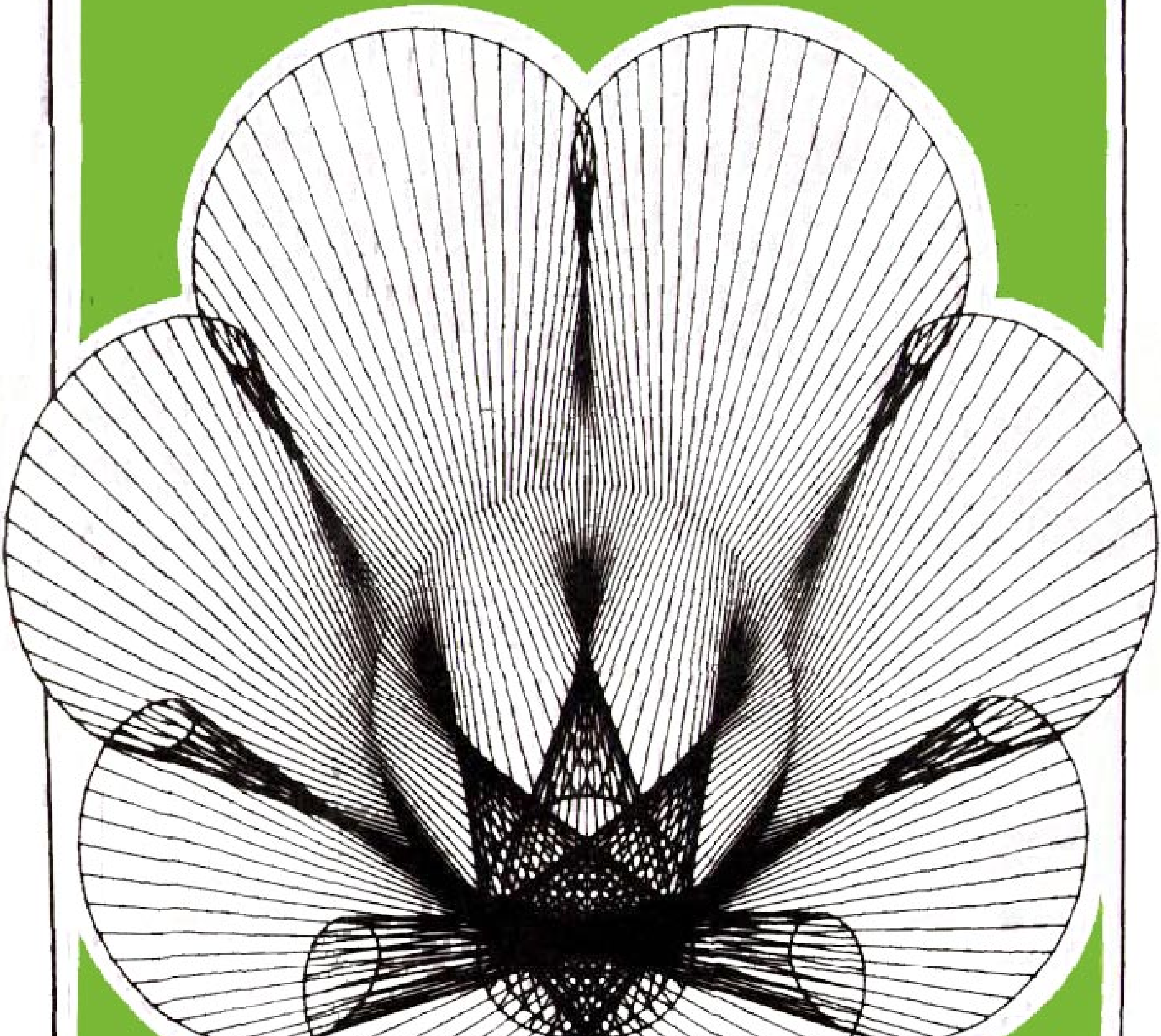
PROG/80

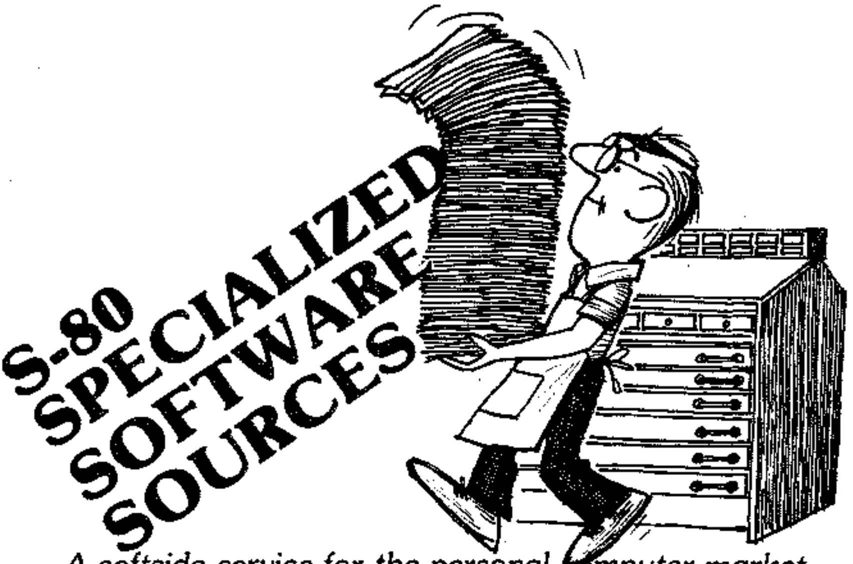
DEDICATED TO THE SERIOUS PROGRAMMER

APRIL 1980 **VOL. 1, NO. 7** **$3.00**

Devices (2), Redefining Level II Keyboard, Writing System Tapes, Forum 80, and much more!



S-80 SPECIALIZED SOFTWARE SOURCES



A softside service for the personal computer market

Quite often we receive submissions that are not commercially marketable. For example, someone may have a useful program for rabbit breeding, but we do not have enough demand for it to justify the cost of reviewing, advertising, and publishing it. A medium is needed for the exchange of information on specialized programs.

Therefore we will maintain a catalog of sources for this software and publish it quarterly at a nominal cost. If you would like to list your software (FREE!), send this form to:

S.S.S.S.
P.O. Box 68
Milford, NH 03055

This is a medium for original authors only. Name and address of the programmer must appear in ad. We reserve the right to refuse any ad, and will publish only ads for non commercial practical applications. SoftSide does not review or warranty these programs in any way.

SUBMISSION FORM:
PLEASE PUBLISH THE FOLLOWING AD IN S.S.S.S.:

NAME
ADDRESS
I CERTIFY THAT I AM THE AUTHOR OF THE ABOVE PROGRAM.

(SIGNATURE)

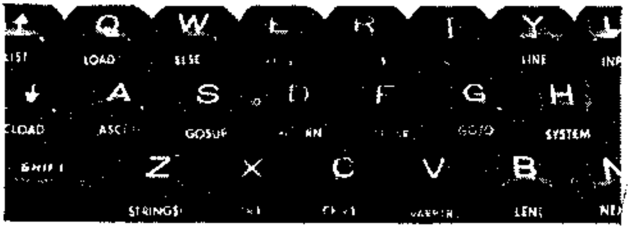
PROG/80

BASIC

DISK

TIME SHARING

BITS


Q
W
Y
LOAD
LINE
A
S
G
H
CLOAD
GOSUB
SYSTEM
Z
X
C
V
B
STRING$
LEN

PROG/80

Publisher
Roger W. Robitaille, Sr.

Business Manager
Elizabeth J. Robitaille

Managing Editor
George W. Blank

Editor
Lance Micklus

Software Editor
James Garon

Time Sharing Editor
Richard Taylor

Copy Editor
Freida Day

Creative Director
Elaine Cheever

Production
Laurie Miller
Alison Willis
Gail Nash

Subscriptions Manager
Diana Bishop

Customer Service
Bette Keenan

PROG/80 is published bimonthly by SoftSide Publications, Milford, New Hampshire. Telephone (603) 673-5144. Subscription rates: USA regular bulk rate - $15 per year. USA first class, Canada, Mexico, APO/FPO, overseas surface mail - $21 per year. Overseas airmail - $27 per year. **Make all remittances payable in US funds.** Mail Subscription inquiries to PROG/80 Subscriptions Manager, PO Box 68, Milford, NH 03055.

PLEASE NOTE:

Our WATS line number is:

1-800-258-1790

(In NH call 673-5144)

TOLL FREE
9 AM to 9 PM*
(Monday - Friday)
9 AM to 3 PM
(Saturdays)

It is ONLY to be used for ordering software.

Programmers will not be available on this line.

HOTLINE Hours:

Tuesday Nights
7-10 p.m.*

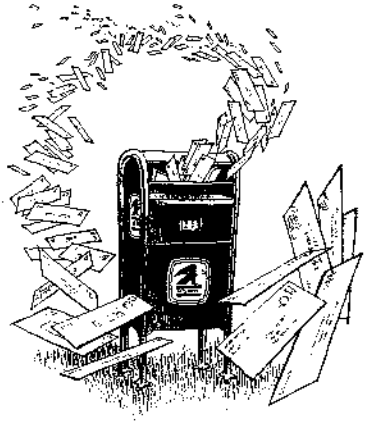
673-5144

Your Cooperation Will Be Most Appreciated

*Eastern Standard Time

Outgoing Mail

by Lance Micklus



This month's $64,000 question is — which operating system will ultimately dominate the TRS-80 Model II market. TRSDOS, CPM, or VTOS — all are, or will be at some point in the future, be competing to be the operating system to use.

Unlike the TRS-80 Model I, the rules of the game are a little different this time. The three most used Model I operating systems are relatively compatible with one another. That is, unless you try to use some specific feature which one system has, but is not in the others. Like OPEN "E". As a software developer, I would not want to write any software that locks me into one particular operating system without a good reason for taking advantage of that feature. In most types of programs, however, you can avoid these system extensions, and write code which will run on all three.

Not so with the TRS-80 Model II. At the machine language level, where I'm doing a lot of work right now, I must work with that operating system, and only that operating system. A Model II ST80 for TRSDOS, will not work at all under Model II CPM.

The Model II is now making some people aware of an important fact. Without the support of software developers, users will not use that operating system. Wouldn't it be interesting if CPM became the standard operating system for the Model II? That would certainly get Radio Shack to think about how important their own software is. The conclusion I think they'll reach is, "so what".

I think, if things reach that point, Radio Shack will just keep selling their Model II's, with Model II TRSDOS, an operating system nobody uses, and smile all the way to the bank. It wouldn't be the first time that's happened. There have been other computer systems that came with their own operating system, even a BASIC, that nobody used.

The thing that will make or break CPM on the Model II are the software distributors. Lifeboat Associates recently went out of their way to see that software developers got CPM for the Model II by selling it at a special price. Clearly, Lifeboat knows that if they're going to sell CPM, and CPM software, the software people, like myself, have to support them. If it works, Lifeboat stands to make a fortune.

Randy Cook, too, realizes that his VTOS, even on the TRS-80 Model I, must get the support of software developers and dealers if it is to survive. And the latest rumor has it that there is a SUPER DOS coming out, and the software dealers and developers are going to be stroked, so they'll support it.

In the end, I think it will turn out this way. Randy Cook is the best DOS author in the country, possibly the world. But, if you read my article called GOING INTO BUSINESS in the EIGHTY (Feb. 1980), then you know that his success is going to be more a matter of how good a business man he is. With all due respect to Randy Cook, he writes a terrific DOS, but a lousy business contract. That's what will decide if the market will go for VTOS, or SUPER DOS.

On the TRS-80 Model II, I think it could go either way. Or, it might turn out to be a split system, with some users running TRSDOS, and others using CPM, and still others going for VTOS (if and when we ever see it). Based on the choice of TRSDOS or CPM, I'll pick TRSDOS, much as I hate it — low as my opinion of it is (which is pretty low, folks).

Look at it this way. Let's say I want to write a payroll program. If I try to write it for the Model II CPM system, then I'm competing with all of the other CPM payroll systems — some of which have been around for years. I've got a better shot on TRSDOS because there is less competition, and what competition there is will be leaving the starting gate the same time I am. Second, everybody who owns a Model II owns TRSDOS, but not everybody with a Model II owns CPM. Third, there is clearly just one single BASIC which is the standard for TRSDOS on the Model II, and everybody owns it. With CPM, there is no standard BASIC. So, I can only market to those CPM people who own the same BASIC as I'm using. If they don't own it, then they have to convert the code over to their BASIC. The fact of the matter is that users want to buy programs that load and go, and do not want to have to sit there for hours on end changing the code to work with their own BASIC.

You might think that VTOS will come in too late in the game to compete against CPM and TRSDOS. I don't think so, not yet. Most Model II users I know are hanging loose, still not strongly committed to one operating system or the other. In fact, many Model II users don't have much software to run on their computers, short of what little they've written themselves. So, the market is still fluid.

VTOS has most of the advantages of TRSDOS in the Model II environment. It could be well supported by software people who would rather work with a new operating system than compete with an old one. Furthermore, VTOS would come with its own BASIC, eliminating the question of which BASIC to write for. Finally, software distributors could get a license for it, so they could support it. The main draw back to it, like CPM, is that fact that not everybod[illegible] will own VTOS.

I happen to have another reas[illegible]n for at least wishing that [illegible]OS becomes the standard ope[illegible]ing system, at least on the TRS-[illegible]. If you trace the roots of T[illegible]OS back, you don't have to lool[illegible]ar to find Randy Cook. But, w[illegible]st people don't know is th[illegible] trace the roots of CPM [illegible] enough, there he is again [illegible] Mr. Randy Cook. He is the father [illegible] Z-80 and 8080 operating systems. I think he should at least get his fair share of the market place.

Outstanding Utilities from RACET Computes

INFINITE BASIC

Adds more than 70 commands to BASIC that can be merged in any combination to make efficient use of memory. Includes matrix read, inverse, transpose, identity, simultaneous equations, scalar, vector, and multidimensional array arithmetic, dynamically reshape, expand, and delete arrays, change arrays in mid program, read and write arrays on tape, copy elements, zero and move arrays. String functions include left and right justify, truncate, rotate, text justification, string centering, delete and insert substring, pack string, convert upper and lower case, translate characters, reverse strings, verify function, test number of occurences, masked string searches, encrypt and decrypt string, compress and uncompress string characters. High speed sort routines for strings and arrays, including multikey sorts, are also part of this package.

For business users, an add on package includes multiple precision packed decimal arithmetic, with up to 127 digits of accuracy, binary search of sorted arrays, insert elements in sorted arrays, automatic page headings, footings, and pagination including forced end of page, and automatic hash for record retrieval. **Infinite BASIC $49.95. Infinite business (add on) $29.95**

COMPROC COMMAND PROCESSOR

Chain multiple steps in disk BASIC upon power up, relocatable key debounce, allows pauses for data entry at specified steps during execution (on cassette for disk systems only) for **$19.95.**

DOSORT

BASIC control program with high speed machine language sort for disk users. Merge and sort files on more than one disk if you have 2 or more drives. Self prompting with manual. Specify 32K or 48K version **$34.95.**

GSF GENERALIZED SUBROUTINE FACILITY

18 machine language subroutines with easy access for BASIC users. Sort 1000 element arrays in 9 seconds, read and write arrays to tape, compress and uncompress data, move arrays in memory, duplicate memory, fast horizontal and vertical lines, 5 routines for screen control. Specify 16K, 32K, or 48K version, for **$24.95.**

REMODEL PROLOAD

BASIC program utility allows you to renumber portions of a program, move portions from one location to another, delete, merge, save and verify combined and changed programs, and create your own library of programs, subroutines, and even data statements. Works on tape or disk systems. Two programs on one tape. Specify 16K, 32K, or 48K (unspecified orders receive 16K) tape **$34.95**

TIMSER

Time series analysis program fits data to 9 different 1st, 2nd, and 3rd order curves. Goodness of fit data, tables, projected data, confidence limits, curve fit, variance, correction factor, seasonal and cyclical variations, inflation corrections are all implemented. Detailed user manual with illustrated examples. Order tape at **$14.95.**

Y-YBAR

Optical system design program allows manipulation of ray heights at lens surfaces using Y-Y Bar diagram method. With documentation on tape for **$14.95.**

DISK SORT/MERGE

DSM IS POWERFUL!!!!!!

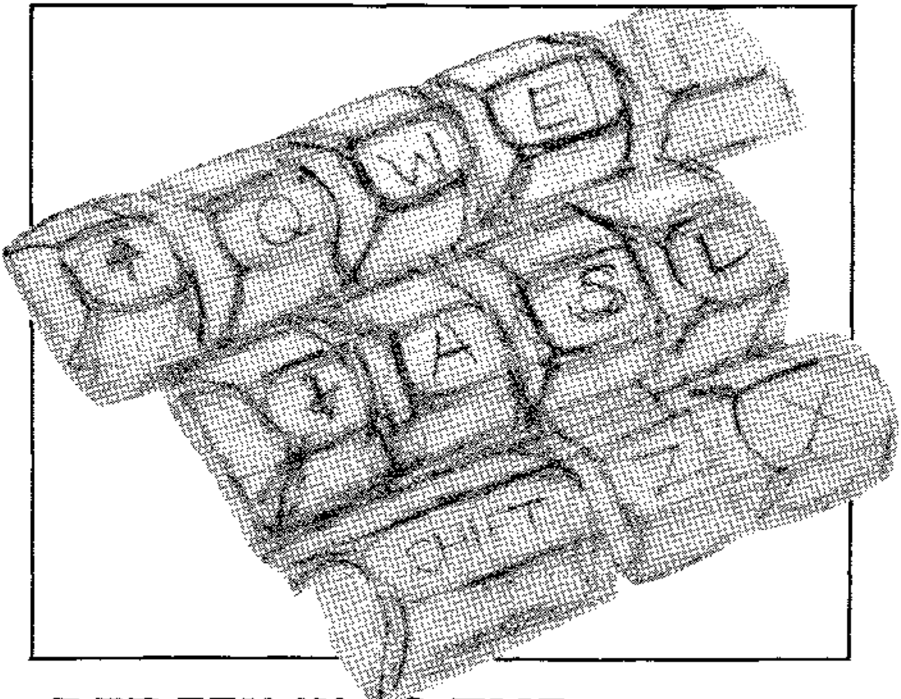
- Sorts large multiple diskette files on a minimum two drive Mod-I disk system.
- All records are physically rearranged — no key files are required.
- Sorts random files created by BASIC, including files containing sub-records spanning sectors.
- Sorts on one or more fields in ascending or descending order. Fields may be character, binary integer, or floating point.
- The sorted output file may optionally have fields deleted, rearranged, or padded.
- Sort commands can be saved for reuse in production applications.
- Single sort, merge, or mixed sort/merge operations may be performed in a single DSM application.
- Sorted output may be written to a new file, or replace the original input file.

DSM IS FAST!!!!

DSM is written entirely in machine language for fast sorting. **$75.00**

TSE The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144

The Software Exchange would like to announce that Microcomputer Applications is an authorized dealer of their software in England. Contact: William F. Jupp, 11 Riverside Court, Caversham, Reading, RG4 8AL, England. Telephone (0734) 470425



REDEFINING THE LEVEL II KEYBOARD

by Phelps Gates

While writing an APL interpreter for the TRS-80, I found that I needed to redefine certain keys: APL uses all four arrows, for example, and you can't get them by pressing keys, even though the video display is capable of printing them (PRINT CHR$(94), etc.). Also, I planned to use shifted letters for some of the APL operators, since the TRS-80 can tell the difference between upper and lower case in keyboard input. But this wasn't going to work unless there was some way to make the video display distinguish upper and lower case...short of getting out the screwdriver, throwing the warranty to the winds, and putting in the lower-case modification.

Fortunately, it's easy to intercept the TRS-80 input and output routines to play around with the value of keys: this article explains how. The possible applications go beyond APL; you can use it in a text editor which feeds a printer with lower-case capability, or redefine keys as graphics characters for games (or artwork). As a bonus, you get a flashing cursor, just like the Model II (or the PET...), and a thoroughly debounced keyboard. The routine described here is slightly different from the one I used in the APL interpreter — it's been modified to work with BASIC programs.

The key to the keyboard routine is location hex 4016-7 which contains the address of the keyboard input subroutine. When you call 2BH for keyboard input, the ROM I/O routine looks in 4016-7 to find the address of the keyboard scan. Normally it finds 03E3 there: this is the address of a routine which gets the ASCII value of the keypress into the A register (or zero if the keyboard is clear). But you can fool the computer into using your own input routine if you put its address into 4016-7: now, when 2BH is called (either by you or by BASIC), control will pass to your keyboard input routine. The video display works the same way; a call to 33H will eventually end up at the routine whose address is in 401E-F (normally 0458H), and you can change this one too.

The following routine sets up the address changes (I've assembled it at 7F80H — MEMORY SIZE 32639 — but it can go anywhere if you make the appropriate changes):

```
7F80 218F7F    SETUP    LD   HL,KEYBO      ;KB routine address
7F83 221640             LD   (4016H),HL    ;to control block
7F86 21C97F             LD   HL,VIDEO      ;video routine
7F89 221E40             LD   (401EH),HL    ;to control block
7F8C C3191A             JP   1A19H         ;back to BASIC
```

If you're starting a machine language program instead of going back to BASIC, replace the last instruction with a jump to your main program. You can't use POKE to put the address of KEYBO into 4016-7, because it's a two-byte address and you can only POKE one byte at a time. As soon as you POKE one of the bytes, 4016-7 will contain a garbage address, and since BASIC is constantly scanning the keyboard it will immediately head for the twilight zone (also known as MEMORY SIZE?).

The actual keyboard intercept routine is pretty straightforward. We begin by calling 03E3 to get the keypress value:

```
7F8F CDE303   KEYBO   CALL 03E3H
```

Now check to see if it's time to flash the cursor on or off:

```
7F92 21BF7F             LD   HL,COUNT      ;count the scans
7F95 35                 DEC  (HL)          ;256 scans done?
7F96 200E               JR   NZ,NOFLIP     ;if not, don't flash
```

COUNT is a byte (at the end of the routine) which counts the number of times the keyboard gets scanned. Conveniently enough, you get a nice flash rate if you flash the cursor once for every 256 scans; so all you have to do is count this byte down to zero over and over. Note that we don't have to save any registers (except A), since the keyboard scan routine is through with them.

Now flash the cursor (if it's time):

```
7F98 F5                 PUSH AF            ;save keypress
7F99 3AC07F             LD   A,(CURSOR)    ;blink the cursor
7F9C EE0F               XOR  0FH           ;flip 80H to 8FH
7F9E 32C07F             LD   (CURSOR),A    ;new cursor
7FA1 2A2040             LD   HL,(4020H)    ;screen location
7FA4 77                 LD   (HL),A        ;cursor to screen
7FA5 F1                 POP  AF            ;restore keypress
```

CURSOR is a byte which alternates between 80H (space) and 8FH (graphics square) by means of the exclusive or instruction. Location 4020-1 hex contains the address on the video screen where the next character is due to be printed.

```
7FA6 B7        NOFLIP   OR  A              ;any key down?
7FA7 C8                 RET Z              ;if not, we're done
```

Now set up for key redefinition:

```
7FA8 21C17F             LD   HL,TABLE      ;conversion table
7FAB 010400             LD   BC,4          ;4 bytes to check
7FAE EDB1               CPIR               ;scan table
7FB0 2005               JR   NZ,DEBOUN     ;NZ if no match
```

The CPIR instruction scans the conversion table (located after this routine), looking for a match with the A register. If it finds one, it sets the zero flag and leaves HL pointing one byte past the matching byte. If it doesn't find a match within 4 bytes (specified by BC), it resets the zero flag. You can customize the conversion table for your own needs: I used the shifted arrow keys for left, right, and down arrows, and the CLEAR key (value 31) to produce an underscore for the APL negation sign.

Now we make the conversion (if necessary) and debounce the keyboard:

```
7FB2 010300             LD   BC,3          ;we're one past
7FB5 09                 ADD  HL,BC         ;up to new value
7FB6 7E                 LD   A,(HL)        ;and convert
7FB7 0E19      DEBOUN   LD   C,25          ;set delay
7FB9 10FE      WAIT     DJNZ WAIT          ;inner loop on B
7FBB 0D                 DEC  C             ;outer loop on C
7FBC 20FB               JR   NZ,WAIT
7FBE C9                 RET                ;all done
```

Starting with a value of 25 for C will loop through the delay 6400 times, which should debounce even the dustiest keyboard.

Now we define bytes for COUNT and CURSOR:

```
7FBF 00        COUNT    DEFB 0
7FC0 80        CURSOR   DEFB 80H           ;alternates 80/8F
```

And finally the conversion table (first the original values, then the converted values):

```
7FC1 18        TABLE    DEFB 24            ;arrow keys (shifted)
7FC2 19                 DEFB 25
7FC3 1A                 DEFB 26
7FC4 1F                 DEFB 31            ;and CLEAR
7FC5 5D                 DEFB 93            ;arrow characters...
7FC6 5E                 DEFB 94
7FC7 5C                 DEFB 92
7FC8 5F                 DEFB 95            ;and underscore
```

We also have to make a few changes to the video printing routine. I decided to print a graphics dot (90H) before shifted letters to distinguish them from unshifted ones. We also need to blank out the cursor when a carriage return or backspace is printed, since otherwise a cursor square sometimes gets stranded at the end of a line; it turns out to be simplest just to blank the cursor every time any character is printed. Finally, we have to backspace twice when we're erasing a shifted character, in order to get rid of the graphics dot; if we required the user to backspace again to get rid of the dot, things would get fouled up in the input buffers.

The following routine intercepts the video output routine to accomplish all this:

```
7FC9 F5         VIDEO   PUSH AF           ;save registers
7FCA D5                 PUSH DE
7FCB DDE5               PUSH IX
```

This time we do have to save the registers, because the video routine which we'll eventually jump to (at 0458H) gets very unhappy if it finds the wrong values there (it doesn't care about HL, though). First we blank the cursor:

```
7FCD 2A2040              LD   HL,(4020H)   ;screen location
7FD0 3620                LD   (HL),20H     ;blank it
```

Now check to see if an extra backspace is needed. We want to backspace twice if the previous character is shifted (and therefore preceded by a dot). Also, of course, the character about to be printed must be a backspace (ASCII 8). So we check for both these conditions. At this point in the I/O routine, the character to be printed is in the C register:

```
7FD2 2B                 DEC  HL           ;back 2 spaces
7FD3 2B                 DEC  HL
7FD4 7E                  LD  A,(HL)       ;a dot?
7FD5 FE90                CP  90H
7FD7 2008                JR  NZ,NODOT     ;if not, forget it
7FD9 79                  LD  A,C          ;C has character
7FDA FE08                CP  8            ;backspace?
7FDC C5                PUSH  BC           ;save C (the char)
7FDD CC5804            CALL  Z,0458H      ;extra backspace
7FE0 C1                 POP  BC           ;restore character
```

We have sneaked in an extra backspace which won't appear in the input buffer, just as the dot we print doesn't appear there; this keeps everything straight. Now print a dot, if necessary:

```
7FE1 79         NODOT    LD  A,C          ;check character
7FE2 FE61                CP  'A'+32       ;carry if unshifted
7FE4 C5                PUSH  BC           ;save character
7FE5 0E90                LD  C,90H        ;graphics dot
7FE7 D45804            CALL  NC,0458H     ;print, if shifted
7FEA C1                 POP  BC           ;restore character
```

Finally we restore the other registers and jump to the ROM video routine to print the character:

```
7FEB DDE1               POP  IX
7FED D1                 POP  DE
7FEE F1                 POP  AF
7FEF C35804              JP  0458H        ;ROM print routine
7F80                    END  SETUP
```

ADVENTURES

by Scott Adams

Adventures on Tape: *Level II, 16K*

Adventure Sampler
Short version of Adventureland, a great way to try out adventuring (but you'll get addicted!) $5.95

Adventureland
An enchanted world with 13 lost treasures guarded by magical beings. $14.95

Pirate's Cove
Clues in blood-soaked books, a pirate and his parrot, alligators, treasure $14.95

Mission Impossible
A chance to prevent the nuclear reactor from being destroyed by a saboteur's time bomb $14.95

The Count
Watch out! you're in Dracula's castle and you must destroy him before he bites you! $14.95

Voodoo Castle
Remove a curse from Count Christo $14.95

Strange Odyssey
Adventure in outer space in the ruins of an ancient alien civilization $14.95

Mystery Fun House
It will take all your ingenuity just to get past the ticket counter, and that's only the beginning! $14.95

Pyramid of Doom
In the shifting sands of the Sahara lies an ancient Pyramid, filled with Egyptian treaasures $14.95

Adventures on Disk, in Sets:

Set 1: Mission Impossible, Pirate's Cove, Adventureland $39.95
Set 2: Voodoo Castle, The Count, Strange Odyssey $39.95

NEW FROM ACORN SOFTWARE PRODUCTS, INC.

LANGUAGE TEACHER SERIES: Spanish or French

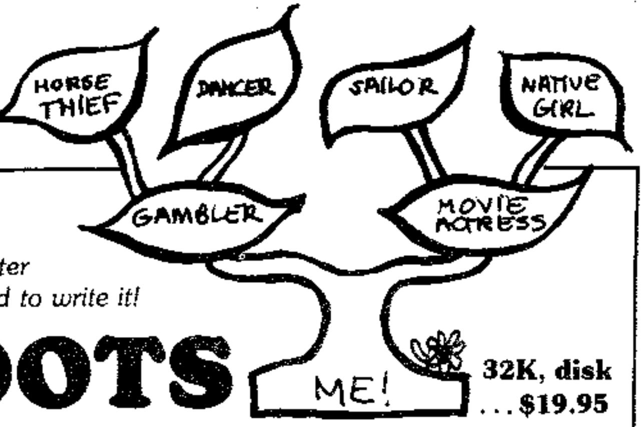
by Cindy and Andrew Bartorillo

Tests you on nouns, verbs, or miscellaneous words; tests your abilities at phrase translations, allowing you to choose whether to translate to English or from it; drills you on verb conjugation. You may choose multiple choice mode or "go it alone". Gives you running percentage of right answers; then retests you on missed answers, gives diagnostic results to show where you need most work, and generates printed tests!

32K, disk only   Specify language   **$19.95 each**

The Software Exchange

6 South Street, Box 68, Milford, NH 03055   603-673-5144



Sooner or later
someone had to write it!

ROOTS

32K, disk
. . . $19.95

Genealogy has become an enormously popular pastime (as well as a lucrative profession for some) in the last few years.

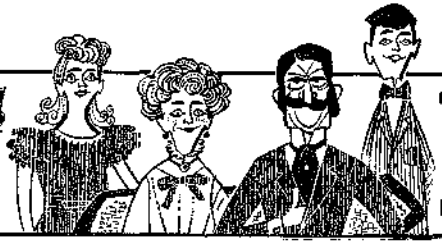
The fascination is endless. Once you've come across your first amazing revelation — that you have a colorful ancestor who was a famous Indian Chief, or an advisor to an English King, or what-have-you — you're hooked, my friend, and as these people who were your progenitors come alive for you, like characters in a play, so does history itself light up like a stage.

ROOTS is an indispensable aid for all genealogists, whether amateur or professional. Up to nine generations of data may be placed into memory per file. Allows display of known data concerning a given individual; display of four-generation pedigree; searches for specific information within a given category (finds all "Smith" last names, or all births in the 1890's, for example); sorts by any selected category; gives printed reports.

The Software Exchange

6 South Street, Box 68, Milford, NH 03055   603-673-5144

TSE



WRITING SYSTEM TAPES

By George Blank

I have long wanted to write a machine language monitor in BASIC to avoid the hassle of system tape loading and demonstrate what a monitor does. Most of the functions of a monitor are quite easy to duplicate with PEEK, POKE, and BASIC arithmetic commands. The one problem was saving machine language programs. Loading was not necessary, for the system command is built into Level II BASIC, but there is no punch command. In addition, BASIC is too slow to keep up with a 500 baud tape.

The first step was to discover the format for system tapes. Thanks to the magnificient instruction manual for TRCOPY ($39.95 from Data Print Publishing) I soon had that

HEADER PORTION OF TAPE

Leader	A string of hex 00
Sync byte	Hex A5
Start code	Hex 55
Program name	Six ASCII bytes
Individual blocks of data	

(Maximum of 256 bytes of data in each block)

INDIVIDUAL BLOCKS OF DATA

Block header code	Hex 3C
Length of block	Hex 01 (Decimal 1) to 00 (Decimal 256)
Start memory for block	(LSB, MSB)
DATA	# of bytes indicated by length of block
Checksum	Total formed by adding memory and data without carrys in single byte

TRAILER BLOCK

End code	Hex 78
Execution address	(LSB,MSB)

Once I knew what had to go on tape, the only problem was to figure out how to do it. Since I was editing our first book, Pathways Through the ROM, which lists what each subroutine in the Level II ROM does ($19.95 from TSE when published), that was also easy. I decided to make use of the following ROM routines:

01F8 HEX	Turns the cassette player off
0212 HEX	Selects cassette one when register A contains 0
0264 HEX	Writes one byte from register A to tape
0287 HEX	Writes a leader and sync byte

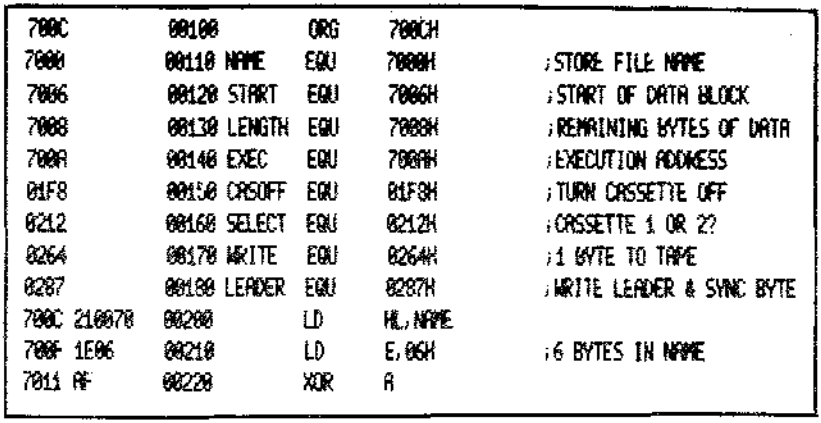
HOW TO USE THE PROGRAM

It is necessary to load the program name into 7000 to 7005 hex, the starting address of the data to be written to tape in location 7006 and 7007, with the least significant byte first, the length of the program in 7008 and 7009, again in least significant, most significant byte order, and then the execution address, also in Z-80 order.

Then press the play and record buttons on your tape recorder and call the routine with a jump to 700C hex. I did not include a routine to input the data in the program as I am using it in a BASIC program, and the BASIC program pokes the data into place.

If you want to relocate this program to another location in memory, you will have to change the address of "NAME", "START", "LENGTH", and "EXEC" in the lines listed in the symbol table at the end of the program. I used relative jumps in the rest of the program, so it can be moved easily.

With this program, you can really save any machine language program if you can find the start, end and execution addresses. The BASIC version included here is a minimal program for writing system tapes. I have modified this routine to use inside a packed string in my monitor-disassembler, "Super Simon", available from TSE for $9.95. It will edit, zero, and save memory, and display it in HEX, ASCII, and Z-80 mnemonics to either the screen or the line printer.



```
700C         00100          ORG    700CH
7000         00110 NAME     EQU    7000H          ;STORE FILE NAME
7006         00120 START    EQU    7006H          ;START OF DATA BLOCK
7008         00130 LENGTH   EQU    7008H          ;REMAINING BYTES OF DATA
700A         00140 EXEC     EQU    700AH          ;EXECUTION ADDRESS
01F8         00150 CASOFF   EQU    01F8H          ;TURN CASSETTE OFF
0212         00160 SELECT   EQU    0212H          ;CASSETTE 1 OR 2?
0264         00170 WRITE    EQU    0264H          ;1 BYTE TO TAPE
0287         00180 LEADER   EQU    0287H          ;WRITE LEADER & SYNC BYTE
700C 210070  00200          LD     HL,NAME
700F 1E06    00210          LD     E,06H          ;6 BYTES IN NAME
7011 AF      00220          XOR    A
```

```
7012 CD1202   00230         CALL   SELECT          ;CASSETTE 1
7015 CD8702   00240         CALL   LEADER          ;WRITE LEADER & SYNC BYTE
7018 3E55     00250         LD     A,55H           ;START CODE
701A CD6402   00260         CALL   WRITE
701D 7E       00270 LOOP1   LD     A,(HL)          ;ONE BYTE OF NAME
701E CD6402   00280         CALL   WRITE
7021 23       00290         INC    HL
7022 1D       00300         DEC    E
7023 20F8     00310         JR     NZ,LOOP1
7025 2A0670   00320 LOOP2   LD     HL,(START)      ;ADDRESS OF DATA BLOCK
7028 ED5B0870 00330         LD     DE,(LENGTH)     ;LENGTH OF DATA BLOCK
702C AF       00340         XOR    A               ;A=0
702D BA       00350         CP     D               ;BLOCK > 256 BYTES?
702E 280D     00360         JR     Z,ENDTST        ;NO, END OF DATA?
7030 24       00370         INC    H               ;ADDRESS OF NEXT BLOCK
7031 220670   00380         LD     (START),HL      ;STORE
7034 25       00390         DEC    H
7035 15       00400         DEC    D               ;LENGTH AFTER THIS BLOCK
7036 ED530870 00410         LD     (LENGTH),DE     ;STORE
703A 14       00420         INC    D
703B 1806     00430         JR     BLOCK
703D BB       00440 ENDTST  CP     E               ;END OF DATA?
703E 2829     00450         JR     Z,TRAILR
7040 320870   00455         LD     (LENGTH),A      ;TRAILR NEXT TIME
7043 3E3C     00460 BLOCK   LD     A,3CH           ;BLOCK START CODE
7045 CD6402   00470         CALL   WRITE
7048 7B       00480         LD     A,E             ;LENGTH OF DATA BLOCK
7049 CD6402   00490         CALL   WRITE
704C 7D       00500         LD     A,L             ;LSB OF BLOCK ADDRESS
704D 4F       00510         LD     C,A             ;BEGIN CHECKSUM
704E CD6402   00520         CALL   WRITE
7051 7C       00530         LD     A,H             ;MSB OF BLOCK ADDRESS
7052 CD6402   00540         CALL   WRITE
7055 81       00550         ADD    A,C             ;CHECKSUM
7056 4F       00560         LD     C,A
7057 7E       00570 DATA    LD     A,(HL)          ;1 BYTE OF DATA
7058 CD6402   00580         CALL   WRITE
705B 81       00590         ADD    A,C             ;CHECKSUM
705C 4F       00600         LD     C,A
```

```
705D 23        00610          INC    HL               ;NEXT DATA ADDRESS
705E 1D        00620          DEC    E                ;COUNT BYTES
705F AF        00630          XOR    A                ;A=0
7060 BB        00640          CP     E                ;LAST BYTE?
7061 20F4      00650          JR     NZ,DATA          ;GO FOR MORE DATA
7063 79        00660 CKSUM    LD     A,C
7064 CD6402    00670          CALL   WRITE            ;WRITE CHECKSUM
7067 18BC      00680          JR     LOOP2
7069 3E78      00690 TRAILR   LD     A,78H            ;END CODE
706B CD6402    00700          CALL   WRITE
706E 210A70    00710          LD     HL,EXEC          ;EXECUTION ADDRESS
7071 7E        00720          LD     A,(HL)           ;LSB
7072 CD6402    00730          CALL   WRITE
7075 23        00740          INC    HL
7076 7E        00750          LD     A,(HL)           ;MSB
7077 CD6402    00760          CALL   WRITE
707A CDF801    00770          CALL   CASOFF
707D C9        00780          RET
0000           00790          END
00000 TOTAL ERRORS
 BLOCK  7043 00460   00430
CASOFF  01F8 00150   00770
CKSUM   7063 00660
DATA    7057 00570   00650
ENDTST  703D 00440   00360
 EXEC   700A 00140   00710
LEADER  0287 00180   00240
LENGTH  7008 00130   00330 00410 00455
LOOP1   701D 00270   00310
LOOP2   7025 00320   00680
NAME    7000 00110   00200
SELECT  0212 00160   00230
START   7006 00120   00320 00380
TRAILR  7069 00690   00450
WRITE   0264 00170   00260 00280 00470 00490 00520 00540 00580
                     00670 00700 00730 00760
```

```
10 REM * SYSTEM TAPE CREATION PROGRAM *
      *  GEORGE BLANK - NO COPYRIGHT *
19 REM               *  SET MEMORY SIZE * CLEAR RESTORE FLAG *
20 CLEAR50:DEFINTN,X,Y:POKE16561,254:POKE16562,111:POKE16553,255
29 REM * POKE SUBROUTINE INTO PROTECTED MEMORY *
30 FORX=28684TO28800:READY:POKEX,Y:NEXT
39 REM * POKE SIX CHARACTER NAME INTO 7000 HEX *
40 INPUT"<FILE NAME>";A$:A$=A$+"      ":A$=LEFT$(A$,6)
50 FORX=1TO6:Y=ASC(MID$(A$,X,1)):POKE28671+X,Y:NEXT
59 REM * POKE START  LENGTH  AND EXEC *
60 INPUT"<STARTING ADDRESS>";M:N=M/256:POKE28678,M-N*256:POKE286
79,N:INPUT"<ENDING ADDRESS>";E:L=E-M:N=L/256:POKE28680,L-256*N:P
OKE28681,N:INPUT"<EXECUTION ADDRESS>";M:N=M/256:POKE28682,M-N*25
6:POKE28683,N
69 REM * TAPE OR DISK BASIC? * APPROPRIATE USR ROUTINE *
70 X=PEEK(16396):IFX=201THENPOKE16526,12:POKE16527,112:X=USR(X):
END ELSECMD"T":DEFUSR=28684:X=USR0(X):END
79 REM     * MACHINE CODE STORED AS DATA *
80 DATA 33,0,112,30,6,175,205,18,2,205,135,2,62,85,205,100,2,126
,205,100,2,35,29,32,248,42,6,112,237,91,8,112,175,186,40,13,36,3
4,6,112,37,21,237,83,8,112,20,24,6,187,40,41,50,8,112
90 DATA 62,60,205,100,2,123,205,100,2,125,79,205,100,2,124,205,1
00,2,129,79,126,205,100,2,129,79,35,29,175,187,32,244,121,205,10
0,2,24,188,62,120,205,100,2,42,10,112,126,205,100,2,35,126,205,1
00,2,205,248,1,201,71,87,66
```

Corrections To "What's In A Name" - February

The following letter was submitted by Ruben Rodriguez, El Paso, Texas...

I enjoyed the articel "What's In A Name" by James Garon because it is one of the articles that has helped me to understand better and fully use the capabilities of my TRS-80.

However I discovered three mistakes that prevent the program from running properly; these are shown below:

10 The 8th entry data should be 254.
70 Change the equals to an asterisk (*)
80 Change the 70 at the end of this line to 80

I hope this information will help other readers to run this program.

VOICE SYNTHESIZER LABORATORY

by John R. Olsen, Jr.

Here is an indispensable aid if you're writing programs using the TRS-80 Voice Synthesizer.

The program displays the correct phonemes on the video screen: you try out various combinations until you get the right words, using all 62 phonemes, even the ones the Synthesizer manual indicates cannot be directly input from the keyboard.

You can listen to your creation as many times as you want without having to type it on the keyboard each time. When you are totally satisfied with it, add it to the inmemory vacabulary. You can also delete words from the memory, when desirable, and before you turn the TRS-80 off, make a copy of the vocabulary on a data tape.

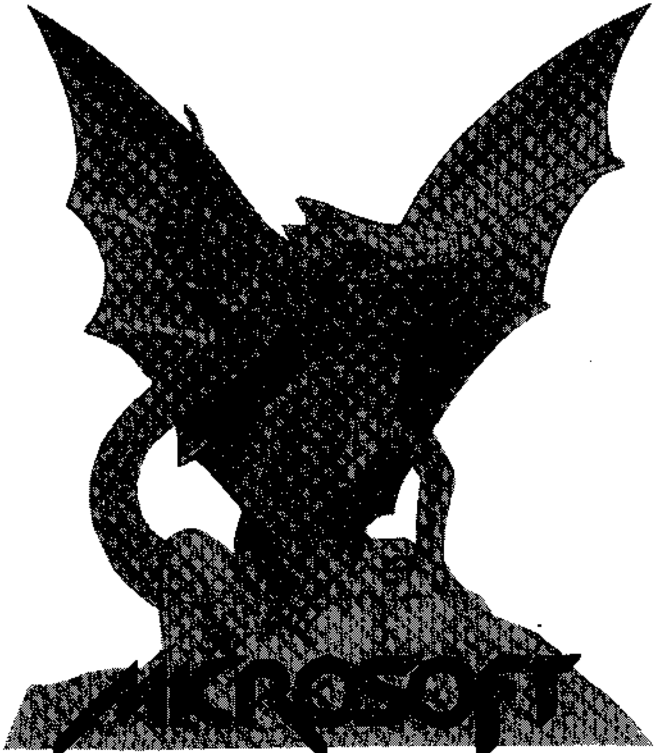
As if all this weren't enough, there's a snappy Pork Pig imitation thrown in for good measure, which signs off the tapes with "TH-TH-TH-THAT'S ALL, FOLKS!"

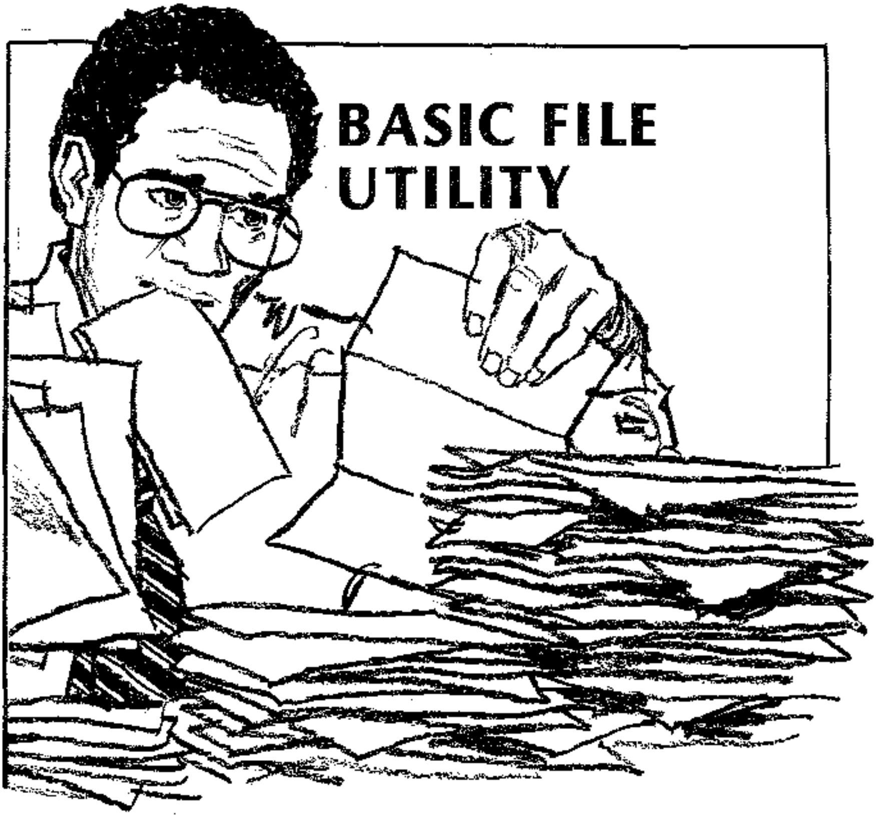
```
60 REM  WRITTEN 3/22/79 BY JOHN R. OLSEN JR.
70 REM                     P.O. BOX 181
80 REM                     NEWBERG, OREGON 97132
90 REM                PHONE:  (503) 538-3031
100 CLEAR(3000):DIMA$(1000):F=0:GOSUB790
110 CLS:PRINT"LONG  SHORT  \   \   \   \    \\   \\   \\     \\
  \   \\  \
120 PRINT"PAUSE PAUSE MAY DAD MOP CALL CLAIM MEET HEAVY OVER PIN
 YOU COLD
130 PRINT"SPACE   0   ";CHR$(94);"@) :9  ";CHR$(93);";A ,12     *
    .E  345   /  ";CHR$(34);"I!#  (   ";CHR$(92);"O(
140 GOSUB750
150 PRINT:PRINT" \\   \\  \    \ \   \\   \    \\   \    \  \  \
   \    \   \
160 PRINT"BOOK TOO CUP ANY BAG CHIP DO BUTTER FAST GO HI JOB KIT
E LET MAN
170 PRINT" $%  'U  678  &   B   C   D  (DT) Q  F   G  H   J   K
   L   M
180 GOSUB750
190 PRINT:PRINT"\    \\ \   \   \   \\  \   \\ \\  \   \   \   \
    \
200 PRINT"NO SING PUT RED SO ASH IT PATH THE VAN WIN YES ZIP AZU
RE
210 PRINT"N   +   P   R   S   > T   =   <   V   W   Y   Z   X
    T+C=CH
220 PRINT@960,"VOICE SYNTHESIZER LAB:";
230 PRINT@704,CHR$(30):PRINT:PRINT:PRINT@704,"-----------------^
  WHICH DO YOU WANT TO DO:  ";CHR$(93);"-----------------";
240 PRINT"1. TRY NEW WORD      2. REPEAT LAST WORD      3. VOCAB
ULARY"
250 PRINT"(PRESS  9  IF THE RED ";CHR$(34);"DEVICE SELECT";CHR$(
34);" LIGHT GLOWS STEADILY)";
260 K$=INKEY$:IFK$=""GOTO260
270 T=100:IFK$="1"GOSUB470:GOSUB760:GOTO230
280 IFK$="2"GOSUB760:GOTO260
290 IFK$="3"GOTO320
300 IFK$="9"PRINT@992,"?";:PRINT@992," ";:GOTO260
310 GOTO260
320 PRINT@768,"1. ADD PREVIOUS WORD    2. SAVE ON TAPE    3. LOA
D FROM TAPE"
```

```
330 PRINT"4. HEAR VOCABULARY    5. DELETE FROM VOCABULARY    9.
RETURN"
340 K$=INKEY$:IFK$=""GOTO340
350 IFK$="1"A$(F)=B$:F=F+1:GOTO230
360 IFK$="2"GOSUB680:GOTO230
370 IFK$="3"GOSUB600:GOTO230
380 IFK$="9"GOTO230
390 IFK$="4"GOTO420
400 IFK$="5"GOSUB440:GOTO230
410 GOTO340
420 T=300:FORW=0TOF-1:H=1:B$=A$(W):GOSUB760:IFK=9W=F-1:K=0
430 NEXTW:GOTO230
440 PRINT@704,CHR$(30):PRINT:PRINT:PRINT@768,"TYPE THE WORD NUMB
ER YOU WANT DELETED (0 TO ESCAPE)";:INPUTX:IFX=0RETURN
450 X=X-1:IF(X<0)+(X>F-1)GOTO440
460 FORW=XTOF-2:A$(W)=A$(W+1):NEXTW:F=F-1:RETURN
470 PRINT@704,CHR$(30):PRINT:PRINT:PRINT@768,"TYPE THE PHONETIC
SPELLING ?  ";CHR$(14);:B$="":FORN=1TO27
480 K$=INKEY$:IFK$=""GOTO480
490 L=ASC(K$):IF(L=8)*(N<2)GOTO480
500 IFL=8THENN=N-2:B$=LEFT$(B$,LEN(B$)-1):GOTO580
510 IFL=13N=27:GOTO590
520 IF(L=10)+(L=91)+(L=9)GOTO480
530 IFL=26L=92
540 IFL=24L=93
550 IFL=25L=94
560 IFL=27L=91
570 B$=B$+CHR$(L)
580 PRINTCHR$(L);
590 NEXTN:PRINTCHR$(15):RETURN
600 D$="LOAD":GOSUB740:IFC$="9"RETURN
610 T=0:PRINT" LOADING...";
620 INPUT#-1,B$:FORW=1TOLEN(B$):Z=ASC(MID$(B$,W,1)):IFZ=97Z=58:G
OTO650
630 IFZ=98Z=44:GOTO650
640 GOTO660
650 B$=LEFT$(B$,W-1)+CHR$(Z)+RIGHT$(B$,LEN(B$)-W)
660 NEXTW:W=F:H=1:GOSUB760:IFLEFT$(B$,8)="<8<8<8<8"RETURN
670 A$(F)=B$:F=F+1:GOTO620
```

```
680 D$="SAVE":GOSUB740:IFC$="9"RETURN
690 PRINT" SAVING...";:FORW=0TOF-1:SV$=A$(W):B$=SV$:T=0:H=1:GOSU
B760:FORX=1TOLEN(SV$):Z=ASC(MID$(SV$,X,1)):IFZ=58Z=97:GOTO720
700 IFZ=44Z=98:GOTO720
710 GOTO730
720 SV$=LEFT$(SV$,X-1)+CHR$(Z)+RIGHT$(SV$,LEN(SV$)-X)
730 NEXTX:PRINT#-1,SV$:NEXTW:PRINT#-1,"<8<8<8<8 <99TS18L0FOUKS":
RETURN
740 PRINT@704,CHR$(30):PRINT:PRINT:PRINT@768,"TO ";D$;", GET CAS
SETTE READY, THEN PRESS ENTER";:C$="":INPUTC$:RETURN
750 FORX=0TO62:PRINT"-";:NEXTX:RETURN
760 PRINT@986,"                                     ";:IFH=1H=0:
PRINT@987,W+1;
770 PRINT@992,"?";B$;" -?";:FORX=0TOT:K$=INKEY$:IFK$="9"K=9
780 NEXTX:RETURN
790 CLS:PRINT@16,"VOICE SYNTHESIZER LABORATORY":PRINT:PRINT:C$="
N":INPUT"DO YOU NEED INSTRUCTIONS";C$:IFLEFT$(C$,1)<>"Y"RETURN
800 PRINT@192,"THIS PROGRAM WILL HELP YOU WRITE PROGRAMS USING T
HE TRS-80"
810 PRINT"VOICE SYNTHESIZER.  YOU WILL NOT NEED TO REFER BACK AN
D FORTH"
820 PRINT"TO YOUR MANUAL FOR THE CORRECT PHONEMES, SINCE THEY WI
LL BE"
830 PRINT"CONVINIENTLY DISPLAYED ON THE VIDEO SCREEN.  YOU CAN T
RY OUT"
840 PRINT"VARIOUS COMBINATIONS UNTIL YOU GET THE ";CHR$(34);"RIG
HT";CHR$(34);" WORDS, USING
850 PRINT"ALL 62 PHONEMES.":PRINT
860 PRINT"YOU CAN LISTEN TO YOUR CREATION AS MANY TIMES AS YOU W
ANT (FOR"
870 PRINT"EVALUATION) WITHOUT HAVING TO TYPE IT ON THE KEYBOARD
EACH"
880 PRINT"TIME.  WHEN YOU ARE SATISFIED YOU HAVE IT PERFECT, YOU
 CAN"
890 PRINT"ADD IT TO THE IN-MEMORY VOCABULARY.":GOSUB1320
900 PRINT"YOU CAN ALSO DELETE WORDS FROM THE MEMORY, IF DESIRED.
  AND"
910 PRINT"BEFORE YOU TURN THE TRS-80 OFF, YOU CAN MAKE A COPY OF
 THE"
920 PRINT"VOCABULARY ON A DATA TAPE, TO BE READ IN NEXT TIME YOU
```

```
 USE THE"
930 PRINT"VOICE SYNTHESIZER LABORATORY.":PRINT
940 PRINT"TO ENTER A WORD, JUST TYPE THE PHONEME SYMBOLS.  WITH
THIS"
950 PRINT"PROGRAM YOU ARE FREE TO USE ANY OF THE 62 SYMBOLS, EVE
N THE"
960 PRINT"ONES THE MANUAL INDICATES CANNOT BE DIRECTLY INPUT FRO
M THE"
970 PRINT"KEYBOARD.  SO THAT THE BACKSPACE KEY REMAINS EFFECTIVE
, YOU"
980 PRINT"WILL HAVE TO USE THE (SHIFT) KEY WITH ANY OF THE FOUR
ARROWS.":GOSUB1320
990 PRINT"TO HEAR YOUR CREATION AGAIN, PRESS  2.  TO WORK WITH T
HE IN-"
1000 PRINT"MEMORY VOCABULARY, PRESS  3.  YOU CAN THEN ADD THE PR
EVIOUS"
1010 PRINT"WORD TO THE VOCABULARY (PRESS 1), STORE THE ENTIRE VO
CABULARY"
1020 PRINT"ON A DATA CASSETTE (PRESS 2), LOAD ADDITIONAL WORDS F
ROM A"
1030 PRINT"DATA CASSETTE (PRESS 3), HEAR THE ENTIRE VOCABULARY (
PRESS 4),"
1040 PRINT"OR DELETE ANY WORD FROM THE VOCABULARY (PRESS 5).":PR
INT
1050 PRINT"WHEN LOADING WORDS FROM A DATA CASSETTE, YOU DO NOT N
EED TO"
1060 PRINT"WORRY THAT EXISTING VOCABULARY WORDS WILL BE LOST.  A
LL THE"
1070 PRINT"WORDS YOU HAVE CAREFULLY CREATED WILL BE SAVED, AND T
HE OTHERS"
1080 PRINT"WILL BE ADDED ONTO THE END OF THEM.":GOSUB1320
1090 PRINT"AS YOU LISTEN TO THE ENTIRE VOCABULARY, YOU WILL NOTI
CE THAT"
1100 PRINT"EACH WORD IS NUMBERED.  THIS IS SO THAT YOU CAN EASIL
Y DELETE"
1110 PRINT"ANY WORD YOU DESIRE, BY REFERRING TO ITS NUMBER.  (BE
 SURE YOU"
1111 PRINT"USE THE CORRECT NUMBER, KEEPING IN MIND THAT THE NUMB
ERS SHIFT"
1112 PRINT"EACH TIME YOU DELETE A WORD).  IF YOU WISH TO STOP TH
E WORDS"
1120 PRINT"WHILE LISTENING TO THE VOCABULARY (TO CHANGE A PHONEM
E, FOR"
1130 PRINT"EXAMPLE) SIMPLY PRESS  9  TO RETURN TO THE COMMAND LI
ST.":PRINT
1150 PRINT"FREQUENTLY, WHEN THE COMPUTER IS FIRST TURNED ON, THE
 VOICE"
1160 PRINT"SYNTHESIZER'S ";CHR$(34);"DEVICE SELECT";CHR$(34);" L
IGHT WILL GLOW CONTINUOUSLY,"
1170 PRINT"AND THE SYNTHESIZER WILL NOT PRODUCE ANY SOUND.  THIS
 IS NOW"
1180 PRINT"SIMPLY A MATTER OF PRESSING  9  AND THE PROBLEM IS SO
LVED.":GOSUB1320
1190 PRINT"ANOTHER NICE FEATURE IS THAT AS A DATA TAPE IS READ I
N, YOU"
1200 PRINT"WILL HEAR ITS CONTENTS SPOKEN ALOUD.  WHEN THE TAPE E
NDS, THE"
1210 PRINT"SYNTHESIZER WILL GIVE A PORKY PIG IMITATION:  TH TH T
H THAT'S"
1220 PRINT"ALL FOLKS!  THIS INDICATES THE END OF DATA, BUT IS NO
T ADDED"
1230 PRINT"TO THE VOCABULARY.":PRINT
1240 PRINT"USING THIS PROGRAM, YOU WILL SPEED UP YOUR CREATION O
F WORDS"
1250 PRINT"AND PHRASES ON THE TRS-80 VOICE SYNTHESIZER.  EVERYTH
ING IS"
1260 PRINT"AT YOUR FINGERTIPS: A COMPLETE LIST OF THE PHONEMES O
N SCREEN"
1270 PRINT"AND THE ABILITY TO SIMPLY USE ALL 62 PHONEMES.  PLUS
THE"
1280 PRINT"ABILITY TO CREATE AND SAVE A VOCABULARY.":GOSUB1320:P
RINT:PRINT
1290 PRINT"FOR YOUR CONVINIENCE, ALL COMMANDS FOR USING THIS PRO
GRAM, ARE"
1300 PRINT"LISTED AT THE BOTTOM OF THE SCREEN AT ALL TIMES, SO Y
OU DO"
1310 PRINT"DO NOT NEED TO MEMORIZE THEM.":GOSUB1320:RETURN
1320 PRINT:INPUT"PRESS  ENTER  TO CONTINUE";A$:CLS:PRINT:RETURN
```


MICROSOFT



BASIC FILE UTILITY

by Clayton Schneider

NOTES ON OPERATION:

The program is self prompting. If you are using only one drive, you may omit lines 20 and 30 and set DR$='0' in line 10.

When converting ASCII files to PENCIL, care should be taken that the target file has no strings longer than 30 or so characters without a space, or PENCIL may malfunction. I.E. 'packed' BASIC files should be expanded with spaces.

PENCIL files of BASIC text lines can be loaded directly by BASIC. The lines must have no imbedded CR's–only one CR at the end of each line. The line numbers may however be out of sequence in the PENCIL file–BASIC will sort them out when it loads them.

Performing conversions on files which are not stored in ASCII format is not recommended (BOMB).

'EXIT TO PENCIL' command assumes of course, that PENCIL/CMD is available on a drive in the system.

NOTES ON THE PROGRAM ITSELF:

Lines 10-40 initialize the system and present the user with a choice for target drive. All subsequent operations will be performed only on that drive. Z$ represents a filename of ZZ00QQ/CES which will be used as a dummy file when executing the conversion to EDTASM utility. If you have any files by this name (????) then you should change the value of Z$.

Lines 50-150 present the program menu. The call to line 920 in 150 produces a flashing cursor on the screen and accepts a single key input returned in IN$. If IN$ passes the value test in line 150, then the value of IN directs the program to the proper branch set in line 160.

Lines 120-250 represent the PURGE portion of the program. Lines 170-200 input the decision for a selective or total purge (of visible, non-protected) files. Line 210 first calls the routine at 760-200 to establish the names of the extant disk files and return them in A$(1) with the number of files in N. Then, if a selective purge was chosen, there is a call to 860-900. This routine moves the flashing cursor to the right of each (non-protected) file shown in the directory and accepts a decision to Kill or Retain each.

If the decision is to retain the file, its' name in (A$(1)) is set to 'null'. Lines 220-230 form a protective 'stop' so that if an incorrect decision was made (or even an unintended decision for a total purge), exemtion of the purge may be aborted. Lines 240-250 do the actual killing of the selected disk files and display what's happening.

Lines 260-340 do the same for the conversion routine that lines 170-230 do for the purge routine above. This portion of the program is common to both the 'TO PENCIL' and 'TO EDTASM' conversions. The correct prompt is chosen by the value of F$ in line 260. Unlike the Purge routine, the conversion routines allow the user NOT to use the directory. In this case only one file name is input and converted (line 290). Line 320 represents the same protective 'stop' as with the purge routine. Line 340 directs the further branch of the program to either PENCIL or EDTASM (the 'GO TO ..10' is a dummy branch, as FC$ will never equal '1' at this point in the program).

Lines 350-420 perform the conversion to PENCIL. The essence of the conversion is to add a 00 byte to the end of the file followed by several spaces (just in case), and to re-name the file to have the extension '/PCL' so it may be read PENCIL. Using NEWDOS, the first can be done by opening the file at the end with the 'E' option and then printing those bytes to it. The routine from 380-400 examines the name of the target file and strips any extension from it (if there is one) and adds the extension /PCL. Then CM$ is built from the old filename and new filename to perform the actual renaming by executing CMDCM$. Lines 350 and 420 construct the loop to convert as many files as were targeted.

Lines 860-900 use the length of the file names found in lines 760-850 above, flash a cursor to the right of the name, and accept a decision to purge or convert. If the file is to be 'passed over' for the particular function, the filename is set to 'null' in A$(1). By comparing the user input to the values S1$ and S2$ which are set in the calling routine, this subfunction can accept a different decision making input for the different functions.

Line 910 is the error trapping routine set in line 760 to catch the possibility of the selected drive being not ready.

Line 920 flashes a graphic character at the current cursor position until an keyboard input is received, at which time it returns to the calling routine.

Lines 930-940 increment the cursor position by the proper spacing found in the directory display.

Lines 430-550 perform the conversion to EDTASM. The essence of the conversion is to add a D3 Hex byte followed by six bytes representing the name of the file to the start of the file and adding a 1A Hex byte to the end. In addition, the five digit line numbers of EDTASM file are offset by 80 Hex from their actual ASCII representations. The conversion is done by writing the new header bytes to a new file named Z$, inputting a line at a time from the old file and modifying the leading 'number' bytes, writing the trailing byte, killing the old file, and finally renaming the new file to the old filename with a new extension ('/SRC'). Lines 460-480 create the six byte filename for the header of the EDTASM file, stripped of its extensions and padded to the right with blanks and write this header to the file Z$. Lines 490-510 input a line at a time of the old file, modify the first five bytes, and write the new line to the file Z$. Line 520 writes the trailing byte, closes the new file, and kills the old file. Line 530 renames the new file to the old filename with the new extension. Lines 420 and 550 construct the loop to convert as many files as are targeted.

Lines 730-750 are the three branches out of the program.

Lines 760-850 call the directory from basic, and then peek at the proper screen locations to construct the names of each file called up by the directory. As the file names are always displayed at the same locations, and never contain spaces, the routine peeks at the first possible location, adds that value to A$(1), increments the location by one, and checks for a space. If a space is found, than the filename has been fully constructed, I is incremented and the 'peek' is incremented to the next possible starting location. If a space is found in the new first location, then all directory entries have been found.

```
1 REM * * * * * * * * * * * * * * *
     *B A S I C  F I L E     *
     *   U T I L I T Y       *
     *   BY CLAYTON SCHNEIDER *
     * * * * * * * * * * * * * *
10 CLS:CLEAR1000:DIMA$(30):CL$=CHR$(29)+CHR$(31):PRINTCHR$(23);
20 PRINT"DRIVE # (0-3) ";
30 GOSUB740:IFIN$<"0"ORIN$>"3"THEN30
40 DR$=IN$:Z$="ZZ00QQ/435:"+DR$
50 CLS:X=15488:I=0:N=0:Y=0:PRINTCHR$(23); :PRINT:PRINT
60 PRINTTAB(4)"BASIC FILE UTILITY":PRINT
70 PRINTTAB(4)"1 = PURGE"
80 PRINTTAB(4)"2 = ASC TO PCL"
```

```
90 PRINTTAB(4)"3 = ASC TO EDTASM"
100 PRINTTAB(4)"4 = EXIT TO BASIC"
110 PRINTTAB(4)"5 = EXIT TO DOS"
120 PRINTTAB(4)"6 = EXIT TO PENCIL"
130 PRINT:PRINT"  PRESS NUMBER DESIRED ";
140 GOSUB740:IFIN$<"1"ORIN$>"6"THEN140
150 FC$=IN$:ONVAL(FC$)GOTO160,250,250,550,560,570
160 PRINTFC$:PRINT:PRINT"SELECTIVE OR TOTAL PURGE (S/T) ";
170 GOSUB740:PG$=IN$
180 IFPG$="T"THEN200
190 IFPG$<>"S"THEN170
200 GOSUB580:IFPG$="S"THENPRINT"PRESS K TO KILL INDICATED FILE,
R TO RETAIN";:S1$="K":S2$="R":GOSUB680
210 PRINT"PRESS <ENTER> TO INITIATE PURGE ON DRIVE ";DR$;",":PRI
NT"PRESS ANY OTHER KEY TO ABORT ";:GOSUB740
220 IFIN$<>CHR$(13)THENGOTO50ELSEPRINTCHR$(27);CL$;
230 FORI=0TON:IFA$(I)<>""THENPRINT"KILLING ";A$(I);:K$=A$(I)+":"
+DR$:KILLK$:PRINTCL$;
240 NEXTI:GOTO50
250 IFFC$="2"THENF$=" TO PENCIL"ELSEF$=" TO EDTASM"
260 CLS:PRINTCHR$(23);"CONVERSION";F$:PRINT:PRINT"USE DIRECTORY
(Y/N) ? ";
270 GOSUB740
280 IFIN$="N"THENCLS:PRINT:INPUT"TARGET FILE ";A$(I):GOTO330
290 IFIN$<>"Y"THEN270
300 GOSUB580:PRINT"PRESS C TO CONVERT FILE";F$;", X TO BYPASS":S
1$="C":S2$="X":GOSUB680
310 PRINT"PRESS <ENTER> TO INITIATE CONVERSION";F$;",":PRINT"PRE
SS ANY OTHER KEY TO ABORT ";:GOSUB740
320 IFIN$<>CHR$(13)THENGOTO50ELSEPRINTCHR$(27);CL$;
330 ONVAL(FC$)GOTO10,340,420
340 FORI=0TON:IFA$(I)=""THEN400
350 PRINT"CONVERTING ";A$(I);F$;:C$=A$(I)+":"+DR$
360 OPEN"E",1,C$:PRINT#1,CHR$(0);"      ";:CLOSE
370 IFINSTR(A$(I),"/")=0THENB$=A$(I)+"/PCL":GOTO390
380 B$=LEFT$(A$(I),INSTR(A$(I),"/")-1)+"/PCL":IFA$(I)=B$THEN400
390 CM$="RENAME "+C$+" TO "+B$:CMDCM$
400 PRINTCL$;
410 NEXTI:GOTO50
```

```
420 FORI=0TON:IFA$(I)=""THEN540
430 PRINT"CONVERTING ";A$(I);F$;:C$=A$(I)+":"+DR$
440 OPEN"I",1,C$:OPEN"O",2,Z$
450 IFINSTR(A$(I),"/")=0THENB$=A$(I):GOTO470
460 B$=LEFT$(A$(I),INSTR(A$(I),"/")-1)
470 PRINT#2,CHR$(211)+LEFT$(B$+"      ",6);
480 IFEOF(1)THEN510
490 LINEINPUT#1,A$:IFLEN(A$)<7THEN480
500 FORS=1TO5:MID$(A$,S,1)=CHR$(ASC(MID$(A$,S,1))+&H80):NEXTS:PR
INT#2,A$:GOTO480
510 PRINT#2,CHR$(&H1A):CLOSE:KILLC$
520 CM$="RENAME "+Z$+" TO "+B$+"/SRC":CMDCM$
530 PRINTCL$;
540 NEXTI:GOTO50
550 CLS:END
560 CLS:CMD"S"
570 CLS:CMD"PENCIL"
580 CLS:PRINTCHR$(23);"PLEASE WAIT":ONERRORGOTO730
590 CM$="DIR :"+DR$:CMDCM$
600 ONERRORGOTO0:PRINT:PRINT"PLEASE WAIT ";
610 A$(I)=""
620 A$=CHR$(PEEK(X+Y)):IFA$=" "THENIFY=0GOTO670ELSEGOTO640
630 A$(I)=A$(I)+A$:Y=Y+1:GOTO620
640 PRINTCHR$(42);:IFPEEK(X+Y+3)=80THENA$(I)=""
650 I=I+1:GOSUB750
660 PRINTCHR$(8);:Y=0:GOTO610
670 N=I-1:I=0:X=0:PRINTCL$;:RETURN
680 IFA$(I)=""THENGOTO720ELSEY=LEN(A$(I))+X:PRINT@130+Y,;:GOSUB7
40
690 IFIN$=S2$THENA$(I)="":GOTO710
700 IFIN$<>S1$THEN680
710 PRINTIN$;
720 I=I+1:GOSUB750:IFI<=NTHEN680ELSEPRINT:PRINTCHR$(31):RETURN
730 PRINTCLS:PRINT"DRIVE NOT CONNECTED OR EMPTY":FORT=0TO1000:NE
XTT:RESUME10
740 IN$=INKEY$:IFIN$=""THENPRINTCHR$(143);:FORT=0TO15:NEXTT:PRIN
TCHR$(8);:FORT=0TO15:NEXTT:GOTO740ELSERETURN
750 IFI=3*INT(I/3)THENX=X+24ELSEX=X+20
760 RETURN
```

What do you do when you finish reading TRS-80 Assembly Language Programming by Bill Barden . . .

and you want to do some serious machine language or assembly language programming?

Buy the

Z80 Software Gourmet Guide & Cookbook

Utility Routines • Stack Operations • Searches and Sorts • Floating Point Routines • Z-80 Instruction Set

New from SCELBI Publications

The index lists over 100 assembly language routines and subroutines that you can use!

$14.95 plus $1.00 shipping

MORE BOOKS FROM TSE

The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144

TRS-80 DISK AND OTHER MYSTERIES

by Harvard C. Pennington

If you are serious enough about disk programming to own RSM-2D or NEWDOS+, then you ought to purchase this book. It explains the organization and features of TRS-DOS 2.0, 2.1, 2.2, NEWDOS, and VTOS, tells you how to use Superzap, RSM-2D, Monitor 3, Debug, Dircheck, and LMOffset, explains the directory track on the disk, file structures (including Electric Pencil files), and even gives detailed procedures for recovery of lost data. While this book is not recommended for the inexperienced user, it is essential for the serious disk programmer. The book is expensive, in 8½ by 11 format with typewritten instead of typeset printing, and only 130 pages, yet the information is worth the price. $19.95
plus $1.00 shipping and handling.

LITTLE BOOK OF BASIC STYLE

John M. Nevison

Ideal reference for BASIC programmer, junior high to research scientist. Indexed, illustrated, 151 pages.

$5.95 plus $1.00 shipping and handling

TRS-80 ASSEMBLY LANGUAGE PROGRAMMING

William Barden, Jr.

Covers Z-80, an introduction to Assembly Language, the Radio Shack Shack Editor/Assembler and T-Bug, and debugging methods; also 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. Well indexed and illustrated.

$3.95 plus $1.00 shipping and handling

TRS-80 INTERFACING

by Jonathan A. Titus.

This valuable reference book for the hardware enthusiast with an understanding of Assembly language programming includes the schematic, printed circuit board artwork, and instructions for building a TRS-80 breadboard, plus 18 experiments to give a solid understanding of interfacing to the TRS-80. $8.95
plus $1.00 for shipping and handling

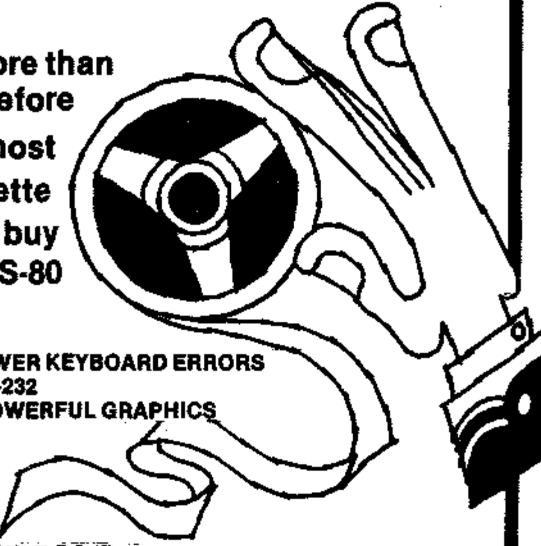
In the beginning there was Level I

THEN THERE WAS LEVEL II

NOW – LEVEL III BASIC

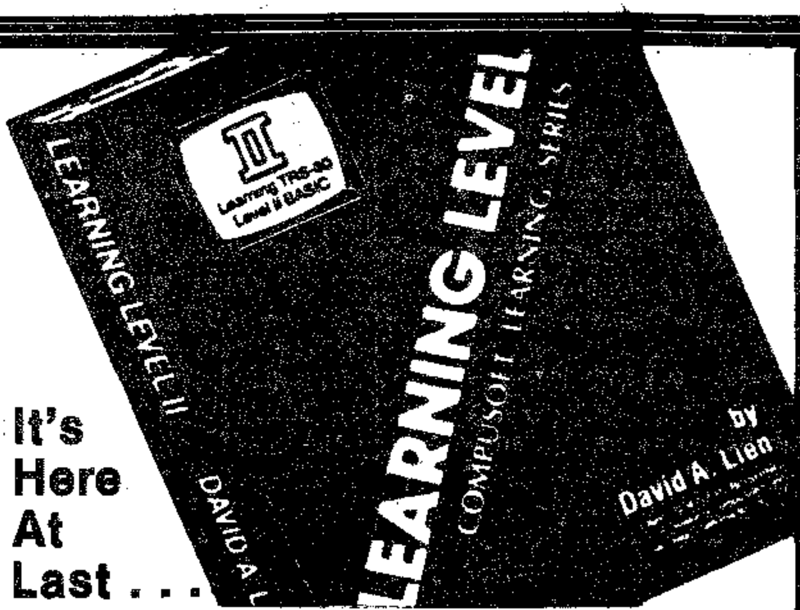
Open the manual and load the cassette. Then get ready to work with the most powerful Basic interpreter you've ever had your hands on . . .Level III Basic for Radio Shack Computers. It loads right on top of the Level II ROM, and in just 5K of space, opens up your capability to new dimensions. For starters, this new cassette-based interpreter gives you most of the commands usually found only in disk BASIC. 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.

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



- **EASIER LOADING, FEWER KEYBOARD ERRORS**
- **BASIC ACCESS TO RS-232**
- **EASIER AND MORE POWERFUL GRAPHICS**
- **10 USR CALLS**
- **ERROR MESSAGES**
- **TIME$**
- **TIMED INKEY**
- **RENUMBER**
- **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.
- **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 includes the Users Manual, a Quick Reference Card, and a preprogrammed cassette tape.

TSE The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144


LEARNING LEVEL II
II
Learning TRS-80
Level II BASIC
LEARNING LEVEL
COMPUSOFT LEARNING SERIES
by
David A. Lien
DAVID A. L

DEVICES (Part 2)

by Lance Micklus

In an earlier issue of PROG/80, I wrote an article about DEVICE and TRSDOS and VTOS. I've gotten so much comment about that article, and so many requests to shed more light on the subject, that I've decided to add this article — a further discussion of DEVICES.

The thing that comes to mind is FORTRAN. FORTRAN is the oldest high level language still in popular use. Micro-computer people today seem to have little idea of what the old computers of 25 years ago were like.

One machine that I used about 12 years ago was the I.B.M. 1620. This machine had about 20K of RAM, a card reader, card punch, and a typewriter (not a Selectric either). At that time, I was an electrical engineering student at Tri-State College. One course in FORTRAN computer programming was required. A few years earlier, I had worked at I.B.M. as a computer operator, mostly on the 1401. I had written a couple of programs at I.B.M., so, I wasn't totally green to computer programming.

At the end of each week, we were required to write a FORTRAN program. After coding the thing on paper, we went down to the key punch room and punched each line of code out on computer cards. The finished program was submitted at the I/O desk. Twice a day, they would take all of the programs submitted and run them on the school's I.B.M. 1620. All of the output was printed on the typewriter. When you went to pick up your output, you got your deck of cards back (assuming the card reader didn't eat them), and the printed output from the typewriter. You then went back to the key punch room and made up new cards to fix the errors, and repeated the process.

Some things should be painfully obvious. First, you really didn't want to write long programs. Second, you wanted to work it all out on paper very carefully before you tried to run your code on the computer. Yet, this is how many computer programs, even in the commercial environment, were written.

FORTRAN is a device driven language. Each device has a number called a LUN (Logical Unit Number). The function of each LUN depended on the machine you were using. The most common arrangements were:

#1 — Console typewriter
#2 — Paper tape
#5 — Card reader
#6 — Line printer
#8 — Card punch

Not all of the LUNS were used. At Tri-State College, the console typewriter was the line printer, and their FORTRAN had been adjusted to cause that to happen. How?

Let's say you want to print some numbers on your line printer. The WRITE verb, in FORTRAN, requires a Logical Unit Number. In this case, it is 6. FORTRAN calls one of its own subroutines called $LUN and passes the line to be printed on to it, along with the operation to be performed. $LUN contains a table with the number of all LUN's defined for that particular computer. Included in the table is a list of what that LUN can and can not do. You can not, for example, read the card punch. In this example, $LUN finds that there is a device #6 on this computer, and device #6 can be used to WRITE. From the table, $LUN pulls out the address of the machine language subroutine and jumps to it. This subroutine was written just for the Tri-State College 1620 computer. The subroutine caused a line of text to be printed on the typewriter, which was our line printer. Had the college later bought a real printer, all they would have to do is change this subroutine and everything would still work just fine.

Suppose I later went and changed the program so that the output went to device #8 instead of #6. Nothing would really change. FORTRAN would still call $LUN, which would locate device #8 in the table. Finding that device #8 can WRITE, it would have jumped to that subroutine instead. The subroutine for device #8 takes a line of TEXT, and causes it to be punched on the card punch.

Now, let's suppose I again changed my mind and decided I wanted to have my output go to device #1. Still no difference — but this time, the output goes to the typewriter. Wait a minute! At Tri-State College, they didn't have a line printer, only a typewriter. Whether I output to device #6 or device #1, I still end up out on the typewriter. It turns out that the subroutine for LUN #1, and the subroutine for LUN #6, are one in the same. LUN #1 will also allow for input from the keyboard of the typewriter. It turns out that since the subroutines for LUN #1 and LUN #6 are the same, YOU CAN INPUT FROM THE LINE PRINTER!

It would not be a good idea to do this, however. One day, you may find a real I.B.M. 1403 chain drive printer sitting there. To make it work, the subroutine for device #6 would be different than device #1 because they are two different machines. There's no way to enter data from a 1403 printer.

Let's stop and think about this for a minute because we have a really beautiful thing here. This means that we can take this FORTRAN compiler and run it on ANY I.B.M. 1620 no matter what kind of hardware is hung on it. The only change we need to make is in the table in $LUN, and we have to supply our own machine language subroutines to support the hardware we have running on our own particular system.

Several years ago, Digital Research developed the CPM operating system for micro-computers. The idea was to develop an operating system which could be used on any 8080, and later Z80, micro-computer. All you had to do was supply the machine language subroutines for your own particular brand of equipment, and make the existence of that subroutine known to CPM. Thus, CPM acts very much like $LUN.

Any program running under CPM does its I/O almost the same way FORTRAN on the 1620 does. It calls CPM, passing to it the identity of the device it wishes to I/O with, the operation to be performed, and the location where the data is to come from or go to. CPM will then check for the existence of such a device in its table, check to be sure the operation is valid for that device, and pass control to the subroutine for that device. By changing the table and the subroutine, we can control what happens in the real world when that device is used. The beautiful thing here is FLEXIBILITY.

One computer running CPM

might have the printer subroutine drive a serial port because the printer is a serial printer. Another computer running CPM might have its printer subroutine drive a parallel port because that's how the printer on that system is connected. The programmer writing under CPM doesn't care what is out there. He is free to change the device anytime he likes.

This kind of flexibility will let you do a lot of things. By patching CPM so the keyboard device is supported by a subroutine which gets its input from a serial port, and patching the video device to output to the same serial board; using a modem, a terminal thousands of miles away could run your computer. (NOTE: In CPM, the keyboard and video are normally the console device). The problem with CPM is that it is harder to learn to use than TRSDOS.

Randy Cook understood the advantages of a device system, and wisely decided to use it for the TRS-80. Devices were given names which began with an asterisk followed by two letters. In addition to this, he made it easy to change the equivalent of the LUN table so the function of a device could be altered, i.e., the printer can really be a disk file spooling the information up, or just a real printer. Likewise, a file can be treated as a device. Sequential files are a good example. Does it really matter if the data you're writing out is really going out to disk, or out to a printer? Usually not. There are special cases.

Consider a PRINT @ in a BASIC program. The new position of the cursor is not sent to the *DO device. Bill Gates at Microsoft pulled a fast one. He just changed the address of the cursor location, by-passing the operating system entirely. What PRINT @ does, in effect, is treat the video monitor as a random access device since we can, at will, cause the data to appear anywhere on the screen.

Disk files are the other special case. They, too, can be randomly accessed. The positioning of the file is not done by transferring data. Rather, it is done by calling routines in DOS which move the position of the file. This isn't any different than what Bill Gates did with PRINT @, except that it is being done by Randy Cook.

The problem with these special cases is that many things in the real world can't do what they do. It's very hard to move the line printer to line 87, then to line 3, and then to line 47.

Suppose we force the issue, and change the operating system so the *DO device is actually going to put anything that was to be put on the screen out to a disk file. It would work, but we would not get the positioning information in a PRINT @ statement. So, if this is something you think you might want to do, don't use any PRINT @'s in your programs.

The same thing is true if we use a device in place of a file. An example might be to take some data that we would normally write out to disk, and have it, instead, printed on the line printer. That's fine, so long as the file is not randomly accessed.

The reason we have several input and output devices on a system is so we can separate different types of data. Obviously, we don't want "PRESS ENTER TO CONTINUE" in the middle of a report. In actual practice, output is formatted and divided up for use by a specific type of physical thing. Certain data is formatted and sent to device *PR because that is the data which we expect the user to print on a line printer. In other words, looking at this from a programmer's point of view, we expect device *PR to really be a printer even though when Joe Blow runs our program, he may have changed things around in his operating system so that device *PR is a PLOTTER.

There is a school of thought that sees this situation as being awkward. But, the device people don't. If the output being sent through device *PR is the wrong kind of data for Joe Blow's plotter, that's Joe's problem. On the other hand, maybe Joe wrote his own clever driver program for device *PR which makes his plotter perfectly happy to operate like a line printer. It turns out that Joe wrote his plotter routine so he plots each letter out on paper in script, using *liquid* ink, so the output looks hand written instead of printer. And, whenever Joe wants, he can reset device *PR back so it sends the data to his Paper Tiger printer.

An even better idea for Joe would be to buy a copy of VTOS. Using the SET command, he can now create a new device called *PP for his plotter. Joe's word processor normally saves the text out to a file. Since the program writes to the file sequentially, would it really matter if the output went to a file or to a device on the system? So, instead of typing TEXT/LST for the output file spec, he types *PP as a filespec. Magic! His nice neat text material is printed on his plotter in script. Or, if he prefers, he could give his word processor the filespec *PR and have his line printer print out his text material. Another thing he can do is give his word processor the file name TEXT/LST. Then from VTOS, he just types COPY TEXT/LST TO *PP, and when that's done, he types COPY TEXT/LST TO *PR. Now, he has his text material in script on the plotter, and then in print on his line printer. And he doesn't have to SET and ROUTE anything.

But Joe really knows how to use a device driven operating system. Instead of using the COPY command twice, as noted above, to go out to his plotter and then to his printer, he can do both at the same time. After spooling his text material to a disk file, Mr. Blow types LINK *PP TO *PR. Next he types COPY TEXT/LST TO *PP. WWWOOOWWW! Now Mr. Blow's text material gets plotted (in script yet) and printed AT THE SAME TIME! It turns out that the text material was really a job resume. After plotting and printing the resume, Mr. Blow realizes that a job resume, made on a plotter in script, isn't going to score any points with the Ajax Tuna Fish Company. Seeing the need for additional copies, on the printer only this time, he types COPY TEXT/LST TO *PR. Now, the job resume is printed on his line printer BUT NOT ON THE PLOTTER.

Why? Because the LINK command Mr. Blow used connects the plotter to the line printer, but not the line printer to the plotter. Thus, the order of the device names are significant. As it stands now, any output to the plotter will also be printed, but any output to the line printer will only be printed.

I freely admit that most people are not doing what I'm doing with my computers. The features I must have, you may be able to do without. And things you must have I can do without. But let me say this: I think that once you start using a device driven system, understand it, and begin to take advantage of all of the things it can do, you may never again want to go back to a dedicated task operating system such as the Model II has.

Editor's Recommendation: This is one of the most valuable programs I own. I have used it for tasks as varied as debugging, disassembling Level II, and storing my copy of Sargon II on disk for fast loading. The only other program I have that compares in value is NEWDOS. — **George Blank**

TSE

The Software Exchange
6 South Street, Box 68, Milford, NH 03055   603-673-5144

RSM 2 (TAPE) $26.95
RSM 2D (DISK) $26.95

RSM 2/2D

by Small Systems Software

This is what you can do:	COMMAND
A - ASCII DUMP:	Display ASCII equivalent of memory block
B - BINARY ARITHMETIC:	Add/subtract in hex and decimal
C - CHECK TAPE:	Check cassette tapes for proper checksum
D - HEX DUMP:	Display Hex equivalent of memory block
E - EDIT:	Enter, examine, or modify memory in hex code
F - FIND 1 BYTE:	Find and display single-byte hex codes
G - GOTO:	Go to and execute program at specified address
H - HUNT 2 BYTES:	Find and display two-byte addresses
I - INITIALIZE/INPUT:	Initialize or input data from port
K - KEYBOARD ECHO:	Type directly to screen or terminal
L - LOAD AND GO:	Load a cassette tape and execute program
M - MOVE:	Move any block of memory to specified location
O - OUTPUT:	Output hex value to specified output port
Q - CHECKSUM:	Compute checksum of specified memory block
R - READ TAPE:	Read cassette tape (header or standard)
S - SYMBOLIC DUMP:	Display memory in ZILOG Z-80 mnemonics
T - TEST MEMORY:	Test memory block and display errors
U - USER:	Allows user to write and execute new commands
V - VERIFY MEMORY:	Compare any two blocks of memory
W - WRITE TAPE:	Write any memory block to cassette tape
X - EXCHANGE:	Interchange any two blocks of memory
Z - ZERO MEMORY:	Write zero or any hex code into memory
@ - BREAKPOINT:	Inserts breakpoint AT specified address
P - PUNCH:	Writes cassette tape in LEVEL-II SYSTEM format
R - READ:	Read SYSTEM tape, display name, start address
L - LOAD:	Loads specified disk sectors into memory block
$ - SAVE:	Saves memory block into specified disk sectors
Y - TRS232:	Set print parameters (baud, etc.) for TRS232
→ - PRINT:	Directs output to printer as well as screen

TYPING TUTOR

by
Roy Groth

You probably have often thought that it would be a lot easier if you didn't have to "hunt and peck" at the typewriter or computer keyboard. If you could only double your typing speed, the job would be done in half the time! And, with increased speed and confidence comes increased accuracy.

Level II, 16K — $19.95

Typing Tutor is a set of programs designed to teach you touch typing — from the basics of learning where the keys are, to practice drills at speeds that would make a Selectric shudder! Each lesson is displayed on the screen, including, at the early stages, a keyboard diagram. You are quizzed and graded, and you progress at your own pace. When you have mastered a lesson, the computer advances to the next, using CAI (computer aided instruction). Progress is fast, painless, and even fun as you keep pushing to break your own record.

So, whether you're going back to school or just want to enter programs from **SoftSide** that much faster, **Typing Tutor** can help.

Order a copy today and be a better typist by next week.

TSE

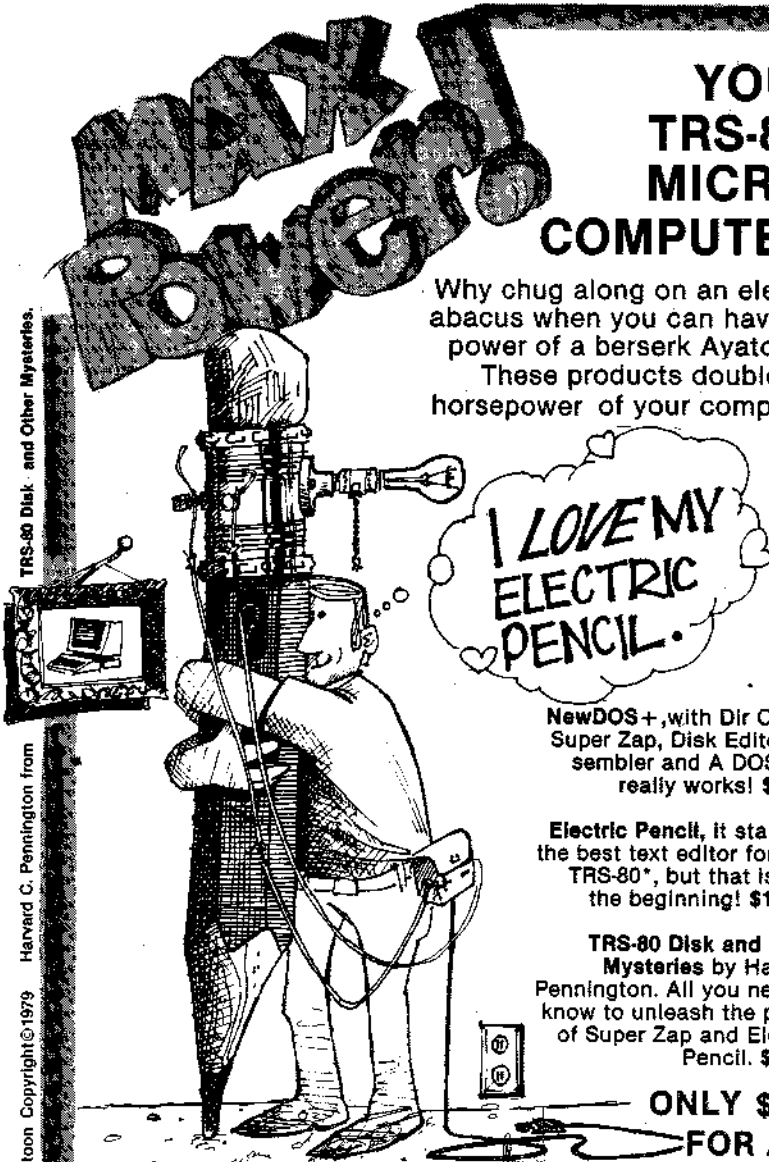
The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144

Programming Problem? Question about a TSE Program?

Try our Hotline...

You can call our programmers direct on Tuesday nights from 7 to 10 PM, (EST)

(603) 673-5144



YOUR TRS-80* MICRO-COMPUTER!

Why chug along on an electric abacus when you can have the power of a berserk Ayatollah! These products double the horsepower of your computer!

NewDOS+, with Dir Check, Super Zap, Disk Editor Assembler and A DOS that really works! **$99.95**

Electric Pencil, it starts as the best text editor for your TRS-80*, but that is only the beginning! **$150.00**

TRS-80 Disk and Other Mysteries by Harvard Pennington. All you need to know to unleash the power of Super Zap and Electric Pencil. **$19.95**

ONLY $269 FOR ALL THREE

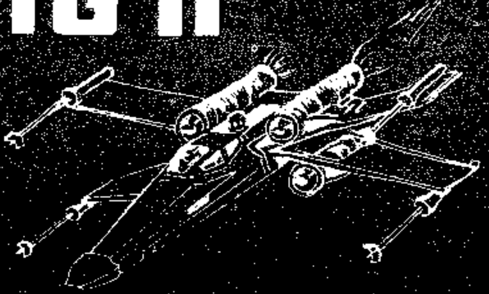
The Software Exchange

6 South Street, Box 68, Milford, NH 03055 603-673-5144

Electric pencil cartoon Copyright©1979 Harvard C. Pennington from **TRS-80 Disk and Other Mysteries.**

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

The Software Exchange

6 South Street, Box 68, Milford, NH 03055 603-673-5144

A NEW USER'S GUIDE TO THE SOURCE

First Time on the Source

by Richard Taylor

Congratulations! You've just received your membership information from THE SOURCE. You have your ID number, and your white three ring binder full of information sent to you by THE SOURCE. What do you do next? Unfortunately if you are like 90% of the people I know, you determine that your RS232 board is not working! (Did you think it was just you?) People are doing strange things to get that little board working. If you are a soldering expert, you can build up all the contacts on the board. However, if you are like me, you will have to live with the fact that every few weeks the contacts will need cleaning. I know one gentleman who insists that the board has to be heated with a hair dryer before it will function. Another person tells me that overheating is the problem and you must relocate your CRT and keep the lid off the RS232 compartment in the Expansion Interface. It seems that a combination of oxidation and a tendency for the board to bow is the main problem. To secure the connection one user suggests bending all of the pins up at a 45° angle. He warns that the board is difficult to reinstall because of the short screws. Some people are convinced that the problem lies on the underside of the plug in question. The plug is symmetrical and contacts the Interface board in the same manner it attempts to contact the RS232 board. If all else fails you will have to open up the bottom of the Interface (voiding your warranty) and remove the plug by taking out the mounting screws that run through the Interface board. This will allow you to remove the plug and clean both sides while bending up all the pins. Building up the solder on the fingers of the Interface board is also recommended. A reliable source tells me that this is the method used by Radio Shack.

I keep my monitor on a shelf above the Interface and I also keep the screws out of the RS232 compartment cover. I find that a light pressure on the contact edge of the board gets mine running. Some people have no problems. If you do, then experiment. Just be aware that the contacts can be worn down easily. The most consistent cure I have heard is to clean the contacts with an ink eraser. This has worked for me, but it is not recommended by Radio Shack.

Everyone agrees that this board was a big design error. I know of two companies coming out with their own versions that will work without the expansion interface. ST80UC and ST80III now incorprate a test for the RS-232-C board as the program initializes. A nice touch. It has saved me some otherwise wasted long distance calls.

Now that you have rebuilt your

Expansion Interface, we can get on to the reason you bought the RS232 in the first place — to access THE SOURCE.

To access THE SOURCE, set your Modem to FULL DUPLEX and ORIGINATE. THE SOURCE recommends an 8 bit word length with parity disabled but a 7 bit word with even parity works fine.

Call the local number THE SOURCE has given you and follow the procedure for either TYMNET or TELENET. If you are using an ST80 series program and signing on through TELENET please answer "Terminal =" with DECWRIII. This will greatly increase your speed for receiving and transmitting. All Nulls are eliminated and that works well with ST80.

THE SOURCE has expanded to two separate systems (10 and 11) and more are planned for the near future. After hooking up to the correct system you will be asked to sign on. Give your account number followed by a space followed by your password. Please note that your password is entered as control characters. They will not print on the screen. This allows you to show off THE SOURCE at your local Radio Shack and still sleep easily knowing hundreds of strangers will not be using your account the following day.

The system will sign you on and tell you what time it is. Then it will just sit there. Don't get jumpy and start hitting ENTER and BREAK. Everything you enter is stored in a buffer and used to answer any subsequent prompts and questions. Depending on how busy the system is, in a few seconds (2 - 45) you will get some introductory information about whats going on such as:

```
YEARNING TO BE A PARTY
ANIMAL? TYPE....DATA TUPPER-
WARE
```

When the announcements stop you will get the system prompt ( ). Typing "DATA TUPPERWARE" will get you just what you think it will. I made up "DATA TUPPERWARE" but you get the idea.

One of the first things you should do on the system is to get acquainted with the MAIL SYSTEM (the backbone of THE SOURCE). One of the best ways to do this is to send me a letter, as I run "SOURCE-80" the TRS-80 Users Group of THE SOURCE. Start by typing MAIL. The system will answer with "Send, read, or scan". Answer "SEND". It then asks "To?" and you answer with the persons account number to whom you wish to send the letter. In this case you would answer TCB575. The next question is "Subject:". You may answer anything you want here up to 122 characters or 132 if you used my DECWRIII hint at sign-on. Hit ENTER to terminate the subject. Now you're on your way. The system says, "Text:". Start typing your message. Just like using ELECTRIC PENCIL right? Wrong!! As you are typing, THE SOURCE is quietly counting up to 122 or 132 characters and then chopping off everything between that character and the next carriage return (ENTER). You must hit ENTER at least every 122 characters. The best habit to develop is that of hitting the ENTER key at the end of every Video line.

Your letter may consist of anything and be of any length. Some of THE SOURCE-80 mail runs 400+ lines. We send entire BASIC programs through the Mail. You can use null lines or anything else with the exception of the @ symbol (more on that in the future). Typing an ESCAPE (control C) ends and sends the letter. For ST80 people, that's your CLEAR key. You may also type ".SEND". All text handling commands begin with a period as the first character of the line so THE SOURCE can recognize them. If things get messed up while you're typing a letter you can always type ".DISPLAY" and your letter will be retyped for you to examine.

After sending the letter you are returned to the "To:" question. Typing "QUIT" will return you to command mode. "QUIT" works almost anywhere you have the opportunity to answer a question. The "BREAK" key for ST80 is also a lifesaver. At the point where you have inadvertently asked UPI for all the stories it sent during 1979, you will be very relieved to know the "BREAK" key will stop it from complying with your request.

A lot of what you do your first time on THE SOURCE will depend on whether or not you have a printer and then upon whether or not you have a terminal program that allows you to use that printer easily. The manual you received from THE SOURCE is just an introduction to the vast amount of information stored in THE SOURCE. I have a book of over 75 pages of printouts taken from THE SOURCE data files and it is not complete! Page nine of your Users Manual shows you an index for SYSINFO. This file on THE SOURCE contains detailed information needed for creating and editing files. It also has the fundamental information needed to program in BASIC and FORTRAN on THE SOURCE. Don't confuse this with DATA SYSDOC. That file lists the manuals available for purchase through your account. The file lists the title and the price and how to order them on THE SOURCE.

Typing "DATA LIBALL" will give you a general index of the data files you might want to read. Each one of these files will lead you to more and more detailed information on the subject.

THE SOURCE is set up for an 80 character line length and a 24 line CRT. This does not work perfectly with the TRS-80. THE SOURCE will pause every 24 lines and wait for you to hit ENTER. At any one of these pauses, you may type "NOCRT" and defeat this pause. This is especially useful if you are printing the information or just loading it into ST80's memory buffer. On the same subject, the MAIL system also pauses and asks "—MORE—". The first time allows you to skip the letter by answering "NO". Thereafter it will ask "—MORE—" every 24 lines. If you are receiving a program from me you will want to defeat this pause. Answer one of the "—MORE—"'s by typing "HARDCOPY" or "NOMORE". The printout will then proceed uninterrupted.

Typing "POST" will put you in the POST SYSTEM. This is a Bulletin Board service and it's organized by category. Only valid catagories such as "TRS-80" and "CLUBS" will be accepted. If you answer the catagory question with "HELP" you will be rewarded with a full list of valid catagories and the number of messages in each. I think you will be amazed at some of the catagories.

At any point in your wanderings around THE SOURCE you may suddenly see the message, "***TCR344 User 22*** following by something like "Hi, want to chat???" This is TCR344 busting in to see if you want to chat. If you do you will have to get out of whatever program you are in and return to COMMAND MODE. Once you get the prompt, type "CHAT TCR344". You are now in direct contact with TCR344. Anything you type appears on his screen and vice-versa. When you finish your thought, hit (ENTER) twice to let him know it's his turn. Chatting is very time consuming but it can be fun. If you type "POST READ CHATTER" you will see messages from other people who wish to chat. If they have left by the time you read their message the system will tell you they're not around.

Notice that line, "POST READ CHATTER"? It's called a COMMAND LINE and it puts you into the program with all input answered. Once the called function is finished you will be returned to COMMAND MODE rather than staying in the program requested. This works with UPI and MAIL also. You may type "MAIL READ" for instance. Typing "MAIL TCB575" places you right into sending a letter to me and when it is sent, returns you to the system prompt.

Typing "ONLINE" gives you a listing of who's on the system at that moment. Unfortunately you can't see who's on the other systems but if you know they are there you can still CHAT with them.

"DATA USEDIR" tells you all about the User Directory. You may add your own name of search through for users with similar interests to your own. No lasts names are allowed. You can search by interest, name, user ID, or State. Please be aware that this search is like any other computer search. TRS-80 will not match TRS 80 or TRS80.

When you send me a letter asking about SOURCE-80 I will send you some of the latest information generated by the group. The information may include entire BASIC programs or programs written in Assembler Source code. The group has developed it's own software to work in conjunction with ST80D or III for disk.This software includes the "BASIC File Utility" program that is being published in this same issue of PROG/80. This program allows you to take an ASCII file generated by ST80 and convert it for instant use with the EDITOR ASSEMBLER. The program will also convert ASCII files for use in the unmodified PENCIL. I say unmodified because you may remember a previous article of mine in which I discussed using ST80D + PENCIL + VTOS 3.0 for total flexibility in editing text and programs generated by ST80. Our group has now developed this same capability for PENCIL in NEWDOS and TRSDOS. I mention only ST80D and ST80III because frankly I am unaware of any other terminal program that comes close to their power. They are fast becoming the standard intelligent terminal software for the TRS-80 and rightly so. Lance Micklus is a member of our group and he takes on all of the difficult questions involving his programs.

That should be plenty of information to get you started on THE SOURCE. The customer service toll-free number is 800-336-3330 and they are very helpful. If you have any questions involving THE SOURCE and the TRS-80 I hope to see them in my mailbox on THE SOURCE - TCB575.

P.S. If you get bored with all the information features you can always type "DATA CHECKERS". Lance Micklus rewrote all of the I/O routines and we now have "THE MEAN CHECKERS MACHINE" running on THE SOURCE. He has kept the screen width to 40 characters so APPLE's and PET's can play too.

Remember the Good Old Days? . . .
You can recapture them!

Level I in Level II

by APPARAT

Level I BASIC interpreter loads in top 4K of any 16K Level II TRS-80. Allows unmodified load, run, and CSAVE of Level I programs — no conversion hassles! Great for teaching beginners, young and old, to program in BASIC.

"System" tape for Level II, 16K — $15.00

The Software Exchange

6 South Street, Box 68, Milford, NH 03055 603-673-5144

THE AUTOMATED DISKETTE DIRECTORY

by George Blank

All you do is insert the diskette in the drive and type the name or number you use for that diskette. The program automatically reads the directory of the diskette, ignoring invisible files like BASIC/CMD and normal DOS files like DISKDUMP/BAS, and stores it in a disk file.

DISPLAY OPTIONS INCLUDE:

$14.95
BASIC
cassette

- **Printout to screen or line printer**
- **Alphabetic sort**
- **Search for single program using INSTR**
  (A search for TRE would find STARTREK and TREES)
- **Index to a single disk**
- **Search and RUN program**

Three programs with instructions for loading on a NEWDOS diskette and instruction manual.

IMPORTANT: This program requires the following minimum system:

At least **TWO** disk drives
NEWDOS by Apparat (Uses CMD"DIR" and OPEN"E")
32K of memory

This Could Be The Handiest Program You Own! By Ken Knecht

Dynamic Data Base

A data base manager is a general purpose file management program that can be set up for different uses. This could be the handiest program you own!

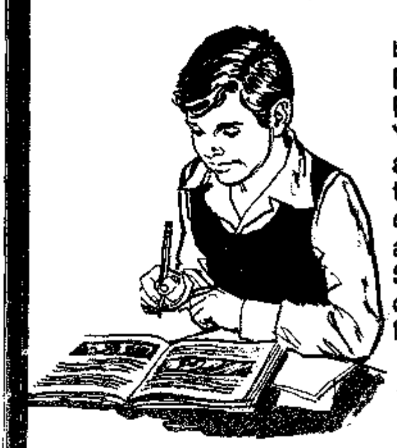
You specify the number of items in the file, whether they are numerical or alphanumeric and how long each item is, and the program creates the file, receives your data, sorts it, searches for information, stores and retrieves from disk, and will even perform arithmetic with data items.

Very easy to learn and use, yet you can use it for mailing lists, hobby collectons, inventory, appointments, accounting, work organizing, radio logbook, billing records, telephone numbers, or a thousand other uses!

Manual and 3 programs in basic on disk for 32K or large systems with one or more drives **only $39.95**

TSE The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144

PENCIL/PAL



by Microcomputer Specialists

Form letter generator for use with the Electric Pencil by Michael Schrayer. You can use a mailing list maintained as a PENCIL file, print letters selectively using selection codes, use custom greetings for each account, and print labels. Requires 32K 1 Disk System, Electric Pencil, and lower case modification kit. (Program 121 lines 5600 Bytes) **$35.**

TSE The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144

PERIODICAL CROSS REFERENCE

by Dave Stambaugh

YOU DON'T HAVE TO BE A LIBRARIAN
Every professional or serious hobbyist who relies on current literature to update his/her knowledge can benefit from this program. Build a file using any subjects or publications you like (18 per file). Allows you to add, delete, edit; search by subject or publication; list on CRT or line printer all or selected entries. SAVE, LOAD with both tape and disk versions.

- **Catalog and classify your library of computer (or other) magazines by individual articles under subject, publication, date, page, and title**
- **List entries by subject or publication to CRT or printer**
- **Maintain data files on cassette or disk (2 versions available)**
- **Edit, delete, add entries at any time**
- **Includes complete instructions**

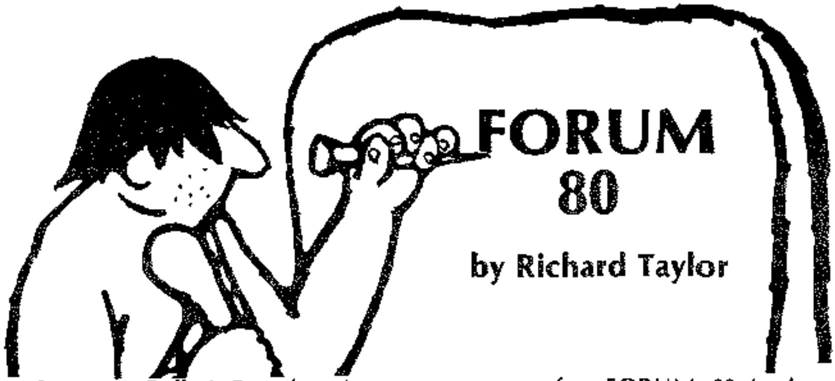
Cassette version: Level II, 16K	**14.95**
Disk Version: Single disk, 32K	**19.95**

(Minimum configurations)

TSE

The Software Exchange

6 South Street, Box 68, Milford, NH 03055 603-673-5144



FORUM 80

by Richard Taylor

Computer Bulletin Board services are springing up all over the country. Some of the most common are called CBBS, ABBS, and FORUM 80. The latter is TRS-80 based and depending on your needs, can offer a wealth of information to the TRS-80 user. If you have a MODEM and only use it to send and receive information with another user, you are missing a lot of fun and information that is just a phone call away.

FORUM 80's operate at 300 Baud using a 7 bit word length for normal transmission and an 8 bit length for graphics. Choose a number from the list below and give it a try.

Set your MODEM for FULL DUPLEX and ORIGINATE and get your terminal program loaded. Dial the number and listen for a high pitched tone. Once you hear the tone, put your phone in the MODEM. If everything is running as it should, you will see "???????". Answer this by hitting the ENTER key (C/R). Next you will see "DO YOU NEED LINEFEEDS". Answer this with "Y" followed by a C/R. The next question is "HOW MANY NULLS (0-50)". Answer "0" C/R. It will then ask you your name and where you are calling from. Once logged on, the system will tell you what functions are available on the system. "HELP" will get you just that, help on what those functions mean. The basic purpose of a FORUM 80 is the exchange of information through questions, answers and announcements. The command "S" will scan through the messages and give you an idea what they are about. Most systems offer "FLAGGED MESSAGE RETRIEVAL". This allows you to hit "F" while the summary is running and "FLAG" the messages you are interested in reading. After the summary is finished, entering "F" will print out all the messages you flagged. If you call the FORUM 80 in Dunstable, Mass. you can see the on-going dialogue between Lance Micklus and his ST80D users. There is some valuable information to be gotten and some fun also. Once you have tried one of these systems your phone bill may never be the same again.

One very exciting feature has just been added to two of the FORUM 80's and I am told many more will be following suit. This feature allows the automatic transmission of programs to ST80D users. It automatically opens and closes the "memory buffer" and prompts you when to save it to disk. I have used this with the FORUM in Dunstable, Mass. and the original FORUM 80 in Kansas City, MO. On both systems it has run error free.

Here is the list of FORUM 80's currently operating in the U.S.:

Orange County, CA	**714-730-1206**
San Francisco, CA	**415-348-2139**
Ventura County, CA	**805-484-9904**
Washington, DC	**202-337-4694**
Ft. Lauderdale, FL	**305-772-4444**
Tampa, FL	**813-223-7688**
Augusta, GA	**803-279-5392**
Chicago, IL	**312-767-0202**
Witchita, KS	**316-746-2078**
Boston, MA	**617-431-1699**
Dunstable, MA	**617-649-7097**
Kansas City, MO	**816-861-7040***
St. Louis, MO	**314-838-7784**
Princeton, NJ	**201-874-6833**
Las Vegas, NV	**702-873-9491**
Memphis, TN	**901-276-8196**
Dallas, TX	**214-288-4859**
Fort Worth, TX	**817-923-0009**
Witchita Falls, TX	**817-855-3916**

*** The original**

Local Systems

Boston, MA **CBBS**	**617-963-8310**
Boston, MA **ABBS**	**617-354-4682**
Anaheim, CA **ABBS**	**714-772-8868**

Other TRS-80 Systems

Olathe, KS

Engineer-80 **913-764-1520**
Programming and information of interest to engineers.

Avionics-80 **913-782-5115**
Information exchange service for those involved in aircraft electronics.

Fairfax, VA

Family Historians Forum **703-978-7561**
The system for anyone interested in genealogy.

Kansas City, MO

Market-80 **816-931-3135**
Message exchange and data for those dealing in commodities.

The list is always growing but each system keeps a current listing of other system phone numbers. Have fun.

INPUT

A COLUMN OF YOUR LETTERS

Dear Sir:

The TRS-80 Model II System Disk File SYS10/SYS has some nice looking edit commands in it. To see them enter (in TRSDOS),

```
LIST SYS10/SYS.EFPN [SLOW,A]
"ENTER"
```

(Look at Records 6 and 7

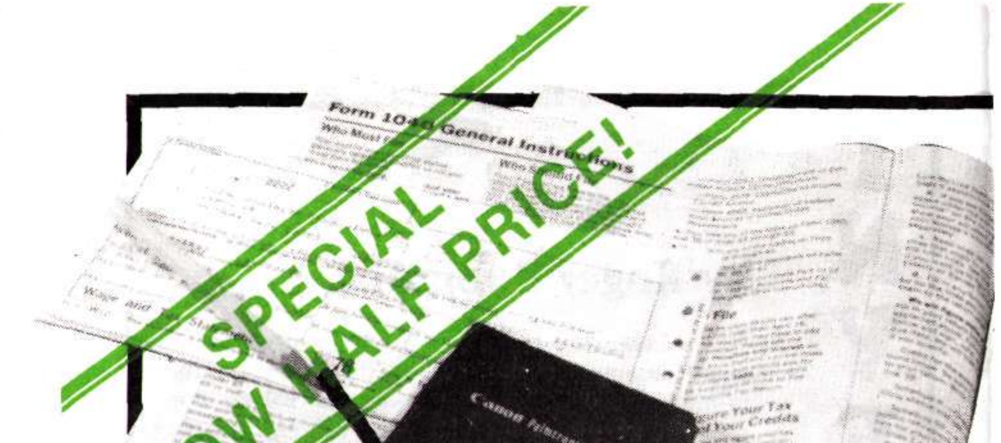
I just hope to find some time to try them out one of these days.

That same password (EFPN, all caps) works for ALL the system files on the Model II I am programming. TRSDOS version 1.1

Happy Programming

D.J. Swindell P.E.

P.S. It only took 2 hours to find with a BASIC program and the 'ON ERROR GOTO' trap statement.


SPECIAL
NOW HALF PRICE!
Form 1040 General Instructions

MICRO TAX II AND III

- Adds Schedule G And Forms 4625, 4726, 6251
- Calculate By Alternate Methods To Get Most Favorable Tax Rate
- Calculate Minimum Tax And Compare With Alternate Minimum Tax

MICRO TAX III

- Adds Schedules C, D, E, SE and Form 2119
- Depreciation Using Straight Line, Declining Balance, And Sum Of Digits
- Taxable Portion of Capital Gains
- Capital Loss Carryover
- Adjusted Itemized Deductions For Alternative Minimum Tax
- Self-Employment Tax

MICRO TAX IV AND V (DISK VERSIONS)

- Includes Form 2210 (Underpayment of Estimated Tax)

PRICES: CASSETTE: Micro Tax I, $25; Micro Tax II, $35; Micro Tax III, $50
DISKETTE:
Micro Tax IV, $100; Micro Tax V, $150

(Add $2 Shipping and Handling)

The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144

STACKING YOUR SUBROUTINES ON THE HEARTBEAT INTERRUPT

by George Blank and John Hartford

How would you like to use your computer for two different tasks at one time? For example, you could print out mailing labels at the same time you are entering new names, or have sound effecis or graphics routines in a game without losing keyboard control!

The secret is to use the heartbeat interrupt to call a machine language subroutine so that the computer alternates between your regular program and the machine language one. In order to do this, you must POKE the address of your subroutine into any convenient spot in memory, and the INSERT the address of your address into the heartbeat interrupt routine.

The computer checks the addresses in the heartbeat routine for a C9 (HEX) return instruction, and if it gets an address instead, goes to that address and loads another address into the program counter. It then executes the routine at the third address.

There are two groups of interrupt addresses. The first group is cycled through so that it is only tested once every forty milliseconds, while the second group is accessed every five milliseconds. For example the real time clock routine is in the first group, and alternates with ? other possible tasks.

The interrupt addresses are at 4500 to 4517 hex. 4200, 4202, 4204, 4208, 420A, 420C, and 420E are the addresses which rotate, while 4210, 4212, 4214, and 4216 are serviced every 5 milliseconds.

Remember that in a Z-80, you must put the least significant byte of each address in the first memory location, and the most significant byte in the next location. Each location holds a two byte address.

GARON'S GOODIES

by James Garon

Repeatable Randomness

Several computers other than the TRS-80 have a capability which ours lacks: the ability to REPEAT a sequence of random numbers. This feature can be quite useful when attempting to debug a program which makes use of the RND function.

Well, we need no longer feel crushed and embarrassed. We too can repeat a random sequence! Try this: Turn off the machine. Turn it on again and enter the following:

```
10    FOR I=1 to 8
20    PRINT I RND (0),
30    NEXT I: PRINT
40    FOR I=16554 to I+2
50    POKE I, 255: NEXT I
60    FOR I=1 to 8
70    PRINT I RND (0),
80    NEXT I
```

As you can see, lines 40 and 50 have reset the random number "seed" to its "Power on" condition without turning off the computer.

Poking values other than 255, 255, 255 into these three locations will give you different sequences which may then be repeated by rePOKEing the same values into the three "seed holders" (locations 16554, 16555 and 16556.

By the way, have you ever wondered what the RANDOM statement actually does? It takes the contents of the **refresh register** and jams it into location 16555 - right in the middle of the three byte seed! The refresh register **must** point to all blocks of memory every few milliseconds, so its contents at any given time are unpredictable.

At any rate, we now see why RANDOM only needs to be executed **once** in a given program.

ZERO SLASH KILLER

by Timothy Smith, Monmouth, Il

Here is an assembly language program to eliminate the slash in the zero on line printer outputs. (Very useful when printouts are to be used by the general public (mailing labels/billings etc.)

Instructions for use:

Program may be loaded or poked in once before needed. Program will function until reset pressed or resetting pokes performed.

Listing

```
    BFF3 F5     Push AF ; Save flags/accumulator
    BFF4 79     LD, A,C; Load character into accumulator
    BFF5 D630  SUB 30H; Subtract value of zero
    BFF7 2002  JR NZ,,BFFB; If not zero go on
    BFF9 OE4F  LD C,4FH; if zero change to letter O
    BFFB F1     POP AF ; recover flags/accumulator
    BFFC
    C38D05     JP 058dH; on to line printer driver address
```

To use: Load beginning of above routine into the lineprinter driver address

4026H = MSB 4027H = LSB (example from above 4026H = F3 4027H = BF)

To reset: Reset driver address to 058D (4026H=8D 4027H=05) Poke and H4026, 141:Poke and H4027,5 or press reset button (all programs stored destroyed)

The program is set up for a 32K system, but can be adapted for 16K or 48K.

The 80-Users of Houston club meets the first Wednesday of each month , at the Bellaire Chamber of Commerce Building, 6900 S. Rice in Bellaire, Texas at 7:30 PM. For more information, call Ben Taylor at (713) 664-5823.

SOFTSIDE ON CASSETTES

If you have less time at the keyboard than you would like to, and have been missing out on some of SoftSide's feature programs, we've got a solution!

NOW AVAILABLE BACK ISSUES W/MAGAZINE $9.95

1979

JANUARY
Round the Horn ● Ten Pin Bowling ● Comput-A-Sketch ● Kiddy Slot

FEBRUARY
● Form 1040 ● Concentration ● Elements Quiz

MARCH Monitor ● Jigsaw Tarot ● Check Balance ● Dive Bomber

APRIL
● Safari ● Rabbits and Foxes ● Personal Finance (Checkfinder) ● Series Circuits ● Don't It Make My Brown Eyes Blue ● Metric/English Converter

MAY
● Dog Star Adventure ● Awari ● Letter Crunch ● Math ● Super Sub

JUNE
● Atlantic Ballon Crossing ● German Word Quiz ● Nim ● Home Appliance Record System ● Entrapment

JULY
● All Star Baseball ● Yahtzee ● Tank ● Spelling Bee ● Collision

AUGUST
● Nuclear Reactor ● Storybook ● Cryptogram ● Shooting Gallery ● Morse Code Tutor

SEPTEMBER
● Engineer ● Treasure Dungeon ● Hangman ● Pyramids

OCTOBER
● Aptitude Test ● Reaction Time ● Battleship ● Westward 1847 ● Tug-O-War

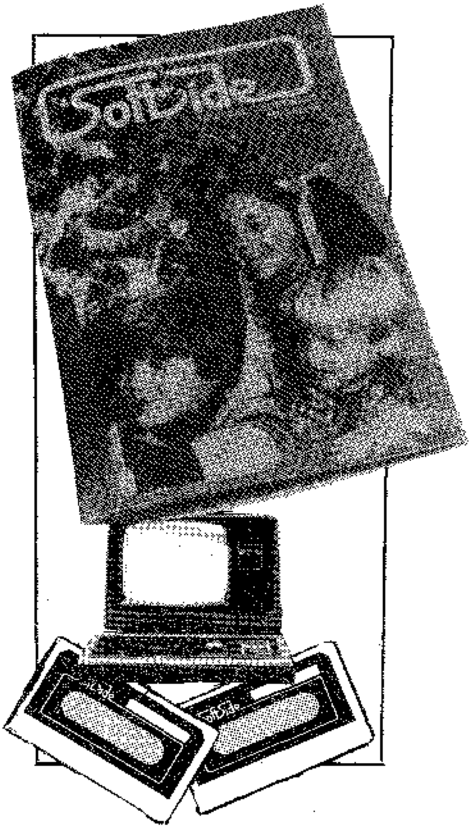
NOVEMBER
● Everest ● Isolate ● Energy Cost ● Picture Puzzles ● Doodle Pad

DECEMBER
● Oil Baron ● Drag Race ● Show and Tell ● Towers of Hanoi ● Slalom

1980

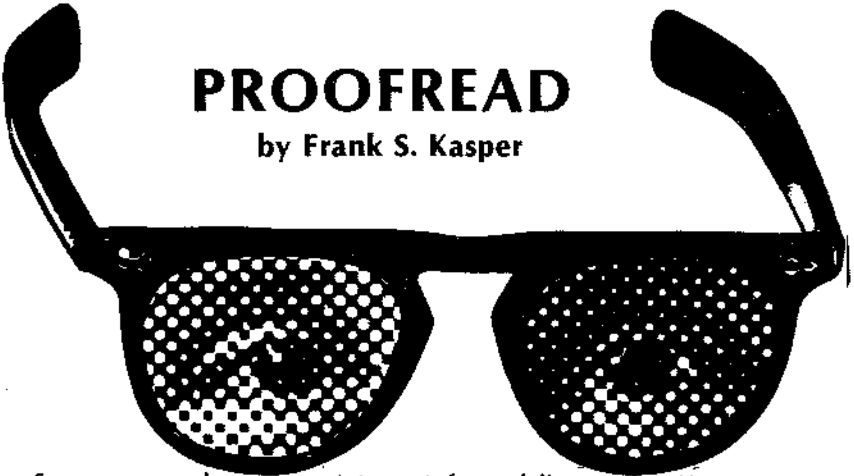
JANUARY
● Moving Maze ● Bomber ● Reflection ● Dante's Inferno ● Duck Hunter

FEBRUARY
Scott Adam's Backgammon ● Parachute ● Deadstick ● Play It Again, Sam ● Deep Six



PROOFREAD

by Frank S. Kasper



Some programs have many statements for each line number. Have you ever had difficulty in getting a program of this kind typed into the computer and up and running? The question that immediately comes to mind is, has the program been copied accurately or does the program have glitches? Don't give up! Try the enclosed program for rapid accurate proofreading.

The program to be proofread must be saved on disk in ASC II form. The printout will be in large 32 character-per-line format in short single statements up to the first colon, semicolon, or ENTER. Each time ENTER key is pressed another statement is displayed.

```
100 REM * PROOFREAD BY F S KASPER                DEC 1979 *
110 REM * THIS PROGRAM HELPS YOU PROOFREAD BASIC PROGRAMS *
        * STORED AS ASCII FILES ON DISKETTE               *
120 CLEAR 500.CLS
130 INPUT"WHAT IS THE FILESPEC OF THE PROGRAM TO BE PROOF-READ";
B$
150 OPEN"I",1,B$
160 IF EOF(1)THEN 260 ELSE LINE INPUT #1,A$
170 S=1
180 S1=INSTR(S,A$,":")
190 S2=INSTR(S,A$,",")
200 IF S1=0 AND S2=0 THEN S1=LEN(A$):S9=1:GOTO230
210 IF S2<>0:IFS2<S1 THEN S1=S2
220 IF S1=0 THEN S1=S2
230 CLS:PRINTCHR$(23):PRINT:PRINTMID$(A$,S,S1-S+1)
240 IF S9=1 THEN S9=0:INPUT ;AS$:GOTO160
250 S=S1+1:INPUT ,AS$.GOTO180
260 CLOSE.END
```

FOR TSE Order Form SOFTWARE ONLY

☐ Accts Rec II Disk 32K	79.95
☐ Adv Pers Fin Disk	24.95
☐ 3 Adventures Disk	39.95
Adventures on tape	14.95
☐ Count ☐ Land ☐	
☐ Voodoo ☐ Pirate's Cove	
☐ Mission Impossible	
☐ Strange Odyssey	
☐ Fun House	
☐ Pyramid Of Doom	
☐ Air Raid	14.95
☐ All Star Baseball	7.95
☐ Amateur Radio Disk	24.95
☐ Amazin' Mazes	7.95
☐ Android Nim	14.95
☐ Appointment Log	9.95
☐ Balloon Race	9.95
☐ Backgammon	7.95
☐ Barricade	14.95
☐ Basic Handbook	14.95 + $1
☐ Basic Statistics	19.95
☐ Basic Styles Handbook	5.95 + $1
☐ Batter Up	5.95
☐ Bee Wary	14.95
☐ Binders	4.95 + $1
☐ Biorhythms	4.95
☐ Breakaway	4.95
☐ Breakaway/Robot	7.95
☐ Bridge Challenger	14.95
☐ Calculator	2.95
☐ Cash Register	9.95
Cassettes	
☐ C-10 10 For	6.50 + $1
☐ C-20 " "	7.50 + $1
☐ Chess Companion	7.95
☐ Concentration	7.95
☐ Cribbage	7.95
☐ Cubes	9.95
☐ Directory Disk	9.95
Diskettes	
☐ Dysan - box of 5	29.95
☐ Verbatim - box of 10	34.95
☐ Diskette Storage Box	5.00 + $1
☐ Dog Star Adventure	9.95
Drill Masters	7.95
☐ German ☐ Italian	
☐ Spanish ☐ Russian	
☐ French ☐ Music Theory	
Electric Pencil	
☐ Tape - 99.00 ☐ Disk - 150.00	
☐ End Zone	7.95
☐ Educator Assistant	9.95
☐ 8080-Z80 Conversion	14.95
☐ Z80 Instruction H'book	4.95 + $1
☐ Electronics Assistant	9.95
☐ Fastgammon	19.95
☐ File Handling	9.95
☐ File Manager 80	49.95

☐ Floppy Armour - box of 5	4.95 + $1
☐ Fortran/Assembler	150.00
☐ Fortran	80.00
☐ Assembler	80.00
☐ General Ledger Disk	79.95
☐ Ham Radio	9.95
☐ Amateur Radio Disk	24.95
☐ Histograph/Scattergram	9.95
☐ Home Financial Man.	9.95
☐ Intro TRS-80 Graphics	7.95 + 1
☐ Inventory II.2 Disk	59.95
☐ Inventory II.3 Disk	79.95
☐ Inventory II Disk	150.00
Inventory S	
☐ Tape - 24.95 ☐ Disk - 39.95	
☐ Journey Center Earth	7.95
☐ Kamikaze	7.95
☐ Keyboard 80	9.95
☐ Kriegspeil	7.95
☐ KVP Extender	24.95
☐ KVP on Disk	29.95
☐ KVP 232	24.95
☐ Learning Level II	15.95 + $1
☐ Level I in Level II	14.95
☐ Level III BASIC	49.95
☐ Lifetwo	14.95
☐ Light Pen	19.95
☐ Machine Lang. Mon. (RSM-2)	26.95
☐ Machine Lang. Mon. (RSM-2D)	29.95
☐ Magic Paper Calculator	14.95
☐ Mail List I Disk	19.95
☐ Mail List II Disk	99.95
☐ Mastermind	7.95
☐ Math Drill	4.95
Mean Checkers Machine	
☐ Tape - 19.95 ☐ Disk - 24.95	
☐ Microchess 1.5	19.95
☐ Microtext Editor	9.95
☐ Moving Signboard	9.95
☐ **NEWDOS**	49.95
☐ **NEWDOS +**	99.95
☐ 9 Games/Preschool Child	9.95
☐ Payroll Disk	59.95
☐ Pentominoes	7.95
Periodical Cross Reference	
☐ Tape - 14.95 ☐ Disk - 19.95	
☐ Personal Finance	9.95
☐ Pork Barrel	9.95
☐ PR Dogfight	7.95
☐ Preflight	20.00
Renumber	7.95
☐ 16K ☐ 32K ☐ 48K	
☐ Three on Disk	25.00
☐ Robot	4.95
☐ 'Round the Horn	9.95
☐ RPN Calculator	9.95
☐ RSM-2	26.95
☐ RSM-2D	29.95

☐ Safari	7.95
☐ Sargon Chess	19.95
☐ Sargon II	29.95
☐ Sargon Handbook	14.95 + $1
☐ Secrets of the Tarot	9.95
☐ Simple Simon	4.95
☐ Sink 'Um	4.95
☐ Slalom	7.95
Small Business Bookkeeping	
☐ Tape - 15.00 ☐ Disk - 25.00	
☐ Journal - 7.00	
☐ Snake Eggs (sound)	14.95
Space Battles	
☐ Tape - 14.95 ☐ Disk - 19.95	
☐ Star Trek III.3	14.95
☐ ST80 Smart Terminal	49.95
☐ ST80D Smarter Disk	79.95
☐ System Copy	9.95
☐ Taipan	9.95
☐ Ten Pin (bowling)	7.95
☐ Text 80 Disk	59.95
☐ 3-D Tic Tac Toe	7.95
☐ Time Bomb	4.95
☐ Time Trek	14.95
☐ Treasure Dungeon	7.95
☐ Treasure Hunt	7.95
☐ Troll's Gold	4.95
☐ TRS-80 Assy. Lan. Pro.	3.95 + $1
☐ Tycoon	7.95
☐ Typing Tutor	19.95
☐ Warfare	7.95
☐ X-Wing Fighter II	9.95
☐ Z-80 Chip Poster	3.99 + $1
☐ Z-80 Gourmet Cookbook	14.95 + $1

The Software Exchange

6 SOUTH STREET MILFORD, NH 03055 Order Number: 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 with 24 - 72 hours. Prices subject to change without notice.

☐ UPS ☐ 1st Class ☐ Charge ☐ COD

Charge card account number

Signature . . . .

Expt. 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. *We are not responsible for typographical errors anywhere in this publication.*

DO NOT FORGET SHIPPING CHARGES

Shipping Charge - Add $1 Disk Charge - Add $5

All C.O.D. and overseas orders must add $5.

SoftSide™

PO Box 68
Milford, NH 03055
"your BASIC software magazine"

Rush me the next 12 issues of **SoftSide**.

☐USA bulk $18 1 yr. ☐$32-2 yrs. ☐CANADA/MEXICO $25 1 yr.
☐USA first class $25 1 yr. ☐OVERSEAS airmail $30 1 yr.
☐APO/OVERSEAS surface $25 1 yr.

Telephone your charge card order! Call our Subscription office Monday through Friday 9:30 to 5:30 (Eastern time) at **603-673-5144**

Credit Card VISA

Exp. Date________ Interbank # (M/C only)________

Signature________

Name ________

Address ________

City________ State ________ Zip________

Please remit in US funds ONLY

Ready to get serious? SUBSCRIBE TO PROG/80 the magazine dedicated to serious programmers...beginners to professionals

SUBSCRIPTION RATES - 6 issues per year

USA
Bulk mail - $15.00
First Class Mail - $21.00
Overseas airmail - $27.00

Canada
Mexico
APO/FPO
Overseas surface mail
$21.00

PROG/80

☐ Check/Money Order enclosed ☐ Master Charge ☐ VISA

SIGNATURE________

ACCOUNT #________

EXP. DATE________ INTER #________

Subscription Order Form

NAME________

ADDRESS________

CITY________ STATE________ ZIP________

Telephone orders accepted for Master Charge or VISA accounts. Call Monday through Friday, 9:30 to 5:30 EST at 603-673-5144

PO BOX 68, MILFORD, NH 03055

PROG/80

A SoftSide Publication
P.O. Box 68, Milford, NH 03055