PROG/80

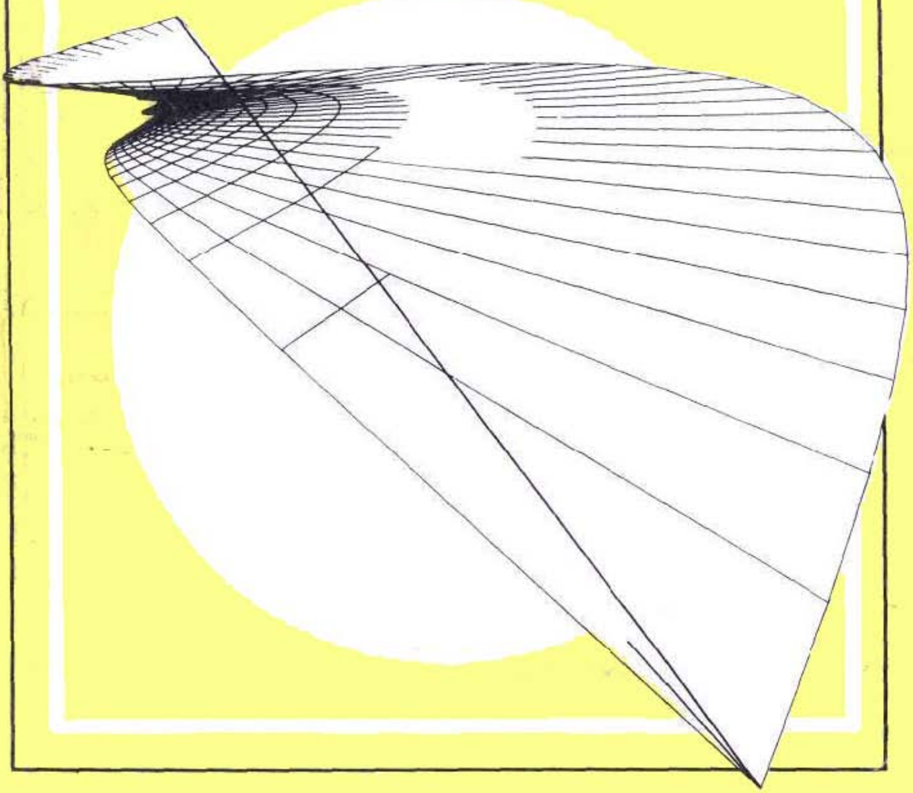
DEDICATED TO THE SERIOUS PROGRAMMER
DECEMBER 1979 $3.00

REVIEWS:

M-80 Morse Code
Model II
Stringy Floppy

$7 Telephone Dialer

Pascal, BASIC, FORTRAN, or Assembler?



PROG/80

DECEMBER 1979 VOL. I, NO. 5

SOFTWARE

HARDWARE

PROG/80

STAFF

Publisher
Roger W. Robitaille, Sr.

Business Manager
Elizabeth J. Robitaille

Managing Editor
George W. Blank

Editor
Lance Micklus

Creative Director
Elaine Cheever

Production
Sharon L. Demmerle
Alison Willis

Marketing Manager
Joseph A. Breton

Subscriptions Manager
Diana Bishop

Customer Service
Bette Keenan

Copy Editor
Freida Day

Accounting
Rita Ellis

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 new WATS line number is:

1-800-258-1790

when installed

It is ONLY to be used for ordering software.

Programmers will not be available on this line.

The WATS line will be open from:

8:00 a.m. to 6:00 p.m.*

ALSO...

During normal business hours, programmers will ONLY be available from:

9-10 a.m. and 5-6 p.m.*

673-5144

New **HOTLINE** Hours:

Tuesday Nights
7-10 p.m.*

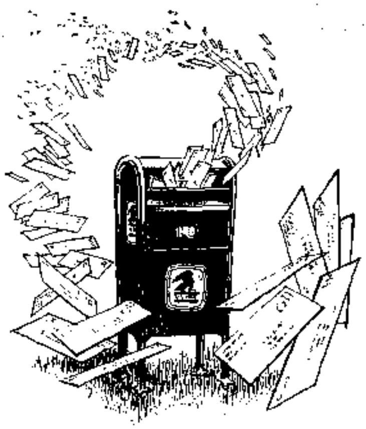
673-5144

Your Cooperation Will Be Most Appreciated

*Eastern Standard Time

Outgoing Mail

by Lance Micklus



This month's Prog/80 features something that we had planned for a long time; a listing for a machine language program. I can't remember now if we ever indicated this anywhere, but Prog/80 is game for machine language, FORTRAN, PASCAL, MMSFORTH, as well as BASIC. As I continue to develop good inside contacts with others, I think you'll find more good meat here in Prog/80 than in any other magazine that caters to programmers, rather than end users. In this magazine we are interested in information and education, rather than end user products - although some of the program listings may be very useful to you.

One of the things I've discovered about machine language programs is that they take up a lot of space. A listing for KVP, for example, would take up three Prog/80 magazines from cover to cover. This is made worse by the fact that we must print them sideways to get all the print on the page. I'm not sure how we'll handle FORTRAN, PASCAL, and FORTH listings. I guess we'll cross that bridge when we get there.

The TRS-80 Model II seems to be on everybody's mind these days. To be honest with you, I haven't been able to devote the time to my Model II that I'd like to right now. But, from a programmer's point of view, I have some opinions on it already. In a nutshell, I don't like it.

Model II TRSDOS is not as good as Model I. Radio Shack seems to have tried to do it themselves, and they just don't have Randy Cook's years of experience in-house. Randy has been working with microcomputers ever since they were invented. He built one of the first microcomputers himself. He had been working on a disk operating system for several years. TRSDOS was the reality of that work.

Randy has worked with several of the Sigma series computers. Their CP-V is the equivalent of TRSDOS. CP-V is considered by many to be the most powerful monitor ever written for a time-sharing computer. It is incredibly fast, in fact, the fastest, and it is DEVICE driven. (TRSDOS does not have to timeshare, but it is DEVICE driven.)

The Model II is a VECTOR driven system. This is a much easier operating system to write, but far less powerful than a DEVICE driven system. A lot of their I/O is a little strange. Their variable length records complicate things considerably, with no real gain in computing power. Microsoft nicely sidestepped the problem by writing all of their files out with a record length of 1. This let them treat files like DEVICES. If Microsoft seems to be having trouble handling files, what about the rest of us?

Model II TRSDOS seems to be hasty and cheap. Corners have been cut in other areas of the machine. One is a packing carton. Many of the Model II's do not work when received because they are shipped by truck and can't stand the bouncing. It is a fact that you can design a shipping carton to send a Model II by truck, and get it there in one piece. Such a carton may cost up to $150, JUST FOR THE CARTON! Radio Shack's answer seems to be to use the cheapest carton, blame the truckers, and then try a more expensive carton until they find one that works. Most companies would go the other way. They would use the expensive carton first, and then move down to the cheaper carton until they find the point where the cheaper carton doesn't work.

The second compromise is in the size of the transformer. There is a school of thought which says that the quality of the design and components is proportional to the size and weight of the power transformer. My Sorcerer has a bigger and heavier transformer than my Model II. But my Sorcerer does not have to power a video monitor and a disk drive.

Being a programmer often is like playing poker. You get dealt a hand, and bet your money. When a new machine comes out by anybody, you have to decide to back it or not. If you back it, and it doesn't sell well, you may end up spending a lot of time and money to develop your products, without a market to sell them in. On the other hand, if you don't back a new system, and it sells well, then you come into the market late, and have to catch up with your competitors.

The fact that I bought a Model II and am developing software for it tells you where I put my money. And I feel comfortable with that decision in spite of all of the Model II's faults, even in spite of Radio Shack's faults. But a word of caution to all of you. Don't assume that each new Radio Shack computer will automatically turn into a pot of gold. Sooner or later, some other company will get their act together, avoid Radio Shack's mistakes, and take control of at least part of the market, if not the bulk of it.

Here's the thought of the month. Zenith now owns Heathkit as of October 1, 1979. Look at the "All - In -One" computer and ask yourself what it would be like with 8 inch disks. Then, there is the large number of Heathkit service centers. Read the description of Heathkit's disk operating system. It's got dynamic file allocation and is DEVICE driven. (I keep harping on that point, don't I?) And what would happen if Zenith sold an improved "All - In - One" computer with 8 inch disks through its distribution system?

Gosh, Hug/80 doesn't sound like a bad name for a Heathkit Prog/80 magazine, does it? ■

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

BASIC STATISTICS

by Steve Reisser

This powerful set of procedures is of use to students, instructors, behavioral and research scientists — anyone applying statistical formulae.

FISHER T-TEST

PEARSON PRODUCT-MOMENT CORRELATION COEFFICIENT

Z-SCORES and STANDARD SCORES

CENTRAL TENDENCY

CHI-SQUARE

SIMPLE ANALYSIS OF VARIANCE

RANDOM NUMBER GENERATOR

RANK-ORDER DATA

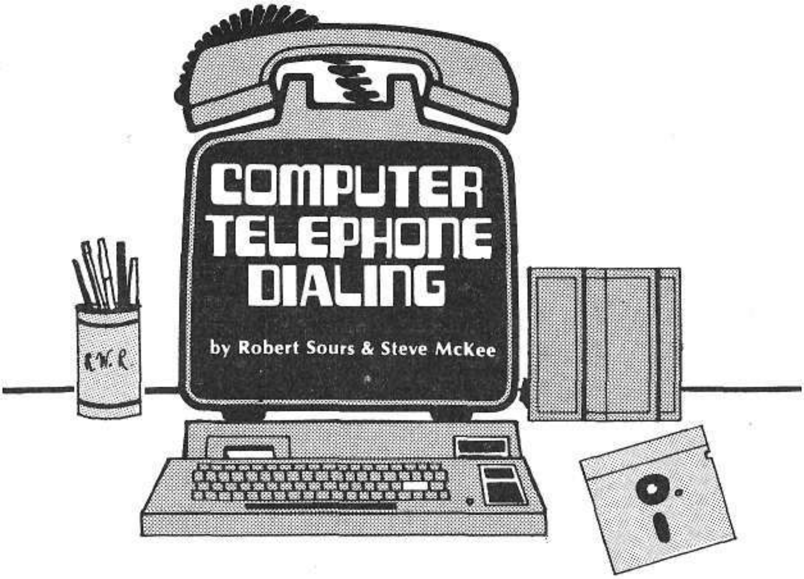
The basic formulae for these major procedures were derived from the textbook, **Elementary Statistics**, by Janet T. Spencer, Benton J. Underwood, Carl P. Duncan, and John W. Cotton. Appleton-Century-Crofts Psychology Series, New York, 1968.

Level II, 16K

$19.95

TSE

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



COMPUTER TELEPHONE DIALING

by Robert Sours & Steve McKee

Computer Telephone Dialing is a program designed to allow the Radio Shack TRS-80 to dial telephone numbers from an in-memory directory. With this program the user builds his own personal telephone directory for those calls made most frequently. The program will allow the user to input and store up to 50 telephone numbers. By entering the proper code number, 1 thru 50, the computer will dial the telephone numbers stored in-memory by the user. The user of this program may also input telephone numbers directly, for those numbers called less frequently. An added feature of this program is that it also times all calls made and stores both the party called and the length of time the user talked. The program has a provision to save the user's previously entered directory on tape for daily use, and the user may also save the calls-made listing on cassette tape for end-of-month auditing of his telephone bill.

When the user begins using this program he is asked to enter first the name of the party to be called; and if the call is local, local toll, or long distance. When the call is local, the code "LO" is entered, which allows the computer to dial 7 digits; when the call is a local toll, the code "LT" is entered, which allows the computer to dial 8 digits (the 8 digits begin 1 plus the called party's 7-digit number). When the call is long distance, the code "LD" is entered, which allows the computer to dial 11 digits, (1 plus the area code plus the called party's 7-digit number).

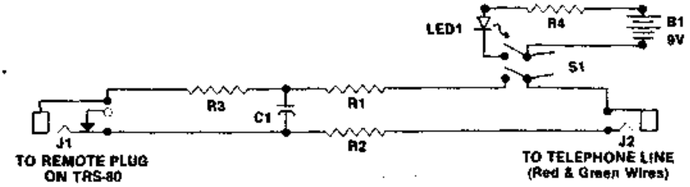
This program dials telephone numbers by the use of the OUT statement in Level II Basic, to turn the TRS-80 relay on and off. The relay must function at a pulse rate of 10 cycles per second with a break-open time of 60%of the closed time during pulsing. The pulse rate has a tolerance of + or -10%.

The TRS-80 is connected to the telephone line by the remote plug.

The remote plug is connected to a telephone interface, which you will need to construct in order to properly match the TRS-80 to the telephone line. The cost to construct the telephone interface is approximately $7.00, using locally available Radio Shack parts. Complete instructions for building the telephone interface and parts listings with Radio Shack part numbers are included in the schematic diagram. You will note that in the diagram a switch is used to open the telephone line from the telephone interface, also a small light is wired with the same switch. The switch is used to disconnect the TRS-80 from the telephone line when not in use. The small LED signals the user that the TRS-80 is connected to the telephone line and is ready for dialing. This switch must be in the OFF position when not using the TRS-80 for dialing. Due to the presence of two diodes across the relay contacts in the TRS-80, failure to turn the switch to the OFF position will result in your telephone ringing only once during an incoming call. Thus the user should note: light ON for dialing, light OFF when not dialing.

Before connecting the TRS-80 to your telephone line it is strongly recommended that you contact your local telephone company in regard to the use of automatic dialing equipment. ■

TRS-80 Telephone Interface Schematic Diagram



PART NUMBER	DESCRIPTION	RADIO SHACK CATALOG NUMBER
R 1	47 ohm ½ watt resistor	271-009
R 2	47 ohm ½ watt resistor	271-009
R 3	470 ohm ½ watt resistor	271-009
R 4	470 ohm ½ watt resistor	271-019
C 1	.1 uf capacitor 50WUDC	272-1069
J 1	Subminiature Phone Jack	274-292
J 2	Shielded Phono Jack	274-346
S 1	Subminiature DPDT Toggleswitch	275-614
B 1	9 U Battery	23-464
LED 1	Flasher LED (red)	276-036
Misc.	9 volt battery clips	270-325
	"mini" utility box	270-230

NOTE: Jacks J 1 and J 2 must be mounted in the plastic sides of the box.

```
5 REM  ****************************************
       **      COMPUTER TELEPHONE DIALING      **
       **             ROBERT SOURS             **
       **                  &                   **
10 REM **             STEVE MCKEE              **
       **           PORTLAND, INDIANA          **
       **               JULY 1979              **
       **                                      **
15 REM **         COPYRIGHT  JULY, 1979        **
       ****************************************
20
25 CLEAR4000:X=0:J=0:DIMN$(101):DIMA$(101):DIMP$(101):DIMQ$(101)
:DIMF$(51):DIMW(51)
30 GOSUB670
35 CLS:INPUT"ENTER TODAY'S DATE <MONTH-DAY-YEAR> NO COMMAS";EE$
40 IFLEN(EE$)>8ORLEN(EE$)<6THEN35
45 GOTO250
50                REM ** DETERMINE LONG DISTANCE OR LOCAL **
55 IFQ$(I)="LO"CLS:GOTO105
60 IFQ$(I)="LT"CLS:PRINT@473,"D I A L I N G":PRINT@538,CHR$(49)+
CHR$(45);+RIGHT$(A$(I),8):B=1:OUT255,7:FORZ=1TO1200:NEXT:GOSUB21
5:GOTO120
65 CLS
70 PRINT@473,"D I A L I N G"
75                REM ** PICKING OUT INDIVIDUAL NUMBERS **
80 PRINT@536,CHR$(49)+CHR$(45),"";A$(I)
85 B=1:OUT255,7:FORZ=1TO1200:NEXT.GOSUB215
90 B=VAL(MID$(A$(I),1,1)):GOSUB215
95 B=VAL(MID$(A$(I),2,1)):GOSUB215
100 B=VAL(MID$(A$(I),3,1)):GOSUB215
105 PRINT@473,"D I A L I N G"
110 PRINT@540,RIGHT$(A$(I),8)
115 OUT255,7:FORZ=1TO1200.NEXT
120 B=VAL(MID$(A$(I),5,1)):GOSUB215
125 B=VAL(MID$(A$(I),6,1)).GOSUB215
130 B=VAL(MID$(A$(I),7,1)):GOSUB215
135 B=VAL(MID$(A$(I),9,1)):GOSUB215
140 B=VAL(MID$(A$(I),10,1)):GOSUB215
145 B=VAL(MID$(A$(I),11,1)):GOSUB215
```

```
150 B=VAL(MID$(A$(I),12,1)):GOSUB215
155 CLS:PRINT@448,"DIALING COMPLETE -- PICK UP RECEIVER"
160 PRINT:PRINT"IF THEY ANSWER HIT THE ";CHR$(34);"A";CHR$(34);"
 KEY."
165 PRINT:PRINT"IF NO ANSWER HIT THE ";CHR$(34);"N";CHR$(34);" K
EY"
170 OUT255,7:FORZ=1TO15000
175 IFINKEY$="A"THEN865
180 IFINKEY$="N"THEN445
185 IFZ>14999THENCLS:PRINTCHR$(23):PRINT"THEY AREN'T GOING TO AN
SWER!":FORZ=1TO1500:NEXT:GOTO445
190 NEXT
195 CLS:PRINT@448,"DIALING COMPLETE -- YOU CALLED"
200 PRINT:PRINTN$(I);"    ";A$(I)
205 PRINT:PRINT"AND TALKED FOR";W(V);"MINUTES":FORZ=1TO2000:NEXT
:GOTO440
210               REM ** DIALING SEQUENCE **
215 IFB=0THENB=10
220 A=B
225 A=A-1:OUT255,7
230 FORZ=0TO30:NEXT
235 OUT255,2:FORZ=0TO18:NEXT
240 IFA=0THEN245ELSE225
245 OUT255,7:FORZ=0TO200:NEXT:B=0:RETURN
250 CLS:PRINT@10,"* *   M E N U   * *":PRINT
255 PRINT" 1 = BUILD DIRECTORY
260 PRINT" 2 = SEE DIRECTORY
265 PRINT" 3 = MAKE CORRECTIONS
270 PRINT" 4 = SAVE DIRECTORY
275 PRINT" 5 = SAVE CALLS MADE
280 PRINT" 6 = LOAD DIRECTORY
285 PRINT" 7 = LOAD CALLS MADE
290 PRINT" 8 = DIAL DIRECT
295 PRINT" 9 = CONTINUE BUILDING DIRECTORY
300 PRINT"10 = LIST CALLS MADE"
305 INPUTK
310 IFK<1ORK>10THEN250
315 ONKGOTO320,440,500,570,595,620,645,770,330,835
320               REM ** CONSTRUCT DIRECTORY **
```

```
325 FORI=1TO50:CLS:GOTO335
330 IFI>49GOTO430ELSED=I-1:FORI=DTO50:CLS
335 PRINT"TO DISCONTINUE BUILDING DIRECTORY TYPE ";CHR$(34);"STO
P",CHR$(34):PRINT"OTHERWISE INPUT THIS CODE NUMBER: ";I;:INPUT"
AND =ENTER=",P$(I)
340 IFP$(I)="STOP"THENP1=I:GOTO430
345 IFVAL(P$(I))<>IPRINT"IMPROPER CODING -- PLEASE TRY AGAIN":GO
TO335
350 PRINT:INPUT"INPUT NAME AND =ENTER=";N$(I)
355 IFLEN(N$(I))<20THENN$(I)=N$(I)+STRING$(20-LEN(N$(I)),"-")
360 IFLEN(N$(I))>20THENN$(I)=LEFT$(N$(I),20)
365 CLS:PRINT"ENTER PROPER CODE:"
370 PRINT:PRINT"LOCAL         - ";CHR$(34);"LO";CHR$(34)
375 PRINT"LOCAL TOLL    - ";CHR$(34);"LT",CHR$(34)
380 PRINT"LONG DISTANCE - ";CHR$(34);"LD";CHR$(34)
385 INPUTQ$(I)
390 IFQ$(I)="LO"ORQ$(I)="LT"ORQ$(I)="LD"THEN400
395 PRINT"IMPROPER CODING -- PLEASE TRY AGAIN":GOTO370
400 PRINT:PRINT"INPUT COMPLETE TELEPHONE NUMBER INCLUDING":INPUT
"AREA CODE AND ALL HYPHENS";A$(I)
405 IFLEN(A$(I))<12PRINT"YOU HAVE LEFT SOMETHING OUT":GOTO400
410 IFLEN(A$(I))>12PRINT"YOU HAVE TOO MANY ENTRIES":GOTO400
415 IFMID$(A$(I),4,1)<>CHR$(45)ORMID$(A$(I),8,1)<>CHR$(45)PRINT"
YOU HAVE LEFT OUT A HYPHEN":GOTO400
420 IFFRE(X$)<100GOTO430
425 NEXTI
430 PRINT"DIRECTORY CLOSED --":INPUT"TO SEE MENU, HIT =ENTER=";
435 GOTO250
440               REM ** LIST DIRECTORY **
445 CLS
450 FORI=1TOP1
455 PRINTP$(I),Q$(I),"   ";N$(I),A$(I)
460 IFI=12ORI=24ORI=36ORI=48THEN915
465 NEXTI
470 PRINT"TO DIAL THE NUMBER ENTER THE CODE NUMBER":PRINT"TO SEE
 MENU TYPE ";CHR$(34);"MENU";CHR$(34);:INPUTP$
475 IFP$="MENU"THEN250
480 FORI=1TOP1
485 IFP$=P$(I)THEN55
```

```
490 NEXT
495 PRINT"IMPROPER CODING":FORZ=1TO1000:NEXT:GOTO445
500              REM ** CORRECTING SEQUENCE **
505 CLS:PRINT"ENTER LINE YOU WANT TO CORRECT BY NAME"
510 INPUTN$
515 FORI=1TOP1:IFN$=N$(I)THEN530
520 NEXTI
525 PRINT"NAME NOT IN DIRECTORY":GOTO555
530 PRINT"ENTER THE CORRECTED INFO: ":PRINT:PRINTCHR$(34);"LO";C
HR$(34);" OR ";CHR$(34);"LT";CHR$(34);" OR ";CHR$(34);"LD";CHR$(
34);" , NAME, TELEPHONE NUMBER"
535 INPUTQ$(I):IFQ$(I)="LO"ORQ$(I)="LD"ORQ$(I)="LT"THENINPUTN$(I
),A$(I)ELSE545
540 IFLEN(A$(I))=12ANDMID$(A$(I),4,1)=CHR$(45)ANDMID$(A$(I),8,1)
=CHR$(45)THEN550
545 PRINT"IMPROPER CODING - PLEASE TRY AGAIN!":GOTO530
550 PRINT"THE DIRECTORY NOW READS: ".PRINTQ$(I);"   ";N$(I),A$(I
)
555 Z=0:INPUT"FOR ANOTHER CORRECTION TYPE 1",Z
560 IFZ=1THEN505
565 GOTO250
570              REM ** SAVING DIRECTORY ONTO TAPE **
575 CLS:INPUT"MAKE PREPARATIONS FOR CASSETTE, WHEN READY HIT =EN
TER=";
580 PRINT"COPYING...."
585 PRINT#-1,P1
590 FORI=1TOP1:PRINT#-1,P$(I),Q$(I),N$(I),A$(I):NEXT
595 CLS.TT=0:PRINT"TO SAVE CALLS MADE TO DATE, INSERT CASSETTE #
2":PRINT"AND =ENTER= THE NUMBER 1 OTHERWISE JUST =ENTER=".INPUTT
T:IFTT<>1THEN250
600 PRINT#-2,EE$,J
605 FORE=1TOJ:PRINT#-2,F$(E):NEXTE
610 PRINT"COMPLETE -- NOTE TAPE LOCATION"
615 INPUT"TO SEE THE MENU, HIT =ENTER=";X.GOTO250
620              REM ** LOADING TAPE TO COMPUTER **
625 CLS:INPUT"WHEN READY, HIT =ENTER=",XX
630 PRINT"INPUTING...
635 INPUT#-1,P1
640 FORI=1TOP1.INPUT#-1,P$(I),Q$(I),N$(I),A$(I):NEXT
```

```
645 CLS:TT=0:PRINT"TO INPUT CALLS MADE TODAY, INSERT CASSETTE #2
":PRINT"AND =ENTER= THE NUMBER 1 OTHERWISE JUST =ENTER=":INPUTTT
:IFTT<>1THEN250
650 INPUT#-2,EE$,J
655 FORE=1TOJ:INPUT#-2,F$(E):NEXTE
660 INPUT"PRESS =ENTER= TO SEE MENU";:GOTO250
665                REM **  INSTRUCTIONS **
670 CLS:PRINT@18,"COMPUTER TELEPHONE DIALING"
675 PRINT"COPYRIGHT  JULY 1979  ROBERT SOURS & STEVE MCKEE  PORT
LAND IN"
680 PRINT:PRINT"*** NOTICE ***
685 PRINT"THIS PROGRAM WAS DESIGNED TO DEMONSTRATE ONE OF THE MA
NY         PRACTICAL USES A MICRO-COMPUTER CAN HAVE FOR BOTH FAMI
LY         AND BUSINESS.
690 PRINT"THE USER BUILDS HIS OWN PERSONAL TELEPHONE DIRECTORY F
OR         THOSE CALLS MADE MOST FREQUENTLY.";
695 PRINT"  BY ENTERING CODE NUMBERS      THE COMPUTER WILL DIAL
THE TELEPHONE NUMBER FOR HIM."
700 PRINT"THE USER MAY ALSO INPUT TELEPHONE NUMBERS DIRECT FOR T
HOSE       NUMBERS CALLED LESS FREQUENTLY."
705 PRINT"THIS PROGRAM ALSO TIMES ALL CALLS MADE AND STORES BOTH
 THE       PARTY CALLED AND THE LENGTH OF TIME THE USER TALKED."
710 PRINT:INPUT"PRESS =ENTER= TO CONTINUE";X$:CLS
715 PRINTCHR$(23):PRINT:PRINT"YOU MUST CHECK WITH YOUR LOCAL  TE
LEPHONE COMPANY BEFORE HOOKINGTHE COMPUTER DIALER TO THEIR    EQ
UIPMENT."
720 PRINT:INPUT"DO YOU WANT DIRECTIONS";Z$:IFZ$<>"YES"THENRETURN
725 CLS:PRINT:PRINT"THE PROGRAM BEGINS BY ASKING TODAY'S DATE.
THIS IS INPUTTED AT THIS TIME AND WHEN YOUR PHONE CALLS MADE ARE
 LISTED,  THAT DATE WILL APPEAR ON THE UPPER RIGHT HAND PORTION
OF THE SCREEN."
730 PRINT:PRINT"A FEW THINGS TO WATCH OUT FOR IS:"
735 PRINT"      1 - AFTER YOU BEGIN BUILDING YOUR DIRECTORY, DO N
OT CALL    FOR THAT FUNCTION AGAIN DURING ANY ONE PROGRAM OPERATI
ON.  IF    YOU DO YOU WILL LOSE WHATEVER DIRECTORY YOU HAVE CONST
RUCTED."
740 PRINT"THERE IS A SPECIFIC FUNCTION CALL FOR CONTINUING WITH
A          DIRECTORY ALREADY STARTED."
745 PRINT:INPUT"PRESS =ENTER= TO CONTINUE";X$:CLS
```

```
750 PRINT:PRINT"     2 - YOU MAY ENTER UP TO, BUT NOT MORE THAN
50 TELEPHONE    NUMBERS IN YOUR DIRECTORY.";
755 PRINT"  THERE IS A PROVISION IN THIS PROGRAMTO KEEP YOU FROM
 ENTERING TOO MANY NUMBERS BY GIVING YOU A      =DIRECTORY CLOSE
D= INPUT WHICH WILL NOT ALLOW YOU TO ENTER ANY  MORE NUMBERS."
760 PRINT:PRINT"     3 - WITH A FULL DIRECTORY OF 50 TELEPHONE N
UMBERS YOU MAY  MAKE 50 TELEPHONE CALLS BEFORE YOU MUST SAVE THE
M."
765 PRINT:PRINT"     4 - AT THE END OF EACH DAY OR AFTER 50 COMP
LETED PHONE     CALLS, YOU WILL HAVE TO SAVE CALLS MADE AND RERU
N THE PROGRAM   THEN REENTER YOUR DIRECTORY.":PRINT:INPUT"PRESS
=ENTER= TO BEGIN";A$:RETURN
770 CLS:IFJ=>50PRINT"YOU CANNOT MAKE ANY MORE CALLS UNTIL":PRINT
"YOU HAVE SAVED THOSE ALREADY MADE":INPUT"PRESS =ENTER= TO GO TO
 MENU";:GOTO250
775 PRINT:PRINT"INPUT COMPLETE TELEPHONE":PRINT"NUMBER: ";:INPUT
A$(I)
780 IFLEN(A$(I))=12ANDMID$(A$(I),4,1)=CHR$(45)ANDMID$(A$(I),8,1)
=CHR$(45)THEN790
785 PRINT:PRINT"PLEASE TRY AGAIN!  CHECK AREA   CODE AND HYPHENS
.":GOTO775
790 CLS:PRINT"ENTER PROPER CODE: "
795 PRINT:PRINT"LOCAL          - ";CHR$(34);"LO";CHR$(34)
800 PRINT"LOCAL TOLL    - ";CHR$(34);"LT";CHR$(34)
805 PRINT"LONG DISTANCE - ";CHR$(34);"LD";CHR$(34)
810 INPUTQ$(I)
815 IFQ$(I)="LO"THENCLS:GOTO105
820 IFQ$(I)="LT"THENCLS:PRINT@473,"D I A L I N G":PRINT@538,CHR$
(49)+CHR$(45)+RIGHT$(A$(I),8):B=1:OUT255,7:FORZ=1TO1200:NEXT:GOS
UB215:GOTO120
825 IFQ$(I)="LD"THENCLS:GOTO70
830 PRINT:PRINT"IMPROPER CODING - PLEASE TRY AGAIN":GOTO795
835             REM ** LIST CALLS MADE **
840 CLS:PRINT@40,"DATE:  ";EE$:FORE=1TOJ
845 PRINTF$(E)
850 IFE=12ORE=24ORE=36ORE=48THEN930
855 NEXTE
860 PRINT:INPUT"HIT =ENTER= TO RETURN TO MENU";X$:GOTO250
865 CLS:PRINTCHR$(23):PRINT@448,"WHEN YOU HANG UP,":PRINT"HIT TH
E "CHR$(34);"P";CHR$(34);" KEY."
```

```
870 PRINT@94,"TIMING: "
875 FORZ=0TO32000
880 W(V)=INT((Z/805+.005)*100)/100
885 PRINT@110,W(V)
890 IFINKEY$="P"THEN900
895 NEXT
900 J=J+1:F$(J)=N$(I)+CHR$(32)+CHR$(32)+A$(I)+CHR$(32)+CHR$(32)+
STR$(W(V))+CHR$(32)+"MINUTES"
905 IFJ>49CLS:PRINT"YOU MUST STORE CALLS MADE BEFORE CONTINUING"
 INPUT"PRESS =ENTER= TO CONTINUE";X$:GOTO250
910 GOTO195
915 PRINT"INPUT -CODE NUMBER- TO DIAL:   ";CHR$(34);"C";CHR$(34)
," TURN PAGE.   ";:INPUTT$
920 IFT$="C"CLS:GOTO465
925 P$=T$:GOTO480
930 PRINT:INPUT"PRESS =ENTER= TO TURN PAGE";X$:CLS:GOTO855
```

LOST -- ONE AUTHOR

We published a piece called CASSETTE CONTROLLER in the July issue of Prog/80, by John D. Eaton. We want to send Mr. Eaton a check but alas, his mailing address is nowhere to be found. If you're out there somewhere, John, please get in touch so we can pay our just debts.

Disk Payroll

Features:

- * Accepts any number of employees
- * Prints check summary on line printer
- * Prints out quarterly summaries

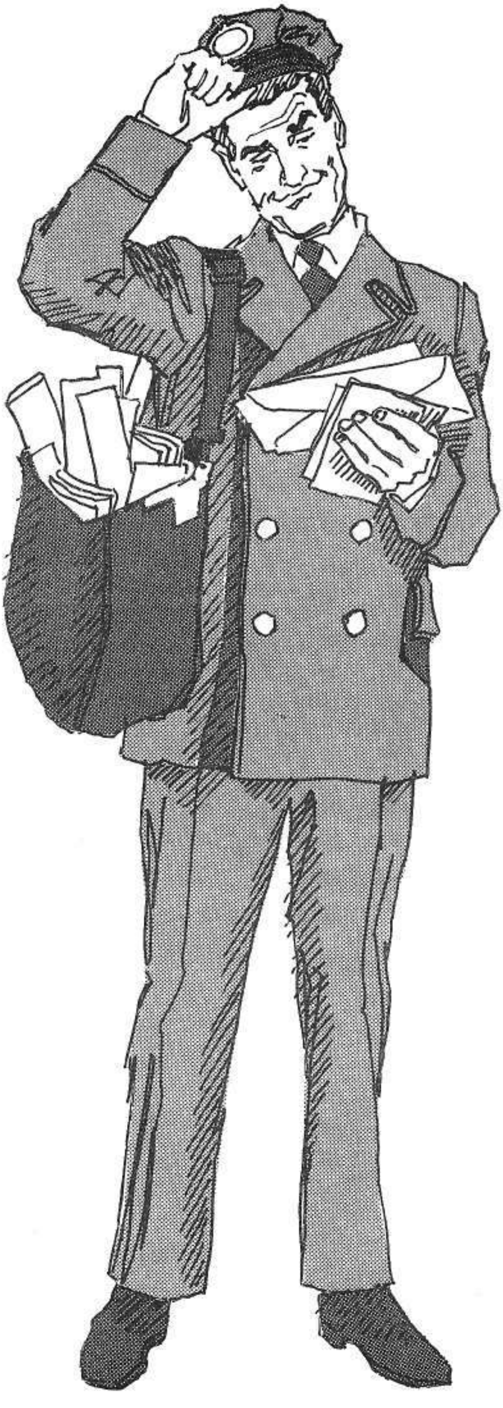
SAMPLE OUTPUT:

```
        EMPLOYEE QUARTER-TO-DATE SUMMARY FOR PAYROLL PAY 2

-------------------------------------------------------------------------
              Employee              Gross   Fed.    FICA   State  City   Other
-------------------------------------------------------------------------
David E. Smith     456-78-9999     500.00  123.81  30.00  20.00  10.00  0.00
Cindy D. Taylor    444-55-6666     500.00  123.81  30.00  20.00  10.00  0.00
Philip G. Jones    454-54-5454     280.00   27.78  16.00  11.20   5.60  0.00
Jane Doe           555-55-5555     240.00   23.81  14.40   9.60   4.80  0.00
J.T. Johnson Jr.   511-11-5111     400.00   34.12  24.00  16.00   8.00  0.00
Mary M. Marks      400-00-0004     260.00   34.12  16.80  11.20   5.60  0.00

        CHECK INFORMATION LISTING FOR PAYROLL PAY 2   (NOVEMBER 14, 1978)
-------------------------------------------------------------------------
  Name             Soc Sec No.  Gross   Fed.    FICA   State  City   Other  Net
-------------------------------------------------------------------------
David E. Smith     456-78-9999  500.00   12.81  30.00  20.00  10.00  0.00   316.19
Cindy D. Taylor    444-55-6666  500.00  123.81  30.00  20.00  10.00  0.00   316.19
Philip G. Jones    454-54-5454  280.00   27.78  16.80  11.20   5.60  0.00   218.62
Jane Doe           555-55-5555  240.00   23.18  14.40   9.60   4.80  0.00   188.02
J.T. Johnson Jr.   511-51-5111  400.00   56.11  24.00  16.00   8.00  0.00   295.89
Mary M. Marks      400-00-0004  280.00   34.12  16.80  11.20   5.60  0.00   212.28

-------------------------------------------------------------------------
  Company                       2200.00 388.81  132.00 88.00  44.00  0.00  1547.19
-------------------------------------------------------------------------

            SUMMARY OF ALL EMPLOYEES (ACTIVE AND TERMINATED)
-------------------------------------------------------------------------
        Gross        Fed        FICA         State         City         Other
-------------------------------------------------------------------------
QTD    2200.00      388.81      132.00        88.00        44.00          0.00
YTD    6515.00     1121.92      402.50       363.00       128.00         40.00
```

Level II, 16K $59.95

The Software Exchange

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



HARDSIDE

**** EFFECTIVE: DECEMBER 1, 1979 ****

	LIST PRICE	OUR PRICE	USED OFFER	USED PRICE
TRS-80 COMPUTERS:				
Level-I 4K	$499	$449	$300	$400
Level-I 16K, w/keypad	$729	$659		
Level-II 4K	$619	$559	$350	$500
Level-II 16K		$669	$450	$625
Level-II 16K, w/keypad	$849	$769	$475	$675
EXPANSION INTERFACE:				
. . . . OK	$299	$269	$165	$230
. . . . 16K	$448	379/403	$245	$340
. . . . 32K	$597	479/537	$325	$440
DISK DRIVES:				
Percom, TFD-100, 40-track		$399		
Percom, dual TFD-100		$795		
Percom, TFD-200, 77-track		$675		
2-drive cable		$29		
4-drive cable		$39		
Radio Shack, -0	$499	$469	$290	$390
Radio Shack, -1, 2, 3	$399	$459	$270	$370
PRINTERS:				
Line Printer III	$1999	$1849		
Centronics 779-2	$1598	$1095	$655	$850
Line Printer II	$999		$545	$750
Centronics 730	$999	$899	$545	$750
Centronics P1 *	$499	$419	$275	$380
Quick Printer II	$219	$197	$125	$165
QP-II Exp. Int. Cable	$20	$18		
* Cable required	$39	$35		
ACCESSORIES:				
Telephone Interface	$199	$179	$100	$150
16K Memory Kit		$99		
16K Memory Kit for E.I.		$95		
RS232-C Interface	$99	$89	$55	$75
TRS-232 Interface		$49		
Data Dubber		$49		

Prices do **not** include shipping

COD orders require
25% cash deposit

Prices subject to change
without notice

HardSide's new retail location has finally opened!
Come See Us!

6 South Street Milford, New Hampshire 03055 (603) 673-5144

***Radio Shack and TRS-80 are trademarks of Tandy Corporation.**

CHOOSING THE RIGHT LANGUAGE

by Lance Micklus

Which computer language is the best language? The answer is, "None of them." In any specific situation, one language may be better than another. The proper choice can make the difference between a well written program and a poor one. There are times when even Level I BASIC is the best language to choose.

One very good reason for selecting BASIC as the language to write in is because it's the only language you know. There's nothing wrong with that, but it might be much better if you had a selection greater than one.

There are several popular languages you can run on the TRS-80. These are BASIC, PASCAL, FORTRAN, and machine language. Under TRS-80 CPM, you can also get COBOL. We will not include COBOL in this discussion, because so few people have it. And then there is MMSFORTH, which does not have the wide acceptance the others have, either. Although that may change, we will also leave FORTH out of our discussion.

PASCAL appears to be a big up-and-coming language, but the TRS-80 version, distributed by FMG, is a disappointment. It has single precision math, and is thus worthless for any serious work with numbers. Second, PASCAL, like other fringe languages, lacks the good I/O handling that BASIC, FORTRAN, and COBOL have. What many people are calling PASCAL, is PASCAL plus the I/O handlers of another language, often FORTRAN. That gives you a combination of PASCAL with FORTRAN format statements. The third problem with TRS-80 PASCAL is that it is still under development. It costs you $150 to buy a prototype. I still haven't gotten my instruction manual and one of the books I ordered with my PASCAL package. So, for the time being, we will have to forget about PASCAL.

That narrows the logical choices for the TRS-80 programmer to BASIC, FORTRAN, and machine language. These are all good choices. Any can be run on a TRS-80 without special operating systems which must be purchased separately. And all can be loaded and run on either tape or disk systems. Thus, most people could put programs written in any of these languages to work immediately.

Many TRS-80 programmers have not had experience with FORTRAN. And that's a shame. FORTRAN is an old language, but it's still nice to work in.

FORTRAN's advantages are numerous: it's fast, the coding technique is protected, it has powerful library functions, it is an excellent number cruncher, its functions are complete subroutines instead of single lines, and it interfaces easily with machine language subroutines. (The last feature eliminates loading machine language programs, entering BASIC, and trying to tie into the machine language programs with USR functions, especially if you're passing several arguments to and from the machine language routines.)

FORTRAN does have its bad points. Unlike BASIC, FORTRAN programs must be written with a text editor program. The code must then be saved on disk. Now the FORTRAN compiler must be called to create a /REL file. If syntax errors are found, you must load the source code back into the text editor, and repeat the process. After eliminating the syntax errors, you must then LINK the /REL file to create machine language code which can be executed.

Debugging your FORTRAN program can be a difficult, time-consuming task. There are no line numbers in the machine language code, so it's difficult to figure out where in memory anything is. There's no way to start and stop the program at will, as there is in BASIC, just by pressing the (BREAK) key, so it's hard to figure out what your variables are doing.

Some of the large timesharing computers have special debug options in their FORTRAN programs which help overcome these problems. But the TRS-80 FORTRAN is minimal, so you're left to your own devices to find problems. Unfortunately, in order to keep the FORTRAN compiler and run time package small, Microsoft had to leave out a number of error checking features which normally would identify problems for you.

A case in point is THE MEAN CHECKERS MACHINE which developed an interesting problem when the forced jump option was added. After coding in all of the additional instructions, and taking care of syntax errors, I finally got a good compile and link. I loaded TMCM into memory. When the checker board appeared on the screen, I entered my move. It was rejected. No move I could make was considered legal by the computer. So I typed UMOV, which lets the computer take its turn first at the beginning of a game. In less than a second, the computer declared itself the winner even though not a single piece had been moved. It prompted me to hit the (ENTER) key and it came back all sad because it lost the game. But wait a minute, nobody had even made a single move!

It took me eight hours to find the problem. One of the arguments being passed to a subroutine was the constant 1. This gets passed to another subroutine, which passes it to a third subroutine. The third routine changes the constant 1 to a 2, but later puts it back to a 1. That is a no-no. It does prove, indeed, what Murphy's Law says about computer programs. "ALL CONSTANTS ARE VARIABLE, AND ALL VARIABLES ARE CONSTANT."

O.K. I can take a hint. I changed the argument to a variable whose value was 1. The variable worked like a constant and the problem was solved.

The worst problem was a subroutine called TITLE. TITLE was written in machine language. All it does is display THE MEAN CHECKERS MACHINE title and copyright on the screen for 8 seconds, and then returns to the calling FORTRAN program. No arguments are passed. TITLE is a very simple little program. Yet, it blew up THE MEAN CHECKERS MACHINE. Why?

The problem had nothing in the world to do with the TITLE subroutine. One of my variables was never set to zero. It turned out that whenever I loaded TMCM, that memory location just happened to be zero'd. Luck! But when I added CALL TITLE to the program, it shifted that variable's location three bytes higher, and it wasn't zero there any more.

Now that we know what a terror FORTRAN can be, let's look at BASIC. We can code and run immediately. This makes it a lot easier to work on and debug a program. We can easily set break points with the STOP command and check variable. In BASIC, all variables are variables, all constants are constant, and all variables start out being zero.

But BASIC does have its weak points. The biggest problem with BASIC is the fact that it's slow.

Before writing THE MEAN CHECKERS MACHINE in FORTRAN, I had a working version written in BASIC. The BASIC version took 10 seconds to generate a playing board, while the FORTRAN version takes about a quarter of a second. Playing at an IQ of 2, the BASIC version took five minutes to find a move. The FORTRAN version takes 40 seconds and does 50% more work.

The final alternative to the TRS-80 programmer is to write in machine language, such as the Radio Shack Editor Assembler and the Microsoft M80 assembler and L80 linker.

There are two major advantages with machine language. One is fast execution speed, and the other is that the memory requirements are usually smaller. No compiler can write code as well as the shrewd and cunning programmer. In addition, the code is kept rather secret because you can't list it very easily. Your source code can be loaded with tons of remarks, and is self-documenting, yet the remarks never appear in the finished product because remarks generate no code in memory.

In machine language, you are actually controlling the machine. In BASIC, you are spending most of your time solving a problem.

EDTASM is the Radio Shack Editor/Assembler which is also available on disk from Apparat. It is an in-memory program, which means the maximum size of the program you're building is limited by the amount of memory in your system. Being both editor and assembler in one, you can code and then compile without reloading. This makes errors easy to fix. Once the syntax errors are fixed, you can assemble the machine language code, save your source code, and run. Because it is an absolute assembler, you know where everything in memory is from the listings you can get on your printer. This makes debugging a lot easier.

The major drawback to EDTASM is that it will not assemble large programs. ST-80D is a prime example. It was written with an earlier version of EDTASM which was modified to run under TRSDOS. Although it has fewer features than the Apparat one, it does have more memory to work with. Even with a 48K system, ST-80D completely fills the memory of the machine. The Apparat EDTASM will not assemble ST-80D because it runs out of memory. This means that little more can be added to ST-80D because there is just no more room to assemble it.

Microsoft's M80 assembler is disk-oriented. It is limited not by memory, but how much room you have on your disk drives. Just on that basis alone, it will assemble a much larger program than EDTASM. However, by writing several machine language programs and linking them together, huge machine language programs can be assembled. This is, in fact, how Level II BASIC, which is 12K, was assembled.

A second powerful feature of M80 is conditional assembly. This lets you write code which can be assembled or not, at will. Using this technique, you can write two routines, one for a Level II machine, and another for a disk TRS-80, and include both in the same code. By changing one true or false condition, you can assemble either version.

But M80 does have its drawbacks. Like FORTRAN, you have to use an editor to code, an assembler to get a /REL file, and then the linker to get machine language code. In other words, there are a lot more steps.

If you're lucky enough to know three computer languages, how do you decide which language to use? The best way I can answer that is to give you some examples, based on my own experience.

STAR TREK III was written in BASIC for a very good reason. At the time I started to write the program, I still had a Level I machine, my EDTASM was on back order, and FORTRAN was still in the works. If I had to do it over again today, I would code it a little differently, but I would still use BASIC.

BASIC executes the program fast enough; time is not a major problem. BASIC has all the computing power needed to make the program run. Of all of the languages, BASIC is the easiest to write code for, and debug. And, from a marketing standpoint, people prefer games written in BASIC rather than some other language which they can't understand.

THE MEAN CHECKERS MACHINE was written in FORTRAN mostly to get an increase in speed. Like BASIC, FORTRAN can handle arrays very nicely. On the other hand, FORTRAN is much easier to code than machine language, especially for a program which is so complex. Believe me, it isn't easy to code a 13.5 K machine language program.

KVP was written in machine language to get control of the machine. There are things you can do in machine language that you can't do any other way. Also, KVP can be in the machine running while the user runs his own program. Thus, there's no problem having the user integrate his program with the KVP program.

MASTERMIND II is an example of a program written in the wrong language. There's nothing wrong with it. It plays an almost unbeatable game, and the display is nice. But, it would best be written in FORTRAN which would still give it the speed, but make the coding job much easier. You have to remember that in machine language, you have to do all of the work. There's no such thing as a PRINT command in machine language. You've got to run the printer yourself. ■

Lifeboat Associates, specialists in microcomputer disk software, is proud to offer the first professional disk-based language and utility package for the Radio Shack TRS-80 computer. Written by Microsoft, creators of Level II BASIC, the package runs on a TRS-80 system with 32K RAM, one or more drives and TRSDOS. The software is supplied on diskettes and consists of:

FORTRAN a true relocatable machine code compiler for ANSI FORTRAN X3.9 (except COMPLEX variables).

MACRO ASSEMBLER a disk-based macro assembler utilizing Zilog mnemonics and producing relocatable code.

LINKING LOADER a link-edit and load FORTRAN and assembler modules for execution.

SUBROUTINE LIBRARY a complete library of subroutines existing as relocatable linkable modules for FORTRAN or assembler programs - e.g., double precision, square root, natural log, transcendentals, etc.

DISK TEXT EDITOR to create and modify FORTRAN and assembler programs as disk files: also can be used as a general purpose text editor for correspondence and other documents.

TRS-80* FORTRAN SOFTWARE PACKAGES

This high-powered professional software package with full documentation is available at the DISCOUNT PRICE OF **$150.** PER COMPUTER SYSTEM

The Macro Assembler, Loader, Editor, and Cross Reference Utilities alone **$80.**

The Fortran Compiler, Loader, Editor, and extensive library of scientific functions alone **$80.**

TSE

The Software Exchange

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

* TRS-80 is a registered trademark of Radio Shack and Tandy Corporation.

S.T.A.D.

Symbolic Trace and Debug

A powerful monitor for the TRS-80 with special ability in Tracing and Debugging. Single Step through machine language programs or set up to three breakpoints, and look at this display format!

```
AF   BC   DE   HL   IX   IY   AF'  BC'  DE'  HL'  SP   PC
0044 0000 C000 B77C 6433 FFFF 0102 0000 4000 3FC0 41FC 4400
4400    LD     A,93
```

All the power of regular monitors as well. Look at these commands:

A FIRST(0) LAST(FFFF)	ASCII dump
A FIRST 0	formatted ASCII dump
B	start of branch table
B VALA	display in decimal
B VALA VALB(0)	hex arithmetic
C	check system tape
D FIRST(0) LAST(FFFF)	dump hex
E FIRST(0)	edit memory
F FIRST LAST VALUE	find byte
G BRKPTS (3 max.)	set breakpoints, continue
H FIRST LAST VALUE	find word
I PORT	read port
K	keyboard echo
L	load system tape
L SECTOR MEMORY COUNT(1)	load from disk
M FIRST LAST BLOCK	move memory
N	display symbol table
N 0	symbol table to tape
N VALUE	define value for symbol table
N FIRST 0	define start symbol table
O PORT VALUE	write to port
P	initialize memory blocks
P ENTRY	write memory blocks and start
P FIRST LAST	define a memory block
Q FIRST LAST	calculate checksum
R	display / modify registers
S FIRST LAST OPTION(0)	disassembler
T COUNT OPTION(6)	trace instructions
U FIRST COUNT OPTION(0)	unformatted tape I/O
V FIRST LAST BLOCK	verify memory
W SECTOR MEMORY COUNT(1)	write to disk
X FIRST LAST BLOCK	exchange memory
Z FIRST LAST VALUE(0)	zero memory

16K, 32K, and 48K versions on one cassette for $24.95
(Use from tape in Level II or as a command file from Disk)

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

SARGON II

DEVICES

by Lance Micklus

Disk users are aware of a DOS command called DEVICE. Almost no one understands what DEVICE means, including Radio Shack, and just about everybody considers it a meaningless command. According to the latest information I have, Radio Shack has now dropped DEVICE, LINK, and ROUTE from the TRSDOS vocabulary, and no longer plans to support these features. Microsoft, I am told, is rewriting their instruction manuals for their FORTRAN package, to eliminate all reference to DEVICES.

DEVICES are one of the most powerful features in the Model I operating system. The idea is not new. Anyone who uses a DEVICE oriented mainframe computer will tell you how hard it is to program on a system which is vector driven, such as the Model II TRSDOS.

What's wrong here is this. First, DEVICES were never implemented in TRSDOS. So, it doesn't work. Second, many TRS-80 users have never used any other type of computer system. So, they don't know what they are missing.

VTOS 3.0, commonly called DOS 3.0, by Randy Cook, the original author of TRSDOS, is what TRSDOS 2.2 would have been, complete with operating DEVICE commands, had Cook and Radio Shack not parted ways. To date, it is the only TRSDOS compatible operating system that supports all of the DEVICE commands. Now that the DOS software is there to support them, the DEVICE commands start to make a lot of sense.

What exactly is a DEVICE?

It is something which does one of two things, or both. First, it will provide, on request, one byte of data. Second, it can be sent one byte of data. Once sent, the data is gone and cannot be changed. The physical thing which is the DEVICE is not known to the operating system - nor does it care. So, the first thing that we learn about DEVICES is that one DEVICE is just like another. The only difference is that some DEVICES are input only, others output only, and some are input and output.

The fact that all DEVICES appear to be the same is what makes them so very powerful, as we shall see.

Since computers usually have several DEVICES, we must distinguish one DEVICE from another. To do this, we give each DEVICE a name. In our case, a DEVICE name is made up of three characters. The first character is **ALWAYS** an asterisk symbol. The next two characters are letters. Thus, we have DEVICES whose names are *KI, *DO, and *PR as standard DEVICES and names, even in Level II BASIC without a disk.

If we output a byte of data to a DEVICE, how does it get to the physical thing we want it to?

The secret is in a machine language program which is called a DRIVER. The DRIVER is the interface between the operating system, and a physical thing. A physical thing might be a keyboard, video monitor, printer, cassette recorder, and so on. These are all physical things we can look at, touch, hold, etc. Or, to put it another way, and this is a poor way to define it, a physical thing is something that can break and will cost you money to fix.

The DRIVER defines the DEVICE. It knows what the DEVICE can and cannot do. You cannot output a byte to a keyboard, or input a byte from the printer.

Another property of the DRIVER is the fact that it may service more than one physical DEVICE. The same DRIVER may be used to service either cassette machine #1 or #2, even though these are two different physical things.

The third thing common to all DEVICES is that each one has its own Data Control Block, or DCB. This is nothing more than several bytes of information that is used by the DRIVER to do whatever it does. All DEVICES, have one thing in common. The very first byte identifies what the DCB is for. Based on that information, the rest of the bytes in the DCB are assigned. Except for disk files, the second and third positions in the DCB are the address of the DRIVER which uses the DCB.

To input or output to a DEVICE is very simple. We simply load the DE register with the address of the DCB for that device. If we are going to output to the DEVICE, we put the byte in register C. Then we call address X'1B'. If the zero flag is NOT set, then there was an error and register A contains the error code. To input from a DEVICE, just load registers DE with the DCB address, and call address X'13'. If the zero flag is NOT set, there was an error and register A contains the error code. If the zero flag was set, then register A contains the byte which was input.

Now that's power. It means that you, as a machine language programmer, do not have to worry about DRIVERS, whether you are dealing with a printer or a file. You treat them all the same way, as DEVICES, and let the operating system do the work. The DCB and the DRIVER can be anywhere in memory.

This is the secret to making KVP work. KVP is nothing more than three DRIVERS - keyboard, video, and printer. All KVP does is change the address of the drivers in the *KI, *DO, and *PR DCB's.

Here's something else to think about. In BASIC, did you ever stop to think that PRINT, LPRINT, and PRINT# (for file I/O) all have the word PRINT in them? BASIC handles all of these statements exactly the same way. The only difference is in which DCB it uses to output the bytes.

And how about this? Take RSM2D and look at Microsoft's M80 assembler program. Please notice that nowhere does it call addresses X'4436', X'4439', and X'443C'. In other words, it appears, at first, that it never reads or writes to files. **BUT IT DOES!** You will find calls to X'1B' and X'13'. That's right. It does not input or output to files. It inputs and outputs to DEVICES.. **AND A FILE CAN BE TREATED AS IF IT WERE A DEVICE.** Therefore, input and output can be processed by anything **-ANYTHING!**

Now, use RSM2D and follow a CALL 33H. First, it loads DE with the address of the *DO DCB. Next, it jumps to address X'1B'. At this location, the DE register is assumed to have the address of the DCB. It now sets up the registers to indicate an output function.

We could put the *DO DCB any place in memory, in which case we bypass the first step by loading DE with the address of our newly moved *DO DCB. Instead, we just call X'1B' directly. The only advantage in keeping the *DO DCB in its present location is it saves the trouble of loading register pair DE with the DCB address.

What does all of this mean to the end user who is running his payroll? It means that a summary report is sent, not to the line printer, but to a DEVICE. Thus, the user, when he runs the program, can decide where he physically wants the output to go. In a typical situation, a payroll program makes only one summary report. Suppose, on some special occasion, you need five reports. Then, from the DEVICE you normally send the output to, ROUTE the output to a file. After running your payroll program, RESET the DEVICE to CLOSE the file. Now, PRINT the file five times.

Here's something else you can do. You can make your own DEVICES up. A case in point was THE MEAN CHECKERS MACHINE. The source code file for all of the machine language subroutines is 32 granules. This part of the program was written for the M80 Microsoft Assembler. To get a listing from the assembler, you would normally specify a /LST file. After assembly, you would just print the /LST file.

There were two problems with this. First, the /LST file produced is more than 67 granules. The second problem is the way the listing appeared on my printer. I kept my listings bound in report folders. Unfortunately, the binding blocked the leftmost edge of the paper from view, so I couldn't always see all of the information on the page. What I needed was the ability to tab each line 10 spaces before anything was printed on a page.

The solution to the first problem was very simple using DOS 3.0. Rather than specifying a /LST file, I specified a DEVICE, which was *PR. That made it possible to print this rather large listing with no problem at all.

The solution to the second problem was a little more work, but was worth the trouble. Because the problem of the left hand margin comes up so often, I simply created my own new DEVICE which I called *LM. The letters are my initials. If your name is George Blank, then I guess you can change the DEVICE to *GB. Why not?

The assembly listing at the end of this article shows you how I wrote the DRIVER program from my new *LM DEVICE. It was written for the Microsoft M80 assembler, but I don't think you'll have any trouble modifying it for EDTASM. If you decide to put this little program together for yourself, remember one thing. Since this is a DRIVER, the COMMAND file you create must have the file extension /DVR and not /CMD. What I did was create this program under file name of SPACE/CMD and then renamed the file SPACE/DVR. Don't let the extension throw you. DRIVERS are nothing more than command files, they just use a different file extension.

To create the DEVICES *LM, you type the following line from VTOS 3.0's prompt of "DOS READY":

```
SET *LM SPACE 10
```

What happens is the following: First, the SET command will load the file SPACE/DVR. Then, it will locate a place in memory where the DCB is to be built. Control will then be turned over to my DRIVER as follows. HL will point to the next ASC character on the input line. In this case, it will be pointing to the "1" in "10". DE will point to the address in memory where I should build the DCB for my DRIVER. My program looks at the number 10, or whatever number was typed in, which is the number of spaces to TAB at the beginning of every line. This number is put into my DRIVER routine. The DRIVER routine is then loaded into high memory, and HIGH$ is reset to protect my DRIVER's memory from use. My program also builds the DCB, specifying that this DEVICE is output only, with the second and third bytes of the DCB being the address of my DRIVER.

Be sure you're clear on this. The program I made is **NOT** the DRIVER. Rather, it is a program which **BUILDS** a DRIVER. The name of the DEVICE is defined by the command you give DOS 3.0. SET *LM means you want to name your DEVICE *LM. If you type *GB, then your DEVICE will be named *GB. So, anybody can use my routine and call it anything they want.

Once control returns back to DOS 3.0, the DEVICE command should now show my new *LM DEVICE in the list.

I'd like to make a subtle, but important point about my DRIVER. It outputs to the *PR DEVICE. You should not think of it as outputting to the printer. It might very well be that the *PR DEVICE is not a printer. So, what the *LM DEVICE really does is output to the *PR DEVICE anything it receives. Anytime a RETURN or FORM FEED character is sent, it will add a specified number of spaces after those characters.

Now, my problem is solved. I have the M80 Assembler list to DEVICE *LM, but leave DEVICE *PR tied to my printer. M80 then lists on my printer, and every line is moved to the right 10 spaces. When I put the listing in my report folder, a large left hand margin is under the binding, and I can read all of the listing.

Thanks to the DEVICE operation of VTOS 3.0, you can use my little program as a model, and go mad creating your own DEVICES. How about making one that converts all lower case letters to capital letters, but places a graphic symbol next to each character that was changed? That way you know which letters were lower case in the original output without an upper-lower case capability.

Here's another interesting idea. How about building a device which encrypts any data sent to it, and another device which decrypts. This would give you a system where you could read and write to a disk file in scrambled form, yet your BASIC program, communicated through your DEVICES, always is seeing real information (i.e. decrypted).

Now, here's the source code listing for SPACE/DVR which you can use as a model to make all of this possible with VTOS 3.0. ■

```
SPACER MACRO-80 3.20   05-MAR-79        PAGE    1-1
COPYRIGHT 1979 BY LANCE MICKLUS, INC.

 0000'
                  00100  .COMMENT $
                  00150       THIS PROGRAM CREATES A NEW SYSTEM DEVICE WHICH
                  00200  WILL OUTPUT TAB SPACES TO THE *PR DEVICE AFTER
                  00250  EVERY CARRIAGE RETURN OR FORM FEED CHARACTER.
                  00300  THUS, THE PRINTED MATERIAL IS SHIFTED RIGHT TO
                  00350  INCREASE THE LEFT HAND MARGIN.
                  00400  $
                  00450  ;
 000D             00500  CR      EQU    0DH
 003B             00550  PRTOUT  EQU    0003BH         ;OUTPUT BYTE TO LP
 1E5A             00600  FIGURE  EQU    01E5AH         ;ASC TO BINARY CONV.
 4049             00650  HIGH$   EQU    04049H         ;TOP OF USEABLE DOS MEM
 402D             00700  DOS     EQU    0402DH         ;RETURN TO DOS SYSTEM
 0019             00750  OFFSET  EQU    ENDMEM-SPC     ;OFF SET CONSTANT
                  00800  ;
 0000'            00850  ASEG
                  00900  ORG     7200H                 ;BEGINNING OF PROGRAM
                  00950  ;
```

```
SPACER MACRO-80 3.20   05-MAR-79        PAGE    1-2
COPYRIGHT 1979 BY LANCE MICKLUS, INC.

                               01000   INIT:
 7200    D5                    01050           PUSH    DE              ;DE POINTS TO THE DCB
 7201    7E                    01100           LD      A,(HL)          ;HL POINTS TO NEXT
 7202    FE 0D                 01150           CP      0DH             ;PART OF ASC
 7204    CA 402D               01200           JP      Z,DOS           ;INSTRUCTION WHICH
 7207    CD 1E5A               01250           CALL    FIGURE          ;CALLED PROGRAM.
 720A    7B                    01300           LD      A,E             ;THIS IS THE NUMBER
 720B    B7                    01350           OR      A               ;OF SPACES TO TAB.
 720C    CA 402D               01400           JP      Z,DOS           ;IF ZERO DO NOTHING.
 720F    32 7241               01450           LD      (TAB+1),A
 7212    2A 4049               01500           LD      HL,(HIGH$)      ;COMPUTE POSITION IN
 7215    11 001E               01550           LD      DE,OFFSET+5     ;MEMORY AND PROTECT
 7218    ED 52                 01600           SBC     HL,DE           ;IT FROM THE SYSTEM.
 721A    22 4049               01650           LD      (HIGH$),HL
 721D    23                    01700           INC     HL
 721E    E5                    01750           PUSH    HL              ;SAVE THIS, IT'S THE
 721F    EB                    01800           EX      DE,HL           ;VECTOR ADDRESS.
 7220    21 7234               01850           LD      HL,SPC          ;MOVE THE ROUTINE
 7223    01 001A               01900           LD      BC,OFFSET+1     ;TO THAT ADDRESS.
 7226    ED B0                 01950           LDIR                    ;VECTOR ADDR > DE
```

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

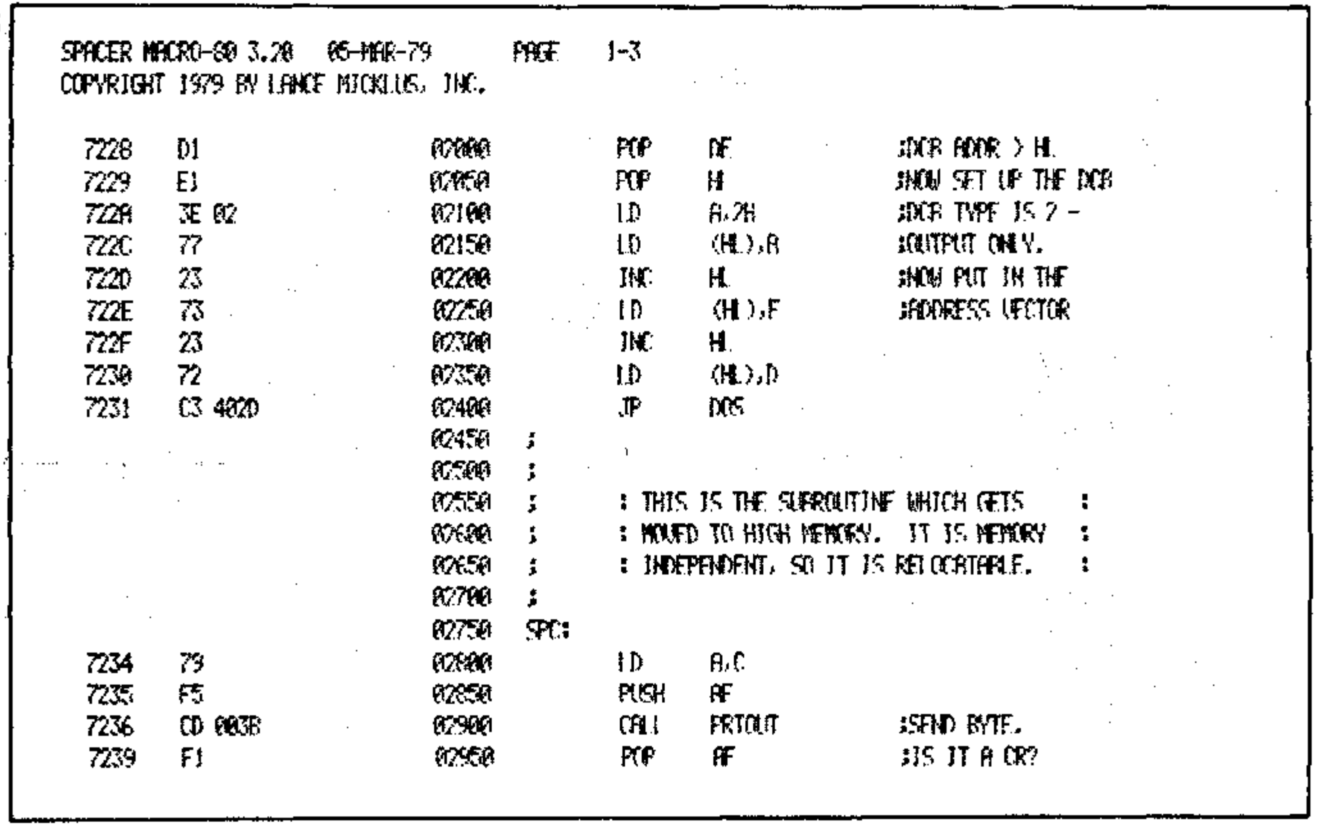
The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144

RSM 2 (TAPE) $26.95
RSM 2D (DISK) $29.95

RSM 2/2D

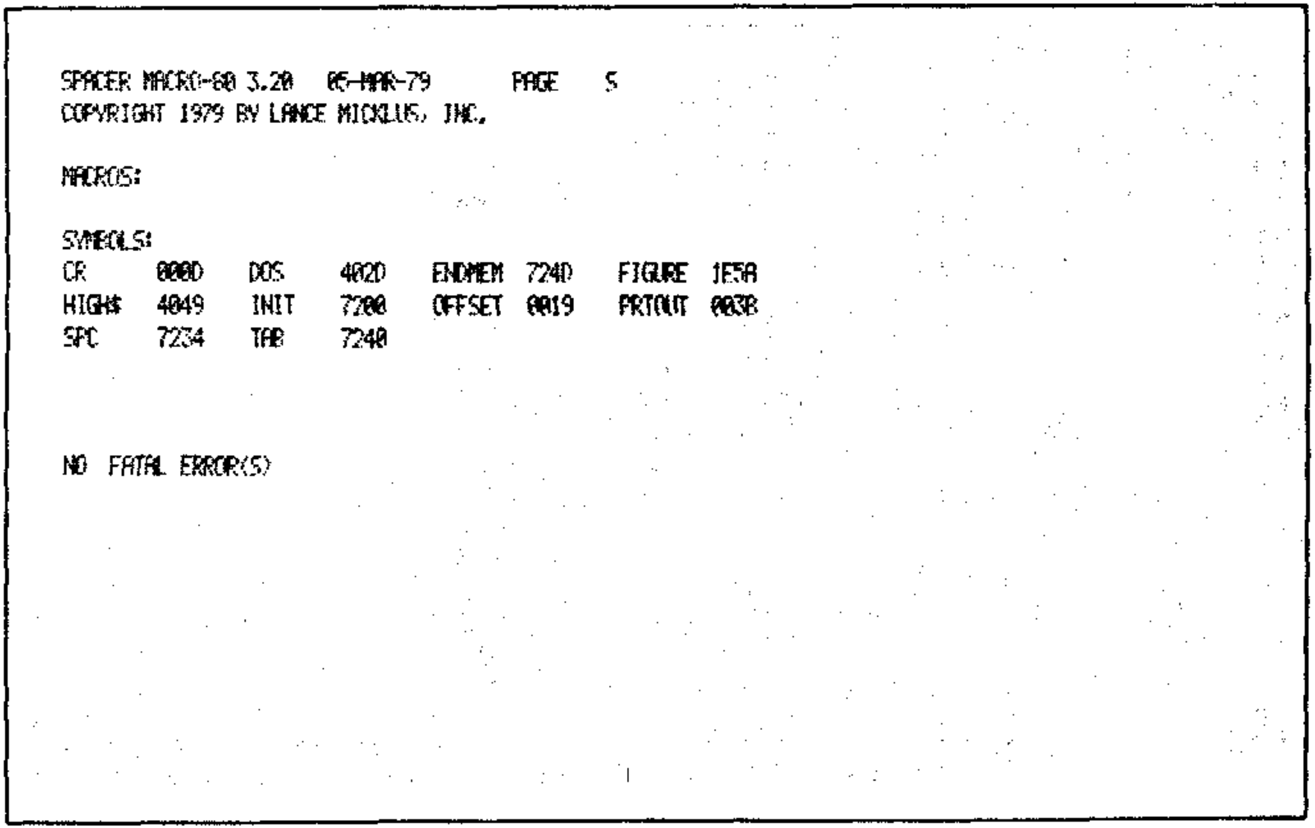
by Small Systems Software

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



```
SPACER MACRO-80 3.20   05-MAR-79        PAGE    1-3
COPYRIGHT 1979 BY LANCE MICKLUS, INC.

 7228   D1               02000          POP    DE            ;DCB ADDR > HL.
 7229   E1               02050          POP    HL            ;NOW SET UP THE DCB
 722A   3E 02            02100          LD     A,2H          ;DCB TYPE IS 2 -
 722C   77               02150          LD     (HL),A        ;OUTPUT ONLY.
 722D   23               02200          INC    HL            ;NOW PUT IN THE
 722E   73               02250          LD     (HL),E        ;ADDRESS VECTOR
 722F   23               02300          INC    HL
 7230   72               02350          LD     (HL),D
 7231   C3 402D          02400          JP     DOS
                         02450   ;
                         02500   ;
                         02550   ;      : THIS IS THE SUBROUTINE WHICH GETS     :
                         02600   ;      : MOVED TO HIGH MEMORY.  IT IS MEMORY   :
                         02650   ;      : INDEPENDENT, SO IT IS RELOCATABLE.    :
                         02700   ;
                         02750   SPC:
 7234   79               02800          LD     A,C
 7235   F5               02850          PUSH   AF
 7236   CD 003B          02900          CALL   PRTOUT        ;SEND BYTE.
 7239   F1               02950          POP    AF            ;IS IT A CR?
```

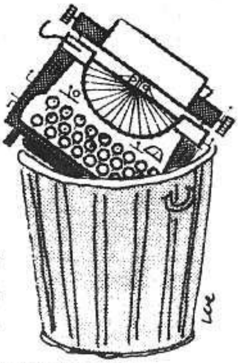
```
SPACER MACRO-80 3.20   05-MAR-79       PAGE    1-4
COPYRIGHT 1979 BY LANCE MICKLUS, INC.

 723A   FE 0C         03000           CP     0CH         ;IS IT FORM FEED?
 723C   D8            03050           RET    C
 723D   FE 0E         03100           CP     0EH
 723F   D0            03150           RET    NC
                      03200   ;
                      03250   TAB:
 7240   06 14         03300           LD     B,20D       ;NUMBER OF EXTRA SPACES
 7242   C5            03350           PUSH   BC
 7243   3E 20         03400           LD     A,20H       ;SPACE CHARACTER
 7245   CD 003B       03450           CALL   PRTOUT
 7248   C1            03500           POP    BC
 7249   10 F7         03550           DJNZ   TAB+2
 724B   C9            03600           RET
 724C   00            03650           NOP
                      03700   ENDMEM:
                      03750   ;
                      03800           END    INIT
```



```
SPACER MACRO-80 3.20   05-MAR-79        PAGE    S
COPYRIGHT 1979 BY LANCE MICKLUS, INC.

MACROS:

SYMBOLS:
CR      000D    DOS     402D    ENDMEM  724D    FIGURE  1E5A
HIGH$   4049    INIT    7200    OFFSET  0019    PRTOUT  003B
SPC     7234    TAB     7240



NO  FATAL ERROR(S)
```

ELECTRIC PENCIL!

Effortless typing is here!



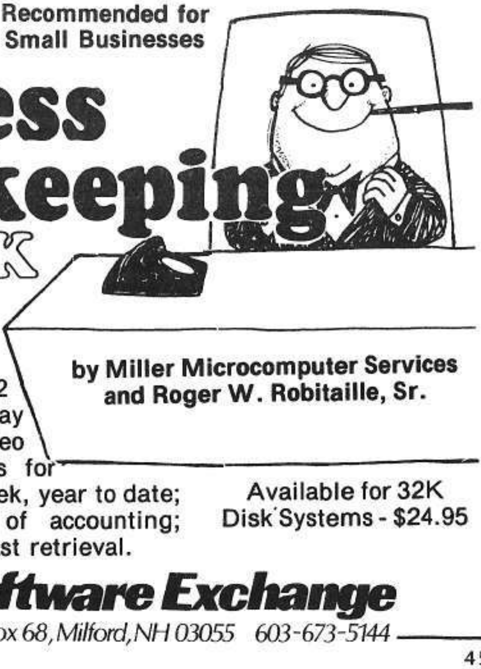
The Electric Pencil by Michael Shrayer is a true word-processing program for the TRS-80. Enter your manuscript, and let your computer do the work. Editing? Just position the cursor with the arrow keys . . . one-key commands let you change, delete, or insert. Fully adjustable margins, left/right justification, variable spacing, page headings, and much more! Save and recall your text with tape or diskfiles. **Typing** everything from letters to reports is fast and incredibly easy using The Electric Pencil.

Level II, 16K tape - $100.00
Disk version - $150.00
(comes on tape)

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

Small Business Bookkeeping FOR DISK

Recommended for Small Businesses

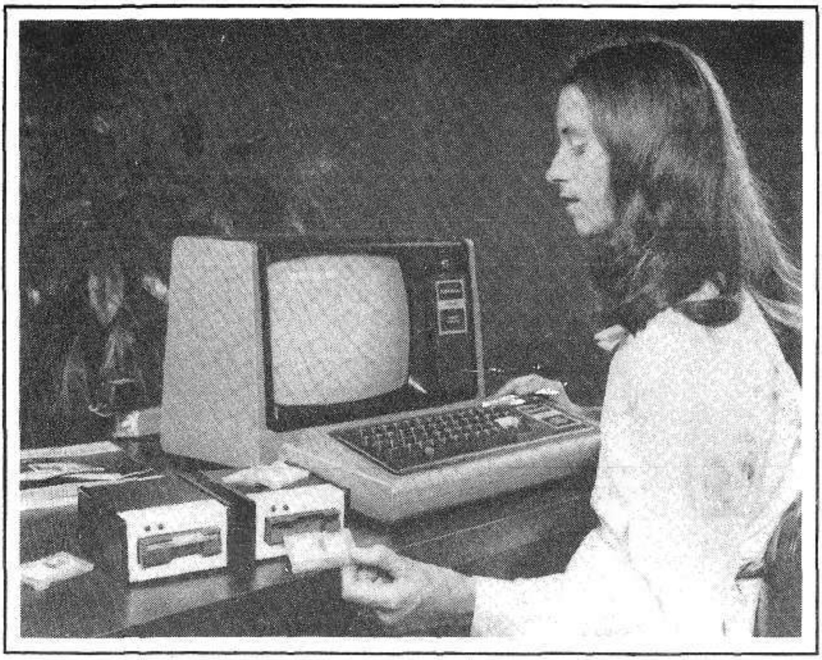


Based on the well-known Dome Bookkeeping System. Posts expenses to as many as 42 accounts (which you may customize). Produces video and line printer reports for year to last week, this week, year to date; supports cash system of accounting; stores data on disk for fast retrieval.

by Miller Microcomputer Services and Roger W. Robitaille, Sr.

Available for 32K
Disk Systems - $24.95

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



AN ALTERNATIVE TO A DISK SYSTEM

by Wynne Keller

In a recent editorial GB told of his love-hate relationship with a disk system for his TRS-80. Perhaps many readers feel a disk is the only option for their system if they want to leave the aggravations of a cassette system behind. At $300-400 for the disk and $100 for the Disk Operating System, not to mention $300 for the expansion interface, there has got to be a cheaper way -- and there is. There is now available a machine called the Exatron Stringy Floppy (ESG) which will do almost everything a disk can do at a fraction of the cost, not to mention headache. We have both systems: a disk at work and ESF at home, and prefer the ESF.

The ESF's merits are fast, reliable loads, and low cost. If you also use Level III Basic with it, you have almost all the commands a disk gives (the major exception being file handling capability). Exatron has recognized the importance of Level III with its system, and sells them as a unit: ESF for $200, Level III for $50, total $250. ESF loads a 16K program in 24 seconds. Failure of a load or save is very rare. It uses very thin tape in a business card sized wafer, connected to form a continuous loop so there is no more fast-forward hassle to find the right program. You simply shove the wafer in the slot, type @LOAD1 (or 2 or whatever number you have given the program) and the machine searches for the right program and loads it. The speed and simplicity of this system will prevent the headaches that formerly seemed to be a painful part of owning a computer.

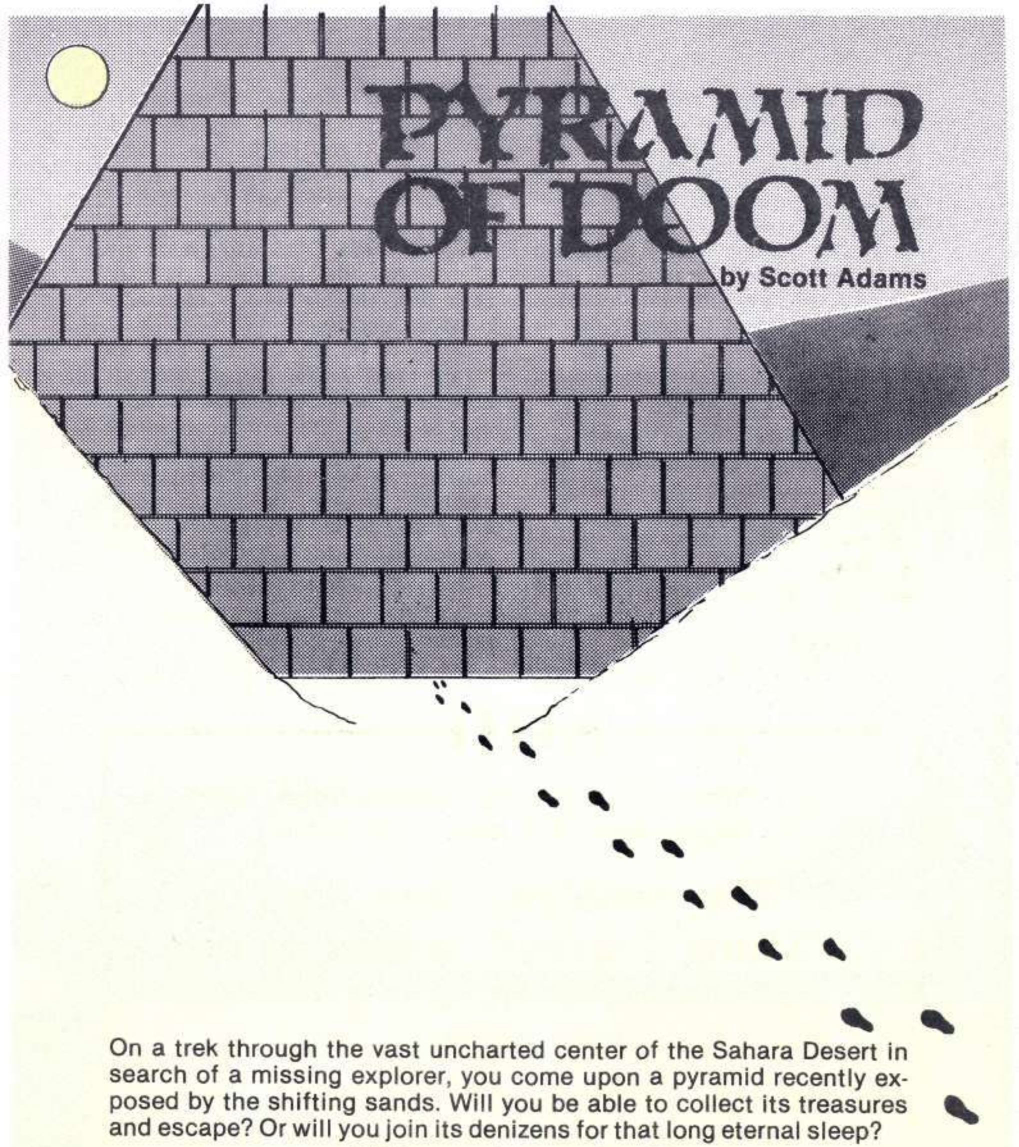
If you load in Level III basic you acquire many DOS commands, and in addition you have abbreviated typing, which DOS does not. (Abbreviated typing means certain letters plus a shift will print more complicated commands such as shift E = EDIT). This can be quite a time saver, once you have learned the letters. There is a catch to Level III however: It uses 5K of memory. Therefore, it doesn't really get used as much as you might like unless you have the expansion interface. We don't have one, so for my longer programs I have to regretfully set Level III aside. DOS also uses a lot of memory: the difference is that with disk the expansion interface must be purchased. With ESF it is optional. If money is not so easy to scrape together you can get the ESF and Level III as one stage in your upgrade, and the expansion interface as the next stage later. After buying it all, you will still have spent $250 less than a disk would cost, and if you decide to live without the expansion interface, it is $550 less.

To find out more about ESF, write or call the company at: Exatron Corporation, 3555 Ryder Street, Santa Clara, CA 95051, tel. 1-800-538-8559.

NOTE

On "String Crasher", by Clayton Schneider
(September 1979 issue - PROG/80)

The problem described in "String Crasher" seems to be associated with some defective Level II ROMs, and appears only at particular memory sizes. However, this is the kind of error that can drive you nuts because it is so hard to find. It is worthwhile to be aware of it.



On a trek through the vast uncharted center of the Sahara Desert in search of a missing explorer, you come upon a pyramid recently exposed by the shifting sands. Will you be able to collect its treasures and escape? Or will you join its denizens for that long eternal sleep?

16K System Cassette $14.95

Want to know what adventure is before you part with that much money? Try the Adventure Sampler! Adventure 0 is a mini version of Adventure 1, offering hours (instead of weeks) of challenge for only $5.95.

Warning; This program may be addictive, leading to prolonged and habitual adventuring.

TSE THE SOFTWARE EXCHANGE
6 South Street Milford, NH 03055

VARIABLE UNIQUENESS AND VALIDITY

by Jeff W. Collins

This program was designed to help keep that new program idea from running amok because of syntax or other problems caused by your innocently-chosen variable names.

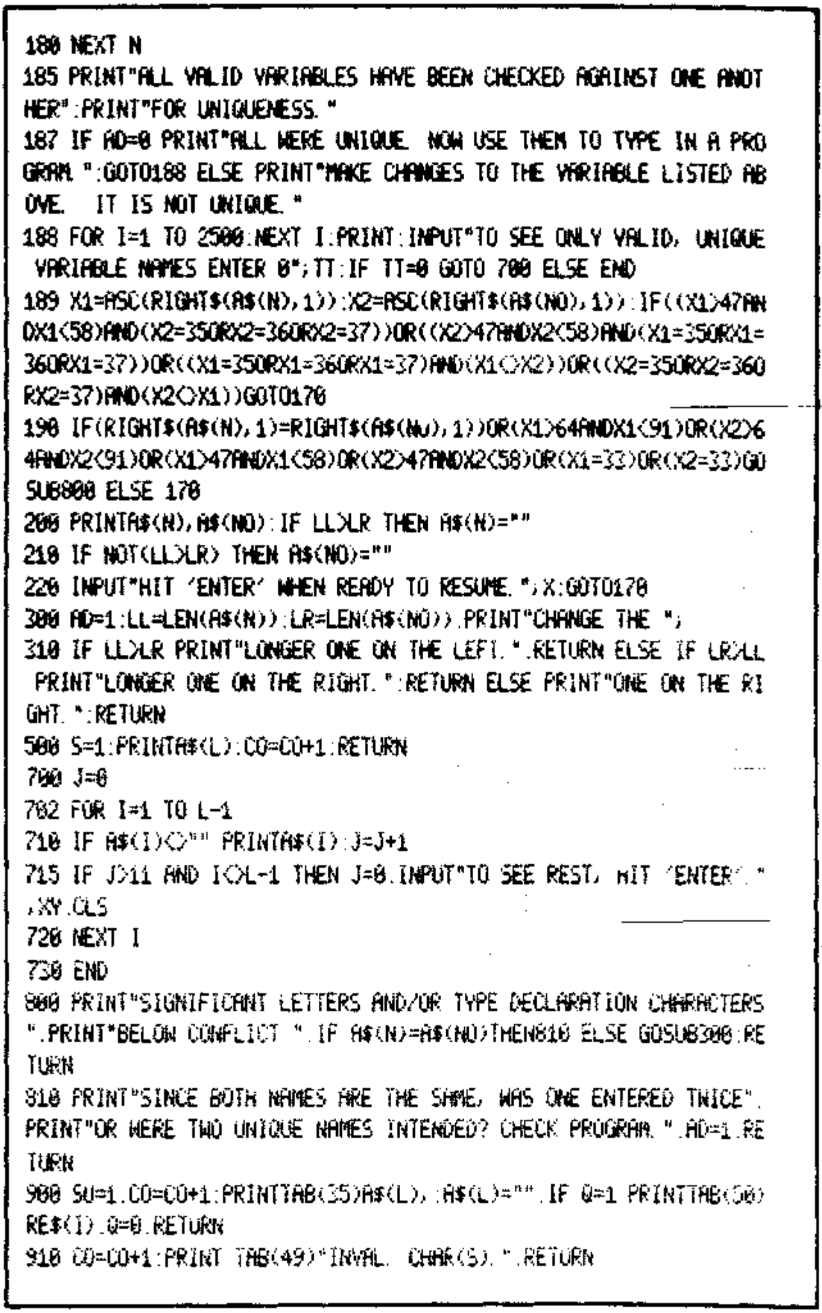
By checking your intended names on this program for validity and uniqueness before running and debugging your program, you have the peace of mind of knowing that no invalid name, or - worse because it's hard to spot - two names with the same significance to the computer, will cause problems.

So, before running your brand-new program, run your intended variable names through this name-checker first. It could be very valuable to you if doing it saves you the later time and frustration of trying to find out what might be wrong in a long, multiple statement line with a lot of levels of parentheses, and especially if you like to use long variable names for clarity instead of the shorter, more symbolic names . . . or even if you simply use a lot of short names in a long program.

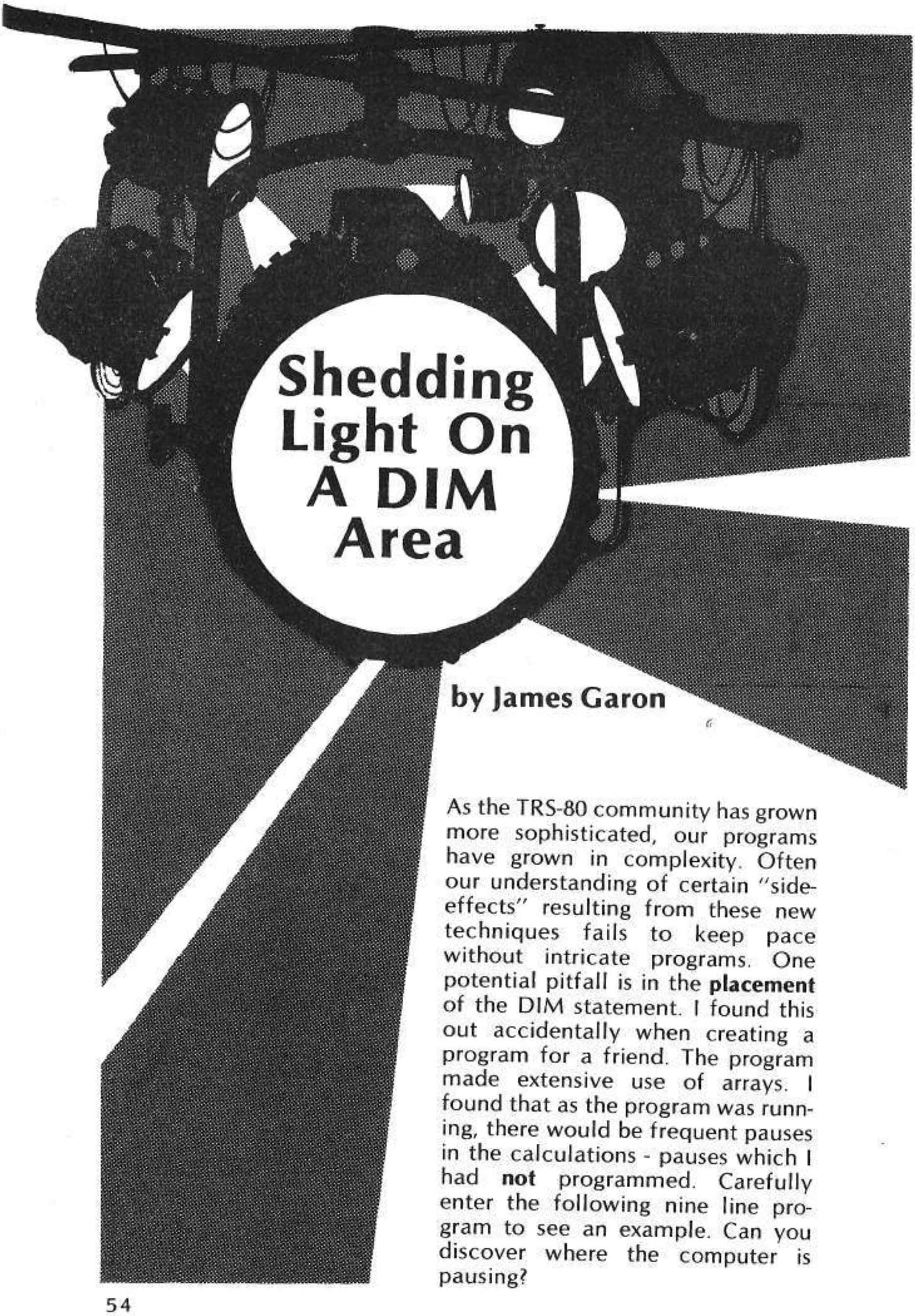
It could possibly spare you the indignity of making bad logic modifications to an otherwise workable program due to false assumptions about the cause of the results you got from your program. Finally, and unphilosophically, did you know that LESS is valid, but that MORE isn't?

```
0 CLS
1 PRINT"THIS PROGRAM CHECKS VARIABLE NAMES YOU MIGHT USE IN A PR
OGRAM."
2 CLEAR 2510
3 DEFINT A-Z
4 DIM RE$(110),A$(100)
5 '---A VARIABLE NAME-CHECKING PROGRAM--CHECKS FOR VALIDITY AND
      UNIQUENESS OF EACH NAME ENTERED. 16K. BY JEFF W. COLLINS,
      WHEELING, W. VA.  MAY '79
6 PRINT"WHEN LONG NAMES ARE ENTERED, ALLOW TIME FOR PROCESSING."
7 FOR I=1 TO 2000:NEXT I:PRINT:INPUT"WHEN READY TO START, HIT 'E
NTER'",X
8 CLS
9 FOR I=1 TO 110
10 READ RE$(I)
11 NEXT I
```

```
12 PRINT"ENTER EACH VARIABLE NAME SUCCESSIVELY. ENTER * WHEN DON
E."
13 FOR K=1 TO 100
15 D$=INKEY$:IF D$="" THEN 15
16 IF D$="*" THEN 28
17 IF ASC(D$)=13 THEN PRINT:GOTO27
19 PRINTD$;
21 A$(K)=A$(K)+D$:D$=""
22 GOTO 15
27 NEXT K
28 K=K-1
29 CLS:PRINT"VALID VARIABLE NAMES","INVALID NAMES",TAB(50)"RESER
VED WORD"
30 FOR L=1 TO K
31 IF A$(L)="" NEXT L ELSE R=LEN(A$(L)):RL=ASC(A$(L))
32 R$=RIGHT$(A$(L),1):RR=ASC(R$)
33 L$=LEFT$(A$(L),1):M2$=MID$(A$(L),2,1)
34 IF CO>11 AND SU=1 PRINT"MAKE A NOTE OF ANY INVALID NAMES."
35 IF CO>11 INPUT"TO SEE REST, HIT 'ENTER'.",XY:CLS:PRINT"VALID
VARIABLE NAMES","INVALID NAMES",TAB(50)"RESERVED WORD":CO=0
36 IF R=1 AND NOT(RL>64 AND RL<91) GOSUB900:GOTO94
37 IF R=1 AND RL>64 AND RL<91 GOSUB500:GOTO94
38 IF R=2 AND RL>64 AND RL<91 AND RR>47 AND RR<58 GOSUB500:GOTO9
4
39 IF R=2 THEN IF (RR=33 OR RR=35 OR RR=36 OR RR=37) AND RL>64 A
ND RL<91 GOSUB 500:GOTO94
40 RM=ASC(M2$)
41 IF R=3 AND RL>64 AND RL<91 AND RM=RL AND RR>47 AND RR<58 GOSU
B500:GOTO94
42 IF R=3 AND(RR=33 OR RR=35 OR RR=36 OR RR=37) AND RL>64 AND RL
<91 AND RM=RL GOSUB 500:GOTO 94
43 IF R=3 AND(RL>64ANDRL<91)AND(RM>47ANDRM<58)AND(RR=33OR RR=35
OR RR=36 OR RR=37)GOSUB500:GOTO94
44 IF R=3 AND(RL>64ANDRL<91)AND(RM>47ANDRM<58)AND(RR>47ANDRR<58)
GOSUB500:GOTO94
45 FOR V=2 TO R-1:IF MID$(A$(L),V,1)<>L$ THEN 47 ELSE NEXT V
46 IF(RR=33 OR RR=35 OR RR=36 OR RR=37) OR (R$=L$) GOSUB500:GOTO
94
47 FOR G=1 TO R:H=ASC(MID$(A$(L),G,1)):IF NOT(H=33 OR H=35 OR H=
36 OR H=37)AND NOT(H>47ANDH<58)AND NOT(H>64ANDH<91)GOSUB900:GOTO
94 ELSE NEXT G
64 FOR I=1 TO 110
65 FE=LEN(RE$(I)):IF FE>R THEN 92
66 FOR J=1 TO R
67 IF FE>R+1-J THEN 92
80 B$=MID$(A$(L),J,FE):IF B$=RE$(I) THEN Q=1:GOSUB900:GOTO 94
81 XX=ASC(B$)
82 IF NOT(XX>64 AND XX<91)AND NOT(XX>47 AND XX<58) AND(XX=33 OR
XX=35 OR XX=36 OR XX=37)THEN Y=Y+1
90 NEXT J
91 IF Y>1 GOSUB900:GOTO 94 ELSE Y=0
92 NEXT I
93 S=1:PRINTA$(L):COUNT=COUNT+1
94 NEXT L
100 DATA @,ABS,AND,ASC,ATN,CDBL,CHR$,CINT,CLEAR,CLOSE,CLS,CMD,CO
NT,COS,CSNG,CVD,CVI,CVS,DATA,DEFDBL,DEFFN,DEFINT,DEFSNG,DEFUSR,D
EFSTR,DELETE,DIM,EDIT,ELSE,END,EOF,ERL,ERR,ERROR,EXP,FIELD
110 DATA FIX,FOR,FRE,GET,GOSUB,GOTO,IF,INKEY$,INP,INPUT,INSTR,IN
T,KILL,LEFT$,LET,LSET,LEN,LINE,LIST,LOAD,LOC,LOF,LOG,MEM,MERGE,M
ID$,MKD$,MKI$,MKS$,NAME,NEW,NEXT,NOT,ON,OPEN,OR
120 DATA OUT,PEEK,POINT,POKE,POS,PRINT,PUT,RANDOM,READ,REM,RESET
,RESTORE,RESUME,RETURN,RIGHT$,RND,SAVE,SET,SGN,SIN,SQR,STEP,STOP
,STRING$,STR$,TAB,TAN,THEN,TIME$,TROFF,TRON,USING,USR,VAL,VARPTR
,TO,SYSTEM,RUN
131 IF SU=1 PRINT"MAKE A NOTE TO CHANGE 'INVALID' VARIABLE NAMES
 "
132 IF S<>1 PRINT"NO VALID VARIABLE NAMES WERE ENTERED.":END
135 PRINT"WHEN READY TO CHECK VALID VARIABLE NAMES FOR"
136 INPUT"DUPLICATE SIGNIFICANCE HIT 'ENTER'.";X
140 FOR N=1 TO L
150 FOR NO=1 TO L
155 IF N=NO THEN 170
156 IF A$(N)="" OR A$(NO)="" THEN 170
160 IF LEFT$(A$(N),2)=LEFT$(A$(NO),2) THEN 189
165 IF LEN(A$(N))<3 AND LEN(A$(NO))<3 AND LEFT$(A$(N),1)=LEFT$(A
$(NO),1) THEN IF (RIGHT$(A$(N),1)="!") OR (RIGHT$(A$(NO),1)="!")
 THEN 189
170 NEXT NO
```



```
180 NEXT N
185 PRINT"ALL VALID VARIABLES HAVE BEEN CHECKED AGAINST ONE ANOT
HER":PRINT"FOR UNIQUENESS."
187 IF AD=0 PRINT"ALL WERE UNIQUE. NOW USE THEM TO TYPE IN A PRO
GRAM.":GOTO188 ELSE PRINT"MAKE CHANGES TO THE VARIABLE LISTED AB
OVE.  IT IS NOT UNIQUE."
188 FOR I=1 TO 2500:NEXT I:PRINT:INPUT"TO SEE ONLY VALID, UNIQUE
 VARIABLE NAMES ENTER 0";TT:IF TT=0 GOTO 700 ELSE END
189 X1=ASC(RIGHT$(A$(N),1)):X2=ASC(RIGHT$(A$(NO),1)):IF((X1>47AN
DX1<58)AND(X2=35ORX2=36ORX2=37))OR((X2>47ANDX2<58)AND(X1=35ORX1=
36ORX1=37))OR((X1=35ORX1=36ORX1=37)AND(X1<>X2))OR((X2=35ORX2=36O
RX2=37)AND(X2<>X1))GOTO170
190 IF(RIGHT$(A$(N),1)=RIGHT$(A$(NO),1))OR(X1>64ANDX1<91)OR(X2>6
4ANDX2<91)OR(X1>47ANDX1<58)OR(X2>47ANDX2<58)OR(X1=33)OR(X2=33)GO
SUB800 ELSE 170
200 PRINTA$(N),A$(NO):IF LL>LR THEN A$(N)=""
210 IF NOT(LL>LR) THEN A$(NO)=""
220 INPUT"HIT 'ENTER' WHEN READY TO RESUME.";X:GOTO170
300 AD=1:LL=LEN(A$(N)):LR=LEN(A$(NO)):PRINT"CHANGE THE ";
310 IF LL>LR PRINT"LONGER ONE ON THE LEFT.":RETURN ELSE IF LR>LL
 PRINT"LONGER ONE ON THE RIGHT.":RETURN ELSE PRINT"ONE ON THE RI
GHT.":RETURN
500 S=1:PRINTA$(L):CO=CO+1:RETURN
700 J=0
702 FOR I=1 TO L-1
710 IF A$(I)<>"" PRINTA$(I):J=J+1
715 IF J>11 AND I<>L-1 THEN J=0:INPUT"TO SEE REST, HIT 'ENTER'."
;XY:CLS
720 NEXT I
730 END
800 PRINT"SIGNIFICANT LETTERS AND/OR TYPE DECLARATION CHARACTERS
":PRINT"BELOW CONFLICT ":IF A$(N)=A$(NO)THEN810 ELSE GOSUB300:RE
TURN
810 PRINT"SINCE BOTH NAMES ARE THE SAME, WAS ONE ENTERED TWICE":
PRINT"OR WERE TWO UNIQUE NAMES INTENDED? CHECK PROGRAM.":AD=1:RE
TURN
900 SU=1:CO=CO+1:PRINTTAB(35)A$(L);:A$(L)="":IF Q=1 PRINTTAB(50)
RE$(I):Q=0:RETURN
910 CO=CO+1:PRINT TAB(49)"INVAL. CHAR(S).":RETURN
```



Shedding Light On A DIM Area

by James Garon

As the TRS-80 community has grown more sophisticated, our programs have grown in complexity. Often our understanding of certain "side-effects" resulting from these new techniques fails to keep pace without intricate programs. One potential pitfall is in the **placement** of the DIM statement. I found this out accidentally when creating a program for a friend. The program made extensive use of arrays. I found that as the program was running, there would be frequent pauses in the calculations - pauses which I had **not** programmed. Carefully enter the following nine line program to see an example. Can you discover where the computer is pausing?

```
10 CLS:DEFSTRA:R=0:Q=R:DIM A(4500):A(0)=STRING$(7,32):A(1)="=ENT
ER=
20 CLS:PRINT"PRESS =ENTER= FOR DEMONSTRATION OF DIM DELAY
30 Q=1-Q:PRINT@6,A(Q);:FORR=1TO20:IFINKEY$<>CHR$(13)NEXT:GOTO30
40 CLS:PRINT"X =";:X=2:PRINTX:X1=X
50 PRINT"TWO PLUS TWO IS":Y=X+X:PRINTY:Y1=Y
60 PRINT"DID ";:A="":PRINT"YOU ";:B=0:PRINT"KNOW ";:C=0
70 PRINT"YOUR ";:D=0:PRINT"COMPUTER ";:E=0:PRINT"WAS ";
80 F=0:PRINT"SO ";:G=0:PRINT"S";:H=0:PRINT"L";:I=J:J=K
90 PRINT"O";:K=C:L=K:M=B:PRINT"W?"
```

Each time a new variable is placed on the left of an equal sign (=), the program pauses. Why? The culprit is the innocent-looking DIMension statement in line 10. This particular DIM statement causes 13,500 bytes to be reserved for string array variables. Now array variables (that is, DIMensioned variables) are stored **above** all non-array (or simple) variables. Once you have used the DIM statement, every new simple variable has to find a home. The computer must make room for the new variable by moving all 13,500 bytes of array. This is what takes all that time and causes those unexpected delays.

Is there a solution? Yes. Just **mention** each simple variable you will be using at least once **before** the DIM occurs. Add the following line to the program above:

```
5 B=0:C=B:D=B:E=B:F=B:G=B:H=B:I=B:J=B:K=B:L=B:M=B:X=B:X1=B:Y=B:Y
1=B.A$="
```

This time there is no delay when the program is RUN. If you "listen" to the program on an AM radio (Insert a ' or REM at the beginning of line 5) you will hear the distinctive sound of "creeping arrays" during each pause. Listen for this sound in your own programs. When you hear it, hit BREAK (or you might try listening for it with TRON). Notice which variables appear to the left of the equal sign (=) and **mention** them **before** the first DIM statement. You should notice a rewarding increase in speed. ■

PRINT SPOOLER for the TRS-80 Microcomputer and a Parallel Line Printer

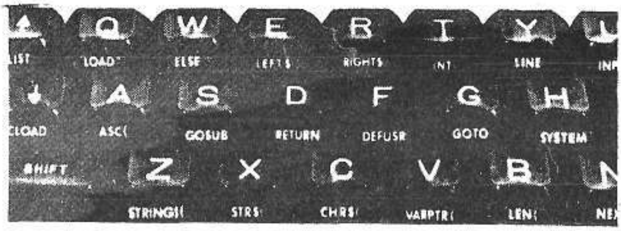
This utility will allow you to print out a data file from disk at the same time you are running a program for something else on your computer. Now you don't have to stop working on your program to get a listing.

Procedure: Save file to be printed as an ASCII file.
Run the SPOOL routine from DOS
Tell SPOOL the name of the file to print out.
Load BASIC and run your other program.

Restrictions: BASIC execution speed reduced 30%
Printer operates 35% slower
File to be printed must be ASCII file
Any printer commands in program you are running will be printed to screen instead of printer
Will NOT work under NewDOS (TRSDOS 2.3 and VTOS 3.0 will work)

Price $19.95 on tape, $24.95 on cassette

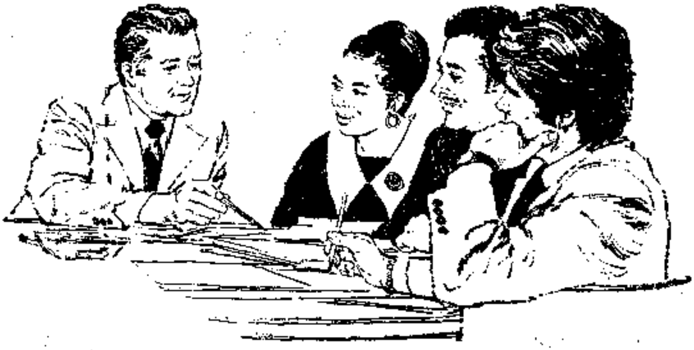
TSE
The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144



Q
W
E
R
T
Y
LIST
LOAD"
ELSE
LEFT$(
RIGHT$(
INT
LINE
A
S
D
F
G
H
CLOAD
ASC(
GOSUB
RETURN
DEFUSR
GOTO
SYSTEM
SHIFT
Z
X
C
V
B
STRING$(
STR$(
CHR$(
VARPTR(
LEN(

PREFERENCE POLL

by George Blank



As the pastor of a modest size church with a full size program, one of my chief responsibilities is to make the best use of the leadership talents of our members. This year my Radio Shack TRS-80 Computer was a big help in matching people's talents and preferences to the job they do for the church.

There are nine persons on our church board, and each one has a second responsibility. The board consists of three classes of three persons each. One class is elected each year, and at that time we rotate the committee assignments. Here are the positions that have to be filled:

- Clerk of Session
- Chairman of Worship Committee
- Chairman of Christian Education Committee
- Chairman of Stewardship Committee
- Chairman of Property Committee
- Chairman of Major Mission Fund Committee
- Chairman of Budget Committee
- Chairman of Nominating Committee
- Chairman of 100th Anniversary Committee

Normally we meet at the manse for coffee and cookies on a Sunday afternoon and ask people to volunteer for committee assignments. Last year the process took an hour and a half. This year we were done in half an hour, leaving us more time to enjoy our refreshments and each other's company. The difference, a saving of ten man (woman) hours, was the contribution of this program.

Before the meeting, I entered into the computer the names of the people on the board and the assignments. As each one arrived, I asked them to be seated at the computer, and press the number next to their name. As soon as they did that, the computer displayed, beneath their name, a list of the nine positions. They were asked to pick the position that they would like most, and type the number beside it. Each time they made a selection, the position was removed from the screen and they were asked to select their preference among the remaining assignments. This continued until each person has picked six choices. When all members of the board had picked their selections, I typed F for "Finished" and the computer displayed the first position, and under it a list of the persons who picked that assignment. The list was displayed in order, from first choice through sixth choice, and a number in front of the name indicated each person's ranking of the assignment. If one or more persons had selected that assignment as first choice, we selected them for the assignment. Otherwise we turned to second choice, and once, to third choice. If two or more had selected the same one, we took time for them to negotiate the assignment between themselves. As soon as one position was filled, I touched a key and the next assignment was displayed in the same way. This continued until all positions were filled.

The advantages of this method were that each person's preferences were displayed and considered on each assignment, we did not have to wait for anyone who was reluctant to volunteer to speak up, the assignments were made with a great deal of satisfaction to the people receiving them, and the computer proved to help people instead of controlling them.

As valuable as this program was for its intended use, that is only a tiny fraction of its potential. There are many other activities for which it is ideally suited. It can help a board choose between different courses of action, by displaying each person's opinion of each action in a clear format. It is extremely well suited to the values clarification method of moral education. It even makes a good party game to help people get acquainted. Beyond those three additional uses, there are many more, but three examples should stimulate your imagination.

Choice Between Different Courses Of Action.

The church board has the responsibility of determining where to send mission money that was contributed above and beyond the $6000 that our church has pledged for general mission causes. As we ask for suggestions, we are given six. Before we determine how to allocate the money, we poll each member of the board on these six options:

A local Seminary.
A more distant Seminary that the Pastor graduated from.
A local College.
Another local College.
Specific support for a missionary.
World hunger relief.

Values Clarification.

As the youth group gathers for the Sunday evening meeting, each member is asked to choose the goals that appeal to them most from this list:

Money
Fame
Power
Strength
Friendship
Love
Health
Peace
Education
Respect

After each member has made their selections, each option is displayed in turn and persons who chose that option are asked to explain why they chose it. This way the young people get to examine what they value, compare their values with other people's values, and hopefully, to understand better why they value what they value.

Party Game.

The exercise above would serve as a good mixer, but to give a more exotic touch to a party, ask each person to select six jobs, in order of preference, from this list:

Lion tamer
Kamikazi pilot
Radical terrorist
Mercenary soldier
High wire artist
Missionary to headhunters
Stunt man
Bomb disposal

Information About The Program.

There are a large number of REMarks in the program listing, and it should be easy to modify. The INKEY$ instruction in Radio Shack Level II Basic detects a key that is pressed and accepts the value without a carriage return. If your BASIC does not have this function, you may substitute an INPUT statement. If you choose this option, you can delete the routine at 5000 which checks for a two digit number if a 1 is pressed, the two statements that call that routine, and the buffer routine at the end of the option selection. The two INKEY$ instructions there are part of the buffer, and serve to clear the 2 key rollover on the Radio Shack computer.

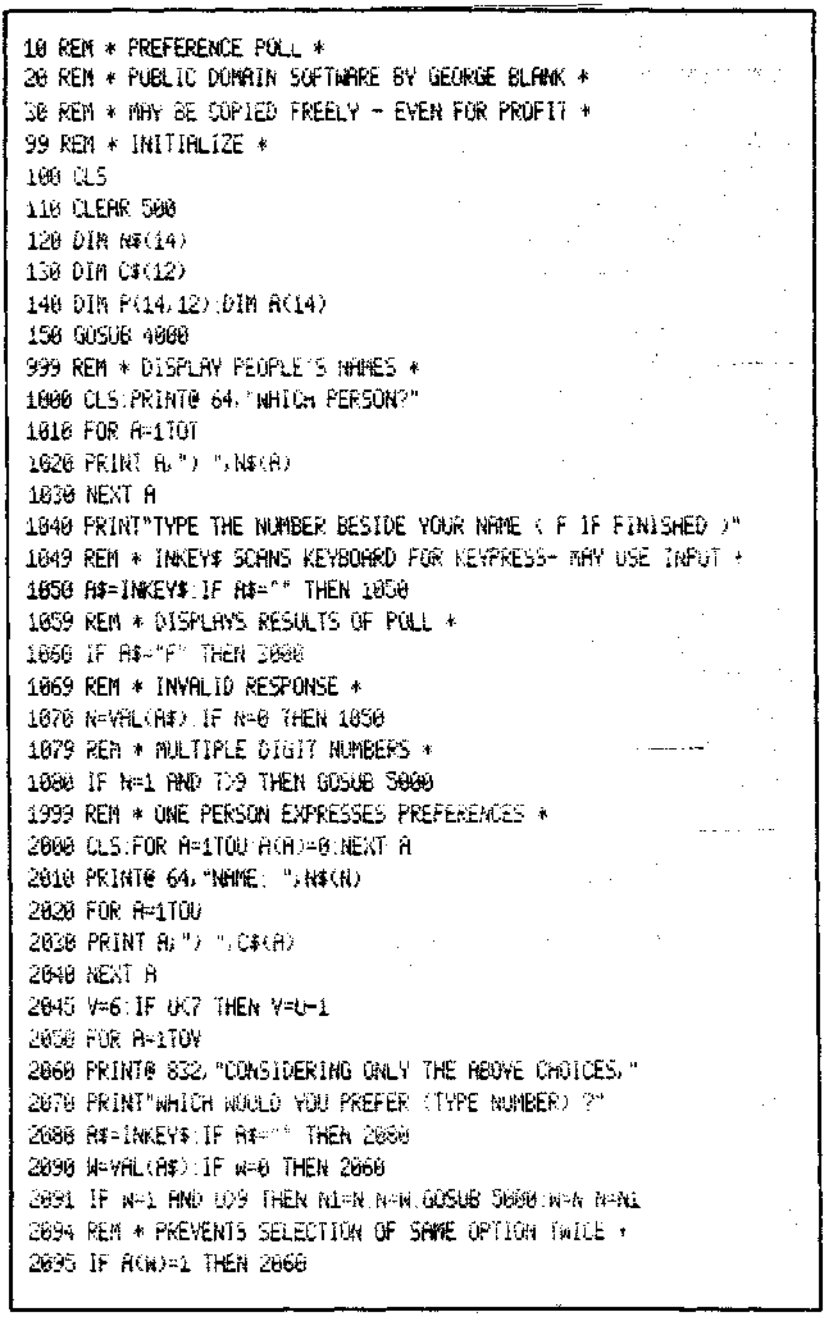
The most logical change people would be likely to make is to change the number of options selected, either to require each person to rank every option, or to limit the options selected to a smaller or larger number. This is easy to do by changing the value of V in line 2045. When a person has selected V options, the computer calls the name routine for the next person.

Players can change their minds and revise their choices simply by typing the letter beside their names a second or third time. This is the way to deal with mistakes. Otherwise, they would have to go completely through the selection routine each time. The results display can be called at any time, even if all players have not made selections. If people arrive late, just cycle through the results, let them enter their preferences, and call the results again. The name selection routine is called at the end of the results routine, and all data is saved.

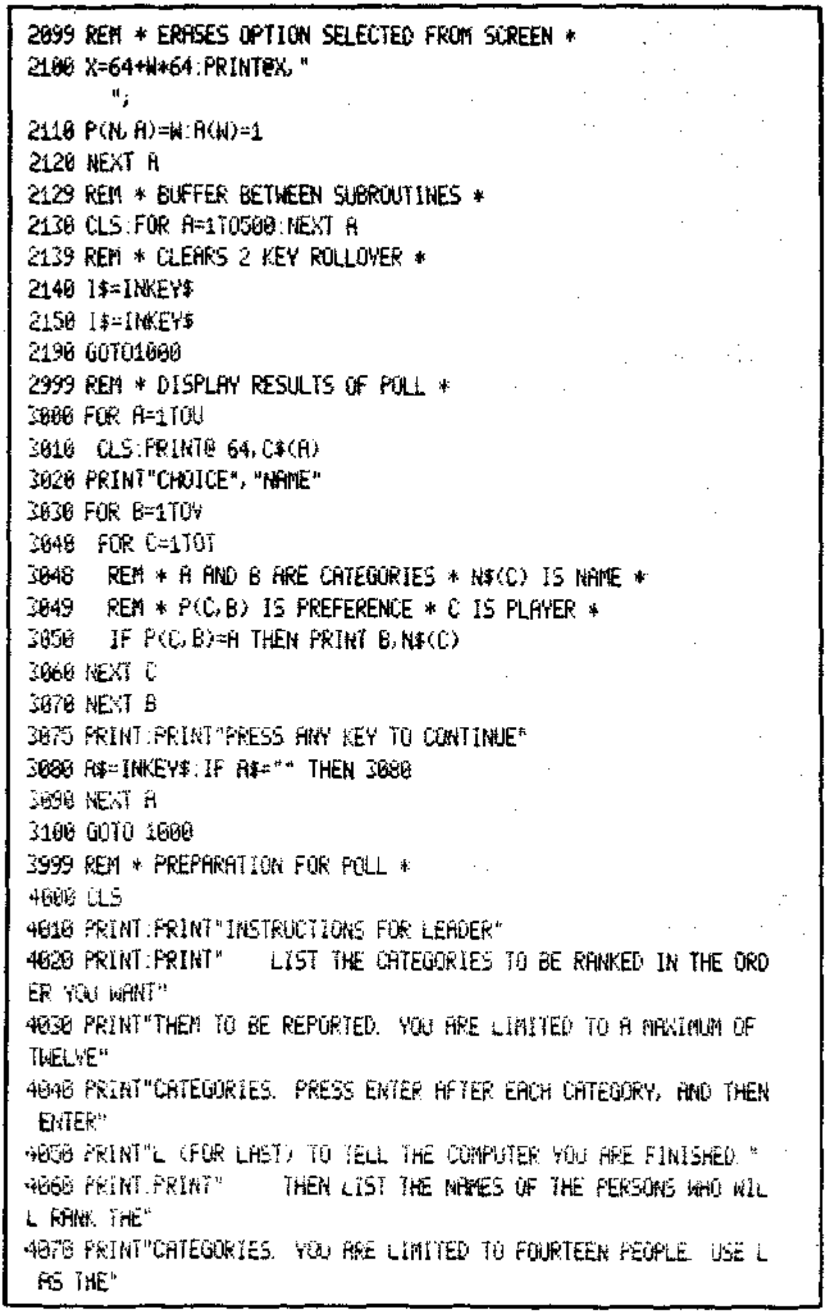
If someone whose name has not been entered comes in, you can break into the program to add a new name. This is how it would be done on the Radio Shack computer, assuming that the person is named BOB and he is the 10th person:

```
        Press BREAK
        Enter
T=10:N$(10)="BOB":CONTINUE
```

None of the other data would be lost if this is done correctly. ■



```
10 REM * PREFERENCE POLL *
20 REM * PUBLIC DOMAIN SOFTWARE BY GEORGE BLANK *
30 REM * MAY BE COPIED FREELY - EVEN FOR PROFIT *
99 REM * INITIALIZE *
100 CLS
110 CLEAR 500
120 DIM N$(14)
130 DIM C$(12)
140 DIM P(14,12):DIM A(14)
150 GOSUB 4000
999 REM * DISPLAY PEOPLE'S NAMES *
1000 CLS:PRINT@ 64,"WHICH PERSON?"
1010 FOR A=1TOT
1020 PRINT A;") ";N$(A)
1030 NEXT A
1040 PRINT"TYPE THE NUMBER BESIDE YOUR NAME ( F IF FINISHED )"
1049 REM * INKEY$ SCANS KEYBOARD FOR KEYPRESS- MAY USE INPUT *
1050 A$=INKEY$:IF A$="" THEN 1050
1059 REM * DISPLAYS RESULTS OF POLL *
1060 IF A$="F" THEN 3000
1069 REM * INVALID RESPONSE *
1070 N=VAL(A$):IF N=0 THEN 1050
1079 REM * MULTIPLE DIGIT NUMBERS *
1080 IF N=1 AND T>9 THEN GOSUB 5000
1999 REM * ONE PERSON EXPRESSES PREFERENCES *
2000 CLS:FOR A=1TOU:A(A)=0:NEXT A
2010 PRINT@ 64,"NAME: ";N$(N)
2020 FOR A=1TOU
2030 PRINT A;") ";C$(A)
2040 NEXT A
2045 V=6:IF U<7 THEN V=U-1
2050 FOR A=1TOV
2060 PRINT@ 832,"CONSIDERING ONLY THE ABOVE CHOICES,"
2070 PRINT"WHICH WOULD YOU PREFER (TYPE NUMBER) ?"
2080 A$=INKEY$:IF A$="" THEN 2080
2090 W=VAL(A$):IF W=0 THEN 2060
2091 IF W=1 AND U>9 THEN N1=N:N=W:GOSUB 5000:W=N:N=N1
2094 REM * PREVENTS SELECTION OF SAME OPTION TWICE *
2095 IF A(W)=1 THEN 2060
```



```
2099 REM * ERASES OPTION SELECTED FROM SCREEN *
2100 X=64+W*64:PRINT@X,"
      ";
2110 P(N,A)=W:A(W)=1
2120 NEXT A
2129 REM * BUFFER BETWEEN SUBROUTINES *
2130 CLS:FOR A=1TO500:NEXT A
2139 REM * CLEARS 2 KEY ROLLOVER *
2140 I$=INKEY$
2150 I$=INKEY$
2190 GOTO1000
2999 REM * DISPLAY RESULTS OF POLL *
3000 FOR A=1TOU
3010  CLS:PRINT@ 64,C$(A)
3020 PRINT"CHOICE","NAME"
3030 FOR B=1TOV
3040  FOR C=1TOT
3048   REM * A AND B ARE CATEGORIES * N$(C) IS NAME *
3049   REM * P(C,B) IS PREFERENCE * C IS PLAYER *
3050   IF P(C,B)=A THEN PRINT B,N$(C)
3060 NEXT C
3070 NEXT B
3075 PRINT:PRINT"PRESS ANY KEY TO CONTINUE"
3080 A$=INKEY$:IF A$="" THEN 3080
3090 NEXT A
3100 GOTO 1000
3999 REM * PREPARATION FOR POLL *
4000 CLS
4010 PRINT:PRINT"INSTRUCTIONS FOR LEADER"
4020 PRINT:PRINT"    LIST THE CATEGORIES TO BE RANKED IN THE ORD
ER YOU WANT"
4030 PRINT"THEM TO BE REPORTED.  YOU ARE LIMITED TO A MAXIMUM OF
TWELVE"
4040 PRINT"CATEGORIES.  PRESS ENTER AFTER EACH CATEGORY, AND THEN
 ENTER"
4050 PRINT"L (FOR LAST) TO TELL THE COMPUTER YOU ARE FINISHED."
4060 PRINT:PRINT"     THEN LIST THE NAMES OF THE PERSONS WHO WIL
L RANK THE"
4070 PRINT"CATEGORIES.  YOU ARE LIMITED TO FOURTEEN PEOPLE.  USE L
 AS THE"
```

```
4080 INPUT"LAST NAME. PRESS ENTER TO BEGIN";A$
4099 REM * LIST CATEGORIES *
4100 CLS
4110 U=0
4120 U=U+1.PRINT"CATEGORY NUMBER",U;:INPUT C$(U)
4130 IF C$(U)="L" THEN C$(U)="":U=U-1.GOTO 4150
4135 IF U=12 THEN 4150
4140 GOTO 4120
4149 REM * EDIT CATEGORIES *
4150 CLS
4160 FOR A=1TOU
4170 PRINT A;C$(A):NEXT A
4180 INPUT"TO CHANGE ANY CATEGORY, ENTER NUMBER (0 IF NO CHANGES
)",A
4190 IF A=0 THEN 4300
4200 IF A>12 THEN 4150
4209 REM * ADD A CATEGORY *
4210 IF A>U THEN U=A
4220 PRINT"NEW CATEGORY NUMBER",A;
4230 INPUT C$(A)
4240 GOTO 4150
4299 REM * LIST NAMES *
4300 CLS
4310 T=0
4320 T=T+1
4340 PRINT"NAME OF PERSON NUMBER";T;
4350 INPUT N$(T)
4359 REM * ADD A NAME *
4360 IF N$(T)="L" THEN N$(T)="":T=T-1:GOTO 4400
4370 IF T=14 THEN 4400
4380 GOTO 4320
4399 REM * EDIT NAMES *
4400 CLS:FOR A=1TOT
4410 PRINT A,N$(A)
4420 NEXT A
4430 INPUT"TO CHANGE A NAME, ENTER NUMBER ( 0 IF NO CHANGES)";A
4440 IF A=0 THEN 4500
4450 IF A>14 THEN 4400
4460 IF A>T THEN T=A
```

```
4470 PRINT"NEW NAME NUMBER",A;
4480 INPUT N$(A)
4490 GOTO 4400
4500 RETURN
4999 * NUMBERS 10 - 19 *
5000 FOR D=1TO500.NEXT D
5010 A$=INKEY$
5020 IF A$="" THEN 5090
5030 N=10+VAL(A$)
5090 RETURN
```

MACROTRONICS M-80

by Donald W. DeJarnette

One of the most fascinating and useful devices to be developed as a peripheral to the TRS-80 Level II, 16K is the Macrotronics M-80, a system that will decode and transmit Morse, Baudot, and ASCII. This unit is designed primarily to interface with an amateur radio transceiver. With very few modifications the M-80 can be adapted to function with a general coverage shortwave receiver. It is only necessary that the receiver have a good product detector BFO (beat frequency oscillator) and can maintain a degree of selectivity and frequency stability (4kHz at 6dB down * selectivity and 1kHz * frequency stability).

M-80 software consists of a cassette BASIC language program and a machine language subprogram. The software is well supported with documentation. A machine language keyboard scan is utilized to prevent keybounce associated with the TRS-80.

Interfacing is accomplished through a printed circuit board that attaches to the 40 pin TRS-80 bus. The circuit board contains the electronics necessary to demodulate the pulsed tone audio signal and transmit the signal to the computer. This circuit also produces the transmitter keying for CW (continuous wave Morse) and RTTY (radioteletype; Baudot, ASCII). M-80 is optically isolated from the TRS-80. ASCII RTTY at 110 band is possible through the M-80 with the additional M-800 system installed.

The M-80 PC board is attached to the speaker or earphone terminals of the shortwave receiver. A CW or RTTY signal is decoded by tuning the received signal to match the tone input frequency of the M-80. I have found the best filter for CW on my receiver (Radio Shack DX-300) is the upper sideband filter. Conversely, the lower sideband filter does a better job on RTTY. The input frequency of the M-80 is adjustable through the use of a trimmer potentiometer. The pot varies the input frequency of the PLL (phased lock loop) demodulator integrated circuit. When your receiver reaches the input frequency a light emitting diode is activated and will remain on for the duration of the pulse. An audio signal is also provided with an optional speaker.

Morse CW reception and transmission is variable from 1 to 399 words per minute. Baudot RTTY is possible at 60, 66, 75, or 100 words per minute. Transmission of RTTY requires the use of a FSK (frequency shift converter) or an AFSK (audio frequency shift converter). Keyboard input is buffered to 255 characters.

RTTY FEATURES

- Send CW Identification
- Reverse mark and space tones
- Reverse shift on space
- Converts to 32 characters per line
- Creates and sends "canned" message

CW FEATURES

- Code practice mode
- Changes CW spacing
- Changes keying polarity
- Converts to 32 characters per line
- Creates and sends "canned" message

SPECIAL FEATURES

- Line printer driver: With the addition of the optional MLK-1 loop keyer module, the M-80 can be used to drive inexpensive surplus Baudot printers such as Teletype Models 15, 19, 28, and 32. The Model 15 RO is currently available surplus for $95. A driver program is included with the M-80 instruction manual.

You can output directly to Centronics type printers (Radio Shack) without the optional MLK-1 by utilizing the following:

***OUTPUT TO PRINTER- POKE 16414,141: POKE 16415,5**
***TO RETURN TO SCREEN* POKE 16414,88: POKE 16415,4**

• I/O board: The M-80 can be used to control various peripheral devices. The PC board, with software support, can switch up to 5 relays (3 solid state, 2 mechanical). These 5 relays can be used to drive higher voltage relays that can control appliances, lighting, timers, heating and cooling systems, etc. A port driver is included with the M-80.

I encountered two minor problems when placing the M-80 into operation. The volume level required to demodulate the signal varied with the received signal strength. This meant that on weak or fluctuating signals I was constantly adjusting the volume level. The problem was solved by the addition of an automatic level control removed from a junked cassette tape recorder. The second problem encountered was inadequate audio filtration. This caused "hits" in both the CW and RTTY modes. While the M-80 functioned well without an audio filter, perfect copy is attainable with one installed. Several commercial audio filters are available including models from Autek Research and MFJ Enterprises. For those who would prefer to construct their own audio active filter, an excellent schematic is included in Radio Shack's "Integrated Circuit Projects - Volume 2". This circuit employs a twin-T bandpass filter in conjunction with a 741 operational amplifier integrated circuit.

The M-80 also has provisions for the attachment of a RTTY terminal unit (demodulator). When connected, the TU replaces both the audio filter and the PLL demodulator. This unit will deliver excellent RTTY under marginal signal conditions. ■

M - 80 $149.00 **MLK - 1 $29.00** **M - 800 $99.00**

Available from: Macrotronics, Inc., P.O. Box 518, Keyes, CA 95328

Teletype Model 15 RO - $95.00.
Available from:

- Atlantic Surplus Sales, 3730 Nautilus Avenue, Brooklyn, NY 11224
- MFJ Enterprises, P.O. Box 494, Mississippi State, MS 39762
- Autek Research, Box 512E, Sherman Oaks, CA 91403

For Machine Language Programming

BEGINNERS

TRS-80 ASSEMBLY LANGUAGE PROGRAMMING: The best current introduction to machine language at a bargain price. This is not a series of easy lessons, but an intermediate level discussion of what you can do in machine language.

$3.95 + $1 shipping

JOURNEYMEN

Z-80 INSTRUCTION HANDBOOK: Reference guide for the Z-80, giving the instructions and telling briefly how to use them.

$4.95 + $1 shipping

Z-80 SOFTWARE GOURMET GUIDE AND COOKBOOK: Well over 100 utility, search, sort, and arithmetic routines, plus how to use them and what they do. Well written and easy to understand.

$14.95 + $1 shipping

MASTERS

SARGON - A COMPUTER CHESS PROGRAM: Program Sargon into your computer using the Assembler, or use the listings to understand how it is done. Many useful subroutines.

$14.95 + $1 shipping

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

HEXADECIMAL TO DECIMAL CONVERTER

by Bill Everett

Has inflation driven your budget up to the point where you get tired even writing the numbers? Would you like to hide your hobby expenses from your spouse? Why not keep your records in the hexadecimal numbering system? This program will help you convert your records back and forth.

Of course, if you are unimaginative, you could use the program to convert numbers for machine language programming.

```
10 ' BY BILL EVERETT
20 '    11036 SE 30TH
30 '    BELLEVUE, WASH. 98004
40 'DATE LAST MODIFIED 4-8-79
50 '
60 'CONVERSION FOR HEX TO DECIMAL OR DECIMAL TO HEX
70 '
80 CLS : B$="XXXX" : H$="0"
90 PRINT@448,;:INPUT "DO YOU WANT TO CONVERT HEX (H) OR DECIMAL
(D) NUMBERS";A$
100 CLS:IF A$="H" THEN 110 ELSE IF A$="D" THEN 240 ELSE PRINT "'
H' OR 'D' PLEASE" : GOTO 90
110 'HEX TO DEC
120 PRINT : INPUT "WHAT HEX NUMBER OR XXXX TO GO TO DECIMAL";C$
130 IF C$=B$ THEN PRINT : GOTO 240
140 IF LEN(C$)>4 THEN PRINT "NUMBER TOO LARGE" : GOTO 120
150 IF LEN(C$)<4 THEN C$=H$+C$ : GOTO 150
160 GOSUB 370
170 A$=D$ : GOSUB 420 : D$=A$
180 A$=E$ : GOSUB 420 : E$=A$
190 A$=F$ : GOSUB 420 : F$=A$
200 A$=G$ : GOSUB 420 : G$=A$
210 D=(VAL(D$)*4096) + (VAL(E$)*256) + (VAL(F$)*16) + VAL(G$)
220 PRINT C$;" HEX  =  ";D;"  DECIMAL" : PRINT
230 GOTO 120
240 ' DEC TO HEX
```

```
250 INPUT "WHAT DECIMAL NUMBER OR 9999 TO GO TO HEX";D
260 D=INT(ABS(D))
270 IF D=9999 THEN 110
280 IF D>65535 THEN PRINT "NUMBER TOO LARGE" : GOTO 250
290 X=D
300 GOSUB 510 : G$=A$
310 GOSUB 510 : F$=A$
320 GOSUB 510 : E$=A$
330 GOSUB 510 : D$=A$
340 C$=D$+E$+F$+G$
350 PRINT D;"  DECIMAL  =  ";C$;"  HEX" : PRINT
360 GOTO 250
370 D$=LEFT$(C$,1)
380 E$=MID$(C$,2,1)
390 F$=MID$(C$,3,1)
400 G$=RIGHT$(C$,1)
410 RETURN
420 IF ASC(A$)>=48 AND ASC(A$)<=57 THEN RETURN
430 IF ASC(A$)<65 OR ASC(A$)>70 THEN PRINT "BAD NUMBER" : GOTO 1
10 : 'JUMP BACK INSTEAD OF RETURN BAD!!
440 IF A$="A" THEN A$="10"
450 IF A$="B" THEN A$="11"
460 IF A$="C" THEN A$="12"
470 IF A$="D" THEN A$="13"
480 IF A$="E" THEN A$="14"
490 IF A$="F" THEN A$="15"
500 RETURN
510 X=X/16 : X1=(X-INT(X))*16
520 X2=X1-INT(X1)
530 IF X2>=.5 THEN X1=INT(X1)+1
540 X=INT(X)
550 IF X1<=9 THEN A$=RIGHT$(STR$(X1),1) : RETURN
560 IF X1=10 THEN A$="A"
570 IF X1=11 THEN A$="B"
580 IF X1=12 THEN A$="C"
590 IF X1=13 THEN A$="D"
600 IF X1=14 THEN A$="E"
610 IF X1=15 THEN A$="F"
620 RETURN
```

MODEL II
The First Look

by Lance Micklus

There's some good news and some bad news inside the Radio Shack Model II. Although I've only had my machine for a few days, I can already tell you what you'll find when you get yours.

We were told that Model II BASIC would be upward compatible with Model I Disk BASIC. Radio Shack seems to have created a very wrong impression here. It is no more compatible than Model I Disk BASIC is with Microsoft's full 16K CPM BASIC on the Sorcerer.

For one thing, the graphics are set up much differently. The Model II has 32 graphic characters which are printed like any other character. SET and RESET graphic commands do not exist.

Noticeably missing are PEEK and POKE, usually standard in any big size BASIC. Moreover, you have no INP and OUT to get to the serial ports.

There are a lot of new BASIC commands which really don't add much. For example, SPC which prints spaces; SPACE$ which prints spaces; STRING$ which will print spaces; and TAB, which prints spaces. I would rather have TAB and STRING$, and drop the other two in favor of PEEK and POKE.

On the plus side is the fact that many DOS commands can be executed from BASIC, just like NEWDOS. Also, there is a very powerful command called SWAP which exchanges the value of two variables. SWAP is a programmer's delight.

At the DOS level, there are some goodies too. You must always enter the current date when you power up the system. Model II TRSDOS has a smart calendar which gives you not only the date, but which day of the year it is (like the 275th day of the year), and the day of the week. Also nice is the fact that the realtime clock is not affected by disk operations, thanks to DMA.

Another nice surprise is the fact that the directory tells you the creation date of the file. Personally, I think it should give you the date the file was last changed.

The FREE command gives you a map of which tracks are used, and which are free. Interesting, but I'd rather have it work like Model I TRSDOS and have another command (MAP maybe?) to give me the disk layout.

One thing really impressed me very much. I guess my eight years in television are showing now, but the Model II video monitor has a good DC restore in it. That means the picture does get dark if most of the screen gets whited out. This is a problem with the Model I monitor, especially if you are using a lot of graphics.

It does have nice upper/lower case letters with good resolution. You can also display any letter or graphic in black on white, or white on black. There is a 40 character per line mode, too. At the moment, there is no way to turn it on from BASIC, however.

For the machine language programmer, the Model II is a horror show. Calling sequences to the DOS will be changed in later releases of Model II TRSDOS. So, if I market an ST-80D for the Model II, I will have to provide every user with an update when new releases of TRSDOS come out.

If you are planning to run BASIC programs on your Model II, and most of you will, I hope you ordered yours with 64K of memory. TRSDOS and BASIC take up around 30K!!! And speaking of BASIC programs, you're going to have a hell of a time trying to transfer them from your Model I to your Model II. Of course, ST-80D will fix that up when it is ready, which won't be for a couple of months.

On a more detailed level, Radio Shack changed the specifications for TRSDOS on both versions of the TRS-80. A very useful feature, found only in VTOS 3.0, is device independent I/O. This means you can set up the DOS to send all output for the printer to a disk file. Later on, you can print the disk file, or kill it if the run was bad. It saves a lot of decision trees and time. Unfortunately, Radio Shack is dropping this feature, which is too bad.

Speed is emphasized in the Radio Shack ads on Model II. Well — don't expect miracles. Yes, it runs faster in memory, but it won't blow you away. Disk I/O speed is no better than the Model I if you are using non-Radio Shack drives and have modified the DOS to run at the faster access speeds (see October PROG. 80). The truth is, we really can't take full advantage of the DMA.

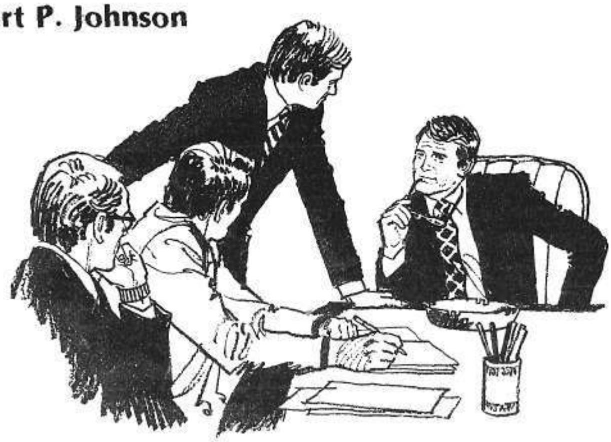
The keyboard and the screen are not memory-mapped. Each uses an LSI (Large Scale Integrated) chip to do its job, leaving the Z-80A free to be a computer. Good thinking, Radio Shack. A nice feature is that the keyboard is connected by a DIN connector, so you can put the keyboard as close or as far away from the monitor as you want. You can even put the keyboard on your lap.

Over all, the TRS-80 Model II looks like a winner for Radio Shack. But its success is going to depend more on company policy than anything else. Specifically, they have to realize that they can't do it alone. Without all of the Level IV's, SoftSide magazines, Small Business System Groups, and Lance Mickluses — the Model II will never grow and develop into the excellent computer system it was designed to be. As is often the case, the engineers can build it, but management may not know how to sell it. And that's what's going to make or break the Model II. ■

SOFTWARE REVIEW

Blaiseing Along In Pascal

by Robert P. Johnson



Now that you have mastered Level I, Level II, and perhaps DISC BASIC, what new worlds remain to be conquered? For TRS-80 owners, Computer Information Exchange, Inc. has made available, at very low cost, an introduction to the greatest programming language since sliced bread" — **Tiny Pascal.** This scaled down version of Pascal was originally developed by Kin-Man Chung and Herbert Yuen and published in the September, October and November, 1978, issues of Byte magazine. In intent, Tiny Pascal bears the same relation to Pascal that Tiny BASIC does to the full-blown versions of BASIC (Level I vs. Level II). Thus Tiny Pascal (thru C.I.E.) offers the budding programmer the opportunity to become acquainted with Pascal without the major investment for 48K memory, 2 disc drives, and $150 for the UCSD version of the big Pascal. People's Pascal I & II run on the Level II, 16K, TRS-80.

C.I.E., Inc. offers two versions of Tiny Pascal: People's Pascal I (under license from Pipe Dream Software), a Pascal compiler written in BASIC for $15.00; and People's Pascal II (under license from Super Soft) in machine language for $23.50. Initially C.I.E. informed callers that the two versions were identical, except that:

1. The machine language version compiles faster, and
2. The machine language version (People's Pascal II) included 32K and 48K compilers on the cassette.

As it turns out, the preceding statements are false. There are no 32K and 48K compilers on the flip side of People's Pascal II. Instead you get a backup copy of side A. Second, the monitor commands differ between the two versions. Third, the People's Pascal I compiler is (gasp!) a **line-numbered** version of Tiny Pascal. So your purchase decision will have to be made on grounds other than just price (e.g.,flip a coin, or that you don't have a DOS that allows you to copy machine language tapes onto your disk).

Now that you have parted with $23.50 for People's Pascal II what do you get when you tear open the package? You get a copy of TRS-80 Computing (Vol. 1, No. 4) that is devoted entirely to People's Pascal (both versions), a slip of paper telling you that you didn't get the big compilers (until you read this, you thought you were getting the big compilers too), a pamphlet explaining the superiority of C.I.E.'s cassettes, and a cassette containing People's Pascal II. Also, you now have problems.

First: Volume One, Number Four of **TRS-80 Computing** is NOT an "Idiot's Guide to Tiny Pascal". Both the BASIC and machine language versions are discussed – but not well identified as to which is which.Second: if you don't already know how to program in Pascal, this 16 page pamphlet will not teach you how (well, maybe it will teach some of you). The situation is made worse by the lack of good introductory books to Pascal. Unlike BASIC, FORTRAN, COBOL, etc. there are not hundreds of books on Pascal. Maybe next year . . . Hopefully, the existence of People's Pascal will encourage the writing of such books –or even Pascal tutorials in some of the magazines (hint-hint PROG/80!). Alternatively, you can lurk around your local Apple II dealer and see what they come up with in the way of a basic Pascal manual.

Moving right along, you load your People's Pascal II tape to see what you've got. If you are like me ("immediate gratification"), you will soon conclude that you have a %&# faulty cassette. From experience, I can assure you (almost) that the cassette is O.K. The Pascal compiler does not start to load until the tape has run for almost a minute. Finally, success! The Pascal package, plus a sample program, are loaded. Following the directions in TRS-80 Computing, you compile and run this program and are treated to an interesting graphics display . . .

You: "Look, dear; what fast graphics!"

Wife: "Is that it? Big deal!"

After you run this program a few more times you can load the next demonstration program, compile it, and run it. This program plots Hilbert curves. Not being a matheematician, I don't know what Hilbert Curves are, what they're for, or why I'm looking at them. Nobody tells me either. O.K., on to the last demonstration program: the game **Blockade.** If you've played this game before, you may enjoy it. Especially after you puzzle through the program listing on page 4 of **TRS-80 Computing** and figure out how to control the drawing of the lines. If you've never played Blockade and don't know Pascal, it is a mildly pleasant exercise in detection to figure out what is going on. It is not clear why these programs are included. Presumably they demonstrate some qualities of Pascal - but what?

To sum up - should you buy this version of Tiny Pascal? Sure, why not? In fact, buy **both** People's Pascal I and II. The reasons are numerous:

- The cost is low ($39.00 total for both).
- You're going to want to learn another language anyway. It is depressing having everyone badmouthing the only language you know. Strike back! Now you can badmouth FORTRAN and APL.
- Remember all those articles in the magazines on Pascal that you didn't read because you only had BASIC? now you can go read them. It will give you something to do until David Lien writes a Pascal manual for the TRS-80.
- Now you can stand up to other microcomputer owners. No color capability? No high resolution graphics? So what! Now for a few paltry cents, YOU CAN RUN PASCAL!! (well, sort of).
- If you thought Adventure was a challenge, learning Tiny Pascal on your own will **really** captivate you.
- The People's Pascal software does what it says it will do. It will give you the capability of running Pascal (and learning in the process). Learning it is up to you. In a world of misleading advertising, People's Pascal delivers what it promises (but no more). ■

People's Pascal I and People's Pascal II, available from computer Information Exchange, Inc.; P.O. Box 158, San Luis Rey, CA 92068 ($15.50 and $23.50, includes postage & handling)

ABOUT OUR COVER...

"An error. The source of an error can be human, it can be in the computer, which occasionally has an electronic failing, or it can be in the plotter, which may have difficulty in reading the computer-produced magnetic tape."

Fig. 29, Page 17 from the book "Computer Graphics" by Melvin L. Prueitt.

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

NEW! utility package from NEPENTHE PROGRAMS

Programming utility for professional programming.

File Manager 80 will organize tape and disk file routines, print a dictionary of items specified by you, edit items already specified and create record layouts using the items you have described. It will even create the code for your input/output routines, including format, sub-record, get, and put routines, and store them as temporary files to merge into your programs. It will maintain a dictionary of the subroutines you create and the variables used.

Minimum configuration is one disk and 32K of memory, but two disks are strongly recommended. Comes on disk with manual for $49.95.

TSE Order Form

Special prices in effect 60 days from mailing

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

☐ Check ☐ VISA ☐ Master Charge

☐ Money Order

*** DOES NOT APPLY TO HARDWARE**

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

Not responsible for typographical errors

THE SOFTWARE EXCHANGE

6 SOUTH STREET MILFORD, NH 03055

Order toll free: 1-800-258-1790

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

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

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

All prices are subject to change without notice. We are not responsible for typographical errors, including incorrect prices.

Charge card account number

Signature..

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

Charge customers: Please fill in account information above and below

Name..

Address..

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

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

PROG/80

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

U.S. Postage
Paid
- Bulk Rate -
Permit No. 21
Milford, NH 03055