINT@971,"X(#####) Y(#####)";
1090 GOSUB1100:PRINT@1005,"      ";:PRINT@1004,CC;:GOTO730
1100 N=N+1:SX=SX+K:SY=SY+L:AA=AA+(K*K):BB=BB+(L*L):XY=XY+(K*L)
1110 IFQ=<4GOTO730
1120 XD=SQR((AA-((SX*SX)/N))/(N-1))
1130 YD=SQR((BB-((SY*SY)/N))/(N-1))
1140 CV=1/(N-1)*((XY)-((1/N)*(SX*SY)))
1150 CC=CV/(XD*YD):RETURN
1160 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 ";
1170 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).";
1180 PRINT"  STANDARD DEVIATIONS (IN THE CORRELATION FORMULAS) A
RE CALCULATEDUSING THE N-1 METHOD."
```

```
1190 PRINT:INPUT"PRESS ENTER TO CONTINUE";A:CLS
1200 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.  ";
1210 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)."
1220 INPUT"PRESS ENTER TO CONTINUE";A:RETURN
1230 FORX=1TO2000:NEXT:RETURN
```

Accounts Receivable II

HEBBLER SOFTWARE SERVICES

A comprehensive accounts receivable program with billing package offering menu oriented operation, audit trail with running balance for each account, date, description and exact amount for every filed transaction, special input procedures, automatic error checks — uses random data files.

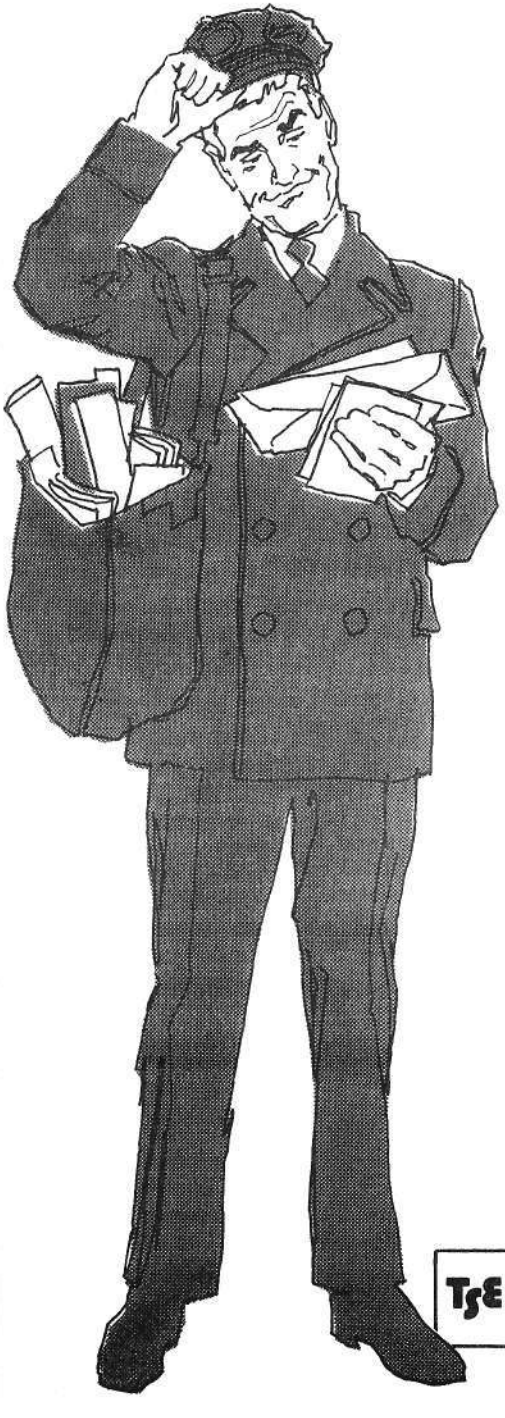
The package which allows you to:

- Maintain receivables files on 200 accounts
- Add new accounts any time
- Change information
- Perform selective information search
- Assign terms
- Print listing of overdue accounts
- Print statements automatically for unpaid accounts
- Print a custom message on statements
- Print mailing labels
- Print an accounts receivable summary for all accounts or unpaid accounts only
- Post charges and credits at the keyboard

Package includes one master diskette, one data diskette, and in depth instruction manual. Requires TRS-80 with 16K memory, two disk drives, and line printer. **$79.95**

TSE TRS-80 Software Exchange

17 Briar Cliff Drive Milford, New Hampshire 03055



MAIL LIST II

by BUS-80

IDEAL for all sorts of small mailing applications, such as small businesses, clubs, churches; for advertising, newsletters, announcements, press releases -- endless possibilities. We use it for a 15,000-name mailing list, yet it is perfect for lists as short as 100 names! You can store 1000 records per data disk, use as many disks as you like . . .

Each record includes:

RECORD NUMBER
RECORD CODE
COMPANY NAME
NAME
ADDRESS
CITY/STATE/ZIP
PHONE NO.
GREETING
PRODUCT CODES
DATE

Utilities include SORT, MERGE, MOVE, BREAK, EXAMINE, and UPDATE.

Prints lavels 1, 2, or 3 across.

Sequential file structure makes the most efficient use of disk space: all alphabetic items can be as long as necessary.

Allows data entry on a 4K, Level II cassette system.

2 Disk Drive, 32K minimum $99.95

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

TALKING BANKO

by Lance Micklus

The first time you show off your TRS-80 Voice Synthesizer, there are two things your friends will be sure to say:

"Why does it have a German accent?"

"That's neat, but what are you going to do with it?"

I haven't any idea why it sounds like an old German college professor, and when I first got mine, I had no idea what I was going to do with it either.

The instruction book provided is well-written, but I wish Radio Shack had spent more time explaining how to get the thing to say stuff that isn't in the book.

Teaching a computer to speak is an art. It isn't hard, but it requires practice. I quickly discovered that four-letter words (unprintable here!) are simple to program. Getting the thing to say my name took me about half an hour. It turns out that **Lance** is L @ I#CNS. And my girlfriend's name (**Dianne** with two N's) is D2AY @ (N. **WOW!!!**

If all this makes no sense to you, maybe I should explain how to program a voice synthesizer. The unit is capable of producing sixty-two **phonemes**. Supposedly, using these sixty-two different sounds, you can simulate any word in the English language.

The unit plugs into the expansion card on the rear of the TRS-80, or the screen printer connection on the expansion box. It constantly watches for any letter put on the video screen in the last thirty-two positions. (This is called the window). A PRINT @ 992, for example, will put a character or line of characters in the window. The characters can also be POKEd into the window. This means that the unit will work with either Level I or Level II. I feel this is senseless, since not many Level I machines are going to have a $400 voice box on them. It should have been set up only for Level II to read a port like a printer. This window concept has problems, as we shall see ...

The voice synthesizer will ignore anything on the screen, so video data will not set it off. When a question mark character appears in the window, it is **open**. The device select light turns on. Now, anything printed inside the window goes to the voice box and it pronounces the proper phoneme. After you print all the phonemes into the window, you print another question mark character within the window to **close** it.

The voice box itself can store up to thirty-two phonemes on a first in, first out basis. If you try to put more phonemes in than there is room for in the stack the box gets confused.

Here's the trouble with this set-up: if you have the unit on and LIST a BASIC program, a question mark is bound to run through the window, causing the voice box to try to pronounce the last half of every line it sees. It speaks lousy BASIC.

But the worst part is that there's no way to know when the stack is full, or how fast to put new phonemes into the stack, so by trial and error you set up timing loops.

If Radio Shack had gone to the port method, all these problems would be non-existant. You'd be able to list to your heart's content. And, as with the printer port, you could have tested the port for a busy condition to avoid overloading the stack. No more timing loops. But it's too late to change that now, so we'll make it work as it is.

APPLICATIONS

While visiting my computer store several weeks ago, I mentioned I had a Voice Synthesizer. I was told the store had recently sold one for the S-100 bus. (I think it was the CT-1 model you've no doubt seen advertised in magazines.) Being curious, I asked what type of computer it was to be used on. Would you believe it ... an IBM 370! Yep, with an S-100 bus. Not an IBM S-100 bus — IBM hasn't been able to design such a thing yet.

What in RAM's name would you want to do with a voice synthesizer on an IBM 370? Among the many jobs 370's have is that of building security. They want it to do more than buzz when there's a fire ... they want the computer to **tell** a security guard that there is a fire and **where the fire is**, in English!

In computer games a voice box can add a whole new dimension. I'm now toying with a new Concentration game that speaks. Educational programs can be even better with verbal reinforcement, also. But, of all the TRS-80 users, it's the business people who need talking computers the most. Watch someone entering data through the keyboard. Now think what it would be like if the computer could give the typist verbal reinforcement ... especially if the eyes must leave the screen momentarily.

I plan to add the voice feature to my own Deluxe Personal Finance program so that when I'm cancelling out checks from the bank I can look at the checkbook while the computer reads the check out to me.

THE GAME OF BANKO

BANKO was my first computer game. First written on the Casio programmable calculator and called LUCKY ELEVEN, it was later rewritten for the SIGMA/6 and was played extensively by University of Vermont students. From there, it was renamed DOUBLE OR NOTHING, then once again rewritten in Level I BASIC (later upgraded to Level II BASIC) and renamed to BANKO.

I have now brought it back out again, this time adding voice capabilities to it.

BANKO is similar to Blackjack. In fact, I was originally trying to write Blackjack way back on the Casio. Briefly, you get a random number which is added to the total in your hand. You keep getting numbers until you quit or break eleven.

BANKO is an easy game to learn. It's fun to play and can be played by a wide range of people, from small children to adults. My son was six when he first started playing BANKO. Not only did he enjoy it, it was great practice for performing simple arithmetic drills in his head without the boring, "HOW MUCH IS 3 + 2" type of question. So, the game can be educational ... if you're six years old or so.

THE PROGRAM

Sometimes it can be dangerous to fool around with other people's programs. BANKO is one of them. You'll notice it contains its own random number generator. This is more than a leftover from the old Casio calculator ... it's an important part of the game. The random number generator only needs to be seeded, which is done at the start. Beyond that, Level II's RND function isn't used because it produces uniformly distributed random numbers — which means you have just as much of a chance of getting a 9 as you do a 2. BANKO's random number generator is biased to the low end. It will produce random fumbers from 1 to 11, but it's more likely to give you a small number than a large one. In other words, 11's occur less often than 2's.

To make the computer play a good game, you must have good strategy, right? WRONG! BANKO does something that is very funny. It mimics you — it plays the same strategy you play. In theory, then, it should play as well as you, and win half the time, right? WRONG! BANKO's dealer always goes last, so if you bust, it can take advantage of your mistakes and try to pull ahead. It seems it should play a little better than you. At least it should win more often then you do. However, in practice, it seems to work out the other way. Your odds of winning are about sixty percent.

BANKO simply keeps a sort of average of when you stopped in the variable Q. Q is modified so that it won't fall less than 6, or more than 10. The computer's logic is also set up so that if drew four times in this hand, and the computer's hand is less than seven, it will always draw again,

regardless of Q. This is because the odds of drawing a high number and losing are smaller than the odds of drawing a low number and getting bonus points.

The voice routines are at the end of the game. BANKO is set up so it can be played either with or without the voice box. The voice routines have much more power than is needed to play BANKO, but the game makes a good frame with which to demonstrate the routine.

The idea here is to build a string with the phonemes needed to pronounce a number or a phrase containing a number. After VO$ is developed, just do a GOSUB 40080. **NOTE:** You need to set SS=-1 when you start up the program. The subroutine at 40080 will POKE the phonemes into the video memory, as well as taking care of opening and closing the window. The speed at which the phonemes are POKE'd is controlled by setting VW. The larger VW is, the slower the phonemes get POKE'd.

The second part of the voice routines creates the phonemes needed to pronounce the number. The phonemes are **added** to VO$. The main line must clear VO$ as needed. Just set VO equal to the number whose phonemes you wish the computer to pronounce. There are three calls you can make depending on how the number is to be pronounced:

GOSUB 42080 will pronounce the number VO as if it were dollars and cents. In other words, if VO=2.58, the computer will add to VO$ the phonemes to say **"Two dollars and fifty-eight cents"**

GOSUB 42320 will make the computer pronounce the numbers one digit at a time. If VO=352, then the computer will add phonemes to VO$ to say, **"Three five two"**

The third call, which is used by BANKO, pronounces the number simply as a number. If VO=34, it will add to VO$ the phonemes needed to make it say **"Thirty-four"**

It should be immediately apparent how powerful these voice routines are. They can easily be transplanted into your own programs to make the programs talk.

In the case of my checking account program, I would want to have the computer give me the check number one digit at a time, but I'd like the value in dollars and cents. An if it did only that, and said nothing more, it would greatly improve the usefulness of the program.

The Radio Shack Voice Synthesizer retails for about $400, which is no small sum. Personally, I think that in time, devices like it will become commonplace on computers — nearly as common as printers. After all, they are output devices.

```
10 SS=-1 : VO$="H;PP.0K;(MP#(TT @  * +   L@ I#(NS0MI)KL%S"
20 VW=10 : GOSUB 40080
30 END
```

```
100 REM  ================================================
120 REM  BANKO-T                  VERSION 2.0
140 REM  BY LANCE MICKLUS         WINOOSKI, VERMONT 05404
160 REM  TRS-80                   16K LEVEL II BASIC
180 REM  COPYRIGHT                MARCH 1979
200 REM  *     *     *     *     *     *     *     *     *
220 REM  FOR OPTIONAL USE WITH
240 REM         THE RADIO SHACK VOICE SYNTHESIZER
260 REM  ================================================
280 REM
300 CLEAR 500 : RANDOM
320 CLS : PRINT@320,"" : INPUT"ARE YOU USING THE TRS-80 VOICE SY
NTHESIZER (Y OR N)";A$ : A$=LEFT$(A$,1)
340 IF A$="Y" THEN SS=-1 ELSE 400
360 INPUT"IS THE DEVICE SELECT LAMP OFF (Y OR N)";A$ : A$=LEFT$(
A$,1)
380 IF A$="N" LET VO$="?" : GOSUB 40000 : FOR K=0 TO 2000 : NEXT
 K
400 CLS
420 E = RND(10000)
440 Z = 30
460 T = 0
480 W = 0
500 Q = 7
520 VW=12
540 PRINTTAB(23)"WELCOME TO BANKO"
560 PRINT TAB(23)"BY LANCE MICKLUS"
580 PRINT TAB(16)"-------------------------"
600 PRINT
620 PRINT"THE GAME IS SIMILAR TO BLACK JACK.  YOU DRAW NUMBERS ST
OPPING"
640 PRINT"BEFORE THE TOTAL POINT VALUE EXCEEDS 11.  IF YOU GO OVE
R 11,"
660 PRINT"THEN YOUR HAND WILL BE ZERO'D.  IF YOU DRAW 5 TIMES WIT
HOUT"
680 PRINT"GOING OVER 11, THEN YOU'LL RECEIVE BONUS POINTS EQUAL
TO THE"
700 PRINT"VALUE OF YOUR HAND AT THE TIME YOU END YOUR TURN."
720 PRINT
740 PRINT"THE SCORE STARTS AT 30 POINTS.  IF IT GOES TO 60, YOU W
IN."
```

```
760 PRINT"BELOW ZERO, YOU LOSE. AT THE END OF EACH TURN, THE POI
NT"
780 PRINT"DIFFERENCE BETWEEN THE TWO HANDS WILL BE ADDED OR SUBT
RACTED"
800 PRINT"FROM THE SCORE."
820 PRINT
840 VO$="W)%LKAM0T#(U0B3#NK$  8@Y@@M@=E@D L3R  PL #SR #D@=E@INST
RAK550NS  PR#@50=E03NT3R@KEY@W##N@YOR@R@D&@T#(UPL@Y"
860 GOSUB 40080
880 PRINT"WISH YOURSELF GOOD LUCK, THEN HIT (ENTER) WHEN READY T
O PLAY."; : B$=INKEY$
900 IF INKEY$="" THEN 900
920 REM
940 REM     THE HUMAN TAKES A TURN
960 REM   ------------------------
980 T = T + 1
1000 D = 0
1020 A(0) = 0 : A(2) = 0
1040 CLS
1060 PRINT@967,"PRESS Q TO QUIT THE GAME, D TO DRAW AGAIN, S TO
STAY";
1080 VW=0 : VO$="*T50Y$R0T3RN" : GOSUB 40080
1100 PRINT@ 25,"USER'S TURN"
1120 PRINT@ 148,"DRAW"; : PRINT@ 164,"HAND"
1140 GOSUB 3620
1160 D = D + 1
1180 A(0) = A(0) + N
1200 PRINT TAB(20) N;
1220 PRINT TAB(36) A(0); : IF NOT SS THEN 1340
1240 VO$=". U0DR#(U " : VO=N : GOSUB 42500
1260 IF N<>A(0) LET VO$=VO$+"  YOR0H@ND0. S0NP#0" : VO=A(0) : GOS
UB 42500
1280 GOSUB 40080
1300 IF N<>A(0) FOR K=0 TO 300 : NEXT K
1320 IF D=5 AND A(0)<12 LET VO$="Y#(U0G2T0<=E0B$N@S0PO*NTS" : VW
=10 : GOSUB 40080
1340 IF A(0) < 12 THEN 1420
1360 A(0) = 0 : A(2) = 0
1380 GOSUB 4220 : GOSUB 3940
1400 GOTO 1640
1420 PRINT TAB(50)" ";
```

```
1440 IF D > 4 THEN A(2) = A(0)
1460 O$=INKEY$ : K=0
1480 O$=INKEY$ : IF O$="" AND K=200 THEN K=0 : GOSUB 4340 ELSE K
=K+1
1500 IF O$<>"Q" AND O$<>"D" AND O$<>"S" THEN 1480 ELSE PRINT
1520 IF O$="S" VO$="YOR05T0YIN+" : GOSUB 40000 : GOTO 1640
1540 IF O$="Q" THEN 3000
1560 GOTO 1140
1580 REM
1600 REM     DEALER'S TURN
1620 REM   ----------------
1640 CLS
1660 A(1) = 0 : A(3) = 0
1680 D = 0
1700 Q = (Q + A(0)) / 2
1720 IF Q < 6 THEN Q = 6
1740 IF Q > 10 THEN Q = 10
1760 IF SS LET VW=0 : VO$="*TS0M30V0TJRN" : GOSUB 40000 : FOR K=
0 TO 300 : NEXT K
1780 PRINT@ 24,"DEALER'S TURN"
1800 PRINT@ 148,"DRAW"; : PRINT@ 164,"HAND"
1820 GOSUB 3620
1840 A(1) = A(1) + N
1860 D = D + 1
1880 PRINT TAB(20) N;
1900 PRINT TAB(36) A(1) : IF NOT SS THEN 2020
1920 VO$="80Y0DR%(U0" : VO=N : GOSUB 42500
1940 IF N<>A(1) LET VO$=VO$+"  M30Y0HEND0.S0N%W0" : VO=A(1) : GO
SUB 42500
1960 GOSUB 40000
1980 IF N<>A(1) FOR K=0 TO 400 : NEXT K
2000 IF D=5 AND A(1)<12 LET VO$="80Y0G2T0(=E0B$N#S0P0+NTS" : VW=
10 : GOSUB 40000
2020 IF A(1) < 12 THEN 2100
2040 GOSUB 4240 : GOSUB 3940
2060 A(1) = 0
2080 GOTO 2320
2100 IF D = 5 THEN A(3) = A(1) : GOTO 2200
2120 IF (D = 4) * (A(1) < 7) THEN 1820
2140 IF Q <= A(1) THEN 2200
2160 IF A(1) - A(0) - A(2) > 2 THEN 2200
```

```
2180 GOTO 1820
2200 PRINT@ 921,"DEALER STAYS"
2220 VW=0 : VO$="=E0D.L3R0ST@YS" : GOSUB 40000
2240 IF SS THEN FOR K=0 TO 500 : NEXT K ELSE FOR K=0 TO 1000 : N
EXT K
2260 REM
2280 REM     END OF TURNS SEQUENCE
2300 REM   ------------------------
2320 CLS
2340 PRINT@ 345,"SUMMARY #";T
2360 PRINT@ 461,"USER'S HAND";A(0);
2380 PRINT@ 481,"DEALER'S HAND";A(1)
2400 PRINT@ 524,"BONUS POINTS";A(2);
2420 PRINT@ 546,"BONUS POINTS";A(3)
2440 Z = A(0) + A(2) - A(1) - A(3) + Z
2460 PRINT@ 666,"SCORE:";Z : IF NOT SS THEN 2640
2480 VO$="AFT3R0" : VO=T : GOSUB 42500 : VO$=VO$+" R5E ND"
2500 IF T<>1 LET VO$=VO$+"S"
2520 VO$=VO$+" <=E0SKOOR0.S0" : VO=Z : GOSUB 42500
2540 VW=10 : GOSUB 40000 : VO$="" : FOR K=0 TO 300 : NEXT K
2560 IF Z>30 LET VO$=VO$+"  Y$R03H@@D"
2580 IF Z<30 LET VO$=VO$+"  80YM03H@@D"
2600 IF Z=30 LET VO$=VO$+"  WEAR0T8**D"
2620 VW=20 : GOSUB 40000
2640 IF (Z >= 20) * (Z <= 40) THEN W = 0 : GOTO 2960
2660 IF Z <= 0 THEN 3200
2680 IF Z >= 60 THEN 3340
2700 IF (W = 0) * (Z < 10) THEN W = 1 : GOTO 2760
2720 IF (W = 0) * (Z > 50) THEN W = 1 : GOTO 2840
2740 GOTO 2960
2760 PRINT@ 906,"CAREFUL TURKEY, OR YOU'RE GONNA LOSE !!!"
2780 VO$="K@RF3L0T3RKEY  OR0Y$R0G$N20L('S" : VW=15 : GOSUB 40000
2800 IF SS THEN FOR K=0 TO 750 : NEXT K ELSE FOR K=0 TO 1250 : N
EXT K
2820 GOTO 2960
2840 VO$="DONT0SM3IL0Y3T  Y@(U0STIL0N D0" : VO=60-Z : GOSUB 4258
0
2860 IF Z=59 THEN VO$=VO$+"PO*NT" : B$="POINT" ELSE VO$=VO$+"PO*
NTS" : B$="POINTS"
2880 VO$=VO$+"0T@(U0WIN"
```

```
2900 PRINT@900,"DON'T SMILE YET. YOU STILL NEED";60-Z;B$;" TO WI
N!!!"
2920 VW=15 : GOSUB 40000
2940 GOTO 2800
2960 IF SS THEN FOR K=0 TO 750 : NEXT K ELSE FOR K=0 TO 1500 : N
EXT K
2980 GOTO 980
3000 REM
3020 REM     END OF THE GAME.
3040 REM   ------------------
3060 REM
3080 CLS
3100 PRINT@ 320,"GAME ENDED AFTER";T-1;"TURNS BY SORE LOSER."
3120 PRINT"DEALER WINS BY DEFAULT."
3140 VO$="  Y%(U0QU*T  =E0D#*L3R0WINS083*0DEF;LT" : VW=10 : GOSU
B 40000
3160 GOTO 3300
3180 REM * DEALER WINS *
3200 PRINT
3220 PRINT"YOU TURKEY!!!  YOU LET A STUPID COMPUTER LIKE ME BEAT
 YOU."
3240 PRINT"BETTER LUCK NEXT TIME."
3260 VO$="  Y%(U0T3RKEY" : VW=20 : GOSUB 40000
3280 VO$=" Y%(U0L@T00Y0STUPAD0K;MP.UT3R LA*K0M. B.T0Y%(U   B38T3
R0LAAK0NA5KS0T2AYM" : VW=15 : GOSUB 40000
3300 GOTO 3420
3320 REM * USER WINS *
3340 PRINT
3360 PRINT"CONGRATULATIONS.  YOU WON THE GAME."
3380 PRINT"HOPE YOU ENJOYED YOURSELF.";
3400 VO$="  K$NGR0TY%(UL80YS58N5  Y%(U0WAN0<=E0G0M  H$P0Y(U0INJ0
Y00YORS4LF" : VW=15 : GOSUB 40000
3420 IF SS FOR K=0 TO 500 : NEXT K
3440 VO$="  D#(U0Y%(U0WANT0T%(U0PL0Y00G0N  PR0S W;5##0OR0INN" :
VW=10 : GOSUB 40000
3460 PRINT@ 0,"DO YOU WANT TO PLAY AGAIN (Y OR N)"; : A$=INKEY$
3480 A$=INKEY$ : IF A$="" THEN 3480
3500 IF A$<>"N" Z=30 : T=0 : W=0 : GOTO 900
3520 REM  END OF PROGRAM
3540 END
3560 REM
```

```
3580 REM     NUMBER GENERATOR
3600 REM   --------------------
3620 IF E > 11.9999 THEN E = E / 11.9999 : GOTO 3620
3640 N = INT(E)
3660 X = E
3680 L = INT(1.4427 * X) + 1
3700 E = 0.693147 * L - X
3720 A = 1.32988E-3 - 1.41316E-4 * E
3740 E = (((A - 0.166665) * E - 1) * E + 1)
3760 A = 2
3780 IF L <= 0 THEN A = 0.5 : L = -L : IF L = 0 THEN RETURN
3800 FOR X = 1 TO L
3820 E = A * E
3840 NEXT X
3860 RETURN
3880 REM
3900 REM     FLASH OVER ON THE SCREEN
3920 REM   --------------------------
3940 FOR J = 1 TO 4
3960 PRINT@ 88,">>> OVER <<<";
3980 FOR K = 0 TO 100 : NEXT K
4000 PRINT@ 88,CHR$(30);
4020 FOR K = 0 TO 100 : NEXT K
4040 NEXT J
4060 RETURN
4080 '
4100 REM LINK TO GAMES MONITOR
4120 END
4140 REM
4160 REM  OVER VOICE MESSAGE
4180 REM  ------------------------
4200 REM
4220 VO$="Y%(U0BAST" : VW=0 : GOTO 40080
4240 VO$="AYY0BAST" : VW=0 : GOTO 40080
4260 REM
4280 REM  HURRY UP
4300 REM  --------
4320 REM
4340 VW=15 : ON RND(3) GOTO 4360, 4380, 4400
4360 VO$="H2ARY80P  2AYM0W0YTIN+" : GOTO 40080
4380 VO$="80YM0L('SIN+0M80Y0P0YS58NS  L33T50GOWW" : GOTO 40080
```

```
4400 VO$="80Y8H8VHNT0G2T62LL6D8Y  T8K8YOR8T3RN" : GOTO 40000
40000 REM
40020 REM  LOAD SYNTHESIZER WITH VO$ - VW IS LOADING SPEED
40040 REM  ------------------------------------------------
40060 REM
40080 IF S5 THEN VS%=PEEK(16383) : POKE 16383,63 : POKE 16383,32
 ELSE RETURN
40100 FOR VX=1 TO LEN(VO$)
40120 POKE 16383,ASC(MID$(VO$,VX,1))
40140 FOR VC=1 TO VW : NEXT VC
40160 NEXT VX
40180 POKE 16383,32 : POKE 16383,63 : POKE 16383,32
40200 POKE 16383,63 : POKE 16383,ASC("-") : POKE 16383,63 : POKE
 16383,VS%
40220 RETURN
42000 REM
42020 REM  VO$ SET TO PRONOUNCE VO NUMBER IN DOLLARS & CENTS
42040 REM  ------------------------------------------------
42060 REM
42080          VZ(6)=SGN(VO) : VZ(5)=ABS(VO)+.005 : VO=FIX(VO) :
 IF VO=0 LET VO=VZ(6)*INT((VZ(5)-INT(VZ(5)))*100) : GOTO 42180
42100 GOSUB 42580 : IF VE RETURN
42120 VO$=VO$+"80;L3" : IF INT(VZ(5))=1 THEN VO$=VO$+" " ELSE VO
$=VO$+"S "
42140 VO=INT((VZ(5)-INT(VZ(5)))*100) : IF INT(VO)=0 RETURN
42160 VO$=VO$+"8ND8"
42180 VZ(5)=VO : GOSUB 42580 : VO$=VO$+"85ENT" : IF ABS(INT(VZ(5
)))<>1 LET VO$=VO$+"S"
42200 RETURN
42220 REM
42240 REM  MAKE VO$ THE PRONOUNCIATION OF EACH NUMBER
42260 REM  ------------------------------------------
42280 REM
42300          VZ$=STR$(VO)
42320 FOR VP=1 TO LEN(VZ$)
42340 VP$=MID$(VZ$,VP,1) : VO$=VO$+" "
42360 IF VP$="." LET VO$=VO$+"PO+NT" : GOTO 42480
42380 IF VP$="E" LET VO$=VO$+"*.&8T(U8<=E" : GOTO 42480
42400 IF VP$="+" LET VO$=VO$+"PLAS" : GOTO 42480
42420 IF VP$="-" LET VO$=VO$+"MI8+NAS" : GOTO 42480
42440 IF VP$=" " THEN 42480
```

```
42460 VO=VAL(VP$) : GOSUB 42580
42480 NEXT VP : RETURN
42500 REM
42520 REM  ADD NUMBER PRONOUNCIATION TO VO$
42540 REM  ------------------------------
42560 REM
42580 IF NOT VQ GOSUB 43220
42600 IF VO<0 LET VO=ABS(VO) : VO$=VO$+"MB*NRS"
42620 REM  TRILLION
42640 IF VO>999E12 THEN VO$="=E0NAMB3R0ES0TUUB.G0FORME0TUU0PRON2
NS" : VE=-1 : GOTO 42980 ELSE VE=0
42660 IF VO<1E12 THEN 42740
42680 VZ(4)=VO : VO=(INT(VO/1E12)) : GOSUB 43020 : VO$=VO$+"0TRI
LE8N "
42700 VO=VZ(4)-(INT(VZ(4)/1E12)*1E12) : IF VO=0 THEN 42980
42720 REM  BILLIONS
42740 IF VO<1E9 THEN 42820
42760 VZ(3)=VO : VO=(INT(VO/1E9)) : GOSUB 43020 : VO$=VO$+"0BILE
8N "
42780 VO=VZ(3)-(INT(VZ(3)/1E9)*1E9) : IF VO=0 THEN 42980
42800 REM  MILLIONS
42820 IF VO<1000000 THEN 42900
42840 VZ(2)=VO : VO=INT(VO/1000000) : GOSUB 43020 : VO$=VO$+"0MI
LE8N "
42860 VO=VZ(2)-(INT(VZ(2)/1000000)*1000000) : IF VO=0 THEN 42980
42880 REM  THOUSANDS
42900 IF VO<1000 THEN 42960
42920 VZ(1)=VO : VO=INT(VO/1000) : GOSUB 43020 : VO$=VO$+"0=AU50
ND "
42940 VO=VZ(1)-(INT(VZ(1)/1000)*1000) : IF VO=0 THEN 42980
42960 GOSUB 43020
42980 RETURN
43000 REM  HUNDREDS ROUTINE
43020 IF VO<100 THEN 43140
43040 VZ(0)=VO : VO=INT(VO/100) : GOSUB 43140 : VO$=VO$+"0HUNDR9
D"
43060 IF INT(VZ(0)/100)*100<>VZ(0) LET VO=VZ(0)-(INT(VZ(0)/100)*
100) : VO$=VO$+"0AND0" : GOSUB 43140
43080 VO=VZ(0) : RETURN
43100 REM  LESS THAN ONE HUNDRED ROUTINE
43120 REM  IF LESS THAN 20 DIRECT OUTPUT
```

```
43140 IF VO<20 LET VO$=VO$+VX$(VO) : RETURN
43160 VO$=VO$+VX$((INT(VO/10)-2)+20)
43180 IF INT(VO/10)=VO/10 RETURN
43200 VO$=VO$+"0"+VX$(VO-(INT(VO/10)*10)) : RETURN
43220 DIM VX$(27) : VO=-1
43240 REM  ZERO TO TWENTY
43260 VX$(0)="ZSERO" : VX$(1)="N2(N" : VX$(2)="T#(U"
43280 VX$(3)="(=RE" : VX$(4)="FO2R" : VX$(5)="F2@V"
43300 VX$(6)="SIKS" : VX$(7)="S3VHN" : VX$(8)="@#T"
43320 VX$(9)="N8*N" : VX$(10)="T3N" : VX$(11)="EL3VHN"
43340 VX$(12)="TWRLV" : VX$(13)="(=IRT.N" : VX$(14)="FORT.N"
43360 VX$(15)="FIVT.N" : VX$(16)="SIKST.N" : VX$(17)="S3VHNT.N"
43380 VX$(18)=".T.N" : VX$(19)="N8&T.N" : VX$(20)="TW3NT#Y"
43400 REM  THIRTY TO NINTY
43420 VX$(21)="(=IRT#Y" : VX$(22)="FORT#Y" : VX$(23)="FIFT#Y"
43440 VX$(24)="SIKST#Y" : VX$(25)="S3VHNT#Y" : VX$(26)="@#T#Y"
43460 VX$(27)="N8*T#Y"
43480 RETURN
```

THE HARD SIDE OF SOFTSIDE ?

NEW! TRS-80 Computers 10% off!

Effective July 1, Radio Shack dropped the price of TRS-80 equipment. On the same date, we received authorization to sell the complete TRS-80 line, at 10% below the new low Radio Shack price. This all new equipment, with Radio Shack warranty and service, is now made more affordable than ever!

If your're looking to save even more, consider Hardside's used equipment department for prime condition, previously owned TRS-80 equipment.

Do you have something to sell? Hardside will buy any used Radio Shack TRS-80 equipment in good condition. Refer to our price list below.

So . . if you're looking to buy or sell TRS-80 equipment, look to us first!

	NEW		USED	
UNIT	NEW R/S LIST PRICE	HARDSIDE DISCOUNT PRICE	WE'LL PAY	SELL
Level I 4K	$499	$449	$275	$350
Level II 4K	$619	$557	$350	$425
Level I 16K	$729	$656	$350	$450
Level II 16K	$849	$764	$500	$650
Level II 16K no keypad			$450	$600
Expansion Interface				
.....OK	$299	$269	$175	$225
.....16K	$448	$403	$225	$300
.....32K	$597	$537	$275	$375
Disk Drives 0	$499	$449	$300	$399
1	$499	$449	$275	$375
(Percom disk drives -		$399)	-	-
Printers Friction feed*	$1299	$1169	$650	$800
Tractor Feed*	$1559	$1403	$750	$900
Line Printer II	$999	$899	New Item Not available	
Quick Printer I*	$499	$449	$250	$325
Quick Printer II	$219	$197	$125	$175
*(Requires cable	$40	$36)		
RS-232C	$99	$89	$50	$75
Telephone Int. I	Discontinued		$50	$75
Telephone Int II	$199	$179	$100	$150

HARDSIDE - Your market for new and used TRS-80 equipment

Either way, give us a call at 603-673-5144. Today!

The prices shown may fluctuate as the market shapes up. So, if you want to sell your TRS-80, we'll pay - CASH. And, if you're looking for the best selection in hardware for TRS-80, we have it!

PLOT
A MACHINE LANGUAGE VERSION OF SET [X,Y] FOR THE TRS 80

by James E. Randall

Machine language programmers may have use for an equivalent of the Level II SET(X,Y) command callable as a subroutine. This article describes such a subroutine which will turn on a graphics block at a given X-coordinate (0 to 127 from the left edge) and at a given Y-coordinate (0 to 47 from the top). The operation consists of setting up the HL register pair to point to the appropriate Video RAM address and using the Z-80 SET b, (HL) command to set the bit which corresponds to the location of the graphic cell in the 2 x 3 matrix which is to be turned on.

The 1,024 bytes of Video RAM starting at decimal 15,360 correspond to 64 "character positions" on each of 16 lines with the first byte mapping the upper left corner of the TRS-80 screen. Upon power-on, pressing the **CLEAR** key, or issuing a CLS command these memory locations will have their high-order bits cleared to 0, a code by which the graphics generator interprets the low-order bits as ASCII characters. When the high-order bit is set to 1 the low six bits indicate which graphics cells are turned on in a 2 x 3 matrix at the character location. (see the attached Figure 1.) This gives the 128 (hortizontal) by 48 (vertical) grid characteristic of the TRS-80. One of the subroutines (CLRS) given here clears the screen by setting all locations in Video RAM to 128; the other (PLOTXY) picks up the X and Y arguments, finds the appropriate memory location by dividing Y by 3 and X by 2, and then sets the bit within the RAM using their respective remainders.

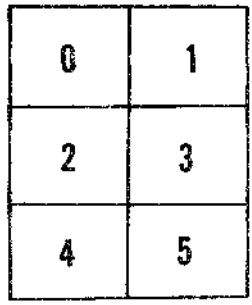
In this version of PLOTXY the X and Y coordinates have previously been loaded into locations XPOS and YPOS (alternatively, they could have been

in Z-80 registers when the subroutine was called.) The Y is divided by subtracting out 3 until the remainder is zero or negative, the quotient is used as a counter for indexing register HL by 64's starting at memory location 15,360. When this is done HL points to the RAM giving information about the character position at the left edge of the correct one of the 16 lines. The character location within that line is obtained by adding XPOS/2 into HL so that the latter now points to the RAM to be modified.

The remainder of YPOS/3, multiplied by 2 and temporarily saved in register C, is added into the remainder of XPOS/2 to indicate which of the six cells in the matrix is to be turned on. To correspond to the format of the SETb, (HL) command this number must be shifted left three places and then loaded into the second byte of the operation code. If the one wished to RESET(X,Y) the same sequence would be involved except that the op code would have to correspond to the Z-80's RES b,(HL) instead.

Testing of this subroutine can be done with Level II by calling CLRS using the USR(0) command, setting XPOS and YPOS with POKE commands, and then calling PLOTXY at its address in higher memory. Note that Level II READY may overwrite many Video RAM locations where the subroutine has placed graphic blocks.

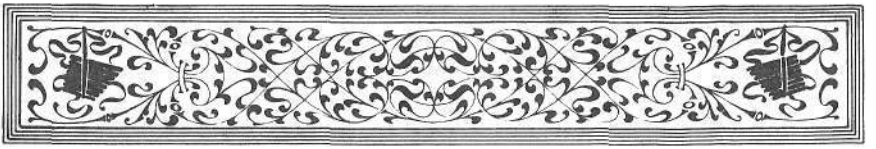
Figure 1: Each of the 16 lines has 64 character locations containing the 2 x 3 matrix as shown. The numbers correspond to the bit which is set in Video RAM for that graphic cell to be on. The high-order bit must also be on to have graphics rather than ASCII displayed.




```
                00100 ;PLOT ROUTINE
                00110 ;LIGHTS DOT AT (XPOS),(YPOS); ORIGIN=UP,LEFT
                00120 ;USES SET B,(HL)  WITH HL=RAM FOR CHAR. LOC.
                00130 ;                        B =CELL WITHIN 2X3 GRID
                00140 ;VIDEO RAM MUST BE PRESET TO 'GRAPHICS CLEAR'=128
                00150 ;  SINCE BASIC CLEARS WITH ASCII SPACE (32)
                00160 ;
61A8            00170         ORG     25000
                00180 ;
                00190 ;JUMP TABLE ENTRY; PARAMETERS
61A8 C3B061     00200         JP      PLOTXY  ;DOT AT X,Y
61AB C3EE61     00210         JP      CLRS    ;CLEAR SCREEN TO ASCII 128
```

```
61AE 00        00220 XPOS    DEFB    0       ;0-127 FROM LEFT
61AF 00        00230 YPOS    DEFB    0       ;0-47  FROM TOP
               00240 ;
               00250 ;DIVIDE YPOS BY 3 TO FIND VERTICAL
               00260 ;COORDINATE OF CHARACTER LOCATION
61B0 0600      00270 PLOTXY  LD      B,0     ;CLR COUNTER
61B2 3AAF61    00280         LD      A,(YPOS)
61B5 04        00290 PLT0    INC     B       ;QUOTIENT
61B6 D603      00300         SUB     3
61B8 B7        00310         OR      A       ;CHK ACCUM
61B9 2806      00320         JR      Z,PLT1  ;ZERO
61BB F2B561    00330         JP      P,PLT0  ;TILL NEGATIVE
61BE C603      00340         ADD     A,3     ;REMAINDER
61C0 05        00350         DEC     B       ;QUOTIENT
               00360 ;B=00-15 LINES DOWN; A=0,1,2 CELL IN LINE
61C1 CB27      00370 PLT1    SLA     A       ;ROTATE LEFT
               00380 ;A= 0, 2, 4
61C3 4F        00390 PLT3    LD      C,A     ;KEEP IN C
               00400 ;SET HL=MEMORY ADDRESS OF LEFT EDGE OF LINE
               00410 ;BY ADDING 64 * (B)
61C4 21003C    00420         LD      HL,15360 ;UPPER LEFT SCREEN
61C7 114000    00430         LD      DE,64   ;CHAR/LINE
61CA 04        00440         INC     B       ;B=01-16
61CB 05        00450 PLT4    DEC     B
61CC 2803      00460         JR      Z,PLT5
61CE 19        00470         ADD     HL,DE   ;INCREMENT BY 64
61CF 18FA      00480         JR      PLT4    ;AGAIN
               00490
               00500  NOW HANDLE X
61D1 3AAE61    00510 PLT5    LD      A,(XPOS)
61D4 CB3F      00520         SRL     A       ;DIV BY 2
61D6 1600      00530         LD      D,0     ;HIGH
61D8 5F        00540         LD      E,A     ;LO=00-63
61D9 19        00550         ADD     HL,DE   ;HL=MEMORY LOCATION
               00560 ;NOW SET BIT FOR CELL IN MATRIX
61DA 3AAE61    00570         LD      A,(XPOS) ;AGAIN
61DD E601      00580         AND     1       ;LEFT OR RIGHT HALF
61DF 81        00590         ADD     A,C     ;BIT NO. TO SET
61E0 CB27      00600         SLA     A       ;MOVE FOR OP CODE
61E2 CB27      00610         SLA     A       ;11 BBB 110
```

```
61E4 CB27     00620          SLA     A
61E6 F6C6     00630          OR      3060
61E8 32EC61   00640          LD      (OP+1),A
61EB CBC6     00650 OP       SET     0,(HL)  ;SETS BIT IN RAM
61ED C9       00660          RET             ;BACK
              00670 ;
              00680 ;
              00690 ;
              00700 ;
              00710 ;CLEARS OUT VIDEO RAM WHICH HAS 32=SPACE
              00720 ; BY PUTTING IN "GRAPHICS SPACE" =128
              00730 ;
61EE 21003C   00740 CLRS     LD      HL,15360 ;UPPER LEFT SCREEN
61F1 3680     00750          LD      (HL),128 ;GRAPHICS BLANK
61F3 11013C   00760          LD      DE,15361 ;DESTINATION ADDR.
61F6 01FF03   00770          LD      BC,1023  ;COUNTER
61F9 EDB0     00780          LDIR            ;FROM (HL) TO (DE)
61FB C9       00790          RET             ;BACK
              00800 ;
              00810 ;
0000          00820          END
00000 TOTAL ERRORS
CLRS   61EE 00740   00210
OP     61EB 00650   00640
PLOTXY 61B0 00270   00200
PLT0   61B5 00290   00330
PLT1   61C1 00370   00320
PLT3   61C3 00390
PLT4   61CB 00450   00480
PLT5   61D1 00510   00460
XPOS   61AE 00220   00510 00570
YPOS   61AF 00230   00280
```



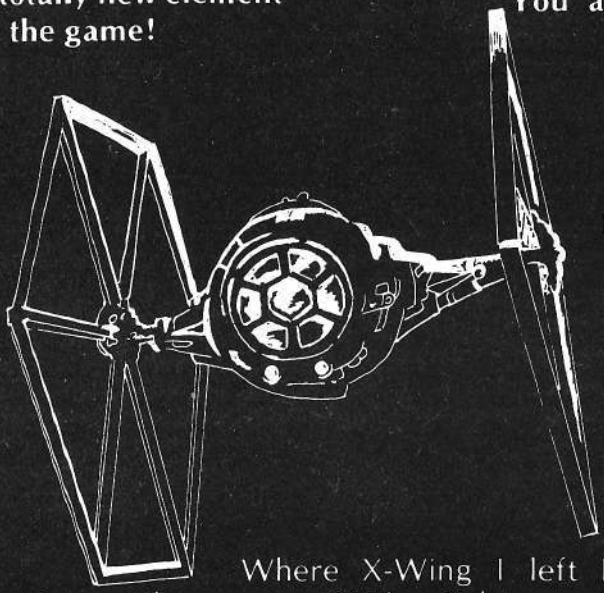
X-WING II

by Chris Freund

For the thousands who have enjoyed X-Wing Fighter, X-Wing II presents a totally new element in the game!

You are Pilot of an X-Wing fighter ...

Your Mission, Destroy the Death Star!



Where X-Wing I left Death Star looming on the screen, **X-Wing II** lets you guide your fighter into the trench, find the exhaust port, aim and fire — all the while avoiding enemy fighters. Excellent graphics, 12 levels of play, and extensive INKEY$ commands make this one of our most exciting "real-time" games.

Level II, 16K — $9.95

TSE TRS-80 Software Exchange

17 BRIAR CLIFF DRIVE MILFORD, NEW HAMPSHIRE 03055

CASSETTE CONTROLLER

This month's construction project should appeal to tape pullers of all ages - a control box permitting visual and audio monitoring of CLOADing cassettes, as well as manual start/stop control of the tape transport.

by John D. Eaton

No question that Radio Shack has a system that is fast becoming the standard of the personal computing industry. Despite its success, I must say that I am positively puzzled at two obvious and aggravating nuisances created by the Shack engineers.

It didn't take very long after I had my TRS-80 Level I uncrated, plugged in and running to discover the "plugging" problem with the REM jack on the cassette recorder. It took me about as long to discover the volume level problem when I had received my CPU back from the Level II upgrade.

Frankly, I had felt that the REM jack plugging problem was bad enough but, after attempting to CLOAD and convert Level I tapes to Level II, I had seriously considered returning to my neighborhood Radio shack and asking that my Level I be reinstalled.

After ordering the Level II upgrade I had read an article written by George Blank, in which he described converting Level I tapes to Level II through the use of an AM radio to monitor volume level. He did his usual outstanding job, and the method he describes works. However, the problem of volume control is not only present during conversion, but ever-present.

I really didn't want to make an AM radio a component of my personal computer; after all, the little bugger was about to make doubting Thomas's out of most of my friends (plug and unplug - CLOAD and RECLOAD - wind - rewind - over and over.) You know the friends who always ask, "What will it do?" - and you really want to show them something.

Anyway, back to the care and feeding of our TRS-80 . . . after more research than I care to admit, and just before I went back to get my Level I, I discovered that a VU meter could be used to read the volume level if it was connected in series, in line between the CPU and CTR 41. As I was putting this idea into production, I discovered that I could also solve the

REM jack problem just as easily. Moreover, neither addition would interfere with the TRS-80's operation nor the "shack's" warranty - (important - especially since my first Level II installation was defective). The complete unit fits compactly in a 2¾" x 5¼" box (see Figure 1.)

PROBLEM #1

My friends wonder why I have to keep running my tapes over and over . . . while I show them "the things it will do" . . .

FIX #1

Well, maybe fix is not the most appropriate term. Fix or not, I believe you will find this addition a helpful tool.

We are now told that the CTR-41 volume setting for Level I programs is 7 to 8 while for Level II the setting should be 4 to 5. (Slightly lower for the CTR-80 - ed.) After an evening of playing with the VU meter a reading of (16) appears to be an ideal volume setting to successfully CLOAD into Level II. The CTR-41 volume setting may vary from 3 to 9, but if my old VU meter is reading a (-6) I am CLOAD(ED).

I got all the parts needed for this project from my neighborhood shack. I used a bakelite box with aluminum cover (part #270-230), VU meter (part #22-053), shielded cable with miniature phone plug on one end and miniature phone jack (female) on the other (part #42-2472), and one bag of assorted vinyl grommets (part #64-3025). The 72" shielded cable is much longer than needed but some of the excess is used in the other projects.

Start the project by drilling the two mounting screw holes for the VU meter, next cut out a round hole to accept the body of the meter. Drill ¼" hole at each end of bakelite box, install 1 grommet in each hole. Next, cut cable ends off leaving about 12" of cable on each end (see Figure 3). Insert cut ends through grommets and pull enough cable through to enable you to strip the insulation off the cable. Using a sharp knife, cut a circle around the cable, taking care not to cut through shielding (copper wire woven around center lead); also make sure that the cut you make around the cable is continuous. Now pull outer insulation off, then strip down the copper shielding; pull around to one side and twist between fingers to make one (stranded) wire. Now you must strip the insulation off the center lead, again being careful not to cut too deeply into this lead. Pull insulation off, and twist lead to make firm wire (see Figure 3A).

You will need to take care to avoid the center lead and shielding making contact. Therefore, you will need to strip off more of the outer insulation than that which is stripped from the center lead (see Figure 3A).

After stripping both cable ends you are ready to make connections to the VU meter. First twist both shielded leads of the two cables together to make one lead, now connect this lead to the (-) negative

terminal of the VU meter; second, twist the center leads together similarly and connect them to the positive (+) terminal (Figure 2). This makes a series connection through the VU meter.

Operation:

Plug the miniature (male) plug from the CPU pigtail into the miniature (female) jack on the cable end coming from the VU meter, then connect the miniature (male) plug on the cable end coming off the VU meter into the "ear" jack on the CTR-41 recorder. You know have all connections complete (see Figure 1).

Load a cassette (with program on it) in the CTR-41 and begin to CLOAD as usual. Notice the VU meter when program data begins to be encountered. It abruptly moves from -20 toward the (+) range. If you move the volume control on the recorder notice that the reading of the VU meter is changed.

Move the volume from high to low and also observe the asterisks on the monitor.

By slowly moving the volume control and watching the asterisks sign for program being read, you will observe that the CPU reads information from the recorder at a VU meter reading of between -5 and -7, with -6 being ideal.

I have found that the volume setting per tape may differ greatly, I suppose, depending on quality of tape, source of recording, etc., so I have found that I cannot rely on a volume setting of 7-8 for Level I or 4-5 for Level II. As a matter of fact, my machine language Level II conversion program (from Radio Shack) CLOAD(S) at a VU meter reading of -6, when the volume control is set at 7.

Once you have adjusted the volume control so that the VU meter registers between -5 and -7 (there will be some movement in this range), rewind tape to start position and CLOAD.
Watch the video screen for the usual flashing asterisks. If the computer does not give signs of CLOAD(ING), adjust volume slightly (some programs are almost impossible to load). When CPU will accept the tape, it will be a good idea to write VU meter reading and volume setting on cassette. I have found these notes to be life savers for my commercial machine language programs.

Troubleshooting:

If your VU meter just sits there during CLOAD, you either have a blank tape, do not have the recorder in the play mode, have not properly connected the leads to the VU meter, or have a short in the connection, (shielding and center lead wires touching each other at meter or where the cables are stripped) (see Figure 2.)

If all of the checks show a dead meter, take it back to the Shack for exchange.

You will wonder how you ever survived before you had your VU meter.

PROBLEM #2

My friends question the credibility of my computer when I am seen fumbling with the REM plug - while trying to show them what "will it do" . . .

FIX #2

In Figure 2 you can see a Double Pole/Double Throw switch R.S. #275-666, in the corner of the box. Note that the cable leads from the CTR-41 REM are connected to the **center** set of terminals, the terminals on your **far right** are shunted (shorted together) and the third set on your **far left** of the switch are used to connect cable leads to the TRS-80 through the subminiature jack. I used 2 pieces of cable left over from Fix #1, a subminiature plug R.S. #274-289, subminiature phone jack R.S. #274-292. The male plug was installed on one end of one cable, and the female phone jack was installed into the side of the box housing. Care should be shown to assure that cable shielding is always connected to shielding and the center lead likewise. If the drawing is followed carefully, you should have no problems.

Start by drilling one hole in the top of the box to accept the DPDT switch, next drill one hole in each end of the box, one for plug (male) cable going to the REM on the CTR-41, and the other for a subminiature (female) phone jack. Cut a 12" piece of cable (left over from Fix #1); strip back both ends as described above, feed one end through the grommet which has been installed at the switch end of the box (Figure 2); connect the shielding lead to one side of the center poles on the switch, and the center lead to the other. Then take any small piece of wire, the center lead from the cable will do, and shunt the two terminals nearest the grommet (see Figure 2).

Now cut a 4" piece of cable and strip each end. Connect the shielding lead to the third set of terminals **on the same side** as the shielding from the recorder is connected; then connect the center lead to the other terminal. When connecting the subminiature jack, keep in mind that the tip of the plug represents the center lead, so make your connections so that when the plug is installed in the jack the top of the plug will complete the center lead circuit.

Finally, install the subminiature plug on the cable end coming from the center terminal of the switch - again, remember that the center lead connects to the top of the plug and the shielding lead acts as ground to the rest of the housing.

The soldering of the plug and jack are not easy for the novice, so you may need those extra parts and cable left from Fix #1.

Operation:

When the DP/DT switch is thrown in one direction the cassette recorder is connected to the CPU for auto control, when thrown in the opposite direction **you** have control of the recorder. You see

the REM plug does nothing more than open the circuit to the CTR-41 motor, now you can use it to your advantage. By shorting the two terminals on the switch as you have done, your switch acts as the REM was intended - a remote switch.

Troubleshooting:

If you have followed both the written and schematic directions, there should only be two possible sources of bugs: 1) bad solder connections; go back and check them, and 2) the female subminiature phone jack. Some female jacks have 3 terminals, use only the two that contact the plug when inserted. The third acts as a switch but is not used here.

Option:

For those of you who just like to add something to anything, every now and then, let's add a 3.2 ohm speaker (part #40-1201) to our VU meter circuit just in case we want to check on the sound quality on our tape - without unplugging the ear plug.

Install a second switch (DP/DT R.S. #275-666) in line with the VU meter (Figure 4). Connect the switch terminals as in the remote control circuit, except rather than shorting the terminals on the far right, match shield to shield, center lead to center lead, and connect both leads to a 3.2 ohm speaker.

Operation:

The DP/DT switch will put the CTR-41 and CPU on line when thrown in one direction and will put the CTR-41 and speaker on line in the opposite. If you install the option your unit will look something like Figure 5. I housed my speaker separate with a jack connection so I may use it for other purposes. You could just as well use a larger box and install everything in one unit. To each his own!

One final word on cassette tapes - if you haven't yet, purchase a good cassette head cleaner/demagnetizer and use it often; it will make a difference.

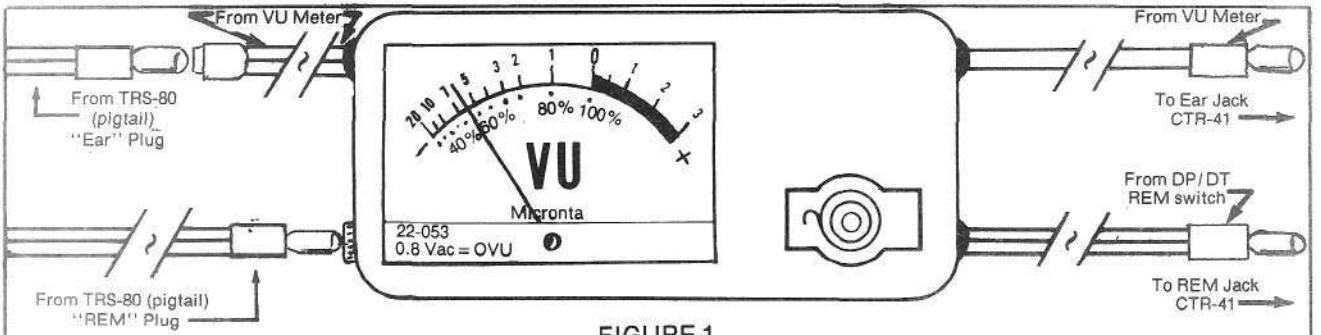


FIGURE 1

WITH SPEAKER OPTION

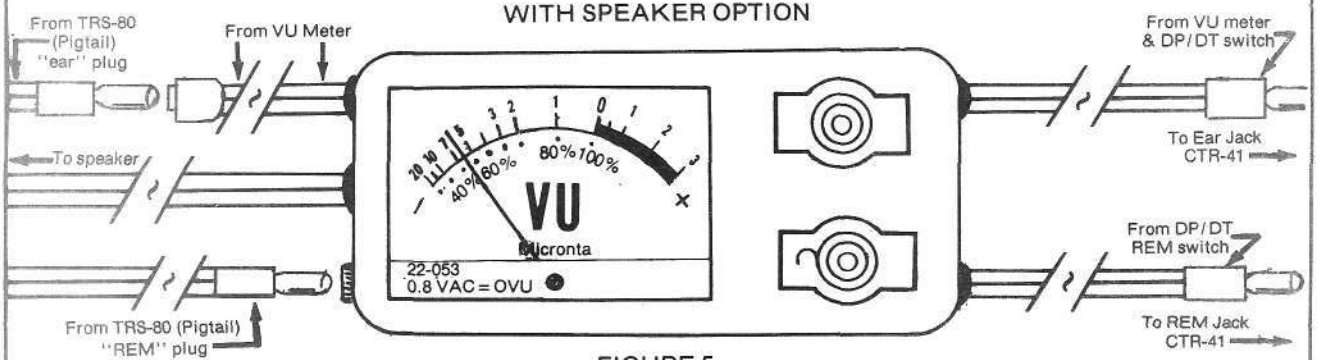


FIGURE 5

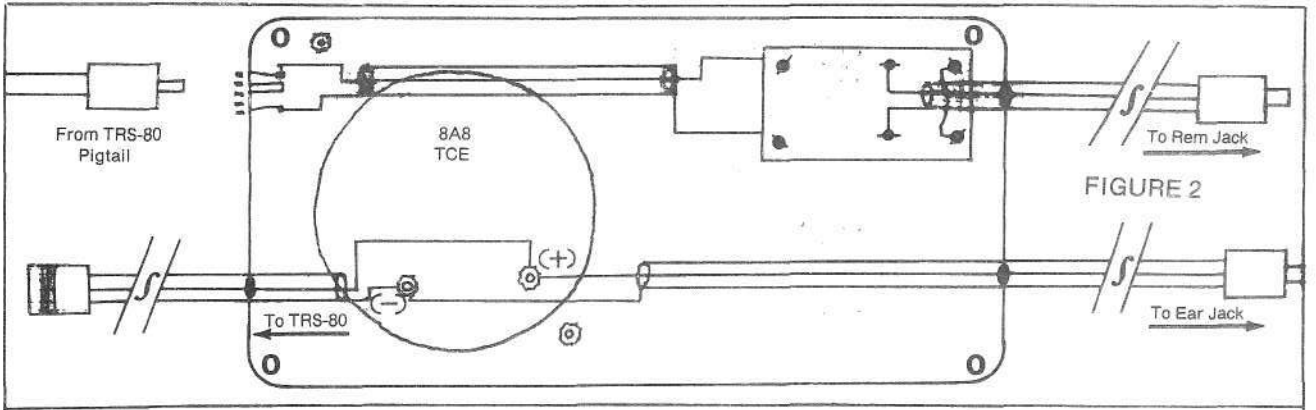


FIGURE 2

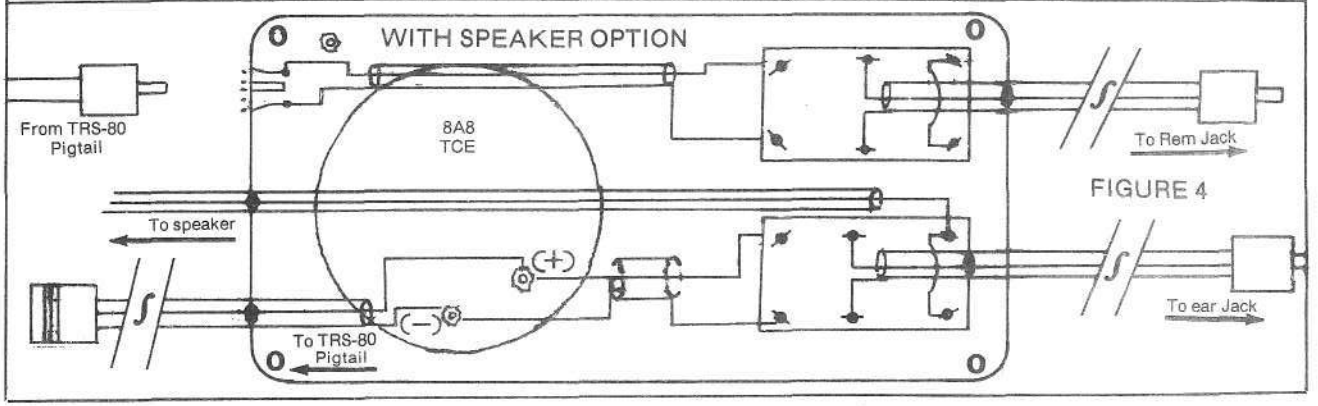


FIGURE 4

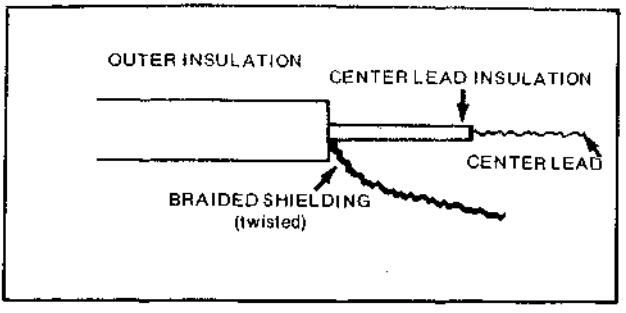


FIGURE 3A

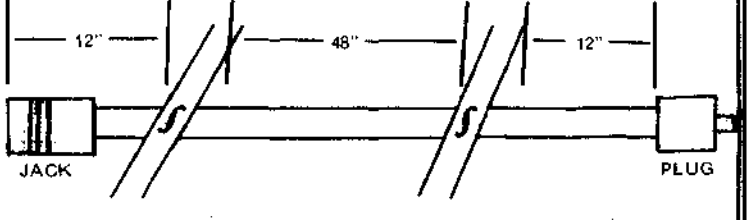


FIGURE 3

INVENTORY 2.3

The programming of the FUTURE is here NOW

Proper inventory management is the backbone of a profitable business, yet it's very difficult to keep current on price increases, shrinkage, low-on-stock itmes and profitable items versus losers, without an efficient and prompt method of surveying your inventory levels at any given time. The Inventory 2.3 System provides the small-to-medium volume business with an efficient method of establishing and maintaining inventory records. Features of Inventory 2.3 include:

- The ability to establish over 2200 inventory item records per clean diskette. Each inventory item record contains data on the vendor, quantity on hand, cost per unit, retail price per unit, reorder points, quantities purchased and sold, and sales history.
- File search by inventory item description or batch/sequence number assignment.
- System operation with 1 to 4 user selectable disk drives.
- Informative reports analyzing both inventory costs and supplies.
- Sophisticated recovery routines with each program to catch data entry errors and to prevent computer "lock-out" through lack of proper line printer interface, etc.
- A looped program format to allow access to any sub-programs whenever a different inventory file function is desired.

A comprehensive manual guides you step by step during your first-time run of the system; your conversion of data from a manual system to the computer system; and regular run procedures throughout the year (update file, add or delete items, monitor activity monthly or weekly, check for low stock, run inventory control reports).

Requires a minimum system configuration of 32K, Level II TRS-80 microcomputer with at least two mini-disks and line printer. Two disks are recommended. $79.95

TSE TRS-80 Software Exchange

17 Briar Cliff Drive Milford, New Hampshire 03055

FORUM-80

A new service for the computer hobbyist is FORUM-80, a network of interactive computer-based message systems. Use of the network is free; required equipment is a modem, RS-232C, and a terminal program such as ST-80 or ST-80D. Simply set your RS-232 to 8 bit word, 300 baud, 1 stop bit, no parity, and dial up one of the numbers below. When you hear a high-pitched tone, place the handset on the modem and press "ENTER". The systems are self-prompting and a lot of fun!

Kansas City, MO	816-861-7040
Chicago, IL	312-925-0259
Ft. Worth, TX	817-923-0009
Orange County, CA	714-730-1206

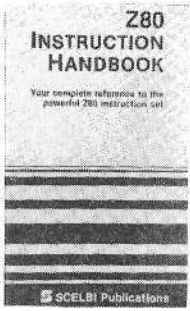
(New systems come on line each month. Try one of the numbers above for a listing of a FORUM-80 in your area.)

TRS-80 Users Group Information

The Central Alabama Microcomputer Society meets on the third Tuesday of every month at the Goodwyn Community Center.

For additional information contact Lewis Garrison (205) 272-8462 or Walter Bray (205) 272-3621.

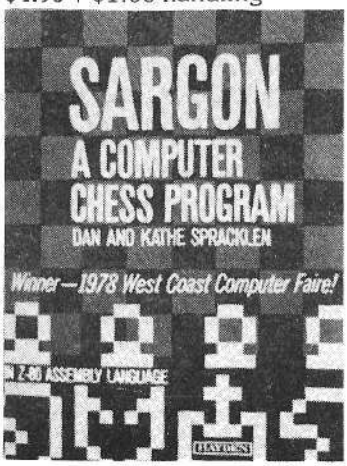
Reference



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 + $1.00 handling

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

NEWDOS +

If NEWDOS is the Cadillac of disk-operating systems, then NEWDOS + has to be the Ferrari. NEWDOS + retains all the features of the original NEWDOS, and adds the following utilities:

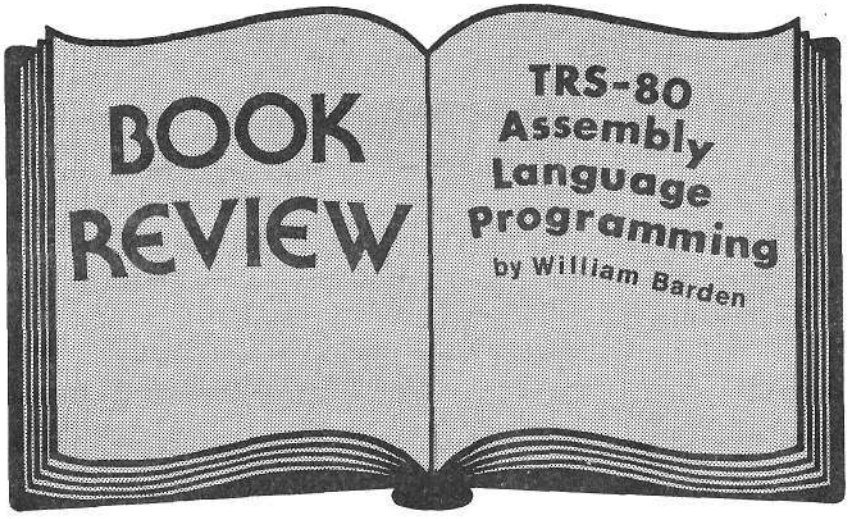
- Editor-assembler for disk
- Disassembler (Z80 machine code)
- LM Offset-allows transfer of any system tape to a disk file (automatically relocated)
- BASIC1-Level one BASIC saved on disk
- LV1DSKSL - not a typo, this saves and loads BASIC1 programs to disk
- DIRCHECK-tests and lists disk directory
- Superzap-display/print/modify any location in memory or on disk

Superzap alone is worth the price of this package. With it, we've quickly recovered lost programs, restored killed data files, and saved many hours of effort. The NEWDOS + manual is another plus: clear and concise, it even includes a byte-by-byte explanation of the directory file ... invaluable if you ever need to save a crashed disk!

The price for all this computer power? That's the best part!
NEWDOS +, Just $99.95

NOTE: Use of this software may require documentation available only with the purchase of Radio Shack TRSDOS 2.1 and/or the Radio Shack Editor/Assembler

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



by George Blank

This book is a major event for TRS-80 programmers. Due to the limited instructions supplied with Radio Shack's Editor/Assembler, there has been an urgent need for a more detailed explanation of Z-80 Assembly Language. I am aware of three other authors presently working on similar books, and have been asked by a pubisher to write one myself. (I declined.)

Bill Barden has excellent credentials for the task. He is well-known to hobbyists as the editor of the Orange County TRS-80 User's Group Newsletter and the author of **The Z-80 Microcomputer Handbook** published by Howard W. Sams & Co. He has an excellent grasp of the subject, and this book will become an indispensable aid to most TRS-80 Assembly Language programmers.

The book is organized into three sections; General Concepts, Programming Methods, and Appendices. The first section covers the Z-80, an introduction to Assembly Language, the Radio Shack Editor/Assembler and T-Bug, and debugging methods. The second section explains how to move data, the use of arithmetic, compare, logic, and bit operations, shifts, strings, tables, input and output, and 12 commonly used subroutines. The appendices list the Z-80 instructions by group and in a breakdown of mnenonic, code and format, description, and flags affected. There is a good index as well.

TRS-80 Assembly Language Programming is less technical and easier to read than **The Z-80 Handbook,** but it is not easy reading. The approach is a cross between a tutorial method and a reference manual, which does make it difficult to use for learning Assembly language programming, but excellent for later reference. There is just enough humor included to prevent the average person from burning out their brain deciphering the technical details. It helps to be able to read on the level of a college

textbook, to understand the hexadecimal number system, and to have a minimal understanding of electronics if you desire to learn from this book. The average thirteen year old ham radio operator will probably be able to devour it at breakfast, but the average adult with a little programming experience in BASIC may feel lost.

Those who do take the trouble to think through the examples and follow the text should be rewarded with a general understanding of the Radio Shack Editor/Assembler and the inner workings of the Z-80. Those who can already use the Editor/Assembler, but who do not feel proficient, should receive a great deal from Mr. Barden.

The book is well presented, though the type is a bit dense and the illustrations limited. There are a few typographical errors, but they are minor. A technical error on page 43 promises a routine to add a number to register A, then gives an example showing how to add a number to register B. I have two complaints about ommissions in the book. Direct memory access is not supported, as "the TRS-80 does not currently utilize it." That is not encouraging to those of us who have ordered the TRS-80 Model II, which does use DMA. The other omission I feel is even more serious. Mr. Barden does not give any help with TRS-80 Assembly language Disk I/O operations. Another feature that would be desirable would be a guide to using subroutines in the BASIC ROM, with a list of important ones, especially since Mr. Barden's library of subroutines is quite limited.

I have a philosophical argument with the author when he encourages self-modifying code on page 177. There is already enough of a breeding ground for bugs in assembler operations without leaving the door open for more trouble. For all these reasons, I do not feel that this is the ultimate book on Assembly language for the Z-80, and I still believe there is a good market for a tutorial handbook on the level of the Level I BASIC User's Manual.

Even with these reservations, Mr. Barden has done a real service to TRS-80 owners, only exceeded by Radio Shack's generosity in making the book available at 5000 locations at the incredible bargain price of $3.95. At that price, you can't go wrong. If you own or are even considering Radio Shack's Editor/Assembler, you should buy this book. It is available at you your local Shack with the catalog number of 62-2006.

GWB

TRS-80 MUSIC

by James Garon

The TRS-80 is a miniature radio station that's always broadcasting and any AM radio placed at the right-hand side of the keyboard will verify this fact. With any program running, tune the station selector for clearest reception. Every program produces a unique pattern of noises. You can harness the noises to produce recognizable tunes. The following program is an example:

LINES 1-5:

When using FOR/NEXT loops to create music on the Level II TRS-80, it is necessary to use **integer variables** for the loop counter(s). To hear why this is the case, turn your radio on and type:

FOR I! = 1 to 3000: NEXT **ENTER**

Sounds like a dying chicken, right? Now type:

FOR I% = 1 to 3000: NEXT **ENTER**

This time we hear a much steadier tone. The reason for this is that integers (like I%) are stored in a relatively simple form, while single precision variables (like I!) are stored in a more complicated type of scientific notation. The larger a single (or double) precision variable is, the more work Level II must do to use that variable. The harder Level II works, the lower the note produced. This explains the falling pitch in the first example.

The variables L, and Q through Z will take on fractional values during program execution, therefore they are not defined to be integers. The jump to line 300 allows us to use small line numbers for the "notemaking" subroutines in lines 10 through 210. Level II can "find" small line numbers faster than large ones. To hear this, type:

0 GO TO 0 **Yes Virginia, there IS a line number zero!**

And then type:

RUN

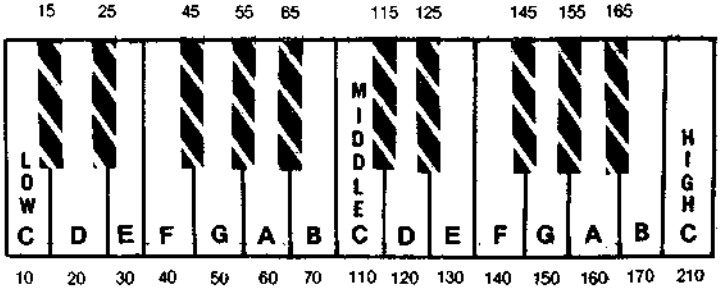
Note the high pitch. Press **BREAK**; remove the line by typing 0 **ENTER**
And now try:

200 GO TO 200 **Be sure to remove the line after listening**

Those two extra digits make quite a difference in pitch!

LINES 10-210;

These 25 subroutines produce two octaves of notes. They are organized in order of increasing pitch as shown:



Note that all "C" notes end in 10 (10, 110, 210), all "D" notes end in 20 and so on, and that "black keys" have line numbers between the corresponding white keys on either side.

These routines use the variable K as the loop counter. All unnecessary spaces have been omitted to minimize stray clicks and pops between notes. (This explains what a friend refers to as my "FORK routines".)

The content of each loop is chosen not for its meaning, but for its difficulty. It takes BASIC a certain amount of time to process a colon (:), a shorter time to handle each character of a REM statement and a much shorter time to deal with spaces (blanks). As we've seen, the longer a loop takes, the lower the note produced. If you're typing this program yourself, be sure to type exactly what is illustrated.

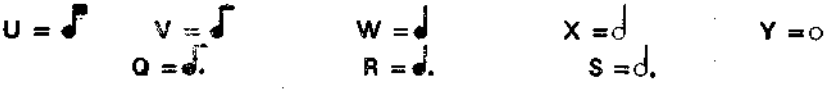
The variable L is used to control the length of each note. A note such as "LOW C" at line 10 requires half as many complete loops to last as long as "MIDDLE C", (one octave higher) at line 110. This is why the numbers which multiply L increase from 12 to 48.

LINES 300-950:

U represents the length of the shortest note. By changing this value, we can change the tempo of the entire song. If you read music, think of U as a sixteenth note.

V is twice as long as U (an eighth note), W is twice as long as V (a quarter note), X is twice as long as W (a half note), Y is twice as long as X (a whole note). Also, Q is 50 percent longer than V (a dotted eighth note),

R is fifty percent longer than W (a dotted quarter note), S is fifty percent longer than X (a dotted half note).



These values are computed and saved prior to starting the song to minimize noisy calculations during the performance.

M is used to keep track of the verses. In line 350 for example, the program branches to 370 if M = 1 or to 380 if M = 2. If M is zero, no branch is made, and line 360 is executed. This allows **two** line numbers to specify a **three-way branch.** Line 390 starts the music. Two pieces of information are required for each note: the duration (L) and the pitch (subroutine line number). Thus the first note is G and it is a sixteenth note. W need not specify note length (L) until there is a change in duration. The third note is "MIDDLE C" and lasts for a whole note (L = X: GOSUB 110).

Lines 650 and 670 temporarily increase the tempo. Line 820 takes advantage of six consecutive notes which have the same duration. The ONGOSUB lets us compress this section significantly.

```
1 DEFINTA-P:DEFSNGL:GOTO300
2 SOFTSIDE PRESENTS:
3 YESTERDAY - BY JOHN LENNON & PAUL MCCARTNEY
5 PROGRAM AUTHOR - JAMES GARON - ANAHEIM, CA
10 FORK=1TO12*L
11 :::: REM
12 :::REM//
13 NEXT:RETURN
15 FORK=1TO13*L
16 :::  REM//
17 :::  REM///
18 NEXT:RETURN
20 FORK=1TO14*L
21 ::REM//////
22 :::REM//////
23 NEXT:RETURN
25 FORK=1TO15*L
26 ::REM/////
27 :::REM
```

```
28 NEXT:RETURN
30 FORK=1TO16*L
31 ::  REM////
32 ::  REM////
33 NEXT:RETURN
40 FORK=1TO17*L
41 ::  REM
42 :: REM
43 NEXT:RETURN
45 FORK=1TO18*L
46 : REM////
47 :: REM////
48 NEXT:RETURN
50 FORK=1TO19*L
51 :   REM
52 ::REM
53 NEXT:RETURN
55 FORK=1TO20*L
56 :REM//////
57 : REM/////
58 NEXT:RETURN
60 FORK=1TO21*L
61 :      REM
62 :REM
63 NEXT:RETURN
65 FORK=1TO22*L
66 REM////////
67 :REM/////
68 NEXT:RETURN
70 FORK=1TO23*L
71 :REM
72            REM
73 NEXT:RETURN
110 FORK=1TO24*L:::REM//////////
112 NEXT:RETURN
115 FORK=1TO26*L:::REM/////
116 NEXT:RETURN
120 FORK=1TO28*L::REM////////////////
121 NEXT:RETURN
125 FORK=1TO30*L
```

```
126 REM///
127 REM
128 NEXT:RETURN
130 FORK=1TO32*L: REM//////////////////////
131 NEXT:RETURN
140 FORK=1TO34*L: REM//////////////////
141 NEXT:RETURN
145 FORK=1TO36*L:   REM///////////
146 NEXT:RETURN
150 FORK=1TO38*L:   REM////////
151 NEXT:RETURN
155 FORK=1TO40*L:REM//////
156 NEXT:RETURN
160 FORK=1TO42*L:REM///
161 NEXT:RETURN
165 FORK=1TO44*L:REM
166 NEXT:RETURN
170 FORK=1TO46*L
171  :NEXT:RETURN
210 FORK=1TO48*L::NEXT:RETURN
300 U=1.5
310 V=U+U:W=V+V:X=W+W:Y=X+X:Q=U+V:R=V+W:S=W+X
500 FORM=0TO1
510 ONMGOTO520,530:CLS:PRINT@18,"YESTERDAY

YESTERDAY, ";:GOTO540
520 PRINT"

SUDDENLY, ";:GOTO540
530 PRINT"
YESTERDAY, ";
540 L=V:GOSUB50:GOSUB40:L=X+R:GOSUB40
550 ONMGOTO560,570:PRINT"ALL MY TROUBLES SEEMED SO FAR AWAY":GOTO580
560 PRINT"I'M NOT HALF THE MAN I USED TO BE":GOTO580
570 PRINT"LOVE WAS SUCH AN EASY GAME TO PLAY
580 L=V:FORI=1TO6:ONIGOSUB60,70,115,120,130,140:NEXT:L=R:GOSUB130:
L=V:GOSUB120:L=X+W:GOSUB120
590 ONMGOTO600,610:PRINT"NOW IT LOOKS AS THOUGH THEY'RE HERE TO STAY":
GOTO620
600 PRINT"THERE'S A SHADOW HANGING OVER ME":GOTO620
```

```
610 PRINT"NOW I NEED A PLACE TO HIDE AWAY
620 L=V:FORI=1TO6:ONIGOSUB120,120,110,65,60,50:NEXT:L=R:GOSUB65:
L=V:GOSUB60:L=W:GOSUB60
630 ONMGOTO640:PRINT"OH I BELIEVE IN YESTERDAY.";:GOTO650
640 PRINT"OH YESTERDAY CAME SUDDENLY.":GOTO650
650 L=W:GOSUB50:GOSUB40:L=V:GOSUB60:L=R:GOSUB50:L=V:GOSUB20:L=W:
GOSUB40:L=V:GOSUB60:L=X+W:GOSUB60:IFM=2THEN730
660 NEXT
670 FORM=0TO1:ONMGOTO680:PRINT"
WHY SHE HAD TO GO I DON'T KNOW";:GOTO690
680 PRINT"I SAID SOMETHING WRONG NOW I LONG ";
690 L=X:GOSUB60:GOSUB60:L=W:GOSUB120:GOSUB130:GOSUB140:L=V:
GOSUB130:GOSUB120:L=R:GOSUB130
700 ONMGOTO710:PRINT"SHE WOULDN'T SAY.":L=V:GOSUB120:L=W:GOSUB110:
GOSUB120:L=Y:GOSUB60:GOTO720
710 PRINT"FOR YESTERDAY.":L=V:GOSUB120:L=W:GOSUB110:GOSUB130:
GOSUB140:GOSUB110:GOSUB65:GOSUB60
720 NEXT:GOTO510
730 CLS:PRINT@460,CHR$(23);:L=W:FORI=1TO5:L=L/.9:READA$:PRINTA$;:
ONIGOSUB40,60,50,20,40:NEXT:READA$:PRINTA$;:L=W:GOSUB60:READA$:
PRINTA$:L=Y:GOSUB60:CLS
900 DATAYES,T,E,R,D,A,Y...
```

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