Basic Computing

The TRS-80 User Journal

8 COMPLETE GAME PROGRAMS

SOFTWARE JARGON EXPLAINED

OVATION: NEW TANDY 2000 SOFTWARE

FULL-FUNCTION MODEL 100 DISASSEMBLER

SORTED DIRECTORIES FOR TRSDOS 2.0

TELEPHONE DIALING PROGRAM

GAMES

HIGH-SPEED GRAPHICS

XXX

0 60959 02 0 318585

$3 per copy / February, 1984

The BEST goes one step beyond!

Thousands of you have already experienced the speed and power of the DOSPLUS* IV disk operating system for the TRS-80** Model 4 microcomputer. Now Micro-Systems Software introduces: DOSPLUS IV/A.

DOSPLUS IV/A is the same great disk operating system as the DOSPLUS IV already released, but with one big PLUS! DOSPLUS IV/A includes the M-ZAL*** Disk Editor/Macro-Assembler. The best disk operating system for the Model 4 combines with the best editor/assembler to bring you all the development software you should ever need for the Model 4 at one low price.

DOSPLUS IV retailed for $149.95, but DOSPLUS IV/A will retail for only $169.95! The greatest software combination imaginable for only a $20 increase in price! No-one should be without this package! Existing DOSPLUS IV owners can upgrade their packages for only $25.00.

Micro-Systems Software's DOSPLUS IV brings unmatched speed and power to the Model 4, and now MSS brings the TRUE MODEL 4 EDITION of M-ZAL, (a regular $149.00 value) the world's best machine language development system. Now there is NO other choice when seeking the best Model 4 operating system...

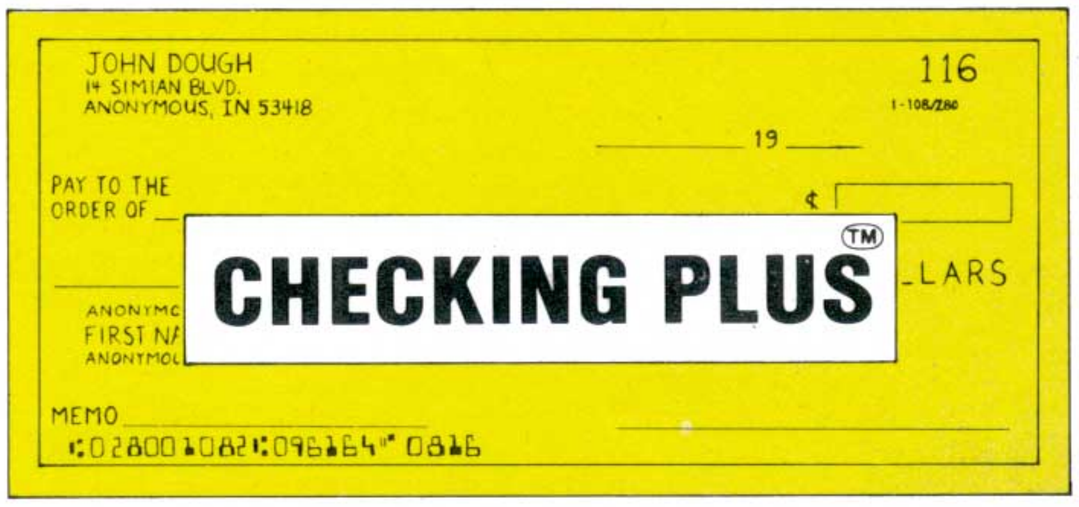
Dealer inquiries invited.

Trademarks:
* "DOSPLUS" and "6.0 PLUS" are trademarks of Micro-Systems Software Inc.
** "TRS-80" and "TRSDOS" are trademarks of Tandy Corporation
*** "M-ZAL" is a trademark of Computer Applications Unlimited

MICRO-SYSTEMS SOFTWARE, INC.
4301-18 Oak Circle, Boca Raton, Florida 33431, Telephone: (305) 983-3390
Orders Only 1-800-327-8724

ANYONE CAN USE OUR SOFTWARE!

MAIL PAC II and CHECKING PLUS represent a new generation of computer software. Our software doesn't even need a manual! Anyone can quickly learn to use either package by following simple, explicit on-screen instructions. But, for those of you who insist -- we've included a complete user's manual as well.



MAIL PAC II FEATURES:

High Capacity --

Stores from one to one million names in Zip Code, Numerical or Alphabetical order. The only limitation is your disk storage space.

Flexibility --

Prints your mailing list on 1, 2, 3, or 4-across labels (with up to 5 user-defined lines on each label) or as a compact, user-designed directory. The record length is completely user-defined, and each field within each of your records is completely variable (allowing storage of any number of characters for any particular address entry). Devote fields to telephone numbers, codes, or even special messages related to each particular name on file.

Powerful --

A built-in word processor allows you to create personalized form letters for each address on your list (or just a particular group of addresses).

Ease of Use --

Create new mailing lists, review existing lists, handle changes of address, delete cancelled names, sort lists, and purge duplicate names from your files. Complete on-screen instructions tell you in plain english exactly how to accomplish all of these tasks.

MAIL PAC II $99.95

CHECKING PLUS FEATURES:

Complete Check Register --

Checking Plus stores your entire check register in a disk file, and then uses the data to balance your account, track your expenses, and help you make budget projections. Review the entire checkbook, enter checks, deposits, fees and adjustments, mark outstanding checks when paid, and get an instant cash balance at any time. All data can be viewed onscreen or printed out in report form.

Tax Preparation --

Storage of monthly and yearly totals and other important information aids in income tax preparation, for your personal use or for your accountant.

Handles the Details --

Store names and addresses for frequently written checks, then print checks to fit standard window envelopes, eliminating the need for extra typing. You can even store any comment, explanation or other message (up to 255 characters) related to a particular check.

Automatic Monthly Bill Payments --

Enter amounts and names of payees for all of your fixed-cost monthly payments, and then sit back while the system automatically prints checks.

CHECKING PLUS $99.95

Both Programs Require 48K And At Least One Disk Drive (2 Drives For Higher Storage Capacities). And All Hard Disk Systems Are Supported.

FREE--TRS-80 Mod I,III & 4 programs supplied on DosPlus (minimum system). Complete DosPlus also available.

FOR YOUR TRS-80™ • APPLE™ • IBM PC™ • NEC™ • OSBORNE™ • XEROX™ • KAYPRO™ • TELEVIDEO™ • ZENITH™ • SANYO™
DEC™ • TI PROFESSIONAL COMPUTER™ • SUPERBRAIN JR.™ • EPSON™ • Any CP/M™ Computer

CP/M-based Computers must be equipped with Microsoft BASIC (MBASIC or BASIC-80)

TRS-80 trademark Tandy Corp. - APPLE trademark Apple Corp. - IBM PC trademark IBM Corp. - ATARI trademark Atari, Inc. - OSBORNE trademark Osborne Corp. - XEROX trademark Xerox Corp. KAYPRO trademark Non-Linear Systems, Inc. - TELEVIDEO trademark Televideo Systems, Inc. - SANYO trademark Sanyo Corp. - NEC trademark NEC Corp. - DEC trademark Digital Equipment Corp. ZENITH trademark Zenith Corp. - TI PROFESSIONAL COMPUTER trademark Texas Instruments, Inc. - SUPERBRAIN trademark Intertec Corp. - CP/M trademark Digital Reseaech - EPSON trademark Epson Corp.

OUR SOFTWARE CATALOG

H & E Computronics, Inc., has mailed more than 1 million software catalogs since 1978. Send $2 for our new 64-page catalog today! (We also send you our catalog FREE with every order). DEALER INQUIRIES WELCOME

30-DAY MONEY BACK GUARANTEE

*** ALL PRICES & SPECIFICATIONS SUBJECT TO CHANGE ***
DELIVERY SUBJECT TO AVAILABILITY

H & E COMPUTRONICS INC
50 N. PASCACK RD., SPRING VALLEY, N.Y. 10977

24 HOUR ORDER LINE
(914) 425-1535

ADD $3.00 FOR SHIPPING IN UPS AREAS
ADD $4.00 FOR C.O.D. OR NON-UPS AREAS
ADD $5.00 TO CANADA & MEXICO
ADD PROPER POSTAGE OUTSIDE OF U.S., CANADA & MEXICO

NEW TOLL-FREE ORDER LINE
(OUTSIDE OF N.Y. STATE)
(800) 431-2818

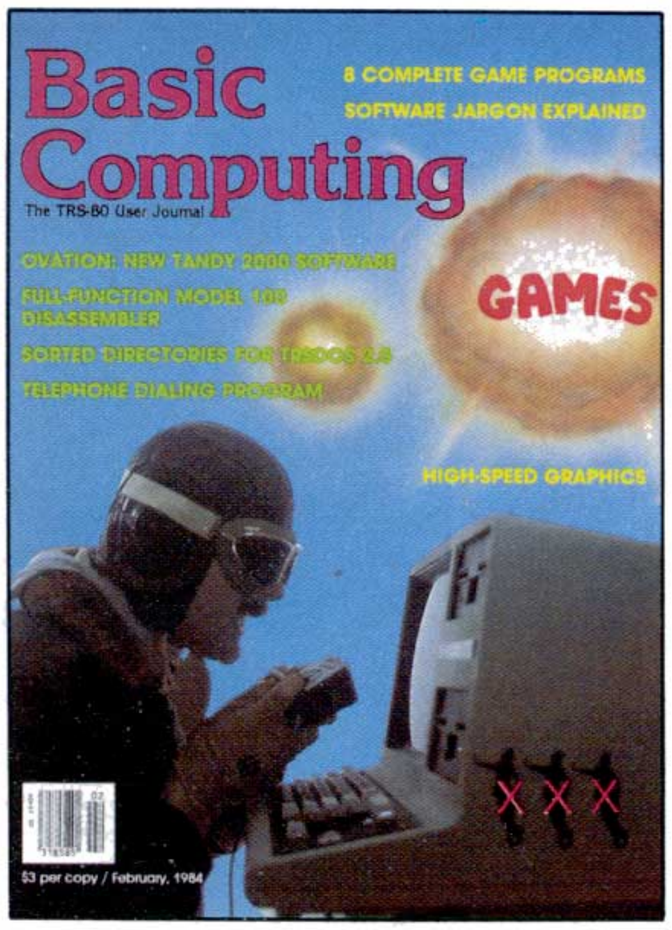
VISA MasterCard AMERICAN EXPRESS

Authors: We constantly seek material from contributors. Send your material (double spaced, upper/lower case, please) and allow approximately 4 to 6 weeks for review. Programs must be supplied in machine-readable form on diskette or tape, clearly marked as to model and operating system. Text files may be on diskette, but please include a hard copy as well. Media will be returned if return postage is provided. Cartoons and photographs are welcome. Generous compensation will be made for non-trivial works which are accepted for publication. Basic Computing pays upon acceptance rather than on publication.

SUBSCRIPTION PRICE: U.S.: $19.97 for one year, $34.97 for two years and $49.97 for three years. **Canada and Mexico:** $29 per year, no two or three year subscriptions are offered. **All other:** $36 per year via surface mail, $78 per year via airmail. Two and three year subscriptions are not offered.

ISSN Publication #0745-9912. Basic Computing is published monthly by 80-Northwest Publishing, Inc., 3838 S. Warner St., Tacoma WA 98409-4698. Printed in the United States of America.

POSTMASTER: Please send change of address form 3579 to Basic Computing, 5615 West Cermak Road, Cicero, Illinois 60650. **Second Class postage PAID at Tacoma, WA and additional entry points.**



Our cover for this month's games issue shows fighter pilot and photographer Frederick A. Johnsen of Tacoma, WA. The background airbrushed artwork is by Randy "Tarkas" Hoar of Centralia, WA.

Basic Computing

The TRS-80 User Journal

TRS-80 is a trademark of the Tandy Corp.

Vol. VII, No. 2 — February, 1984

ARTICLES

REVIEWS

DEPARTMENTS

Basic Computing

The TRS-80 User Journal

Publisher
I. Mike Schmidt

Managing Editor
Cameron C. Brown

Technical Editor
Greg Sheppard

Associate Editors
Terry R. Dettmann
Spencer Hall
Jim Klaproth

Contributing Editors
Timothy K. Bowman
Thomas Quindry
Mark E. Renne

Advertising Coordinator
Catherine Shappee

Advertising Representatives
East of the Mississippi River
Garland Associates:
John A. Garland, Frank Surace
(617) 934-6464

West of the Mississippi River
Catherine Shappee
Advertising Coordinator
(206) 475-2219

Promotion/Circulation
Robert P. Perez

Production
Catherine D. Doud

Accounting
Helen Dalton

Editorial Secretary
Julie Bartz

RENEWING? Check your label to be sure it's correct. For uninterrupted service, include your label with your order.

MOVING? Please enclose your label or write your name and address as it appears on your label.

Name _____

Address _____

City/State/Zip _____

Write in new address:

Name _____

Address _____

City/State/Zip _____

Basic Computing
Subscription Department, 5615
West Cermak Road, Cicero, IL 60650

Editorial

Cameron C. Brown

We have been getting disturbing reports from users. There are more and more complaints about obtaining updates and accurate information from local Radio Shack Computer Centers. In our October, 1983 issue (written in August), we told you that SuperScripsit 1.2 was available. It is now late December and we have found out that SuperScripsit 1.2.03 is now the current version. We applaud Tandy for supporting their product, but wonder why the patches, which were known in Fort Worth in April, are only now coming to light.

We also reported in December that TRSDOS 6.1 was available. One reader was told that it was only for hard disk purchasers and had no application to Model 4 floppy disk users. We found out from other sources that it is valuable to all users. Among other things, it fixes (for some machines), a key-bounce problem that Radio Shack has yet to acknowledge even exists on some Model 4s. We have been told that TRSDOS 6.1 has been shipped with all Model 4s since September, yet some Computer Centers do not make it available to early purchasers. Other readers, who purchased some of the earliest Model 4s, complain that the pages missing in their manuals have never been sent to them. Yes, they are registered owners.

We reported way back in December, 1982 that TRSDOS 2.8 was coming for Model I owners who have a single disk drive and use the Radio Shack Doubler Board. That was based on a letter from Fort Worth confirming that it was coming out. One reader cannot get his Computer Center to admit it ever existed.

I was attempting to update my Model II Scripsit by applying patches that were published in the *TRS-80 Microcomputer News*, Radio Shack's own newsletter. I was led from issue to issue, back to the phrase, "Do not apply these patches unless you have applied the February patches." The problem is that the February patches do not exist. My local store is completely unaware of them. They were kind enough to upgrade my disk at no charge, after I showed them the dead end that I had reached.

Obviously the software registration card is not enough. The newsletter is not enough. Asking a local store for help is not enough. Tandy needs to realize admitting an update exists is a positive act that helps the consumer, not an admission of failure. Every store should prominently post a list of current version numbers. Every time software is sold, its binder should be opened up to insure that the version is up-to-date. Software is what makes computing useful. Customers deserve, and pay for, up-to-date information and knowledgeable sales personnel. Tandy should insure that is what we receive every time we enter a store.

It is reasonable to charge for enhancements and new options. Updates that repair problems should be free but, if it is a matter of money, I would pay. I would rather spend a few dollars to kill a bug now than have it jump out and bite me later.

Letters to the editor

I read Juge's apologia. If confined to Tandy's own "magazine," that's fine. Why is a reputable magazine permitting this self-serving commercialism to creep into its editorial pages? And right after I was shouting, "Bravo" to your editorial on integrity. Very confusing signals you send out.

Lucien R. Greif
Chappaqua, NY

"Tandy Topics" has one of our highest responses to reader interest surveys. We agree with your assessment that it is clearly a Tandy point of view. But what is wrong with that? Mr. Ed Juge has never denied that he is providing material as Tandy sees it. He has always given our readers accurate and honest information, never stating facts he knew to be untrue. Would readers prefer no information at all? We think not. --Ed.

I feel very strongly that the patches, like those published in "Model 4 Tips and Tricks," November, 1983 (page 54) and "Notes, etc.," December, 1983 (page 8), should be well tested by the magazine prior to publishing. My program, which I advertised in two issues and got no response, was for a Model 4 speed-up done right. I hope you can keep this kind of information out of your fine magazine in the future.

Francis A. Desimone
Futuraware
Nashua, NJ

I received my first issue of *Basic Computing* today and already it has come in handy. Better yet, I can already contribute something back. Your note in the December issue about obtaining 4MHz speed in the Model III mode (Model 4III) is quite handy to perk up some aspects of SuperScripsit. However, in order to avoid the switching to the alternate graphics characters caused by the POKE 16912, 64, one should use a POKE 16912,72. I have enclosed a listing of a short program that facilitates rate switching. Listing 1 will work with either Newdos/80 or TRSDOS 1.3 (see line 45). Thanks for the info.

Mike Zarowitz
Department of Biochemistry
Univ. of Minnesota
St. Paul, MN

As with all information regarding patches and changes to programs or operating systems, they are given as is, with no warranty that they will work in all cases. Readers who wish to operate their Model 4 at 4MHz when in Model III mode are encouraged to read this month's "Notes, etc." column before applying any of the published patches. Regarding the first letter, such information is gathered by our readers during their investigations. We have no intention to stop publishing it. If your product did not draw a response, we are sorry. We deliver an audience. The decision to purchase is determined by them. --Ed.

Listing 1

```
5 CLS:A%=PEEK(16912):IF A%
=72 THEN PRINT"In 4 MHz mo
de":GOTO 10 ELSE IF A%=40
THEN PRINT"In 1.7 MHz mode
" ELSE PRINT"Mode Undeterm
inable"
10 PRINT:PRINT"Press <F>as
t or <S>low"
15 PRINT"      <W> for SCR
IPSIT"
20 PRINT"      <E>nd"
25 A$=INKEY$:IF A$="" THEN
 GOTO25 ELSE A%=INSTR("FfS
sWwEe",A$):ON A% GOTO 35,3
5,40,40,45,45,50,50
30 GOTO 25
35 POKE 16912,72:GOTO 5
40 POKE 16912,40:GOTO 5
45 PRINT:PRINT"Loading SCR
IPSIT":IF PEEK(16397)=103
THEN CMD"S=SCRIPSIT" ELSE
CMD"I","SCRIPSIT"
50 END
```

Notes, etc.

Cameron C. Brown

Model 4III Speed

We have reported, more than once, that the Model 4, when running in the Model III mode, can be made to run at a faster clock speed. In the December, 1983 Notes, etc. column, we said to try POKE 16912, 64, but warned about possible disk I/O problems. In the November, 1983 article, Model 4 Hints and Tricks, Al Mashburn pointed out that a POKE &H4210, 249 would also work. (Both tips are to be executed from within BASIC). Mr. Mashburn's tip worked fine on LDOS and NEWDOS, but TRSDOS 1.3 has problems with it. The clock will be displayed, programs will load and run faster, but saving a file can cause the disk to be destroyed. Ours wouldn't even reboot.

Mr. Jerry Latham of Midwest City, OK was able to decipher the problem. He wrote to say: "Location 4210 Hex is an image of the bit configuration that is to be sent out port EC during certain operations. Poking a fixed value into that location can cause unexpected results. The proper way is to examine the contents of the memory location and use the OR command to set the bit you are interested in. To change the clock speed to the 4MHz mode, bit six needs to be altered. To set this bit, without affecting the status of the other bits, the syntax from BASIC would be POKE &H4210, PEEK(&H4210) OR 64. To reset that bit, and only that bit, the BASIC instruction would be POKE &H4210, PEEK(&H4210) AND NOT 64.

The individual bit uses for the location 4210 Hex are given on page 18 of the Model 4 Technical Reference Manual. Bits 7, 5, and 0 are not used. Bit 6 is for CPU fast. Bit 4 enables EX I/O. Bit 3 enables alternate set. Bit 2 is video select mode (32 or 64 characters per line). Bit 1 is cassette motor on." Mr. Latham went on to say that he disagreed with the definition of bit 0 being not used. He has found that setting that bit enables the clock display on the video. The point he makes is well taken. He concluded with, "Poking fixed values into locations can cause other things to happen besides what is desired. By using the OR and the AND NOT instructions, you can control what will happen. With LDOS and TRSDOS 6.x.x, you can bet that memory locations are used as some flag, or the individual bits have meanings assigned to them. In all cases, it is much safer to use the logical operators to set and reset the bits, rather than picking a fixed pattern." Our thanks to Mr. Latham for clearing up the problem.

Good Company Award

Last month, we asked you to let us know which company, or companies, have provided you with excellent service or support. We are not looking for a product recommendation. There are many good software and hardware items out there. What we want to know is which company has earned your praise. To let us know, send in the "user-friendly" reader response card that is in this issue. Be sure that your vote reaches us by April 6, 1984. The results will be in our July issue.

Corrections and Updates

Quick Find, November, 1983, page 88, needs two changes. In line 130 of Listing 2, the numeral 2 should be changed to a quote mark ("). In Listing 4, line 50, the second equal sign, =, should be changed to a minus sign.

Break-Break, December, 1983, page 86, caused some readers to tell us about other methods they have devised. Mr. Robert Bequette recommends POKE 16396, 165. Pressing the break key does nothing, but pressing shift and break will work as normal. Mr. Lucien Greif has an easy hardware change. Just pop the break key-cap off and replace the spring with one that is quite a bit stronger. That way, an operator will have to press down quite hard to invoke the break key.

Basic Bits, December, 1983, page 94, had a bug in line 110 of Listing 2. The code >65 should be changed to >64. As published, the program would not allow the "unkilling" of any file whose name started with the letter A (ASCII 65).

Exploring VisiCalc, December, 1983, had some typos in the next to last paragraph on page 72. The code for the Epson printer is SHIFT@ H0F, not HOF as printed. Also, the reference in that paragraph should be to the LPVIII, not LPVII. We received many letters about that month's column. Be sure to look at this month's Exploring VisiCalc for updates and more information on how to set printer format codes for various models of computers and printers.

Puzzler

Our November, 1983 puzzler asked for a routine that would check a string to see if it was palindromic. We had well over a hundred submissions, and the winner was selected at random from those that were correct. Our congratulations to Mr. Dale Rogerson of Lexington, SC. A very interesting solution came in from Mr. Bill Pringle of West Simsbury, CT. His three pages of code were written on a Model I using Pascal 80, modified, and then run on a Perkin-Elmer 3242 since his Model I did not have a printer. Most solutions involved a few lines of BASIC, but as all puzzler solvers know, there is more than one way to succeed.

This month we want to see if you can write, in BASIC, a program that will reverse a byte (any integer from 0 to 255, inclusive), a bit at a time. For example, the decimal integer 65, is 0100 0001 in binary. Its bit reversal is 1000 0010, the decimal integer 130. Our thanks to Mr. Dave Mc Glumphy for the suggestion. Send your sample code, with examples showing the bit reversal of the numbers 0, 3, 100, and 254. Send solutions to February Puzzler, c/o *Basic Computing*, 3838 So. Warner, Tacoma, WA 98409. Do not send tapes or disks, no material is returned. The winner receives a free, six-month subscription to *Basic Computing*.

Fun 'n games

Some shorties to whet your appetite

Never say die

A strategy game where you must get the right total to win

Models I/III/4III

David Lewis

Tired of games that wear out your fingers without challenging your mind? Is your video screen starved for some three-dimensional action?

Enter Dicegame. A game of strategy, Dicegame is played on a three-dimensional cube. The die, which starts off randomly rolled, is drawn on your screen, and digits indicate the values of the sides. You then enter into pitched battle with the computer.

The die is moved by quarter-turns (that means, no flipping it over); the top value is then added to the running total. The object is to turn the die so that the running total becomes 31, or to turn it so that the computer must go over 31. It is really not as easy as it looks. You will have to contend with spatial distortion and ply-moves, not to mention cosmic radiation.

A Few Words About the Program

The program is very straight-forward. After initialization, turns alternate. The computer does not allow an illegal move (line 120). Wins are checked for in a bit of logical programming in line 5000; score is kept.

The values of the sides are stored in the array A(1 to 6). When a legal move (M) is made, the values are rotated to simulate the turning of the die. They are then replotted.

I had to insert time delays to slow the game down. You may care to delete the FOR . . . NEXT loops in lines 300, 305, and 190.

The computer has some digital-root strategy built-in, so you'll have to think to win. Die-hard gamesters will find Dicegame a challenge. To the rest of us, it will be a dicey challenge.

Program Listing for Dicegame

```
1 ' DICEGAME BY DAVID LEWIS, BOX 88, SHA
DY, NY  12479
10 GOSUB8000 'INTRODUCTION
20 CLS:PRINT@448,;:INPUT"Do you want to
go first";F$:CLS:RT=0:RESTORE:RANDOM
30 GOSUB4500:GOSUB7000 'initialize and d
isplay die and values
40 PRINT@20,"RUNNING TOTAL:";:PRINT@0,"M
E:";:PRINT@50,"YOU:"
50 PRINT@3,CS;:PRINT@54,HS;:PRINT@34,RT;
90 IF LEFT$(F$,1)="N"THEN200 'GOTO COMP'
S MOVE. ELSE, HUMAN'S.
93 '
95 ' HUMAN'S MOVE
100 PRINT@218,"YOUR MOVE";:PRINT@34,RT;
110 PRINT@896,;:INPUT"What number do you
 want on top";M
115 IFM<1ORM>6THEN110
120 IF M=A(1)ORM=A(6)THENFORG=1TO3:PRINT
@218,"*ILLEGAL*";:FORF=1TO100:NEXT:PRINT
@218,"          ";:FORF=1TO100:NEXT:NEXT:
GOTO100
130 GOSUB6000:RT=RT+M'RE-DO DISPLAY
140 GOSUB5000:IFL=-1THENCLS:PRINT@475,"I
 WIN!":CS=CS+1:GOTO5500
        ELSEIFL=1THENCLS:PRINT@475,"YOU
WIN!":HS=HS+1:GOTO5500
190 PRINT@34,RT;:FORG=1TO500:NEXT
195 '
200 PRINT@218,"MY MOVE  ";
210 PRINT@896,STRING$(42,32);:PRINT@896,
"I turn the die to";
230 IFRT=30ANDA(1)<>1ANDA(6)<>1THENM=1:G
OTO300
240 IFRT=29ANDA(1)<>1ANDA(6)<>1THENM=1:G
OTO300 'SELECT 1
250 M=31-RT:IFM<=6ANDRT<31ANDA(1)<>MANDA
(6)<>MTHEN300ELSEM=22-RT:IFM<=6ANDRT<22A
NDA(1)<>MANDA(6)<>MTHEN300
                ELSEM=13-RT:IFM<=6ANDRT<
13ANDA(1)<>MANDA(6)<>MTHEN300
                                ELSEM=4-RT:I
FM<=6ANDRT<4ANDA(1)<>MANDA(6)<>MTHEN300
260 IF A(1)<>1 AND A(6)<>1THENM=1ELSEM=2
295 'MOVE FOUND BY NOW
```

```
300 FORG=1TO500:NEXT:PRINTM;:GOSUB6000 '
RE-DO DISPLAY
305 FORG=1TO500:NEXT
310 RT=RT+M:GOSUB5000:IFL=-1THENCLS:PRIN
T@475,"YOU WIN!":HS=HS+1:GOTO5500
        ELSEIFL=1THENCLS:PRINT@475,"I WI
N!":CS=CS+1:GOTO5500
320 GOTO100
330 '
4495 'INITIALIZATION OF DIE FOR THIS GAM
E
4500 A(1)=RND(6):A(6)=7-A(1)
4510 A(2)=RND(6):IFA(2)=A(1)ORA(2)=A(6)T
HEN4510ELSEA(5)=7-A(2)
4520 A(3)=RND(6):IFA(3)=A(1)ORA(3)=A(6)O
RA(3)=A(2)ORA(3)=A(5)THEN4520ELSEA(4)=7-
A(3)
4524 RT=RT+A(1)
4530 'NOW DISPLAY DIE
4540 READAD,PQ:IFAD=-1THENRETURNELSEPOKE
AD+15360,PQ:GOTO4540
4600 DATA407,160,408,158,409,131,410,131
,411,131,412,131,413,131,414,131
4610 DATA 415,131,416,131,417,131,418,16
3,419,159,420,191,470,184,471,135
4620 DATA 481,184,482,135,484,191,534,19
1,535,131,536,131,537,131,538,131
4630 DATA 539,131,540,131,541,131,542,13
1,543,131,544,131,545,191,548,191
4640 DATA 598,191,609,191,612,191,662,19
1,673,191,675,160,676,159,726,191
4650 DATA 727,176,728,176,729,176,730,17
6,731,176,732,176,733,176,734,176
4660 DATA 735,176,736,176,737,191,738,18
4,739,135
4670 DATA -1,-1
4990 '
4995 'CHECK FOR WIN, LOSS
5000 L=-(RT=31)+(RT>31):RETURN
5490 '
5495 'ASK FOR ANOTHER GAME
5500 PRINT@896,;:INPUT"AGAIN";F$:IFLEFT$
(F$,1)<>"N"THEN20ELSEPRINT"SO LONG.":END
5990 '
5995 'ROTATE CUBE
6000 IF A(2)=MTHENG=A(2):A(2)=A(6):A(6)=
A(5):A(5)=A(1):A(1)=G
6010 IF A(5)=MTHENG=A(5):A(5)=A(6):A(6)=
A(2):A(2)=A(1):A(1)=G
6030 IF A(3)=MTHENG=A(3):A(3)=A(6):A(6)=
A(4):A(4)=A(1):A(1)=G
6040 IF A(4)=MTHENG=A(4):A(4)=A(6):A(6)=
```

```
A(3):A(3)=A(1):A(1)=G
6995 'DISPLAY DIE'S VALUES
7000 PRINT@476,A(1);:PRINT@667,A(2);:POK
E610+15360,A(3)+48:PRINT@658,A(4);:PRINT
@550,A(5);:PRINT@795,A(6);
7010 RETURN
7990 '
7995 'INTRODUCTION
8000 CLS:GOSUB4540:PRINT@602,"DICE";:PRI
NT@666,"GAME";
8010 PRINT@741,"BY DAVID LEWIS";
8030 FORG=1TO1000:NEXT
8050 CLS:PRINT@69,"DICEGAME is a three-d
imensional challenge.":PRINT
8060 PRINTTAB(5)"We play the game with a
 single die, and add points to the runn
ing total by turning the die a quarter-t
urn (that is, movingone of the numbers a
round the sides to the top) and adding t
he value on top to the total.";
8077 PRINT
8080 PRINT:PRINT"The object -- a real ch
allenge to your thinking powers -- is to
 either reach 31 or force me to go above
 31.  I'll try to do the same to you.":P
RINT
8085 PRINT"Even die-hard gamesters will
find DICEGAME a challenge -- for  the r
est of us, it will be a dicey problem!"
8090 PRINT:INPUT"READY";F$:F$="":RETURN
```

The castle

A pocket fantasy adventure

PC-1

Laurence Klein

This PC-1 program, The Castle, is a fantasy adventure program in which a player roams a castle in his quest for the Great Urn, doing battle with any "orcs" he encounters.

The display tells the player his immediate surroundings, using N, S, E, and W for north, south, east, and west, and WAL, HAL, TRN, INT, and URN for wall, hall, turn, intersection, and Urn. To move, the player enters 8 to go north, 2 to go south, 6 to go east, and 4 to go west (following the layout of the keypad). If the player attempts to move into a wall, the display will tell him that he cannot do so.

If the player encounters an orc, he must try to strike him by swinging his sword at one of the ten possible positions of the orc. The player enters this position, using the numbers 0 to 9. If his aim is true, or slightly off, he will vanquish his foe. He has a 40 percent chance of this. If he misses, the orc will attack him, and the player will lose one of his vitality points (he begins the game with 8 vitality points). The battle then continues.

The game ends either when the player finds the Urn, or when he loses all of his vitality points in battle. In either case, the total number of defeated orcs is included in the final display.

Program Listing for The Castle

```
10 LET A$="WAL",B$="HAL",C$="TRN",D$="IN
T",E$="URN"
20 FOR M=27 TO 92:A(M)=1:NEXT M
30 LET A(39)=3,A(40)=2,A(41)=2,A(42)=3,A
(44)=3,A(45)=2,A(50)=2,A(53)=3,A(54)=2
40 LET A(55)=4,A(57)=5,A(58)=3,A(61)=2,A
(63)=2,A(66)=2,A(69)=2,A(72)=3
50 LET A(73)=2,A(74)=4,A(75)=2,A(76)=2,A
(77)=4,A(78)=2,A(79)=2,A(80)=3
60 LET J=5,K=5,G=5,H=0,I=0,U=8,Z=0
70 PAUSE "THE CASTLE":GOTO 150
80 INPUT "WHICH WAY?";G
90 IF G=8 LET H=0,I=-1
100 IF G=5 LET H=0,I=0
110 IF G=6 LET H=1,I=0
120 IF G=2 LET H=0,I=1
130 IF G=4 LET H=-1,I=0
140 IF G=5 GOTO 80
150 L=(J-1)+(K-1)*11+27:IF A(L+H+I*11)=1
 PAUSE "THAT IS A WALL!":G=5:GOTO 80
160 J=J+H:K=K+I:L=(J-1)+(K-1)*11+27:IF A
(L)=5 BEEP 4:PRINT "VICTORY! ";Z:END
170 IF L=40 GOTO 240
180 IF L=53 GOTO 240
190 IF L=58 GOTO 240
200 IF L=61 GOTO 240
210 P=A(L-11),Q=A(L+11),R=A(L-1),T=A(L+1
)
220 PRINT "N:";A$(P);"S:";A$(Q);"W:";A$(
R);"E:";A$(T)
230 G=5:GOTO 80
240 PAUSE "ORC ATTACK!"
```

```
250 W=    (H+J+Z),S=(W-INT(W))*10
260 INPUT "WHERE IS HE?";N
270 O=ABS(S-N):IF O<2 BEEP 2:PAUSE "YOU
GOT HIM!":Z=Z+1:GOTO 210
280 U=U-1:BEEP 1:PAUSE "YOU HAVE ";U;" P
OINTS"
290 IF U=0 PRINT "YOU ARE DEFEATED!  ";Z
:END
300 GOTO 260
```

Magic squares

A graphic remember-the-pattern game

Models I/III/4III

Mike Lynne

Centuries ago, mankind began the unending quest to find the ultimate game. A short time ago, I decided to drop out of this quest and settle for computerizing a favorite game of mine. This game is Magic Squares on Merlin™, a hand-held electronic game.

My program started out as a subroutine for a large program I was going to write called TRS-80 Merlin. I questioned why a person should put six AA batteries in the back of a small, hand-held toy to enjoy hours of endless fun, when he can just program it all into a $2,000 computer and seem so much more intelligent.

The program works the same as Magic Squares, except that it has no sound. Try it. You may like it.

Program Listing for Magic Squares

```
10 REM ******************
20 REM WRITTEN BY MIKE LYNNE
30 REM A.D. 1982
40 REM ******************
50 CLS : CLEAR 100
60 REM ****** INSTRUCTIONS *******
70 PRINT"YOUR GOAL IS TO GET THE
   * * *"
80 PRINT"SQUARES TO MATCH THIS :
   *   *"
90 PRINT"All the outside ones lit
   * * *"
100 PRINT"Press the numbers shown to cha
nge it. Each square controls a    certai
n section of squares."
110 REM ******* READ DATA *********
120 FOR I= 1 TO 9 : READ A(I),B(I),C(I),
D(I) : NEXT
```

```
130 DATA 206,270,275,334,218,282,287,346
140 DATA 230,294,299,358,462,526,531,590
,474,538,543,602
150 DATA 486,550,555,614,718,782,787,846
,730,794,799,858,742,806,811,870
160 GOSUB 180 : IF Q$<>" " THEN 160 ELSE
 210
170 REM ******* SUB ROUTINE *********
180 PRINT:PRINT"Press SPACE bar to conti
nue."
190 Q$=INKEY$ : IF Q$="" THEN 190 ELSE R
ETURN
200 REM ******** GAME STARTS HERE ******
***
210 CLS : PRINTTAB(21)"MIKES MAGIC SQUAR
ES"
220 REM *** DRAW BOARD ***
230 XL=24 : XR=97 : YT=7 : YB=43
240 SET(XL,7) : SET(XL,31) : SET(XR,19)
: SET(XR,43) : IF Z=1 THEN 260
250 IF YT>43 THEN Z=1 ELSE Z=0 : SET(24,
YT):SET(25,YT):SET(73,YT):SET(74,YT):SET
(48,YB):SET(49,YB):SET(96,YB):SET(97,YB)
260 XL=XL+1 : XR=XR-1 : YT=YT+1 : YB=YB-
1 : IF XR=25 THEN 280 ELSE  GOTO 240
270 REM *** READ BOX NUMBERS,PRINT POSIT
IONS ***
280 FOR I=1 TO 9 : READ XX : READ A : PR
INT @ A,XX;:NEXT
290 DATA 7,273,8,285,9,297,4,529,5,541,6
,553,1,785,2,797,3,809
300 REM *** PLACE MARKERS ***
310 FOR I= 1 TO 9 : X= RND(50)
320 IF X<26 THEN P(I)=1 ELSE P(I)=0
330 GOSUB 640
340 NEXT I
350 REM *** WAIT FOR MOVE ***
360 GOSUB 190
370 REM *** FIND KEY PRESSED ***
380 IF Q$="7" THEN 490
390 IF Q$="8" THEN 500
400 IF Q$="9" THEN 510
410 IF Q$="4" THEN 520
420 IF Q$="5" THEN 530
430 IF Q$="6" THEN 540
440 IF Q$="1" THEN 550
450 IF Q$="2" THEN 560
460 IF Q$="3" THEN 570
470 GOTO 360
480 REM ******** TURN BOXES ON/OFF SUB R
OUTINES **********
490 I=1 : GOSUB 640 : I=2 : GOSUB 640 :
I=4 : GOSUB 640 : I=5 : GOSUB 640 : GOSU
B 590 : GOTO 360
500 FOR I= 1 TO 3 : GOSUB 640 : NEXT I :
 GOSUB 590 : GOTO 360
510 FOR I= 2 TO 6 : IF I=4 THEN NEXT I E
LSE GOSUB 640 : NEXT I : GOTO 590
520 FOR I=1 TO 7 STEP 3 : GOSUB 640 : NE
XT I : GOTO 590
530 FOR I=2 TO 8 : IF I=3 OR I=7 THEN NE
XT I ELSE GOSUB 640 : NEXT I : GOTO 590
540 FOR I= 3 TO 9 STEP 3 : GOSUB 640 : N
EXT I :GOTO 590
550 FOR I= 4 TO 8 : IF I=6 THEN NEXT I E
LSE GOSUB 640 : NEXT I : GOTO 590
560 FOR I= 7 TO 9 : GOSUB 640 : NEXT I :
 GOTO 590
570 FOR I= 5 TO 9 : IF I=7 THEN NEXT I E
LSE GOSUB 640 : NEXT I : GOTO 590
580 REM ***** CHECK TO SEE IF YOU WIN **
****
590 IF P(1)+P(2)+P(3)+P(4)+P(5)+P(6)+P(7
)+P(8)+P(9) <> 9 THEN 360
600 PRINT @ 960,"YOU WIN!!   TRY AGAIN <
Y/N> ";CHR$(140);:PRINT@ 987," ";
610 GOSUB 190 : IF Q$<>"Y" AND Q$<>"N" T
HEN 610 ELSE PRINTQ$;:IF Q$="Y" THEN PRI
NT @ 960,"
";:GOTO 310
620 END
630 REM ******** CHANGE VARIABLES/BOXES
SUB ROUTINE ********
```

How To Enter Our Listings

Our program listings come directly from the submissions of our authors. We do not edit them at all (that's why you sometimes see spelling errors in them). We run all submissions and make sure that they do work.

To enter one of the listings given make sure you have the type of computer specified and all necessary programs, operating systems, or hardware that the program uses. Type in the program exactly as it appears in the magazine. Be extra careful so you do not confuse 0 (zero) with O or 1 (one) with I or L. Save the program to tape or disk **before** running it. On long programs it is wise to save it as you go along, thus protecting yourself from having to re-enter the whole program if the lights go out.

Here are some tips to help you catch errors that you may have made in typing. If you get an out of data error, the problem lies in the DATA statements, rarely in the READ line that the computer refers to. Check all DATA lines to see that they are correct and that no commas or values are missing. It might be useful to print each variable after it is read, that way you can follow the computer as it goes through the data. Just insert a :PRINT **variable** right after the READ **variable** command.

Many of our authors use a linefeed, or downarrow, in their programs. If you see lines of code that have many blank spaces and then they begin again on the next line with more code, a linefeed was used. Even if you don't use them, the program will run but the video display may be messed up.

You will find the TRON command helpful in following the program's logic. By turning the trace command on, TRON, you can see what lines are being executed by the program. It is very useful in catching GOTO or GOSUB errors and incorrect references to linenumbers. Don't worry about video formatting when the trace is on, it will be quite messy.

If you find yourself getting TM or type mismatch errors, check carefully the use of the $ symbol. Also look at the beginning of the program to see if you correctly entered the DEFINT or DEFSTR statements.

Function call errors usually occur when a variable has a value that is not allowed. Check all variables that are being used by the function, one of them probably has the wrong value.

If after all that, you can't get it to run, send us a paper listing of your program, what systems you are running it on, and carefully document the error you are getting. We will do what we can to find the flaw. It is very difficult for us to try to help you debug errors over the phone. Check Letters and Notes, etc. in the next few issues for updates or conversions. Many times a reader will tell how to embellish a previously published program.

```
640 IF I=5 THEN 670
650 IF P(I)=1 THEN X=128 : P(I)=0 ELSE X
=191 : P(I)=1
660 PRINT @ A(I),STRING$(9,X);:PRINT @ B
(I),STRING$(4,X);:PRINT @ C(I),STRING$(4
,X);:PRINT @ D(I),STRING$(9,X); : RETURN
670 IF P(I)=1 THEN X=191 : P(I)=0 : ELSE
 P(I)=1 : X=128
680 GOTO 660
```

Plus1

A generate-the-longest-path game with sound and enough twists to make it tough

Models I/III/4III

David Lewis

PLUS1 is adapted from a hand-held electronic game that was popular some months ago. You are pitted against the computer in a battle of wits. The object of the game is to create the longest possible path. There is a catch, though. Every other move is backward.

The computer generates a pattern of blocks on the screen and designates one of them as the starting-block. The journey of a thousand moves begins with a single step. Let's say you move the cursor up (by pressing the arrow key). The computer retraces your path, by going down, and then it adds a move of its own; perhaps to the left. You retrace the path by going right and up, then you make another move of your own. The object is to create the longest possible chain of moves (without making a mistake) in this series of retracing the opponent's path backward and adding your own move.

The Program

Though the game is twisted, the program is rather straight-forward. A sound routine is provided to help you keep track of the moves. (It is general-purpose; you can use it in your own programs.) It is read in from line 15000 on. The program checks the value of the memory address 14400 to see which arrow key is being pressed. The move that each player makes is added to the opposite ends of two strings; the player's move must duplicate one, and the other string is used for the computer to move. The computer chooses its moves randomly. In either case, the validity of the move is checked in lines 1000 on.

I've found PLUS1 an excellent challenge to my thinking abilities. I hope you enjoy it.

Program Listing for PLUS1

```
5 'PLUS1/BAS BY DAVID LEWIS, BOX 88, SHA
DY, NY  12479
7 CLEAR300:GOSUB15000
10 GOSUB10000
20 IFLEFT$(F$,1)="N"THENFORG=1TO200:NEXT
:GOTO350
        ELSE PRINT@92,"YOUR MOVE";:GOTO2
70
190 'HUMAN PLAYER
200 PRINT@92,"YOUR MOVE";:FORG=1TOLEN(P$
)
220 V=PEEK(14400):IFV<>8ANDV<>16ANDV<>32
ANDV<>64THEN220
        ELSEIFV<>ASC(MID$(P$,G,1))THENPR
INT@92,"  SORRY  ";:G=USR(255):FORG=1TO5
00:NEXT:GOTO10
                ELSEB=B+1:PRINT@7,B;:IFB
>HBTHENHB=B:PRINT@58,HB;
250 GOSUB1000:NEXT
260 'ADD MOVE
270 V=PEEK(14400):IFV<>8ANDV<>16ANDV<>32
ANDV<>64THEN270
        ELSE GOSUB1000:IFF<>0THEN270
280 P$=P$+CHR$(V):IFV=8ORV=32THENG=2*VEL
SEG=V/2
285 C$=CHR$(G)+C$
289 FORG=1TO300:NEXT
290 'COMPUTER'S MOVE
300 PRINT@92,"  MY MOVE  ";:FORG=1TOLEN(C$
):V=ASC(MID$(C$,G,1)):GOSUB1000:FORJ=1TO
200:NEXT:NEXT  '***CHANGE VALUE TO 300 F
OR A SLOWER GAME OR TO 100 FOR A FASTER
GAME
340 'RND MOVE TO BE ADDED
350 J=RND(4):V=INT(2[J*4+.5):GOSUB1000:I
FF<>0THEN350
355 C$=C$+CHR$(V):IFV=8ORV=32THENG=2*VEL
SEG=V/2
357 P$=CHR$(G)+P$
360 GOTO200
995 'CHANGE SCREEN
1000 PRINT@128+128*Y+6*X,B$;
1010 IFV=8THENF=(Y-1<0):IFF=0THENY=Y-1:J
=USR(50+256*127)  'UP
1020 IFV=16THENF=(Y+1>6):IFF=0THENY=Y+1:
J=USR(150+256*43)'DN
1030 IFV=32THENF=(X-1<0):IFF=0THENX=X-1:
J=USR(100+256*64)'LT
1040 IFV=64THENF=(X+1>10):IFF=0THENX=X+1
:J=USR(200+256*32)'RT
1100 PRINT@128+128*Y+6*X,E$;:RETURN
10000 CLS:PRINTCHR$(23)
10010 PRINT@476,"U";:J=USR(250+256*50):F
ORJ=1TO200:NEXT
10020 PRINT@474,"L";:J=USR(220+256*50):F
ORJ=1TO200:NEXT
10030 PRINT@478,"S";:J=USR(190+256*50):F
ORJ=1TO200:NEXT
10040 PRINT@472,"P";:J=USR(160+256*50):F
ORJ=1TO200:NEXT
10050 PRINT@480,"1";:J=USR(130+256*50):P
RINT@734,"BY DAVID LEWIS":FORJ=1TO700:NE
XT
10060 CLS:PRINT"Tired of games that flex
 your fingers without using your brain?"
:PRINT:PRINT"PLUS1 is as much of a chall
enge as you will find on the path of lif
e."
10070 PRINT"You are pitted against the c
omputer in a battle of wits.  The  obje
ct of the game is to create the longest
possible path."
10080 PRINT:PRINT"The path starts with o
ne 'stepping-stone.'  You add to it by
  moving in a direction by pressing an a
rrow key.  The computer will then trac
e the path backwards to the starting pla
ce.  It  then adds a square of its own.
"
10090 PRINT"You then trace the path back
wards and add another step of your  own,
 and so on.  The computer will keep trac
k of your score.  Play against a frien
d for maximum competition."
10095 PRINT@960,;:INPUT"PRESS ENTER TO C
ONTINUE";B$
11000 CLS:PRINT"<PLUS1 uses sound to hel
p you follow the action.  Hook the
cassette line into the AUX port on your
cassette player.>"
11070 P$="":C$="":B=0
12000 PRINT:POKE16409,1:INPUT"Do you wan
t to go first";F$
12020 CLS:PRINT"SCORE:":PRINT@30,"PLUS1"
:PRINT@48,"HI SCORE:":PRINT@58,HB
12030 PRINT@991,".";:PRINT@64*9-1,".";
12050 FORX=0TO10:FORY=0TO6:PRINT@128+128
*Y+6*X,B$;:NEXT:NEXT:X=RND(7)+1:Y=RND(3)
+1:PRINT@128+128*Y+6*X,E$;:RETURN
14995 'SOUND ROUTINE
15000 Z$=STRING$(23," "):V=VARPTR(Z$):LS
=PEEK(V+1):MS=PEEK(V+2):L=LS+256*MS:IFL>
32767THENL=L-65536
15015 DATA205,127,10,203,36,69,62,1,211,
255,16,254,69,62,2,211,255,16,254,37,32,
239,201
15020 FORG=LTOL+22:READX:POKEG,X:NEXT:IF
PEEK(16396)=201THENPOKE16526,LS:POKE1652
7,MS ELSE DEFUSR=L
15030 REM USE USR(FREQUENCY+256*DURATION
)
15040 B$=STRING$(3,191):E$=STRING$(3,128
):RETURN
```

Mental madness

What do the figures really say?

Models I/III/4III

Alan Mandell

The next time your friends ask you what your computer can do, run this program for them. See if they can reason out the clues from the screens shown. Some people never can use the spatial clues provided in the screens to figure out a word or commonly-known phrase. The program can liven up an evening when your visitors have played all your computer games.

The only interesting programming used is the technique in lines 100 to 310 where the various screens are presented using set graphics and PRINT@'s. Lines 350 to 360 have the answers entered as backward data items and then MID$ is used to write them correctly to compare with the user's input. This way, a listing of the program will not display the answers in an obvious way.

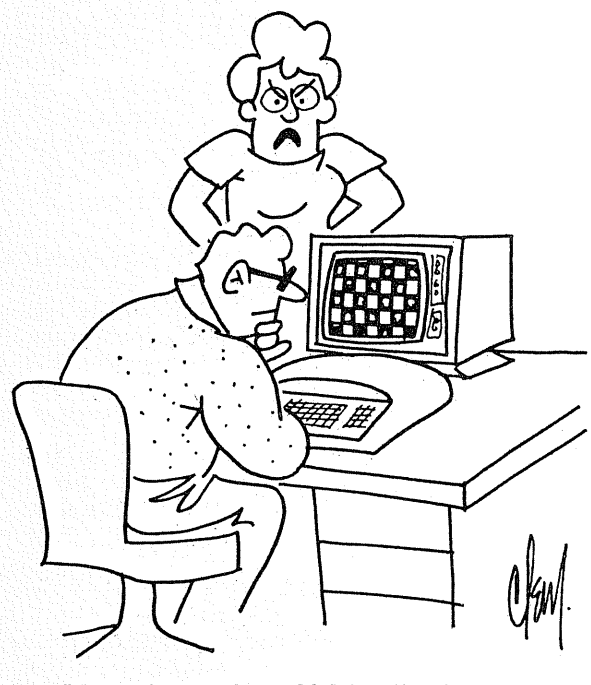
If you want to add more of your own puzzles, use the formats in lines 100 to 310 and don't forget to change your DIMs (line 5), your read data value, and to add the data items spelled backward.

Have fun with the puzzles.

Program Listing for Mental Madness

```
1 REM A FUN CHALLENGE GAME
5 CLS:CLEAR700:DIMM1$(20),M2$(20),M3$(20
):PRINTCHR$(23):PRINT@140,"M E N T A L":
PRINT@266,"M A D N E S S":PRINT@896,"BY
ALAN MANDELL":GOSUB350:CLS
10 PRINT@128,"Would you like directions
(Y/N) ";:INPUTQ$:IFQ$="N"THEN100
15 IFQ$<>"Y"THENCLS:GOTO10
20 CLS:PRINT:PRINT"You will be presented
 with a series of illustrations. You are
 totry to figure out the word or stateme
```

```
nt that they represent.":PRINT:PRINT"The
 clues are the spatial relationships sho
wn in the              illustrations."
25 PRINT:PRINT"You can quit any time."
30 PRINT:PRINT"Press <ENTER> if you give
 up.":GOSUB997:GOTO100
100 I=1:CLS:PRINTCHR$(23):FORX=12TO52:FO
RY=9TO34:SET(X,Y):NEXTY,X:PRINT@394,"S A
 N D";:GOSUB1000:I=I+1
110 PRINT@200,"MAN":FORX=8TO40:SET(X,13)
:NEXT:PRINT@326,"BOARD":GOSUB1000:I=I+1
120 PRINT@200,"STAND":FORX=8TO40:SET(X,1
3):NEXT:PRINT@332,"I":GOSUB1000:I=I+1
130 FORX=8TO68STEP8:FORY=12TO14:SET(X,Y)
:NEXTY,X:Z=262:Z$="READING":FORJ=1TOLEN(
Z$):PRINT@Z,MID$(Z$,J,1);:Z=Z+4:NEXT:GOS
UB1000:I=I+1
135 CLS:IFM2$(I-1)="READING RAILROAD" TH
EN PRINT"VERY CREATIVE THINKING HOWEVER"
:FORT%=1TO1500:NEXT
140 PRINT@202,"WEAR":FORX=8TO40:SET(X,13
):NEXT:PRINT@330,"LONG":GOSUB1000:I=I+1
150 Z=148:PRINT@Z,"R":Z=Z+64:PRINT@Z,"O"
:Z=Z+64:PRINT@Z,"A":Z=Z+64:PRINT@Z,"D":Z
=Z+64:PRINT@Z,"S"
155 PRINT@272,"ROADS";:GOSUB1000:I=I+1
160 Z=148:PRINT@Z,"T":Z=Z+64:PRINT@Z,"O"
:Z=Z+64:PRINT@Z,"W":Z=Z+64:PRINT@Z,"N":G
OSUB1000:I=I+1
170 Z=202:Z$="CYCLE":FORJ=1TO3:PRINT@Z,Z
$:Z=Z+64:NEXT:GOSUB1000:I=I+1
180 Z=202:PRINT@Z,"0":FORX=8TO40:SET(X,1
```



"Why can't you play golf like other husbands?"

```
3):NEXT:PRINT@330,"BS":PRINT@394,"MA":PR
INT@458,"PHD":GOSUB1000:I=I+1
190 PRINT@202,"KNEE":PRINT@266,"LIGHTS":
GOSUB1000:I=I+1
200 PRINT@202,"LE":PRINT@270,"VEL":GOSUB
1000:I=I+1
210 PRINT@202,"IIII":PRINT@266,"OOOO":GO
SUB1000:I=I+1
220 PRINT@26,"CHAIR":GOSUB1000:I=I+1
230 Z=202:Z$="TOUCH":FORJ=1TOLEN(Z$):PRI
NT@Z,MID$(Z$,J,1):Z=Z+64:NEXT:GOSUB1000:
I=I+1
240 PRINT@202,"GROUND":Z=330:FORJ=1TO6:P
RINT@Z,"FEET":Z=Z+64:NEXT:GOSUB1000:I=I+
1
250 PRINT@202,"MIND":FORX=8TO40:SET(X,13
):NEXT:PRINT@328,"MATTER":GOSUB1000:I=I+
1
260 PRINT@202,"ECNALG":GOSUB1000:I=I+1
280 PRINT@202,"GI - GI":FORX=12TO60:SET(
X,13):NEXT:PRINT@332,"CCC":GOSUB1000:I=I
+1
290 PRINT@200,"DEATH ==> LIFE":GOSUB1000
:I=I+1
300 PRINT@200,"OHOLENE":GOSUB1000
310 PRINT@200,"ENOUGH'S ENOUGH":FORT%=1T
O1500:NEXT:GOTO900
350 FORI=1TO20:READM3$(I):NEXT:DATA XOBD
NAS,DRAOBREVO NAM,DNATSREDNU I,SENIL EHT
 NEEWTEB GNIDAER,RAEWREDNU GNOL, SDAORSS
ORC,NWOTNWOD,ELCYCIRT,OREZ WOLEB SEERGED
 EERHT,STHGIL NOEN,LEVEL TILPS
355 DATA SEYE EHT REDNU SELCRIC,RIAHCHGI
H,NWODHCUOT,DNUORGREDNU TEEF XIS,RETTAM
REVO DNIM,ECNALG DRAWKCAB,SAESREVO S'IG,
HTAED RETFA EFIL,ENO NI ELOH
360 FORI=1TO20:FORJ=LEN(M3$(I))TO1STEP-1
:M1$(I)=M1$(I)+MID$(M3$(I),J,1):NEXTJ:NE
XTI:RETURN
900 CLS:PRINTCHR$(23):PRINT@128,"You had
";C;"correct ":FORT%=1TO1000:NEXT:PRINT"
out of";I;"tries !!":PRINT@926,"< - END
- >";:END
997 PRINT@960,"Press the <SPACE BAR> to
go on.";
998 IFINKEY$<>" "THEN998
999 CLS:RETURN
1000 PRINT@768,"Type in your guess ":INP
UTM2$(I):IFM2$(I)=M1$(I)THENPRINT"VERY G
OOD";:C=C+1:GOSUB997:GOTO1010
1005 PRINT"NOPE ==> ";M1$(I):GOSUB997
1010 CLS:PRINT@704,"Had Enough Yet (Y/N)
 ";
1011 KY$=INKEY$:IFKY$=""THEN1011
1012 IFKY$="Y"THEN900
1015 IFKY$="N"THEN1020
1020 CLS:PRINTCHR$(23):RETURN
```

SOFTWARE PRICE WAR

$19.95

Not cheap programs but complete software systems. I believe most software is pirated because it is overpriced and I'm taking this gamble to prove it.

HELP STAMP OUT SOFTWARE PIRATES

I advertised these Systems in **80MICRO** for **$79.00**. I rather sell 1000 systems at **$19.95** than 100 at **$79.00.** These are complete software systems supplied to you on a **TRSDOS 1.3** system diskette (I paid Tandy Corp. a royalty fee for this privilege). All systems fully menu driven — user friendly just insert diskettes, reset and go. Systems are supplied with full documentation (although most users report that they don't read it).

LYNN'S PAYROLL SYSTEM

Complete record keeping & payroll calculation. To include your state's unemployment tax & forms, detailed 941 form, prints paychecks & W-2's.

LYNN'S ACCOUNTS RECEIVABLE SYSTEM

Single entry — open invoice system. Prints invoice and statements. Built in aging report.

LYNN'S CHECK REGISTER SYSTEM

Prints checks. Register for any month, reconciliation statement.

LYNN'S TRSDOS UTILITY PROGRAMS

For TRSDOS 1.3 includes short directory (file names only), cursor directory (allows cursor access to CMD, BASIC, and "DO" Files), and a 2K Print Spooler. Menu installs utilities on existing TRSDOS Diskettes.

LYNN'S E-Z MAIL SYSTEM

Sort by name, zip code, state. 2,500 names. E-Z edit.

All systems are written for Model III or Model IV in Model III Mode.

AND NOW THE BEST PART
ANY SYSTEM ONLY $19.95 EACH.

Plus $3.00 shipping per order. Illinois residents add 5¼% sales tax.

(The **TRSDOS** alone is worth **$14.95**)

All systems phone supported call 815/436-4477 and ask the author JOE LYNN what you want to know. I will assist all original purchasers any way I can.

master charge
THE INTERBANK CARD

Joe Lynn #7

Computer Services
23501 W. Gagne Lane
Plainfield, Illinois 60544
815/436-4477

BANKAMERICARD
VISA

Your computer can learn

Play LASTX and see the computer get "smarter"

Models I/III/4III

A. W. Maddox

We have all played games against our computers and frequently we are soundly thrashed. These programs generally have one thing in common. They all have a prescribed set of actions to follow in a particular situation. They do not alter their basic logic. When you "beat" the program and play the same game again, you will beat it again.

There is, however, a set of programs that alter their logic as a result of wins or losses. These programs are considered to be adaptive or to have artificial intelligence (AI). We define these as a process wherein a computer is instructed to alter its decision logic or variable coefficients as a result of prior events with a goal of improving its performance on subsequent decisions. This means we write the program with an initial set of decision rules and add sufficient coding to allow those rules to be altered. To accomplish this, we need to tell the computer what constitutes a good or bad decision and how to alter its rules as a result of the outcome produced by these decisions.

To illustrate, consider the following simple game. Fifteen objects are placed on a table. These can be toothpicks, paper clips, match heads, or Krugerrands. Call these objects "pips." Each of two players will alternately remove one two, or three pips from those remaining when it is his turn. The object is to remove the pips in such a way that your opponent is forced to take the last pip. I call this game "LASTX."

Play this game with a friend. After several games, each player will begin to recognize certain patterns and will develop a strategy which he hopes will win the game. A player can insure a win if he can remove the second, sixth and tenth pip. Notice that if a player removes the second pip, he is assured the opportunity of removing the sixth, which ensures that he can remove the tenth and force a win. Therefore, a knowledgeablee player wants to start each new game. In a fair game session, players should alternate starting each game.

The program in Listing 1 (LASTX1) will enable you to play this game against the computer. If you let it start, *it will win!*

Before describing the program, we need to examine the two subroutines used. The subroutine at line 4000 checks the legality of the move. That is, only one, two, or three pips can be taken and the player cannot take more than what is available. If all is okay, it sets the variable LF equal to zero.

The subroutine at line 5000 reduces the pip count by the amount taken and checks to see if any pips are left. If there are, it reprints the screen and returns for more play. If they are all gone, it prints out the winner and goes on to another game.

Let's examine the program. After some preliminaries, it reads in its decision rules (lines 60 to 100). The pip counter is set to the original 15 pips at line 100. At line 110, we allow the human to select who is to be first. Entering H allows the human to start. PL keeps track of whose turn it is (PL=1 for TRS-80, PL=2 for human).

Since we don't know the strategy of the human, we just ask him to enter his move. We check it for legality in subroutine 4000 and call subroutine 5000 to update the display and look for a win.

The strategy for the computer is different. Its entire strategy is contained in line 90 which tells the computer how many pips to remove at each board situation. Note that at the beginning of the game (when there are 15 pips available), it is told to take two pips; a winning strategy. When the human selects a move where the computer seemingly can't win, it takes one pip, slowing the game down to allow the human to make a strategic error. The computer selects its move at line 1030 and checks it for legality. The board is adjusted and win status is determined.

In this game, the computer was given precise instructions on the winning strategy. It didn't have to "think" much. Now, consider the same game where the computer is not told precise decision rules for a winning strategy. It must "learn" through playing several games with a "knowledgeable" human player. This is shown in Listing 2 (LASTX2). The following lines are changed or added. (That's the reason for the funny line numbering in Listing 1.)

Line 40: A second dimension of three is added to BD to allow the computer to select one, two, or three at each board situation. CR and CP

are dimensioned to keep track of the pip count and pips taken. Ten moves are allowed for any one game. HR and HP are the same for the human.

Line 104: NM is the number of moves taken during a given game.

Lines 1010 to 1015: Here is where the computer selects the best move it can for a given remaining pip count. It compares each potential move at the given PC with all possible moves trying to find the maximum MX. If it has never faced this situation before, it picks a random move at line 1016. More on this later.

Lines 1034 to 1036: The current pip count and selected move are recorded in CR and CP. If it is time to do so, the move is incremented. Lines 2064 to 2066 do the same for the human.

Lines 7000 to 7050: This is the learning phase of the program. After each game is over, the game is essentially replayed since we kept track of each move by the human and computer. If the computer won, it is rewarded for every move it made (adding one to the value of BD corresponding to the value of PC and the number of pips taken for each move). Since the human lost, one is subtracted from BD for each of its moves. The opposite occurs if the human was lucky enough to win.

Thus, the value of BD is continually adjusted as the games are played. The strategy in lines 1010 to 1015 is to cycle through all prior situations with the current pip count to pick the number of pips that corresponds to the maximum value of BD for the current PC.

Several points need to be brought out. First, a good human player will force the computer to learn fast. This game can be learned after about 15 games. Secondly, the process permits the program to learn from both players. Many AI programs only learn from the winner. This program can learn about twice as fast if both players are considered. Thirdly, there are no provisions to save the BD array (where the accumulated strategy is stored). The program must relearn its strategy each time it is run. You may want to revise the program to save BD after a series of runs and read them in when you rerun the program. To expand on the AI work, however, there is a better way.

You may have wondered what kind of klutz would try to convince you that coding the three DATA lines at 90-92 was the right way to initialize the BD array at 50. You're right if that is all that is to be done. I chose this way so I can demonstrate how an AI program can actually modify itself during the learning process. I take this approach so I don't have to save the values of BD after each game session to remember the current move strategies. Also, this way, I don't have to change any of my program, just add a few lines.

The last item that is read into the program by a READ statement can be found by PEEKing at locations 16639 and 16640. The address of this last DATA element is calculated by line 8000 in program Listing 3 (LASTX3). I know that the start of the DATA statements is 144 from its end by knowing that each BASIC line has a minimum of six bytes of storage for it. These have been explained in many articles, so I'll only summarize them here.

The first two identify the storage location of the BASIC line that follows the current one. The next two are for the current linenumber. The text of the line follows and takes one storage byte for each character or operator recognized by the interpreter. Finally, there is a zero to tell the interpreter that we are done with this line of code. Since there are 150 data storage bytes to store the 45 sets of 50's, minus the six for the system, we have 144 as an offset from where the last DATA element was READ. Wow! Lines 8020 to 8090 decode the BD array in such a way that the subroutine at line 9000 can poke the new values of BD into the original DATA statements.

Exactly what have I done? Load and save a copy of Listing 3 as it stands, or add the changes in lines 7060 to 9070 and save a copy. Now, run this program just like you ran the program in Listing 2. After several games, BREAK and list program lines 90 to 92. You will notice that *the program has*

changed itself based upon the logic of learning. If you save this program and RUN it later, you will see that it starts where it left off after the last game session.

These programs learned by playing many games. Why not let the computer teach itself to play? A final version of LASTX (which does this) requires only six line changes to Listing 3. They are as follow:

```
105 GM=GM+1:IF GM>50 THEN
108
106 A$="C":IF RND(2)=2 THEN
A$="H"
107 GOTO 120
130 IF A$="H" THEN 2005
2005 IF GM<50 THEN PP=RND(3)
:GOTO 2020
2050 GOTO 2005
```

Lines 105 and 2005 allow for a 50-game learning session. The program then reverts back to a two-player game as above. Line 106 ensures that the human and program randomly start each game.

Try this program and you will note that the program does "learn," but when it reverts back to normal play after its 50 games, it doesn't play very well. This is because a random "idiot" taught it to play. It's nice to know that a human is still needed to get it off on the right track.

Listing 1 — LASTX1

```
10 'GAME OF "LAST X" WHERE COMPUTER HAS
SET STRATEGY
20 CLS
30 DEFINT A-Z
40 DIM A$(2),BD(15)
50 A$(1)="---I ":A$(2)="---YOU "
60 FOR I=1 TO 15'
READ IN
70 READ BD(I)'
  STRATEGY
80 NEXT I'
    VALUES
90 DATA 1,1,2,3,1,1,2,3,1,1,2,3,1,1,2
100 PC=15'
 PIP COUNTER
110 INPUT"WHO'S FIRST, <H> OR <C>";A$
120 GOSUB 5010 '
  DISPLAY BOARD
130 IF A$="H" THEN 2010 '
  IF HUMAN, SKIP
1000 PL=1'
*** COMPUTER PLAYER ***
1010 PP=BD(PC)'
  PICK A MOVE
1020 GOSUB 4000 '
   LEGAL CHECK?
1030 PRINT"I'LL TAKE ";PP
1040 FOR I=1 TO 2000'
  SLOW DOWN FOR HUMAN
1050 GOSUB 5000 '
   ADJUST BOARD
2000 PL=2'
*** HUMAN PLAYER ***
2010 INPUT"YOUR MOVE HUMAN, ENTER PIPS";
PP
2020 GOSUB 4000 '
   LEGAL CHECK?
2030 IF LF=0 THEN 2060
2040 PRINT"CHEATER"
2050 GOTO 2010
2060 GOSUB 5000 '
   ADJUST BOARD
2070 GOTO 1000 '
  COMPUTER'S MOVE
4000 LF=0'
*** LEGAL CHECK ***
4010 IF PP>3 THEN LF=1'
   GREATER THAN 3
4020 IF PP<1 THEN LF=1'
   LESS THAN 1
4030 IF PP>PC THEN LF=1'
   NOT ENOUGH LEFT
4040 RETURN
5000 PC=PC-PP'
*** ADJUST BOARD ***
5010 IF PC=0 THEN 6000 '
    EMPTY?
5020 FOR I=1 TO PC
5030 PRINT"X ";
5040 NEXT I
6000 '
*** WIN CHECK ***
6010 IF PC>0 THEN RETURN
6020 PRINT A$(3-PL);"WIN"
6030 GOTO 100
```

Listing 2 — LASTX2

```
10 'GAME OF "LAST X" WHERE COMPUTER LEAR
NS TO WIN
20 CLS
30 DEFINT A-Z
40 DIM A$(2),BD(15,3),CR(10),CP(10),HR(1
0),HP(10)
50 A$(1)="---I ":A$(2)="---YOU "
60 FOR J=1 TO 3:FOR I=1 TO 15'
READ IN
70 READ BD(I,J)'
  STRATEGY
80 NEXT I,J'
    VALUES
90 DATA50,50,50,50,50,50,50,50,50,50,50,
50,50,50,50
91 DATA50,50,50,50,50,50,50,50,50,50,50,
50,50,50,50
92 DATA50,50,50,50,50,50,50,50,50,50,50,
50,50,50,50
100 PC=15'
 PIP COUNTER
104 NM=0'
 MOVE COUNTER
108 A$="C"'
 SET TO ME
110 INPUT"WHO'S FIRST, <H> OR <C>";A$
120 GOSUB 5010 '
 DISPLAY BOARD
130 IF A$="H" THEN 2010 '
 IF HUMAN, SKIP
1000 PL=1'
*** COMPUTER PLAYER ***
1010 MX=0'
MAXIMUM MOVE
1011 FOR I=1 TO 3'
CYCLE THROUGH MOVES
1012 IF MX>BD(PC,I) THEN 1015 '
  THIS ONE BETTER?
1013 MX=BD(PC,I)'
  NO, THIS ONE IS
1014 PP=I'
  SET PIPS TO TAKE
1015 NEXT I'
  ALL DONE
1016 IF MX=50 THEN PP=RND(3)'
  WELL, JUST PICK ONE
1020 GOSUB 4000 '
  LEGAL CHECK?
1025 IF LF<>0 THEN 1000 '
  SORRY, DO AGAIN
1030 PRINT"I'LL TAKE ";PP
1032 IF A$="C" THEN NM=NM+1
1034 CR(NM)=PC'
SAVE PIP COUNTER
1036 CP(NM)=PP'
SAVE THIS MOVE
1040 FOR I=1 TO 2000'
  SLOW DOWN FOR HUMAN
1050 GOSUB 5000 '
   ADJUST BOARD
2000 PL=2'
*** HUMAN PLAYER ***
2010 INPUT"YOUR MOVE HUMAN, ENTER PIPS";
PP
2020 GOSUB 4000 '
   LEGAL CHECK?
2030 IF LF=0 THEN 2060
2040 PRINT"CHEATER"
2050 GOTO 2010
2060 GOSUB 5000 '
   ADJUST BOARD
2062 IF A$="H" THEN NM=NM+1'
INCREASE MOVE COUNTER
2064 HR(NM)=PC+PP'
SAVE PIP COUNTER
2066 HP(NM)=PP'
SAVE THIS MOVE
2070 GOTO 1000 '
COMPUTER'S MOVE
4000 LF=0'
*** LEGAL CHECK ***
4010 IF PP>3 THEN LF=1'
   GREATER THAN 3
4020 IF PP<1 THEN LF=1'
   LESS THAN 1
4030 IF PP>PC THEN LF=1'
   NOT ENOUGH LEFT
4040 RETURN
5000 PC=PC-PP'
*** ADJUST BOARD ***
5010 IF PC=0 THEN 6000 '
    EMPTY?
5020 FOR I=1 TO PC
5030 PRINT"X ";
```

```
5040 NEXT I
6000 '
*** WIN CHECK ***
6010 IF PC>0 THEN RETURN
6020 PRINT A$(3-PL);"WIN"
7000 '
*** LEARNING ***
7010 PRINT"JUST A SEC WHILE I LEARN FROM
 THIS VALUABLE EXPERIENCE"
7020 FOR I=1 TO NM'
       REPLAY GAME
7030 BD(HR(I),HP(I))=BD(HR(I),HP(I))+(3-
2*PL)' HUMAN WINS
7040 BD(CR(I),CP(I))=BD(CR(I),CP(I))-(3-
2*PL)' COMPY WINS
7050 NEXT I
7060 GOTO 100  '
  LET'S PLAY AGAIN
```

Listing 3 — LASTX3

```
10 'GAME OF "LAST X" WHERE COMPUTER MODI
FIES ITSELF
20 CLS
30 DEFINT A-Z
40 DIM A$(2),BD(15,3),CR(10),CP(10),HR(1
0),HP(10)
50 A$(1)="---I ":A$(2)="---YOU "
60 FOR J=1 TO 3:FOR I=1 TO 15'
READ IN
70 READ BD(I,J)'
  STRATEGY
80 NEXT I,J'
    VALUES
90 DATA50,50,50,50,50,50,50,50,50,50,50,
50,50,50,50
91 DATA50,50,50,50,50,50,50,50,50,50,50,
50,50,50,50
92 DATA50,50,50,50,50,50,50,50,50,50,50,
50,50,50,50
100 PC=15'
 PIP COUNTER
104 NM=0'
 MOVE COUNTER
108 A$="C"'
 SET TO ME
110 INPUT"WHO'S FIRST, <H> OR <C>";A$
120 GOSUB 5010 '
 DISPLAY BOARD
130 IF A$="H" THEN 2010 '
 IF HUMAN, SKIP
1000 PL=1'
*** COMPUTER PLAYER ***
1010 MX=0'
MAXIMUM MOVE
1011 FOR I=1 TO 3'
CYCLE THROUGH MOVES
1012 IF MX>BD(PC,I) THEN 1015 '
  THIS ONE BETTER?
1013 MX=BD(PC,I)'
  NO, THIS ONE IS
1014 PP=I'
  SET PIPS TO TAKE
1015 NEXT I'
  ALL DONE
1016 IF MX=50 THEN PP=RND(3)'
  WELL, JUST PICK ONE
1020 GOSUB 4000 '
  LEGAL CHECK?
1025 IF LF<>0 THEN 1000 '
  SORRY, DO AGAIN
1030 PRINT"I'LL TAKE ";PP
1032 IF A$="C" THEN NM=NM+1
1034 CR(NM)=PC'
SAVE PIP COUNTER
1036 CP(NM)=PP'
SAVE THIS MOVE
1040 FOR I=1 TO 2000'
  SLOW DOWN FOR HUMAN
1050 GOSUB 5000 '
   ADJUST BOARD
2000 PL=2'
*** HUMAN PLAYER ***
2010 INPUT"YOUR MOVE HUMAN, ENTER PIPS";
PP
2020 GOSUB 4000 '
   LEGAL CHECK?
2030 IF LF=0 THEN 2060
2040 PRINT"CHEATER"
2050 GOTO 2010
2060 GOSUB 5000 '
   ADJUST BOARD
2062 IF A$="H" THEN NM=NM+1'
INCREASE MOVE COUNTER
2064 HR(NM)=PC+PP'
SAVE PIP COUNTER
2066 HP(NM)=PP'
SAVE THIS MOVE
2070 GOTO 1000 '
COMPUTER'S MOVE
4000 LF=0'
*** LEGAL CHECK ***
4010 IF PP>3 THEN LF=1'
   GREATER THAN 3
4020 IF PP<1 THEN LF=1'
   LESS THAN 1
4030 IF PP>PC THEN LF=1'
   NOT ENOUGH LEFT
4040 RETURN
5000 PC=PC-PP'
*** ADJUST BOARD ***
5010 IF PC=0 THEN 6000 '
   EMPTY?
5020 FOR I=1 TO PC
5030 PRINT"X ";
5040 NEXT I
6000 '
*** WIN CHECK ***
6010 IF PC>0 THEN RETURN
6020 PRINT A$(3-PL);"WIN"
7000 '
*** LEARNING ***
7010 PRINT"JUST A SEC WHILE I LEARN FROM
 THIS VALUABLE EXPERIENCE"
7020 FOR I=1 TO NM'
       REPLAY GAME
7030 BD(HR(I),HP(I))=BD(HR(I),HP(I))+(3-
2*PL)' HUMAN WINS
7040 BD(CR(I),CP(I))=BD(CR(I),CP(I))-(3-
2*PL)' COMPY WINS
7050 NEXT I
8000 AD=PEEK(16639)+256*PEEK(16640)'
  LAST DATA READ
8010 AD=AD-144'
  FIRST OF 50'S
8020 FOR I=1 TO NM'
  DECODE
8030 XR=HR(I)'
    THE
8040 XP=HP(I)'
    HUMAN
8050 GOSUB 9010
8060 XR=CR(I)'
    AND
8070 XP=CP(I)'
    COMPUTER
8080 GOSUB 9010
8090 NEXT I'
     MOVES
8100 GOTO 100  '
      LET'S PLAY AGAIN
9000 IF XR*XP=0 THEN RETURN'
ODD GAME PROTECTION
9010 XB=BD(XR,XP)'
*** POKE ROUTINE ***
9020 PK=AD+50*(XP-1)+3*(XR-1)
9030 LB=INT(XB/10)'
  LEFT BYTE
9040 RB=XB-LB*10'
  RIGHT BYTE
9050 POKE PK,48+LB'
  POKE LB
9060 POKE PK+1,48+RB'
  POKE RB
9070 RETURN
```

Sorted TRSDOS 2.0 Directory

A needed enhancement, written in FORTRAN

Model II

Richard A. Poitras

As many times before, I typed in DIR and the screen filled with the filenames that were on the disk. I jabbed at the hold key in an attempt to keep the filenames from scrolling so fast that I would miss the one for which I searched. With my free hand, I ran a finger down the screen, adding more fingerprints to the ones already there from previous DIR commands. I poked the hold key again and again to view the entire directory. "Had I missed the filename because I had not stopped the scrolling in time?" I thought.

Finally, in frustration, I printed the directory, and through slow finger-scrolling through the listing, I found that the program was, indeed, on that disk.

For those who have spent much time and frustration searching through the directory listing provided by the Model II command DIR, as I have, there is hope to be found in the following text. What I have done is to combine the use of assembly language and FORTRAN to produce two modules which, when combined, provide a useful directory utility.

DIR

The DIR command is issued from the system level. It provides a listing of the directory of programs found on the selected diskette. The listing also includes (let me see, I must look up what all those things are) diskette name, drive number, today's date and time, date of creation, date of update, attributes, file type, record length, number of records used on the diskette, number of extents, granules allocated, sectors allocated, and sectors used.

The DIR command of the Model II provides a wealth of information that can be very valuable to the programmer and the debugger. Rarely is all of that information of much value to the user or operator of the computer. The profuse listing provided with the DIR command might be called "information over-kill" because so much of the information is not only of little value for most applications, it is rather distracting and even confusing. While it is fine to have access to the information, I would prefer that it remain hidden most of the time.

Another pitfall of the Model II DIR command is that the information seems to have no order. Oh, it is displayed in nice, neat columns, but to find the program for which you are looking, you must scan the entire directory. It is not sorted alphabetically, by creation date, or by any other method which might aid us in our plight to find out quickly what is on the diskette.

What is needed is a program that can be called in place of DIR which can provide an alphabetical listing of the directory on the diskette in a selectable drive. Listings 1 and 2, when linked, provide such a program.

Sorted Directory — SD

My approach to the Sorted Directory (SD) program was to utilize two types of programming. First, I used assembly language to take advantage of the supervisor calls provided with TRSDOS. These are subroutines which can be called from assembly language programs which allow the programmer to take advantage of program segments which have already been thoroughly checked out and which perform the desired operations. In conjunction with assembly language, I used FORTRAN to provide easier input from the operator, easier device selection for output and easier modification in the future.

Listing 1 is the assembly language module. This module provides for retrieval of the entire non-system directory and places it in a RAM buffer. The module sorts the directory in ascending or descending order (user selectable) and provides for system error messages. Much of the code is simply setting up for calling TRSDOS supervisor routines. Three of these routines are used in this module: RAMDIR, ERROR, and SORT.

RAMDIR is a routine which gets the selected disk directory (the location labeled DSK: stores selected disk) and places the directory in RAM at selected location (the location labeled BUFFER: stores the directory). The information available is exactly the same as that provided by the DIR command.

SORT is a routine which sorts strings of equal length, starting at a specified position within the string for a key of specified length. The key in this case is the filename. This routine will sort into either ascending or descending order (the location labeled MODE: controls direction). Since we are sorting directory entries which are of equal length (34 bytes each), this is a useful call for the sort. (Even though this sort is a bubble sort, it is fast in machine language.)

ERROR is a routine which provides a display of the error code if an error occurs during retrieval of the directory information. (The most frequent error will probably be due to incorrect drive specification which will produce ERROR 8, Disk Drive Not Ready.)

Listing 2 is the FORTRAN module. FORTRAN was used for this module rather than doing the entire project in assembly language because the input/output is easier from FORTRAN than from assembly language. Also, cleaning up the BUFFER: area is a little nicer in FORTRAN.

The FORTRAN module is composed of seven segments. The first segment (lines 1200-1500) identifies the variables. The EXTERNAL statement identifies the entry point of the assembly language routine. DSK,

MODE, and BUFFER are the same variables as used in the assembly language routine and are included in the COMMON data area in both modules.

Although LOGICAL might at first seem to be an inappropriate type of variable to be used with what are usually integer or string types, in this FORTRAN, logical can include both integers and strings and has the advantage of having each one-byte location addressable by variable name. This reduces the bit-twiddling that is often necessary when using other types to store string information.

The second segment (lines 1900-2200) places blanks at all locations of the buffer in which the directory will be stored. This is necessary to insure that illegal characters will not be found in locations that are not written to the RAMDIR call.

The third segment (lines 2600-4000) reads input from the operator. The operator is requested to enter the drive number, the mode of sorting, and the output device. The output device can be any valid FORTRAN output device. The default devices with FORTRAN are: display = 3, printer = 2, disk = 6-10. (Oh, yes, you can put the sorted directory on disk . . . handy for future use.)

The fourth segment (line 4700) calls the assembly language module which provides the sorted directory entries along with the extraneous information of which we are trying to rid ourselves.

The fifth segment (line 5100) checks to see if an error occurred in the call to DIR0 (the assembly language routine). If an error occurred, the error code has been displayed and display of the directory is inappropriate, so a jump is made to the end of the program.

The sixth segment (lines 5700-6300) removes new, unwanted characters in the BUFFER area which were placed there by the call to DIR0. Since many of these characters may cause FORTRAN I/O errors, they must be replaced with spaces.

The final segment (lines 6800-7800) moves the appropriate characters from BUFFER to BUF which is a print buffer from which the characters are printed. The format for the printing provides for five columns of alphabetized directory entries. Each column contains twenty possible entries and the columns are filled in order. With this method, the entire user directory (96 possible entries) can fit on the display at the same time.

Link/Load

For those who are unfamiliar with compiling, linking and loading, a short explanation is in order.

When programs are written in BASIC, FORTRAN, or in assembly language, the text which we type is known as the *source code*. When a program is run from BASIC, each statement gets converted to *executable* machine instructions each time it is encountered in the program. When programs are written in FORTRAN or assembly language, they are translated to machine executable instructions only once. These instructions are normally stored as a separate file on the disk.

The process of translating FORTRAN programs is called *compiling*. The process of translating assembly language instructions is called *assembling*. Commonly, when programs or modules are assembled or compiled, the code which is produced is *not executable* code. Rather, what is produced is *relocatable* code. This code, when it has been run through a loader, becomes *executable code*. The loader modifies the relocatable code so that it can be placed anywhere in memory. Frequently, the loader has an additional capability. It can combine relocatable modules to form the final program. This process is called linking. A program which performs both the linking and loading operations is known as a linking loader. The Link-80 program is such a linking loader.

The modules for the final SD program must be created, compiled, and linked. Both modules should be created with EDIT-80. The assembly language module should be saved as DD/MAC and the FORTRAN module should be saved as SD/FOR (other names may be used if you make appropriate changes below).

Compiling the assembly language routine is done with the MACRO-80 assembler using the command line M80 DD,=DD. This produces the relocatable module DD/REL.

The FORTRAN module should be compiled with the FORTRAN-80 compiler using the command line F80 SD,=SD. This produces the relocatable module SD/REL.

The two modules can then be linked, loaded and saved with LINK-80 using the command line L80 SD,DD,SD-N-E. This will produce the final executable program SD which can then be executed simply by typing SD and answering the appropriate questions.

Note: If you have only one disk drive, as I do, you will need to copy the DD/REL file to the same diskette that contains the SD/REL file and LINK-80 before link/load.

Summary

This article should provide you with two things. First, you should end up with a utility program which will provide you with a more frequently useful directory than the DIR command. Second, it should be an example of using assembly language in conjunction with FORTRAN to produce useful programs.

If you do not have access to the assembler or FORTRAN compiler, I will provide a copy of the final SD program and the source files for both SD/FOR and DD/MAC on your TRSDOS diskette for $5, covering the cost of copying. Send your 8-inch TRSDOS diskette to Richard A. Poitras, 2505 Highwood Drive, Missoula, MT 59803.

Richard A. Poitras taught high school science and mathematics for eight years. He holds B.A. and B.S. degrees in Microbiology and Computer Science, respectively. He is currently employed at the University of Montana in the Department of Zoology and provides computer programming, computer interface design, and electronic maintenance for the department. He also does programming and consulting on a private basis.

Listing 1

```
ØØ1ØØ   ;This program uses supervisory c
alls to TRSDOS to get the
ØØ2ØØ   ;disk directory in RAM and sort
it into alphabetical order.
ØØ3ØØ   ;It should be linked with SD/FOR
 for total control of directory
ØØ4ØØ   ;listing.  No listing is provide
d by the segment of program
ØØ5ØØ
ØØ6ØØ            ENTRY DIRØ
ØØ7ØØ   DIRØ:    PUSH AF          ;Save st
atus
ØØ8ØØ            PUSH BC
ØØ9ØØ            PUSH DE
Ø1ØØØ            PUSH HL
Ø11ØØ            PUSH IX
Ø12ØØ
Ø13ØØ   ;Get DIR into RAM using supervis
or call RAMDIR
Ø14ØØ
Ø15ØØ            LD A,53
Ø16ØØ            LD IX,DSK        ;Drive #
Ø17ØØ            LD B,(IX)
Ø18ØØ            LD C,Ø
Ø19ØØ            LD HL,BUFFER
Ø2ØØØ            RST 8
Ø21ØØ            JR Z,DIR1        ;Continu
e if no errors
Ø22ØØ   ;        Error occurred...print e
rror code using supervisor
Ø23ØØ   ;        call ERROR
Ø24ØØ
Ø25ØØ            LD B,A
Ø26ØØ            LD A,39
Ø27ØØ            RST 8
Ø28ØØ            LD (IX),99       ;Notify
SD/FOR of error
Ø29ØØ            JR RETURN
Ø3ØØØ
Ø31ØØ   DIR1:    LD BC,34         ;Offset
to next entry
Ø32ØØ            LD A,'#'         ;Char to
 check for at DIR end
Ø33ØØ            LD HL,BUFFER     ;DIR sto
rage area
Ø34ØØ   DIR1_1: CP (HL)           ;Check c
har
Ø35ØØ            JR Z,DIR2        ;Found e
nd of DIR area
Ø36ØØ            ADD HL,BC        ;Next en
try
Ø37ØØ            JR DIR1_1        ;Loop 't
il end found
Ø38ØØ
```

```
03900   ;Sort Directory using supervisor
 call SORT
04000
04100   DIR2:   LD A,56
04200           CCF
04300           LD BC,34        ;Offset
to last valid entry
04400           SBC HL,BC       ;Get fir
st byte of last entry => by HL
04500           PUSH HL         ;Put HL
on stack for move to DE
04600           POP DE          ;Put it
in DE
04700           LD B,1
04800           LD C,34
04900           LD H,(IX+1)     ;Mode fr
om SD/FOR
05000           LD L,15
05100           LD IX,BUFFER
05200           RST 8
05300   RETURN: POP IX
05400           POP HL
05500           POP DE
05600           POP BC
05700           POP AF
05800           RET
05900
```

```
06000           COMMON /TABLE/
06100   DSK:    DB 0            ;Disk #
06200   MODE:   DB 0            ;0=ascen
d sort, not 0=descend sort
06300   BUFFER: DB 0            ;Buffer
extends from here on for 3265 bytes
06400
06500           END
```

Listing 2

```
00100           PROGRAM SD
00200   C
00300   C       This program must be lin
ked with DD/MAC which provides
00400   C       a sorted directory in th
e common TABLE
00500   C
00600   C       The common is as follows
:
00700   C         DSK   = Disk number fo
r directory from this prog.
00800   C         MODE  = Method of sort
; 0 = ascend, not 0 = descend
00900   C         BUFFER= Sorted directo
ry 34 bytes per entry 2-16 file spec
```

```
Ø1ØØØ   C
Ø11ØØ   C
Ø12ØØ           EXTERNAL DIRØ
Ø13ØØ           LOGICAL DSK,MODE,BUFFER(
34,1ØØ),I,J,K,BUF(75)
Ø14ØØ           LOGICAL COLON,DEVICE
Ø15ØØ           COMMON /TABLE/DSK,MODE,B
UFFER
Ø16ØØ   C
Ø17ØØ   C       Blank out buffer so ille
gal characters don't goof up output
Ø18ØØ   C
Ø19ØØ           DO 5 J=1,1ØØ
Ø2ØØØ           DO 5 I=2,16
Ø21ØØ           BUFFER(I,J) = ' '
Ø22ØØ   5       CONTINUE
Ø23ØØ   C
Ø24ØØ   C       Get disk#, sort mode and
 output device
Ø25ØØ   C
Ø26ØØ           WRITE (3,7)
Ø27ØØ   7       FORMAT (' Enter drive #'
,/)
Ø28ØØ           READ (3,1Ø) DSK
Ø29ØØ   1Ø      FORMAT (I1)
Ø3ØØØ
Ø31ØØ           WRITE (3,15)
Ø32ØØ   15      FORMAT (' Ascending sort
 enter Ø, Descending sort enter 1',/)
Ø33ØØ           READ (3,2Ø) MODE
Ø34ØØ   2Ø      FORMAT (I1)
Ø35ØØ
Ø36ØØ           WRITE (3,25)
Ø37ØØ   25      FORMAT (' Output device
#',/,
Ø38ØØ         2  ' 2    = Printer',/,' 3
   = Display',/,' 6-1Ø = Disk files',/)
Ø39ØØ           READ (3,3Ø) DEVICE
Ø4ØØØ   3Ø      FORMAT (I1)
Ø41ØØ
Ø42ØØ   C       Get directory in RAM and
 sort using routine written
Ø43ØØ   C       in assembly language usi
ng supervisor calls.
Ø44ØØ   C
Ø46ØØ   C
Ø47ØØ           CALL DIRØ
Ø48ØØ   C
Ø49ØØ   C       Check for error and term
inate if found.
Ø5ØØØ   C
Ø51ØØ           IF (DSK.EQ.99) GOTO 999
Ø52ØØ
Ø53ØØ   C
Ø54ØØ   C       Remove unwanted characte
rs from listing so display will
Ø55ØØ   C       be easily readable.
Ø56ØØ   C
Ø57ØØ           DO 5Ø J=1,1ØØ
Ø58ØØ           COLON = .FALSE.
Ø59ØØ           DO 5Ø I=2,16
Ø6ØØØ           IF (BUFFER(I,J).EQ.58) C
OLON = .TRUE.
Ø61ØØ           IF (.NOT.COLON) GOTO 5Ø
Ø62ØØ           BUFFER(I,J) = ' '
Ø63ØØ   5Ø      CONTINUE
Ø64ØØ
Ø65ØØ   C
Ø66ØØ   C       Move lines to printing b
uffer and print on device.
Ø67ØØ   C
Ø68ØØ           DO 1ØØ J=1,2Ø
Ø69ØØ           DO 9Ø K=2,16
Ø7ØØØ           BUF(K-1)=BUFFER(K,J)
Ø71ØØ           BUF(K+14)=BUFFER(K,J+2Ø)
Ø72ØØ           BUF(K+29)=BUFFER(K,J+4Ø)
Ø73ØØ           BUF(K+44)=BUFFER(K,J+6Ø)
Ø74ØØ           BUF(K+59)=BUFFER(K,J+8Ø)
Ø75ØØ   9Ø      CONTINUE
Ø76ØØ           WRITE (DEVICE,6Ø) (BUF(K
),K=1,75)
Ø77ØØ   6Ø      FORMAT (' ',(75A1))
Ø78ØØ   1ØØ     CONTINUE
Ø79ØØ   999     END
```

Computer ease

Software is the most important part, but what can it do?

For all readers

Mark E. Renne, Contributing editor

We've looked at most of the hardware parts of the computer in the past few months. Hardware consists of the computer, disk drives, printer, and possibly a modem. Even though these devices are important to the system, the most important part of the computer is the software. This is the program (or programs) that actually makes the computer useful as a problem solver. This month, we'll look at major types of software and what they can do for you.

What do you mean, software's the most important part?

Many people overlook the importance of software. In fact, a computer without the right software is merely a very nice looking box that serves no useful purpose. The fastest CPU in the west is no good without the appropriate program. When buying a computer, always think software first.

What do most people use computers for?

There are as many uses for computers as there are people who use them. The most common uses are word processing, spreadsheet analysis, accounting, education, entertainment, and data base management. These are six very general categories which are each made up of several hundred variations. For example, there are many different accounting packages written specifically for businesses ranging from farming to publishing. There are very few businesses that don't have software specifically written for their needs.

What's a word processor?

A word processor is the answer to a writer's prayers. It allows you to create letters, documents, or manuscripts with much more flexibility than the common typewriter. Words or phrases can be moved throughout the document with the touch of a couple of keys. Lawyers can complete 40-page contracts by simply inserting the "party of the first part" and "party of the second part" a single time. The word processor also takes care of margins and justification. Justification is the process of inserting "white space" between words to make all lines exactly the same length. Take a look at the right and left edges of this column. See how all the lines line up at the edge? There's a computer at work here.

Word processors can also take care of footnotes, quotations, headers and footers. Entire books can be typeset using a computer without ever putting a single word on paper. There are hundreds of word processors, each with its own unique features. Programs are also available to check your spelling and even your grammar. If you write for a living, a word processor is becoming a must. If you type like I do, it's a blessing.

What's a spreadsheet?

This is a computerized version of the columnar pad. Anyone involved with forecasting has become fondly attached to it. This program allows you to do "what if" calculations based on changing external factors. Pretend you happen to own the largest widget manufacturing plant in the U.S. It appears that the price of widget grease may be increasing by 10 percent next year. What price will you have to sell your widgets at to maintain the same profit percentage? What price would you sell them at to keep the same total dollar amount of profit? Can you compete against the Burmese widget?

These questions and many others could be answered quickly by your spreadsheet program. You might even ask for the effect of widget grease varying by nine percent, or seven percent. This type of instant calculation, where all items change in relation to others, is priceless to busy executives. What might have taken hours to calculate can now be done in seconds. Without a doubt, spreadsheets have changed financial modeling forever.

What type of accounting programs are there?

Again, there are hundreds of different accounting packages. General ledgers, accounts receivable, accounts payable, inventory, payroll and cost accounting can all be done by the computer. Software packages are also written to handle these accounting activities for specific types of businesses. There are many medical accounting packages which are tailored especially for the medical profession. Recently, many packages have started popping up for the farming profession where good bookkeeping can make the difference between survival and failure. All of these programs make accounting easier and generate far more reports than any person could be expected to by hand.

A computer, with the appropriate software, can make any business more efficient and profitable. You

must realize, however, that the transition to computerized bookkeeping is not without its problems. You should keep both manual and electronic books for several months to make sure the bugs are out of the system. I always recommend a year just to make sure. Also, if you don't keep manual books in order, don't expect a computer to solve your organizational problems. An organized business profits by the addition of a computer, a disorganized business simply becomes more disorganized at a faster rate with a computer.

What type of educational programs are there?

Computers can be used to augment instruction in just about any subject. They can teach foreign languages, advanced math, elementary math, and even computer science. Note that I said augment. No computer could teach anything without the help of a highly skilled educator. Education is dependent on proper lesson preparation and sequence. Computers will never replace teachers, but computers can serve as another tool for overworked professionals.

Computers in the classroom are exciting for both students and teachers. They can be used to help slower students keep up with the rest of the class. They can also present a challenge to those who are at the top of the class. Computer Assisted Instruction (CAI) is especially useful for instruction that requires repetition. Multiplication practice can be done on the computer while the teacher is free to help other students that may require more personalized instruction. Computers can also be used in the home to educate. Remember, the same rule applies here. Parents can't be replaced by computers, only helped.

But can computers play Space Invaders?

The entertainment value of computers is demonstrated by the great number of video arcades appearing all over the country. The games that you put your quarters into are simply dedicated computers, programmed to beat your socks off. The same games found in these arcades can be purchased for most computers. There are certain advantages to playing games on your computer instead of at the arcade. It's definitely cheaper. Even for the most devoted computer scientist, there's nothing like a good adventure. Good games can educate as well as entertain.

What's a data base and why do they need managing?

A data base is simply a collection of facts or figures. It might be a list of names and addresses or a current inventory of automobile parts. It could be as simple as your Christmas card list, or as complicated as the federal budget. A data base manager, DBM, is simply a program that "manipulates" this data base. A good DBM will allow you to reference your data in hundreds of different ways. You might want a listing of world famous writers that publish in *Basic Computing, National Geographic* and *Newsweek*. Albeit a short list when finished, it would be quite a task to sort through by hand thousands of writers looking for the common variables.

DBM's can also be used to create mass mailings in combination with an excellent word processor. These generate letters you receive telling you about driving down your street in your new car after subscribing to several magazines. These programs are extremely useful and can take the place of several different programs if they're carefully written.

That's a quick survey of the major types of programs available for most computers. The most difficult choice you'll have to make is not which type you need, but which of the several hundred programs within each type you really want. Next time, we'll look at telecommunications and the outside world. Questions are always welcome and so are answers. Remember, computers are fun and understanding them is easier than you think. Happy computing.

Advance to go

Compute the odds for the grand old game of Monopoly™

Models I/II/III/4III/12/16II/Color Computer

Joey Robichaux

Jaded by adventures? Bored stiff with space war shoot-em-ups? Think you'll lose your cookies if you swallow one more power pellet? Then come back — visit the pre-smart bomb, pre-electronic world of game playing . . . back then, deciding what game to play was easy. There was only one king, the rest were pretenders to the throne.

And just what game was played time and time again, above all others? Why, what other game than a cut-throat, no-holds-barred, damn-the-torpedoes marathon session of Monopoly!

Conceived by an unemployed heating and engineering equipment salesman to divert the real-world gloom of the Great Depression, Monopoly has since delighted the millions who have explored this fantasy world of paupers and tycoons.

Now, however, Monopoly's time has passed. Out of touch with our high-tech society, Monopoly must follow the paths of shepherds, horses and buggies, and bustle manufacturers. For gamesters, home computers reign supreme — Monopoly, the king, is dead.

Or is he? Can a royal marriage, computers and the grand old game, produce new insights to renew the pleasures found in a game of Monopoly? Why, of course, as it's been said over and over, computers can do anything!

Computers excel at simulations, especially at mimicking a particular task. Programs can tally occurrences of special events and people can draw conclusions from the results of simulations. For example, a program can simulate a game of Monopoly. The program can "roll" a pair of dice, "move" a playing piece, "dole" out rewards, and "collect" penalties. For a short game or perhaps a few hundred moves, the results can be interesting.

Roll the dice thousands and thousands, or even millions, of times, however, and statistical patterns emerge. These statistical tendencies provide new insights to Monopoly, insights that yield new playing strategies.

One question a Monopoly player might like to answer is, "Are some properties landed on more often than others? There are 40 properties, or squares, on a Monopoly board. If every possible move is equally likely and is unaffected by any outside event, then each square should be visited 2.5% of the time (100%/40 squares).

Program Listing 1 examines this question. Line 70 heads a loop that simulates approximately 10,000 die rolls in Monopoly. Line 60 drives this simulation loop 100 times. This totals a 24-hour marathon session of over a million moves. Change line 60 to FOR N=1 TO 1 for a shorter session. *(As written, Listing 1 will take hours to yield results. —Ed)*

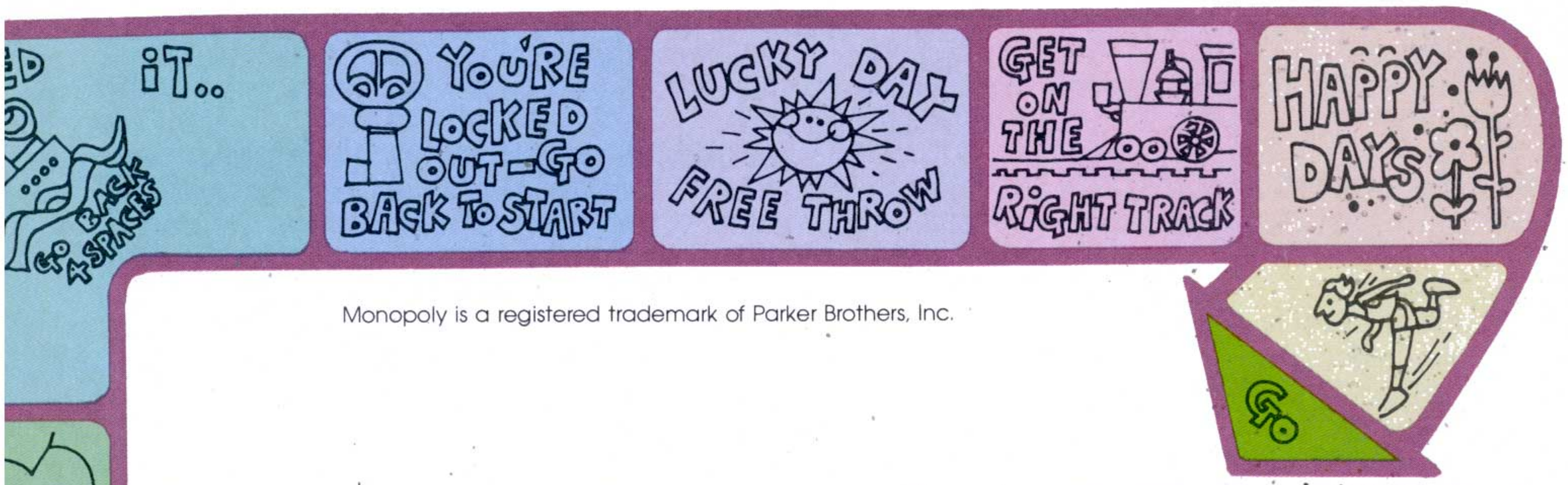
Run the program — the overall results will probably not surprise you, but individual results almost certainly will.

It won't ruin any surprise to say that some squares are landed on more often than others. Since the possible moves are not equally likely, and since moves are affected by outside events (Community Chest, Chance, Go To Jail, etc.), you can easily explain the variety of different landing percentages.

It takes a little more thought to explain some of the results. Remember, if one square is landed on more often than any other square, that square is a starting place more often than any other square. What will that initial move average?

Actually, a Monopoly player really isn't interested in how often a property is visited. He's interested in how much money a piece of property is worth. Does a low rent, high landing percentage property return more cash than a high rent, low landing percentage property (or a medium rent, medium landing percentage property)?

We can also examine each property's potential worth. Once the program determines the landing percentages, it multiplies each percentage by the maximum amount of rent the property can return.



For example, a property that returns $100 in rent (with a hotel, of course) has a landing percentage of 7.5%. Every time an opponent makes a circuit of the playing board, that property will return an average of $7.50.

Once you've determined each property's potential worth, the next question is, "What color group is the most valuable?"

Finally, the questions are all settled. Should you wheel and deal to gather the whole red-yellow side? Should you hoard those orange $500's and hold out for Park Place and Boardwalk? What good are utilities, anyway? Should you trade for a ride on the Reading? Let the other guys worry about it. You'll be a tycoon and tycoons don't wonder, they know.

Program Listing for Advance to Go

```
10 ' PROGRAM NUMBER THREE
20 CLS
30 DIM C(40), P(40), M$(40), PO(40)
40 FOR I=1 TO 40: READ M$(I): NEXT
50 D=0: DB=0  ' DB=CHECK FOR DOUBLES 3 T
IMES IN A ROW
60 FOR N=1 TO 100   'CONTROL LOOP
70 FOR I=1 TO 10000 'CONTROL LOOP ALSO
80 A = INT(RND(0)*6+1): B = INT(RND(0)*6
+1)  ' ROLL DICE
90 IF A=B THEN DB=DB+1: ELSE DB=0  ' DOU
BLES?
100 IF DB=3 THEN D=10: DB=0: GOTO 280  '
GO TO JAIL
110 D = A+B+D: IF D>40 THEN D=D-40  ' MO
VE PLAYING PIECE
120 IF D=7 OR D=22 OR D=36 THEN GOTO 680
  'CHANCE CARD
130 IF D=2 OR D=17 OR D=33 THEN GOTO 770
  'COMMUNITY CHEST
140 IF D=30 THEN C(D)=C(D)+1: GOTO 190
'GO TO JAIL
150 C(D) = C(D)+1  'COUNT NUMBER OF TIME
S LANDED
160 IF L=0 THEN 330  'NOTHING SPECIAL HA
PPENED
170 ON L GOTO 180  , 190  , 200  , 210
, 220  , 230  , 240  , 250  , 260  , 270
180 D=40: L=0: GOTO150   'ADVANCE TO GO
190 D=10: L=0: GOTO280   'GO TO JAIL
200 D=D-3: L=0: GOTO150  'GO BACK 3 SPAC
ES
210 D=39: L=0: GOTO150   'GO TO BOARDWAL
K
220 D=24: L=0: GOTO150   'GO TO ILLINOIS
 AVE
230 D=11: L=0: GOTO150   'GO TO ST. CHAR
LES PLACE
240 D=10: L=0: GOTO280   'GO TO JAIL (AG
AIN)
250 D=40: L=0: GOTO150   'ADVANCE TO GO
260 D=5 : L=0: GOTO150   'GO TO READING
RAILROAD
270 D=DN: L=0: GOTO150   'NEAREST RR OR
UTILITY
280 C(10) = C(10) + 1    'GET OUT OF JAI
L ROUTINE
290 FOR MM=1 TO 3: A=INT(RND(0)*6+1): B=
INT(RND(0)*6+1)
300 IF A=B THEN 90       'ROLL DOUBLES T
O GET OUT
310 NEXT MM              'OF JAIL?
320 GOTO 90
330 NEXT I
340 NEXT N
350 X=0: FOR I=1 TO 40: X=X+C(I): NEXT
  'TALLY TOTAL LANDINGS
360 FOR I=1 TO 40
370 P(I) = C(I) / X      'DETERMINE PERC
ENTAGES
380 NEXT I
390 FOR I=1 TO 40
400 P(I) = INT(P(I)*1000)   'CLEAN PERCE
NTS UP
410 P(I) = P(I)/10
420 NEXT I
430 CLS
440 ' PRINT PROPERTY LANDING AMOUNTS AND
 PERCENTS
450 FOR I=1 TO 10
460 PRINT M$(I),"was landed on";C(I);"ti
```

```
mes. ";P(I);"%"
470 NEXT I
480 PRINT:INPUT"Press <ENTER> for next s
creen";Z$
490 CLS
500 FOR I=11 TO 20
510 PRINT M$(I),"was landed on";C(I);"ti
mes. ";P(I);"%"
520 NEXT I
530 PRINT: INPUT "Press <ENTER> for next
 screen";Z$
540 CLS
550 FOR I=21 TO 30
560 PRINT M$(I),"was landed on";C(I);"ti
mes. ";P(I);"%"
570 NEXT I
580 PRINT: INPUT "Press <ENTER> for next
 screen";Z$
590 CLS
600 FOR I=31 TO 40
610 PRINT M$(I),"was landed on";C(I);"ti
mes. ";P(I);"%"
620 NEXT I
630 PRINT
640 PRINT"Press <Y> to redisplay percent
s, or <N> to continue"
650 GOSUB1360
660 IFK$="Y"THEN430
670 GOTO850
680 K = INT(RND(0)*16+1)     'PULL A CHAN
CE CARD
690 IF K>10 THEN 150         'NO AFFECT
700 IF K<8 THEN L=K+2:GOTO150  'LOAD "ON
 L GOTO"
710 L = 10
720 'FOLLOWING FINDS NEXT RAILROAD OR UT
ILITY
730 IF K=8 THEN IF D<11 OR D>29 THEN DN=
12: ELSE DN=28
740 IF K=8 THEN GOTO150   'ADVANCE TO UT
ILITY
750 IF D=7 THEN DN=15: ELSE IF D=22 THEN
 DN=25: ELSE DN=5
760 GOTO 150                 'ADVANCE TO RA
ILROAD
770 K = INT(RND(0)*16+1)  'PULL COMMUNIT
Y CHEST CARD
780 IF K=1 THEN L=1: GOTO 150  'ADVANCE
TO GO
790 IF K=2 THEN L=2: GOTO 150  'GO TO JA
IL
800 GOTO 150
810 DATA"MEDITERRANEAN ","COMMUNITY CHES
T","BALTIC AVE","INCOME TAX","READING RR
","ORIENTAL AVE","CHANCE","VERMONT AVE",
"CONNECTICUT AVE","JAIL"
820 DATA"ST CHARLES PL","ELECTRIC CO","S
TATES AVE","VIRGINIA AVE","PENNSYLVANIA
RR","ST JAMES PLACE","COMMUNITY CHEST","
TENNESSEE AVE","NEW YORK AVE","FREE PARK
ING"
830 DATA"KENTUCKY AVE","CHANCE","INDIANA
 AVE","ILLINOIS AVE","B&O RR","ATLANTIC
AVE","VENTNOR AVE","WATER WORKS","MARVIN
 GARDENS","GO TO JAIL"
840 DATA"PACIFIC AVE","NORTH CAROLINA ",
"COMMUNITY CHEST","PENNSYLVANIA ","SHORT
 LINE","CHANCE","PARK PLACE","LUXURY TAX
","BOARDWALK","GO"
850 FOR I=1 TO 40
860 READ PO(I)
870 PO(I) = PO(I) * P(I)/100
880 PO(I) = INT(PO(I)*100)
890 PO(I) = PO(I)/100
900 NEXT I
910 CLS
920 PRINT: PRINT: PRINT,"PROPERTY POTENT
IAL WORTH":PRINT
930 PRINT"     The potential worth of an
y property is based on "
940 PRINT"the maximum revenue possible (
property w/hotel, own "
950 PRINT"all the railroads, etc.) multi
plied by the landing"
960 PRINT"probability.  If a property re
turns $100 in rent"
970 PRINT"and there is a 7.5% chance of
landing on that"
980 PRINT"property, then the potential w
orth is $7.50.
990 PRINT:PRINT:PRINT:INPUT"Press <ENTER
> when ready";Z$
1000 CLS
1010 FOR I=1 TO 10
1020 PRINT M$(I);"'s potential worth is
$";PO(I)
1030 NEXT
1040 PRINT: INPUT"Press <ENTER> for next
 screen";Z$
1050 CLS
1060 FOR I=11 TO 20
1070 PRINT M$(I);"'s potential worth is
$";PO(I)
1080 NEXT
1090 PRINT: INPUT"Press <ENTER> for next
 screen";Z$
1100 CLS
1110 FOR I=21 TO 30
1120 PRINT M$(I);"'s potential worth is
$";PO(I)
1130 NEXT
1140 PRINT: INPUT"Press <ENTER> for next
 screen";Z$
1150 CLS
```

```
116Ø FOR I=31 TO 4Ø
117Ø PRINT M$(I);"'s potential worth is
$";PO(I)
118Ø NEXT
119Ø PRINT: PRINT"Press <Y> to redisplay
 worth's, <N> to continue"
12ØØ GOSUB136Ø
121Ø IF K$="Y" THEN GOTO 1ØØØ
122Ø K=1
123Ø FOR I=1 TO 39
124Ø IF PO(I+1) > PO(K) THEN K=I+1
125Ø NEXT
126Ø CLS
127Ø PRINT:PRINT:PRINT
128Ø PRINTM$(K);" has the highest potent
ial worth of $";PO(K)
129Ø PRINT:PRINT
13ØØ INPUT"Press <ENTER> to continue";Z$
131Ø GOTO138Ø
132Ø DATA25Ø,Ø,45Ø,Ø,2ØØ,55Ø,Ø,55Ø,6ØØ,Ø
133Ø DATA75Ø,7Ø,75Ø,9ØØ,2ØØ,95Ø,Ø,95Ø,1Ø
ØØ,Ø
134Ø DATA 1Ø5Ø,Ø,1Ø5Ø,11ØØ,2ØØ,115Ø,115Ø
,7Ø,12ØØ,Ø
135Ø DATA 1275,1275,Ø,14ØØ,2ØØ,Ø,15ØØ,Ø,
2ØØØ,Ø
136Ø K$=INKEY$:IFK$=""THEN136Ø
137Ø IFK$="Y"THEN RETURN:ELSEIFK$="N"THE
N RETURN:ELSEGOTO136Ø
138Ø CLS:PRINT"VALUES BY COLOR GROUP"
139Ø PRINT"  SIDE 1"
14ØØ P1=PO(1)+PO(3):P2 = PO(6) + PO(8) +
 PO(9)
141Ø PRINT"PURPLE = $";P1,"LT BLUE = $";
P2
142Ø PRINT:PRINT"  SIDE 2"
143Ø P1=PO(11) + PO(13) + PO(14): P2 = P
O(16) + PO(18) + PO(19)
144Ø PRINT"VIOLET = $";P1,"ORANGE = $";P
2
145Ø PRINT:PRINT"  SIDE 3"
146Ø P1=PO(21) + PO(23) + PO(24): P2=PO(
26) + PO(27) + PO(28)
147Ø PRINT"RED   =   $";P1,"YELLOW = $";
P2
148Ø PRINT:PRINT"  SIDE 4"
149Ø P1=PO(31) + PO(32) + PO(34): P2=PO(
37) + PO(39)
15ØØ PRINT"GREEN  =  $";P1,"BLUE =$";P2
151Ø PRINT
152Ø PRINT"RAILROADS = $";PO(5)+PO(15)+P
O(25)+PO(35),"UTILITIES = $";PO(12)+PO(2
8)
153Ø GOTO153Ø
```

Full-function Model 100 disassembler

See what is going on inside

Model 100

Joseph L. Hartmann, Jr.

This is a utility program for the Radio Shack TRS-80 Model 100. It takes up about 3.5K bytes, which leaves you with about 1.5K bytes free when you are using an 8K Model 100.

It consists of memory examine, memory modify, disassemble, interpret memory as 16-bit number from 0 to 65535, request new address to memory examine, jump to out-of-line code (if desired by user), return from out-of-line code (if desired by user), and print disassembled output to the printer.

The mnemonics of the disassembler are Zilog mnemonics since they seemed preferable to the Intel mnemonics. The only difference in the mnemonics is that the Zilog "add a,n" are rendered here as "add n." Since the only thing you can add the n to is the a register, we left out this redundancy. You can put it in if you modify the proper string.

The op codes for the 8085 are, for the most part, quite regular, leading to the algorithmic generation of the mnemonics for most cases. This accounts for the small size of this disassembler even though it is written in BASIC.

When outputting to the display, the disassembler requires operator input for every line disassembled. This seemed practical since disassembly is very "thinking oriented." You are constantly asking yourself, "Should I branch, or continue straight through?"

When printout of the disassembly is requested, printing will automatically continue unless an instruction is disassembled, which may result in an out-of-line jump. In this case, the printing stops and a message appears on the display asking you to give the command to jump, continue, return, or go back to non-printing command mode.

The disassembler is decimal

Table 1 — 8085 OP Codes/Zilog Mnemonics

0	1	2	3	4	5	6	7
nop	ld bc,nn	ld (bc),a	inc bc	inc b	dec b	ld b,n	rlca
8	9	10	11	12	13	14	15
illeg	add hl,bc	ld a,(bc)	dec bc	inc c	dec c	ld c,n	rrca
16	17	18	19	20	21	22	23
illeg	ld de,nn	ld (de),a	inc de	inc d	dec d	ld d,n	rla
24	25	26	27	28	29	30	31
illeg	add hl,de	ld a,(de)	dec de	inc e	dec e	ld e,n	rra
32	33	34	35	36	37	38	39
rim	ld hl,nn	ld (nn),hl	inc hl	inc h	dec h	ld h,n	daa
40	41	42	43	44	45	46	47
illeg	add hl,hl	ld hl,(nn)	dec hl	inc l	dec l	ld l,n	cpl
48	49	50	51	52	53	54	55
sim	ld sp,nn	ld (nn),a	inc sp	inc (hl)	dec (hl)	ld (hl),n	scf
56	57	58	59	60	61	62	63
illeg	add hl,sp	ld a,(nn)	dec sp	inc a	dec a	ld a,n	ccf
64	65	66	67	68	69	70	71
ld b,b	ld b,c	ld b,d	ld b,e	ld b,h	ld b,l	ld b,(hl)	ld b,a
72	73	74	75	76	77	78	79
ld c,b	ld c,c	ld c,d	ld c,e	ld c,h	ld c,l	ld c,(hl)	ld c,a
80	81	82	83	84	85	86	87
ld d,b	ld d,c	ld d,d	ld d,e	ld d,h	ld d,l	ld d,(hl)	ld d,a
88	89	90	91	92	93	94	95
ld e,b	ld e,c	ld e,d	ld e,e	ld e,h	ld e,l	ld e,(hl)	ld e,a
96	97	98	99	100	101	102	103
ld h,b	ld h,c	ld h,d	ld h,e	ld h,h	ld h,l	ld h,(hl)	ld h,a
104	105	106	107	108	109	110	111
ld l,b	ld l,c	ld l,d	ld l,e	ld l,h	ld l,l	ld l,(hl)	ld l,a
112	113	114	115	116	117	118	119
ld (hl),b	ld (hl),c	ld (hl),d	ld (hl),e	ld (hl),h	ld (hl),l	halt	ld (hl),a
120	121	122	123	124	125	126	127
ld a,b	ld a,c	ld a,d	ld a,e	ld a,h	ld a,l	ld a,(hl)	ld a,a
128	129	130	131	132	133	134	135
add b	add c	add d	add e	add h	add l	add (hl)	add a
136	137	138	139	140	141	142	143
adc b	adc c	adc d	adc e	adc h	adc l	adc (hl)	adc a

oriented. Since PEEK and POKE work with decimal arguments, why not the disassembler?

The commands which the command processor (lines 40 through 70) recognizes are as follow (note that CR stands for Carriage Return, or the ENTER key on the Model 100):

nnn CR Where nnn is a number from 0 to 255. This causes the number entered to be POKEd into RAM at the location being examined.

DR Just a carriage return causes the next higher memory location to be examined.

h CR This causes the memory being examined to be interpreted as the high byte of a two-byte 16-bit number. The low byte is the previous memory location. These two bytes are interpreted as a 16-bit, unsigned number whose value is from 0 to 65535, inclusive. This value is printed out and the next higher location is examined.

j CR When you ask for a disassembly of a memory location, the result may be a jump (JP), call (CALL), or restart (RST). Many of the jump and call instructions are conditional, which means you must decide whether or not you want to make the jump to see what the code is doing at the jump target. To make the jump, you type j CR and you will see the memory examined at the target of the jump. The j CR command automatically stores the return location, so when you come to the end of the call, you may return from whence you came.

r CR After you have jumped (for

144	145	146	147	148	149	150	151
sub b	sub c	sub d	sub e	sub h	sub l	sub (hl)	sub a
152	153	154	155	156	157	158	159
sbc b	sbc c	sbc d	sbc e	sbc h	sbc l	sbc (hl)	sbc a
160	161	162	163	164	165	166	167
and b	and c	and d	and e	and h	and l	and (hl)	and a
168	169	170	171	172	173	174	175
xor b	xor c	xor d	xor e	xor h	xor l	xor (hl)	xor a
176	177	178	179	180	181	182	183
or b	or c	or d	or e	or h	or l	or (hl)	or a
184	185	186	187	188	189	190	191
cp b	cp c	cp d	cp e	cp h	cp l	cp (hl)	cp a
192	193	194	195	196	197	198	199
ret nz	pop bc	jp nz,nn	jp nn	call nz,nn	push bc	add n	rst 0
200	201	202	203	204	205	206	207
ret z	ret	jp z,nn	illeg	call z,nn	call nn	adc n	rst 8
208	209	210	211	212	213	214	215
ret nc	pop de	jp nc,nn	out n	call nc,nn	push de	sub n	rst 16
216	217	218	219	220	221	222	223
ret c	illeg	jp c,nn	in n	call c,nn	illeg	sbc n	rst 24
224	225	226	227	228	229	230	231
ret po	pop hl	jp po,nn	ex (sp),hl	call po,nn	push hl	and n	rst 32
232	233	234	235	236	237	238	239
ret pe	jp (hl)	jp pe,nn	ex de,hl	call pe,nn	illeg	xor n	rst 40
240	241	242	243	244	245	246	247
ret p	pop af	jp p,nn	di	call p,nn	push af	or n	rst 48
248	249	250	251	252	253	254	255
ret m	ld sp,hl	jp m,nn	ei	call m,nn	illeg	cp n	rst 56

The numbers above each mnemonic are the op codes in decimal. "n" is a byte operand from 0 to 255, inclusive. "nn" is a two-byte operand (low byte for lower address, high byte for higher address) from 0 to 65535, inclusive. Parentheses mean "the contents of." For example, ld hl,(nn) means the contents of memory location nn are loaded into register l and the contents of memory location nn+1 are loaded into register h. Operators with two operands cause data to be moved from right to left, e.g., ld c,d causes the value in the d register to be copied into the c register.

example, to a call routine), you may want to jump back again from whence you came. For example, when you come to the end of your call subroutine, the final command is a ret. Pressing the r CR keys will cause you to return.

m CR m is for minus. This allows you to examine the preceding memory address. Repeated pressing of m CR will allow you to walk backward through memory.

a CR a will prompt you for a new address. Respond to the prompt with a number from 0 to 65535 and you will display the contents of this memory location.

d CR d is for disassemble. This causes the memory location being examined to be treated as an op code and to be disassembled. Op codes are either one, two, or three bytes long, and the disassembly will automatically advance to the next op code. It shows the operand in the case of a two- or three-byte op code in the mnemonic style of Zilog with the exception previously noted.

p CR p is for print. This causes the memory examined to be disassembled and printed on the printer. Disassembly will continue automatically until an instruction is reached which may cause a jump, such as: call c,nn ; ret po ; rst 8 ; etc. When an instruction is disassembled which may result in an out-of-line jump in the actual execution of the code when run on the machine, the printout stops and a message is displayed, asking for your command to continue printing, jump and continue printing, return and continue printing, or quit automatic printing mode.

The structure of the disassembly part of the program is largely determined by the op codes themselves. The structure of the BASIC program makes use of line numbers to highlight its modular nature. Renumbering would make it harder to understand.

The disassembly is structured as a subroutine 400. It is called from line 60 when the user is desiring disassembly output to the display, and from line 520 when the automatic printout has been selected.

Op codes from 64 to 191 are very algorithmic and are disassembled by the lines 1000 to 1020. You will see this regularity mirrored in the op codes provided in Table 1.

Disassembly of the column of eight op codes starting with 0 is handled by line 2000. The column whose eight op codes start with 1 is disassembled by lines 2100 through 2130. The column whose eight op codes start with 2 is disassembled by lines 2200 through 2250, etc.

Disassembly of the column of eight op codes starting with 192 is handled by line 3000. The column whose eight op codes start with 193 is handled by lines 3100 through 3210. The column whose op codes start with 194 is handled by lines 3200 through 3210, etc.

When in the disassembly mode of the program, not every memory location is shown in decimal; those op codes that expect either a one-byte, or a two-byte operand are interpreted and listed as part of the mnemonic, and the program counter is automatically set to the next op code.

Lines 15, 17; 22, 24; and 900 to 934 specify the strings required and read this data into the appropriate string variables. Lines 30 to 110 form the command interpreter and can be considered to be the "main" part of the program. The program forces you to be in the lowercase mode. If you don't like this, you can change it by changing lines 50 through 70 from lowercase to uppercase, or both, if you desire.

The printing is structured so that printing to any device (line printer, com, filename, cas) can be accomplished just by changing line 502. The program as written is for the line printer, but changing line 502 to the device desired allows output to go wherever you specify.

References: Mostek Corp., 1215 W. Crosby Rd., Carrollton, TX 75006, "Z80 Programming Manual," publication number MK 78515. Intel Corp., 3065 Bowers Ave., Santa Clara, CA 95051, "8080/8085 Assembly Language Programming," manual order number 9800940.

Program Listing

```
1 'filename is peek1.do
2 ' COPYRIGHT 1983 BY JOSEPH L.HARTMANN, JR.
```

```
5 PRINT"caps lock off"
10 INPUT"st.adr=";A
12 OPEN"LCD:"FOROUTPUTAS1
15 DATA"b","c","d","e","h","l","(hl)
","a"
17 FORI=0TO7:READC$(I):NEXT
20 CLS
22 DATA" add"," adc"," sub"," sbc","
and"," xor","  or","  cp"
24 FORI=0TO7:READD$(I):NEXT
25 GOSUB900
30 GOSUB330
40 B$=INKEY$:IFB$=""THEN40
42 IFASC(B$)>47ANDASC(B$)<58THEN150
45 IFB$=CHR$(13)GOTO90
50 IFB$="h"GOTO200
55 IFB$="j"THENA=H:PRINT:RR=R:GOTO30
57 IFB$="m"THENA=A-1:PRINT:GOTO30
60 IFB$="d"THENGOSUB400
61 IFB$="a"THEN INPUT"enter new
address";A:GOTO30
65 IFB$="r"THENA=RR:PRINT:GOTO30
70 IFB$="p"THEN GOTO500
90 A=A+1:PRINT
110 GOTO30
150 B1$=B$
155 B$=INKEY$:IFB$=""THEN155
160 IFASC(B$)=13THENPOKEA,VAL(B1$)
:PRINT:GOTO30
165 B1$=RIGHT$(B1$+B$,3):GOTO155
200 GOSUB300
210 PRINTTAB(2)H;:GOTO90
300 H=256*PEEK(A)+PEEK(A-1):RETURN
330 PRINT#1,"(
";:PRINT#1,USING"#####";A;:PRINT#1,")
=";:PRINT#1,USING"###";PEEK(A);:RETURN
400 PRINT#1,"    :";:D=PEEK(A):IFD>63ANDD
<192GOTO1000
410 IFD<64THEN1100
420 I=0:D=D-192
430 IF(D-I)MOD8=0THENONI+
1GOTO3000,3100,3200,3300,3400,3500,3600,
3700
440 I=I+1:GOTO430
500 PRINT:CLOSE1
502 ' OPEN"LPT:"FOR OUTPUT AS 1
503  OPEN"com:57ile"FOR OUTPUT AS 1
505 PRINT#1,
510 GOSUB 330
515 OP=PEEK(A):OA=A
520 GOSUB 400
525 GOSUB700
530 A=A+1:PRINT#1,
540 GOTO510
700 IFOP<192THENRETURN
710
IFOP=201OROP=195OROP=233OROP=205THENGOTO
800
720 IF(OP-192)MOD8=0THEN800
730 IF(OP-194)MOD8=0THEN800
740 IF(OP-196)MOD8=0THEN800
750 IF(OP-199)MOD8=0THEN800
760 RETURN
800 PRINT#1,
805 CLOSE1
810 OPEN"LCD:"FOR OUTPUT AS 1
815 A=OA
820 GOSUB330
830 GOSUB 400
840
PRINT:B$="":INPUT"r=RET,j=JMP,c=CONT,q=Q
UIT";B$
850 IFB$="c"THENLETA=A+1:GOTO500
860 IFB$="j"THENA=H:RR=R:GOTO500
870 IFB$="r"THENA=RR:GOTO500
880 IFB$="q"THEN30
890 GOTO840
900
DATA"nz","z","nc","c","po","pe","p","m"
902 FORI=0TO7:READE$(I):NEXT
904 DATA"bc","de","hl","sp"
906 FORI=0TO3:READF$(I):NEXT
908 DATA"(bc),a","a,(bc)","(de),a","a,(
de)"
910 FORI=0TO3:READG$(I):NEXT
912 DATA"rlca","rrca"," rla"," rra","
daa"," cpl"," scf"," ccf"
914 FORI=0TO7:READI$(I):NEXT
916 DATA" nop","illeg","illeg","illeg","
rim","illeg"," sim","illeg"
918 FORI=0TO7:READJ$(I):NEXT
920 DATA"push bc","call","push
de","illeg","push hl","illeg","push
af","illeg"
922 FORI=0TO7:READK$(I):NEXT
924 DATA"  jp","illeg"," out","  in","
ex (sp),hl","  ex de,hl","  di","  ei"
926 FORI=0TO7:READM$(I):NEXT
928 DATA" pop bc","  ret","  pop
de","illeg","  pop hl","  jp (hl)","  pop
af","  ld sp,hl"
929 FORI=0TO7:READL$(I):NEXT
930 DATA" dec","  inc","  ret","call","
rst"
932 FORI=0TO4:READH$(I):NEXT
934 RETURN
1000 IFD>127THEN1020
1005 IFD=118THENPRINT#1,"halt";:RETURN
1010 PRINT#1,"  ld   ";:D=D-64:F=INT(D/8
):E=DMOD8:PRINT#1,C$(F)
;:PRINT#1,",";:PRINT#1,C$(E);:RETURN
1020 D=D-128:F=INT(D/8):E=D
MOD8:PRINT#1,D$(F);:PRINT#1,"
";:PRINT#1,C$(E);:RETURN
1100 I=0
1110 IF(D-I)MOD8=0THENONI+
1GOTO2000,2100,2200,2300,2400,2500,2600,
2700
1120 I=I+1:GOTO1110
2000 E=INT(D/8):PRINT#1,J$(E);:RETURN
2100 D=D-1:E=INT(D/8):IFEMOD2=0THEN2120
2110 PRINT#1,"  ";D$(0);"  ";F$(2);",";F$(
(E-1)/2);:RETURN
2120 PRINT#1,"  ld  ";F$(E/2);",";:A=A+
2:GOSUB300
2130 PRINT#1,H;:RETURN
2200 D=D-2:E=INT(D/8):IFE>3THEN2220
2210 PRINT#1,"  ld  ";G$(E);:RETURN
2220 A=A+2:GOSUB300
2230 PRINT#1,"  ld  ";:IFE=4 OR
E=6THENPRINT#1,"(";H;")
,";:IFE=4THENPRINT#1,F$(2)
;:RETURNELSEPRINT#1,"a";:RETURN
2240 IFE=5THENPRINT#1,"hl,(";H;")
";:RETURN
2250 IFE=7THENPRINT#1,"a,(";H;")
";:RETURN
2300 D=D-3:E=INT(D/8):IFEMOD2=0THEN2320
2310 PRINT#1,H$(0);"  ";:PRINT#1,F$((E-1)
/2);:RETURN
2320 PRINT#1,H$(1);"  ";:PRINT#1,F$(E/2)
;:RETURN
2400 D=D-4:E=INT(D/8):I=1
2410 PRINT#1,H$(I);"  ";:PRINT#1,C$(E)
;:RETURN
2500 D=D-5:E=INT(D/8):I=0:GOTO2410
2600 D=D-6:E=INT(D/8):PRINT#1,"  ld
";:PRINT#1,C$(E)
;:PRINT#1,",";:PRINT#1,PEEK(A+1);:A=A+
1:RETURN
2700 D=D-7:E=INT(D/8):PRINT#1,I$(E)
;:RETURN
3000 PRINT#1,H$(2);"  ";:PRINT#1,E$(INT(D
/8));:RETURN
3100 D=D-1:E=INT(D/8):PRINT#1,L$(E)
;:RETURN
3200 D=D-2:PRINT#1,M$(0);"  ";:PRINT#1,E$
(INT(D/8));",";:A=A+2:GOSUB300
3210 PRINT#1,H;:R=A+1:RETURN
3300 D=D-3:E=INT(D/8):PRINT#1,M$(E);
3310 IFE=2ORE=3THENPRINT#1,"
";:PRINT#1,PEEK(A+1);:A=A+1:RETURN
3320 IFE<>0THENPRINT#1,;:RETURN
3330 PRINT#1,"  ";:A=A+2:GOSUB300
3340 PRINT#1,H;:R=A+1:RETURN
3400 PRINT#1,H$(3);"  ";:PRINT#1,E$(INT((
D-4)/8));:A=A+2:GOSUB300
3410 PRINT#1,",";H;:R=A+1:RETURN
3500 PRINT#1,K$(INT(D-5)/8);:IFINT(((D-5
)/8))<>1THENPRINT#1,;:RETURN
3510 PRINT#1,"  ";:A=A+2:GOSUB300
3520 PRINT#1,H;:R=A+1:RETURN
3600 PRINT#1,D$(INT((D-6)/8));"
";:PRINT#1,PEEK(A+1);:A=A+1:RETURN
3700 PRINT#1,H$(4);:H=INT((D-7)/8)*
8:PRINT#1,H;:R=A+1:RETURN
```

THE TANDY TRS-80® MODEL

The New Ultra-High Performance MS™-DOS System from Radio Shack

Our remarkable Tandy Model 2000 delivers much more than other 16-bit MS-DOS based computers. More speed. More disk storage. More expansion. Higher resolution graphics. But that's not all. You get the quality and support that have kept Tandy Corporation in the forefront of the microcomputer industry from the beginning.

Higher-Speed 16-Bit CPU

Model 2000 is dramatically faster than other 16-bit MS-DOS computers. In actual benchmark comparisons, the Model 2000 performed almost three times faster than the IBM® Personal Computer. That means all the applications software runs faster, so you get the job done quicker.

Increased Storage Capacity

Each of the two Model 2000 disk drives stores over four times as much information as a standard IBM PC drive, and twice as much as a TI® Professional drive. If you need even more storage, choose our Model 2000 HD with a built-in hard disk.

Get a Better Picture of Things

Model 2000 video monitors and graphics options let you easily create stunning, highly-detailed business charts and graphs. Our color graphics have twice the resolution of the COMPAQ™ Portable or IBM PC color graphics, and twice the number of colors.

TRS-80 Model 2000 With Two Disk Drives

2750⁰⁰

Commerical Lease Available For Only $95 Per Month (Plus Applicable Use/Sales Tax)

TRS-80 Model 2000 HD With Built-In 10-Megabyte Hard Disk

4250⁰⁰

Commercial Lease Available For Only $150 Per Month (Plus Applicable Use/Sales Tax)

Monitor, graphics, Floor stand, Pedestal and Digi™-Mouse not included

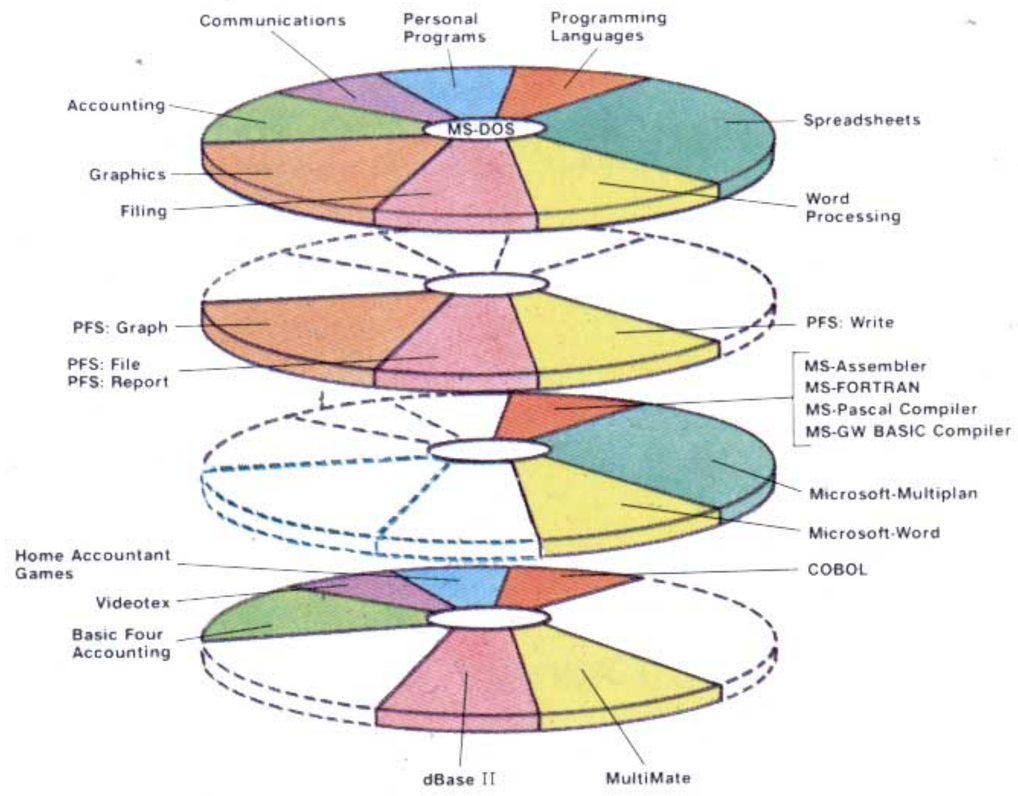
Compare the Model 2000 to the IBM Personal Computer

Price Comparison†	Tandy Model 2000	IBM Personal Computer
Base Unit	$2750	$2104
2nd Drive	Included (720K)	$529 (320K)
Monochrome Monitor	$249	$345
Display/Printer Adapter	Included	$335
128K RAM	Included	$165
RS-232	Included	$120
MS-DOS 2.0	Included	$60
Total Cost*	$2999	$3658
Feature Description	**Tandy Model 2000**	**IBM Personal Computer**
Internal Memory	128K Standard	64K Standard
Disk Capacity Per Drive	720K	160K or 320K (optional)
Microprocessor Clock Speed	8 MHz	4.7 MHz
True 16-Bit Microprocessor	Yes (80186) 16 bit/16 bit data path	No (8088) 16 bit/8 bit data path
User-Available Expansion Slots*	4	2
Grapics Options		
Color Resolution	640 × 400	320 × 200
Number of Colors	8	4
Monochrome Resolution	640 × 400	640 × 200

*Comparable IBM configuration with monochrome adapter and display, communications adapter, two 320K disk drives and 128K RAM. †Manufacturer's pricing as of 9/1/83.

2000 PERSONAL COMPUTER

Model 2000 Adds a New Dimension to the Most Advanced MS-DOS Software Around



Choose From the Hottest Names In Software

Rapidly becoming an industry standard, the MS-DOS operating system lets us bring you the most popular programs around. Programs like dBase II™. Microsoft-Multiplan™. PFS®:File. MultiMate™. And many more already acclaimed by the entire microcomputer industry. The four software "wheels" above illustrate the outstanding programs available. Each "slice" in the top wheel represents typical applications—electronic filing, word processing, spreadsheets, communications and more. The corresponding slices on the lower wheels portray the programs available for each application family. For example, we have an excellent line of small business accounting software from MAI/Basic Four™—general ledger, accounts payable, accounts receivable and more. These flexible programs have already been proven in thousands of applications worldwide. More programs are under development, including Microsoft's state-of-the-art MS-Windows operating environment. You can view several "windows" at once—files, letters, graphs and more—and exchange information between them.

Room to Grow for the Future

Model 2000 has many standard features that cost extra on other computers. Aside from obvious cost advantages, that leaves more expansion slots open for your future needs. Add additional memory, a hard disk drive, our Digi™-Mouse and more. And you can install most options yourself.

A Milestone in Ergonomic Design

Model 2000 is as spectacular outside as it is inside. The attractive modular system adapts to your environment to assure maximum comfort and productivity. With the optional floor stand, Model 2000 occupies much less desk space than other personal computers.

Total Service and Support is Close at Hand

Our extensive network of Radio Shack Computer Centers gives you complete hardware and software support directly from the manufacturer, not an independent computer dealer. On-site and carry-in service is available nationwide.

The Future of Personal Computing

See the TRS-80 Model 2000 and its popular line of software today at over 1000 Radio Shack Computer Centers nationwide. Ask for a hands-on demonstration and you'll never want to let go.

Radio Shack®

The biggest name in little computers®

A DIVISION OF TANDY CORPORATION

Send me the TRS-80 Model 2000 brochure.

Mail To: Radio Shack, Dept. 84-A-595
300 One Tandy Center, Fort Worth, Texas 76102

NAME _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

TELEPHONE _____

BASIC bits

Answers to readers' questions about Scripsit, SuperScripsit and printers

Models I/III/4III

Thomas L. Quindry, Contributing editor

My patch to Scripsit for the Smith Corona TP-1 printer in the August 1983 issue of *Basic Computing* received some helpful responses which I will pass on to you. To refresh your memory, the TP-1 will not recognize two consecutive linefeeds. One of them will be ignored. I gave a patch for Scripsit to get around this. First of all, for the Model III, the addresses I gave for CD3B00 in Model III Scripsit were off by one byte. They should be 606BH and 6073H.

Scott Stallings of Memphis, TN points out a much simpler, though partial, solution for those who don't need automatic double linefeeds and don't want to program the patch. Scott suggests that if you send an unprintable character before the extra linefeed, the linefeed will be recognized. Since the TP-1 printwheels do not have the greater than (>) or less than (<) symbols, either of these can be placed before the extra linefeed. Since Scripsit uses the greater than symbol for format instructions, it is best to use the less than symbol. To get double linefeeds, add in your text *enter enter*.

Don Eberly of Elliott, IL was inspired to write a somewhat more efficient patch for the Model III tape Scripsit, Version 1.0. I haven't tried his patch, but it should do the job. In fact, a close analysis of my original TP-1 patch would indicate that it shouldn't work. I am assured that it does, however, by the person I collaborated with to write it. Don's patch could be easily adapted to any version as mine could. Just follow the directions I gave in the August issue and change ORG addresses.

I use the Model III version of Superscripsit at the office and the Model I version at home. Because of limited disk storage space for text, I added the Radio Shack doubler to my Model I. The Model I version of Superscripsit can be copied to double density using the TRSDOS 2.7DD format but cannot operate with this DOS. I am very interested in ideas for adapting either Superscripsit or the disk operating system for use with my Model I. For instance, can TRSDOS 2.3 be modified for double density with the Radio Shack doubler and would it support SuperScripsit?

—W.B., Reisterstown, MD

The Radio Shack doubler is not like any other doubler for the Model I. This doubler has given fits to most pre-Radio Shack doubler operating systems due to its oddball design. TRSDOS 2.7DD, which is supplied with it, is neither TRSDOS 2.3 nor TRSDOS 1.3 compatible. If you took those two DOS's for the Model I and III respectively, mixed them up, and took bits and pieces out at random for different functions, you would be likely to come up with TRSDOS 2.7DD. Unfortunately, it is not compatible with many of the more popular Radio Shack programs (or other machine language programs for that matter) including Scripsit or SuperScripsit.

TRSDOS 2.3 has been successfully patched for other Model I doublers but I don't think it can be patched easily for the Radio Shack doubler. Though other DOS's may be compatible with this doubler, the only one I am sure about is the current version of LDOS. LDOS 5.x also provides patches to provide compatibility between it and SuperScripsit. LDOS would be a good choice for you because I believe it should also provide compatibility of data files (or Superscripsit files) between the Model I and Model III and you have access to both.

I am using a Model III with an Epson RX-80. While doing a formfeed in BASIC, the paper advances one line at a time in a jerky motion instead of one smooth movement. Using OUT 251,12 (POKE 14323,12 on the Model I instead of LPRINT CHR$(12);

seems to correct this problem, but is there a fix when using programs like Scripsit or Profile III Plus?

—P.E., Green Bay, WI

The reason LPRINT CHR$(12) causes this jerky motion on your RX-80 is because the Model I and III translate this command to multiple linefeeds (CHR$(10)'s) to take you to the top of form. This is a carry over when "dumb" printers didn't have the logic to do a formfeed. Many of today's printers including the Epson MX series, and I presume the RX series, recognize the ASCII 12 as well as ASCII 140 as formfeed. In BASIC, try LPRINT CHR$(140) to get a smooth formfeed.

As for programs like Scripsit and Profile III Plus, if you can determine where in the code the ASCII 12 is sent to the printer, you can patch the program to send ASCII 140.

Printers are getting smarter all the time. The new Riteman printer, an MX-80-compatible, by Infotrunner, smooths out the multiple linefeeds in spite of what the computer sends it. This printer won't recognize the ASCII 140 as a formfeed, so try it out on your printer before you modify your programs.

Both my printer and the Model III have the special characters, { | } ~ [\ ] ^ _ . How can I use them with my word processor?

—N.T., Los Angeles, CA

To use these special characters, press the i and y keys or the I and Y keys down simultaneously. While holding them there, quickly press the k, l, m, n, K, L, M, N, or O key. You will get something like yik {, yil |, yim }, yin ~, YIK [, YIL \, YIM], YIN ^, or YIO _ respectively. The small i and y letters in combination with the k, l, m, or n give you half of the special characters and using the uppercase lock, the capital I and Y in combination with the K, L, M, N, or O give you the others. Go back and edit out all but the special character. You may have to hit the last key more than once to get the special character. This can also be accomplished in BASIC.

Remember to send your requests for future column topics, questions and tips to me, in care of *Basic Computing*, 3838 South Warner Street, Tacoma, WA 98409. Send a self-addressed, stamped envelope and I'll try to give you a personal, handwritten reply as long as the answer is not too long and involved. Problems of general interest may be included in future BASIC bits.

Listing 1

```
;*************************************************************
;*                                                           *
;*  PATCHS       PATCHES MOD III TAPE SCRIPSIT VER. 1.0      *
;*               FOR USE WITH SMITH CORONA TP-1 PRINTER      *
;*                                                           *
;*               ***     DO NOT RUN SCRIPSIT     ***         *
;*               LOAD SCRIPSIT, THEN LOAD THIS PATCH         *
;*                                                           *
;*               WRITTEN BY DON EBERLY  AUGUST 1983          *
;*                                                           *
;*************************************************************
;
        ORG     4EEFH           ;REPLACES CALLS TO ROM'S
        CALL    PATCH           ;$PRCHAR ROUTINE
        ORG     4EF7H
        CALL    PATCH
        ORG     490AH
PATCH   PUSH    HL              ;SAVE REGISTER
        LD      HL,FLAG         ;ADDR OF FLAG
        CP      0DH             ;CARRIAGE RETURN
        JR      Z,EXTRA
        RES     0,(HL)          ;CLEAR FLAG
        JR      PRINT
EXTRA   BIT     0,(HL)          ;TEST FLAG
        JR      NZ,PRINT        ;DONE BEFORE
        INC     (HL)            ;SET FLAG
        CALL    003BH           ;$PRCHAR
PRINT   CALL    003BH
        POP     HL              ;RESTORE REGISTER
        RET
FLAG    DEFB    0
        END     4303H           ;NORMAL SCRIPSIT START
```

In the chips

High-speed graphics

Models I/III/4III

Spencer Hall, Associate editor

The very best way to become acquainted with machine language, no doubt about it, is to copy well-annotated source programs into a computer. The copying process forces one to dwell for some time on each statement in a way that just doesn't come naturally when you're only reading. Good annotations, in the remarks column of the source code, clarify exactly what the source code is doing and why. You see ROM calls used. You see clever manipulations of the screen. You see multiple-use subroutine CALL's which can be copied and/or modified for your own use. Best of all, you soon lost that feeling of strangeness which surrounds source code the first few weeks you use it.

Unfortunately, this approach requires a general familiarity with the hexadecimal system. The Z-80 registers, and the use of an editor-assembler program. Fortunately, graduates of this series are well-armed with this information. Fortunately, also, there are several books available which contain the same material. We've already listed them in past columns, so for now, we'll mention just one more new reference. For our purposes, this is probably the best of all. William Barden, Jr., in his Radio Shack text, *Programming Techniques for Level II BASIC*, offers a brilliantly executed, slam-bam-fast introduction to machine language concepts. It's contained in Chapter 11.

One question in a thought-provoking letter from a reader didn't depend for its answer on past columns. I quote the entire paragraph because the rest of you will probably join in a chorus of agreement.

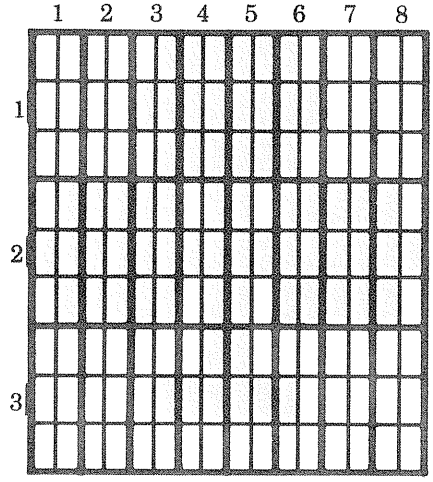
"As I look at assembly language source codes in *Basic Computing*, and in other magazines, I continually ask myself the same question: "Why?" And *how* does the author know what to do? Wish I could be as smart."

The answer, of course, lies in the fact that machine language is only language, but a program is a product of the imagination. How does any author know what to write next? He has a purpose. What he writes next is the code which carries out that purpose.

This month, we want to harness the incredible speed of machine language in order to provide a way for a BASIC program to display a graphic image (of any size) instantaneously on the screen. We'll do it by writing a machine language program using EDTASM, compile it, embed it in our BASIC program in the form of DATA statements to be POKEd in place, and call it with USR.

At the end of our machine language routine, we'll place a "buffer," or string of addresses, containing the bytes which make up the image. To determine these bytes, we'll draw our design on a sheet of cross-ruled paper like the video screen map which is contained in both the Level II and Model III manuals. We can find the codes for these bytes by using the graphics code list which is also printed in both TRS-80 manuals.

Figure 1 — The club figure generated by Listing 2.



For simplicity, we'll use all the bytes in a rectangle surrounding our image (Figure 1). We'll simply place them one after the other in our buffer. How can our machine language routine tell where to write them on the screen or how to arrange them? Let's pass the screen address integer to the subroutine by placing it in parentheses in the USR call, so: X=USR(SA) where SA is the PRINT@ address where we want the image.

Also, let's decide (quite arbitrarily) that the first two bytes of our buffer will contain, respectively, the number of lines and the length of these lines in our image rectangle. For our test, we've drawn a playing card club. Next, using the graphics code list just mentioned, we decipher the byte values of each graphics character in this image. For now, we must write them down on scratch paper because we don't know where the end of our machine code will be in memory. The first byte in this buffer will be 3 and the next one 8, because we have three rows of eight characters each in our image.

Now we can write our code using EDTASM. See Table 1 for comments on the code for Listing 1.

To carry out our plan, we must make a poke version of this compiled

program. This is shown in Listing 2. We need to protect the subroutine or it will be clobbered by the BASIC operating system. The POKEs in line 5 do this. The numbered poked in LSB/MSB format is 28670. Always make this number two less than the starting address of what is protected. That's how BASIC does it. Next, we must tell BASIC where to go when it gets the USR call. The pokes in line 7 do this. Using Disk BASIC, these become either DEFUSR(28672) or DEFUSR (&H7000).

Now we convert the compiled hex bytes from Listing 1 into decimal values and place them in DATA statements. Finally, the number of lines and the length of each are placed in DATA statements followed by the bytes of our image. The arrangement in Listing 2 was chosen for clarity. Actually, all bytes follow each other in memory, as indicated by the loop in line 90 which delivers them. We know they start at 28672. The loop end at 28738 was obtained by simply counting all DATA bytes and adding the total to 28672. Don't forget to subtract one to allow for the fact that we're using both the first and last address!

Note that we can put the code for any image we design right there in those DATA statements. Doing so makes this subroutine usable in any program, for any image, from one byte to a full screen.

You can erase this entire program once you've run it and your image, plus the subroutine to display it, will still be there waiting for a USR call. Have fun by entering the code in Listings 3 and 4. Be sure, of course, that you first *run* Listing 2.

Figure 2 is a screen print, actually a "self-portrait," made with the image maker. First, I listed part of our original BASIC program. After hitting the BREAK key and getting the READY prompt, I typed the command as you see it at the bottom of Figure 2 beginning "POKE

Listing 1

```
                00100 ;*****            IMAGE MAKER             *****
                00110 ;*****          In the Chips #10          *****
7000            00120          ORG     7000H           ;Protect 28672 decimal
7000 E5         00130 START    PUSH    HL              ;Save registers we'll use
7001 D5         00140          PUSH    DE              ;...BASIC interpretor is
7002 C5         00150          PUSH    BC              ;probably using them.
7003 CD7F0A     00160          CALL    0A7FH           ;Get PRINT location in HL
7006 EB         00170          EX      DE,HL           ;Move it into DE
7007 212970     00180          LD      HL,IMAGE        ;Get start of image file
                00190 ; The first byte in file is number of lines in image
700A 7E         00200          LD      A,(HL)          ;Number of lines into A
                00210 ; The second byte is the number of bytes per line
700B 23         00220          INC     HL              ;Next byte in file
700C 4E         00230          LD      C,(HL)          ;Put bytes per line in BC
700D 0600       00240          LD      B,00H           ;Make sure BC is only C
700F 23         00250          INC     HL              ;HL at first image byte
                00260 ; We're now ready to do an LDIR, but this will reduce
                00270 ; BC to zero. We'll need it again. Also starting DE must
                00280 ; be increased by 64 to get next line start. We'll save
                00290 ; both of these on the stack.......
7010 C5         00300 WRITE    PUSH    BC
7011 D5         00310          PUSH    DE
7012 EDB0       00320          LDIR                    ;Write an image line
7014 3D         00330          DEC     A               ;One less line to go
7015 CA2370     00340          JP      Z,EXIT          ;Done? If so, return
7018 D1         00350          POP     DE              ;First loc. of last line
7019 E5         00360          PUSH    HL              ;Must "borrow" HL
701A 214000     00370          LD      HL,40H          ;64 to add to HL
701D 19         00380          ADD     HL,DE           ;result is in HL
701E EB         00390          EX      DE,HL           ;New start is now in DE
701F E1         00400          POP     HL              ;Text pointer back in HL
7020 C1         00410          POP     BC              ;Line length replaced
7021 18ED       00420          JR      WRITE           ;Write another line
7023 E1         00430 EXIT     POP     HL              ;Two dummy POP's to shift
7024 E1         00440          POP     HL              ;stack pointer to BASIC'S
                00450                                  ;register contents.
7025 C1         00460          POP     BC              ;Replace them before we
7026 D1         00470          POP     DE              ;return control to BASIC
7027 E1         00480          POP     HL              ;so it can continue.
7028 C9         00490          RET                     ;Return to calling prog.
7029 03         00500 IMAGE    DEFB    3               ;Image file starts here
7000            00510          END     START
```

28713 . . . This created a buffer containing the whole screen. Now, CLEARing the screen and typing X=USR(15360) caused the whole screen to reappear instantly. Try it. It's a great way to make a HELP screen for use in a BASIC program.

To get a copy of the "self-portrait" for this article, I used the code in Listing 5.

Table 1
Line #/Comment

120: 7000H (28672 decimal) is a nice round starting address popular for the start of subroutines because it's low enough to be used in 16K systems but still allows room for at least a short BASIC program.

130-150: For subroutines used in a BASIC program, always save the major registers and restore them at the end of the subroutine. The BASIC interpreter is actually a machine language program and probably needs these register contents.

160: The routine at 0A7FH fetches the number, SA (or anything else you choose) from "X=USR(SA)" and places it in the HL register.

170-320: These lines set up and execute an LDIR instruction which moves a first image line into the screen memory.

330-350: We're using A to hold the number of lines and after each line is printed, we reduce A and test to see if we've printed the last line yet.

360-420: If we need to print another line, we get its start by adding 40H (64 decimal), the length of a Model I or III screen line, to the previous start. The ADD statement in line 380 is an easy way to perform this addition so we've got to store the contents of HL, and "borrow" it in much the same way we stored all registers for BASIC at the start.

430-480: After the image has been placed on the screen, we've got to get those three values for BC, DE and HL which we PUSHed onto the stack at the beginning. PUSHing register values onto the stack is like dropping them into a hole. We've dropped two more in on top of the original three, so they must be POPped out and thrown away to get at the ones we want. The "dummy" POPs in lines 430 and 440 do this. We could have used any register. The unneeded values appear briefly in HL and are immediately written over. Note that the POP order must be exactly the reverse of the PUSH order, so that correct values will end up in each register. Keeping track of PUSHed and POPped numbers requires concentration.

490: Return to the BASIC program.

500: When we type this source code in, using EDTASM, we don't see the code at the left of this listing so we have no idea where the buffer will begin. In order to have a location for "IMAGE," which is referred to in line 180 above, we must define the next byte after "RET." We name it IMAGE, to match the line 180 reference, and since we know from our graphics layout that there are three lines, we DEFB (DEFine Byte) as 3.

Figure 2

```
A SCREEN PRINT 'CAPTURED' BY THE CODE
WHICH APPEARS IN THE LAST TWO LINES OF
THIS 'SELF-PORTRAIT'

)LIST
5 POKE16561,254:POKE 16562,111
6 POKE16526,0:POKE16527,112
9 '** BYTES FOR THE IMAGE DRIVER PROGRAM **
10 DATA 229,213,197,205,127, 10,235, 33, 41,112,126, 35, 78
20 DATA   6,  0, 35,197,213,237,176, 61,202, 35,112,209,229
30 DATA  33, 64,  0, 25,235,225,193, 24,237,225,225,193,209
40 DATA 225,201
48 '** BYTES FOR THE "CLUB" IMAGE FILE **
49 '        NUMBER OF LINES AND CHARACTERS PER LINE
50 DATA 3,8
BREAK
READY
) POKE 28713,16:POKE28714,64:RA=28715:FOR SA=15360 TO 16383:POKE
RA,PEEK(SA):RA=RA+1:NEXT
```

Listing 2

```
4 '     PROTECT MEMORY AT 2
8670
5 POKE 16561,254:POKE 1656
2,111
6 '     STORE ENTRY POINT F
OR M/L SUBROUTINE (28672)
7 POKE 16526,0:POKE 16527,
112
9 '** BYTES FOR THE IMAGE
DRIVER PROGRAM **
10 DATA 229,213,197,205,12
7, 10,235, 33, 41,112,126,
 35, 78
20 DATA   6,  0, 35,197,21
3,237,176, 61,202, 35,112,
209,229
30 DATA  33, 64,  0, 25,23
5,225,193, 24,237,225,225,
193,209
40 DATA 225,201
48 '** BYTES FOR THE "CLUB
" IMAGE FILE **
49 '     NUMBER OF LINES AN
D CHARACTERS PER LINE
50 DATA 3,8
59 '     FIRST LINE
60 DATA 128,128,174,191,19
1,157,128,128
69 '     SECOND LINE
70 DATA 174,191,191,174,15
7,191,191,157
79 '     THIRD LINE
80 DATA 128,128,136,142,14
1,132,128,128
85 DEFUSR1=28672
89 '     INSTALL SUBROUTINE
 AND IMAGE IN MEMORY
90 FOR RA=28672 TO 28738:R
EAD B:POKE RA,B:NEXT
95 ' Line 100 is for Mod I
.  For Mod III replace Lin
e 100 with          100 DEF
USR1=28672:CLS:SA=15704:X=
USR1(SA)
100 CLS:SA=15704:X=USR(SA)
109 '    USE THE SUBROUTINE
!
110 GOTO 110
```

Listing 3

```
2 GOTO 10
9 Z$=INKEY$:IF Z$="" THEN
9 ELSE RETURN
10 POKE 16526,0:POKE 16527
,112
20 CLS
25 SA=RND(832)+15360
26 ' Line 30 is for Mod I.
  For Mod III replace Line
 30 with          30 DEFUS
R1=28672:X=USR1(SA):N=N+1
30 X=USR(SA):N=N+1
40 Z$=INKEY$:IF Z$="" THEN
 25
50 GOSUB 9:IF Z$="S" THEN
PRINT@960,"PRINTED"N"CLUBS
";:GOSUB 9:GOTO 20
60 GOTO 30
```

Listing 4

```
5 POKE 16526,0:POKE 16527,
112
10 CLS
20 FOR K=2 TO 770 STEP 256
30 FOR J=K TO K+56 STEP 9
35 ' Line 40 is for Mod I.
  For Mod III replace Line
 40 with          40 DEFU
SR1=28672:X=USR1(15360+J)
40 X=USR(15360+J)
50 NEXT:NEXT
60 GOTO 60
```

Listing 5

```
5 ' For Mod III add 5 CLEA
R 100
10 POKE16526,0:POKE16527,1
12
15 ' Line 20 is for Mod I.
  For Mod III replace Line
 20 with          20 DEFUS
R1=28672:X=USR1(15360)
20 X=USR(15360)
40 LPRINTCHR$(27)"B"
50 FOR SA=15360 TO 16383
60 B=PEEK(SA)
70 IF B<32 THEN B=B+64
80 LPRINTCHR$(B);:NEXT
90 CLS
```

Bridge-80

Become a Life Master

Models I/III/4III

Bridge-80 is a program designed both to teach Bridge fundamentals and to allow the experienced player hours of solo practice and fun. Like the popular mechanical game of "Autobridge," you, the player, are always the dealer and you always bid first. The TRS-80 plays your partner and both opponents. You have to take it on faith that the machine does not peek! Bridge-80 plays most popular conventions. Although your opponents do not bid against you, they may double your contract if their strength seems to justify it.

You can make the game as challenging or as easy as you like by how you bid, just as in regular Bridge. For highest scores, bid aggressively. Try to get to slam as often as possible, but watch out . . . the program can frequently defeat overbid contracts.

Bridge-80 is very "user friendly," with several options to enhance its usefulness. Interesting or misplayed hands may be replayed as often as desired. Games can be played either open, with all hands shown, or closed, with only your hand visible during bidding and only yours and dummy's visible during play. The Goren "Point-Count" system is used to evaluate hands and the count for your hand is displayed during bidding. A running score is kept and displayed at all times. You and your "dummy" partner are scored as "We." Partial scores, games, penalties and bonuses are all accumulated (except if you elect to replay a hand already scored, it doesn't count again).

Few instructions are really necessary to use Bridge-80 if you already know how to play Bridge. If not, you'll need a good beginner's book to get started. At least you can learn with no chance of making a fool of yourself or making an enemy of your partner.

Getting Started

Your first response is a single letter to indicate whether you wish the hands to be "displayed" (played open), or to replay the last hand. Any key other than "D" or "R" is assumed to mean a normal new hand. The deck is then shuffled at random and dealt out graphically. Your hand is automatically sorted and the point-count is displayed. Enter a bid by typing a number, followed by the first letter of the suit (or "N" for notrump). There is no need to use the enter key at any time and invalid bids are ignored. Use "P" to pass and "R" to redouble. At any time you can type "@" to cancel the current hand and either replay it or get a new one.

Compared to regular Bridge, you'll notice that with Bridge-80 you generally get much better hands. This is because the computer always picks out the best of the four hands as yours. This seems only fair. After all, it's *your* computer!

Your partner is programmed to respond normally, according to his or her point count and assessment of yours. Partner responds to 2 clubs as forcing, and must show best suit (or 2 diamonds with less than five points). Your partner may respond to your opening 1 notrump with the 2 club Stayman convention. You can ask for aces and kings by using the Blackwood convention (4NT and 5NT) but not the Gerber convention (4 clubs), which is no longer used in the best Bridge-80 circles.

The "Display" mode allows you to see your partner's hand during bidding and all four hands during the play. Display mode is a great way to learn how your partner responds to your bidding since it also shows your partner's point count and his/her assumption about the points in your hand. During play, you can get an appreciation of the art of the finesse and the cross-ruff.

The "Replay" mode lets you recover from a totally misbid mess and start over, or lets you play an interesting hand again with different bidding. You can also use the Replay option to stage a kind of duplicate tournament with a friend in which you both play each hand and try to outscore each other. Since each new deal is totally random, you can replay only the last hand (although you can replay it as often as you like). A reader interested in storing hands for future replay could add code to save array "DL," which has only 52 elements.

The Auction

Bidding is straight-forward. Simply press the correct number key followed by the first letter of the suit. Bridge-80 will not let you bid below the current auction level. Your bids are displayed just above your hand and those of your partner just below his/her hand. All bids are visible throughout the auction. Bidding

and never feel like a dummy

George L. Farnsworth

continues until you pass (press "P") or respond to an opponent's "double" with a "redouble" (press "R," of course), or until your partner passes without an intervening double.

Figure 1 displays the screen at the start of bidding. Your hand appears sorted by suit and number with the point-count on the right. The current score is at upper right and the number of hands played is at top center. Figure 2 shows the screen after four rounds of bidding. All bids are displayed in left-to-right and top-down order.

You may decide to open the bidding by passing if you have a weak hand and your partner may then decide to open. If partner also passes, the hand ends.

Either you, as dealer, or your partner, can win the auction. If your partner wins, Bridge-80 automatically exchanges your hands so that you play the hand. (The computer is always "dummy".)

At any time during the auction or the play of the hand you can cancel and get a new deal or start over by pushing "@."

When the auction is finally over, you get a chance to look over the bidding before play begins. By pushing any key, you can cut this review period short. The final contract is displayed in the lower left-hand corner during play along with any conditions such as DBL (doubled), RDBL (redoubled), or VL (vulnerable), which affect scoring.

The Play

During play, each card appears in the center of the screen near the hand that played it. When it's your turn to play, a question-mark appears, either in front of your hand or dummy's.

Considerable effort has gone into making Bridge-80 easy to use, especially during play. If you or dummy have only one possible card to play, for example, it is played automatically. Otherwise, indicate the card to be played by its value first (2 through 9, A, K, Q, J, or T). If the suit is then unambiguous; either because you must follow suit or because there is only one such card in the hand, you will not be asked for it. Otherwise, enter the first letter of the suit (C, D, H, or S). Bridge-80 won't let you make an illegal play or play out of turn. Your completed tricks are displayed horizontally above your hand and those of your opponents vertically on East's side (left of screen).

Figure 3 illustrates the situation where it is North's turn to play on the third trick of a six-club doubled and vulnerable contract. South's lead was the queen of clubs, East played the ten and it is now your turn to play from the board. Note that the question-mark appears at the location where the played card will be placed. Figure 4 shows the same hand at the start of the fifth trick. In Figure 5, you see the display at the end of the contract. The partnership took all tricks and made their contract with one overtrick.

At the conclusion of each trick, you are given a chance to look at it before it gets gathered in. Just press any key to signal that you're ready. Toward the end of play, it frequently happens that either the declarer or the board is left with all trump while the opponents have none. At that point, Bridge-80 ends the hand, automatically giving you all remaining tricks.

Scoring

The score sheet always appears in the upper right corner of the screen. "We," of course, represents you and your partner. "They" are your Bridge-80 opponents. Since the size of the score sheet is limited, it only displays the total of points toward game (part-score) below the line and all other points (summed) above the line. At the conclusion of each hand, the detailed score for that hand is displayed at lower left, just below the contract.

For hands that are replayed, the results appear, but are not added to the running score.

A word of explanation for those of you whose Bridge is rusty — one hundred points bid and made constitutes a "game," which may take only one hand or several. Points leading up to game are placed "below the line" until game is reached. All other points, such as those from "slams" and "rubbers," are scored above the line and don't count toward game. A slam is a bid of six, called a small slam, or seven, the famous grand slam. Two games constitute a rubber, for which you get a bonus of 700 points (in regular Bridge, you may get only 500 points if your opponents make a game during the rubber).

Club and diamond contracts are worth 20 points for each trick bid and made. Hearts and spades are worth 30 each and in notrump the

first trick is worth 40 and the rest 30 each. Two hearts are therefore worth 60 points and it takes five diamonds to make a game, but only three notrump. A small slam is worth 500 or 750, and a grand slam 1000 or 1500 points, depending on whether or not you are "vulnerable" (have one game toward rubber).

If you fail to make your contract, your opponents earn points above the line on their side (penalties), 50 or 100 points per trick depending on vulnerability.

"Doubling" or "redoubling" does exactly that to the scoring, except that the penalties for undertricks are even worse. You should redouble only when you are very sure of making your contract.

Program Structure

An attempt has been made to keep Bridge-80 modular and structured for easy understanding and user modification. Wherever possible, subroutines and logical expressions have been used to avoid building up confusing nests of IF . . . THEN . . . ELSE structures.

For reasons of speed and economy of memory, Bridge-80 makes frequent use of subroutines and long, dense statements. Listing 1 is a standard LLIST of the program.

Figure 6 outlines the program structure. Use this structure to modify the program to suit your style of play. If, after using the program for awhile, you feel that your partner bids too cautiously, take a look at lines 3000 to 3300. An examination of line 3240 reveals that North passes if MP+PN is less than 20. From Figure 7, you see that these variables are an estimate of your points as shown by bidding and North's points, respectively. Change the "20" in the cited line to "18" if you feel that your partner is passing too many of your one-bids. Similarly, change the "26" in the next line to "24" if you feel that your partner should continue bidding toward game with weaker hands.

In keying the program in from the listing, you may omit comments if you prefer, but be very careful not to change the screen displays. Bridge-80 examines the screen for certain necessary information and unpredictable results may occur if the screen doesn't appear exactly as it expects.

```
BRIDGE-80    1.1                    5                  WE      THEY
GL FARNSWORTH  (C)1983                              2,100      400
```

Figure 1

```
                                                        80         0

                BID : ?
                        CLUBS   : AQ863
                        DIAMONDS: K
                        HEARTS  : AKT              25 POINTS
                        SPADES  : AQJ2
```

```
BRIDGE-80    1.1                    2                  WE      THEY
GL FARNSWORTH (C)1983                               3,960      400
```

Figure 2

```
                                                        0          0

                        3C    4H   5C   PASS
          PASS                                    PASS
          PASS                                    PASS
          PASS                                    PASS
          PASS                                    DOUBLE

                BID :  2C    3S   4S    6C  ?
                         CLUBS   : KQ
VULNERABLE               DIAMONDS: AKT4
                         HEARTS  : K43             24 POINTS
                         SPADES  : AKJ5
```

Listing 1 — Bridge-80

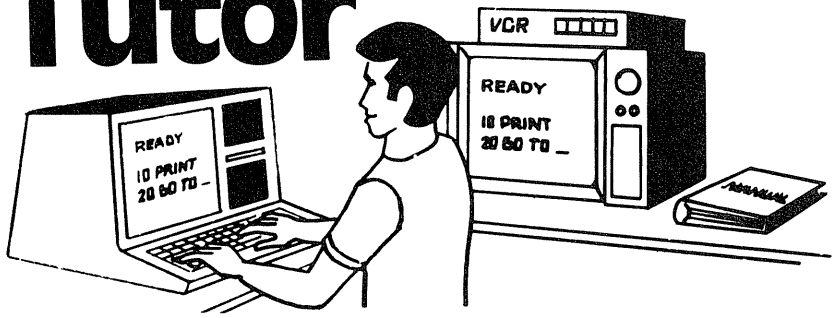
```
10 GOTO 1000 :' ----- BRIDGE-80   (C)198
3 BY GEORGE FARNSWORTH

     2343 ROSEDOWN DR.

     RESTON, VA  22091

20 PRINT@128,CHR$(31);:GOSUB21000:PRINT@
520,"MODE (NORMAL/DISPLAY/REPLAY) ";:AT=
552:GOSUB70 :PRINT@520,BL$;:IFK$="D"THEN
TC=1ELSEIFK$<>R$THENTC=0ELSEAT=616:PRINT
@584,"REPLAY MODE (NORMAL/DISPLAY)":GOSU
B70 :PRINT@584,BL$:PRINT@528,"RE";:TC=(K
$="D"):GOTO2000
30 K$=INKEY$:IF K$<>"" THEN K2=99:RETURN
 ELSE RETURN
40 K$=INKEY$:M$="ANY KEY"
50 FOR K1=0TO10:PRINT@948,M$;BL$;:FORK2=
0TO99:GOSUB 30   :NEXT:IFK$="" THEN PRI
NT@948,"    ";:FORK2=0TO99:GOSUB 30   :N
EXT:IFK$="" THEN NEXT:RETURN
60 PRINT@948,BL$;:IFK$="@"THENN$="CANCEL
LED":GOTO5010 ELSERETURN
70 FORIK=0TO2STEP0:PRINT@AT,"?";:K$=INKE
Y$:IFK$="" THEN PRINT@AT,"*";:NEXT ELSE
IF K$="@" THEN N$="CANCELLED":GOTO 5010
ELSE RETURN
80 PRINT@Y,"#";:Y=Y+1:K$=MID$("CDHS",SK+
1,1):RETURN
90 PRINT@AT,"?";:M$="CARD PLEASE":GOSUB
50   :IF K$=""THEN90   ELSEC$=K$:PRINT@A
T,K$;:RETURN
100 PRINT@AT+1,"?";:M$="SUIT PLEASE":GOS
UB 50   :RETURN
110 PX=P:AX=AT:AT=384:P=E:GOSUB 22000:AT
=429:P=W:GOSUB 22000:AT=AX:P=PX:RETURN:'
==== DISPLAY E-W HANDS
1000 CLEAR200:DEFINT A-Z:CLS:DIM DK(51),
DL(51),H$(3,4),NP(3),TP(3),M(3),B(3),D(4
):HN=1:V1=1.1
1010 FOR P=0TO3:TP(P)=0:NP(P)=0:D(P)=5:N
EXT:D(4)=5:BL$=CHR$(30)
1020 '*** DEAL
1030 PRINT@0,"BRIDGE-80   ";V1,HN:PRINT"
GL FARNSWORTH (C)1983";:GOSUB 20:HX=0:RA
NDOM:PRINT@512,"SHUFFLING ";:FOR CD=0TO5
1:IF CD<26 THEN PRINT CHR$(143)" "; ELSE
 PRINTSTRING$(2,24);BL$;
1040 P=RND(4)-1:IFNP(P)=13THEN1040 ELSEN
P(P)=NP(P)+1:DK(CD)=P*13+NP(P)-1:NEXT
1050 K=15424:GOSUB 22800:FOR CD=0TO51
1060 P=INT(4*RND(0)):IF TP(P)=13 THEN 10
60 ELSE TP(P)=TP(P)+1:DL(DK(CD))=P:NEXT
CD:PRINT@512,BL$;
2000 PRINT@530,"DEALING ";:F$="23456789T
JQKA":N(0)=335:N(1)=677:N(2)=975:N(3)=64
0:K=0:R$="R"
2010 FOR P=0TO3:NP(P)=0:FOR CS=0TO3:H$(P
,CS)="":NEXTCS,P
2020 FOR CS=0TO3:FOR CN=12 TO 0 STEP -1:
CD=CS*13+CN: P=DL(CD):  IF CN>8 THEN NP(
P)=NP(P)+CN-8
2030 PRINT@N(K),CHR$(143);:N(K)=N(K)+2:K
=K+1:IFK=4 THEN K=0
2040  H$(P,CS)=H$(P,CS)+MID$(F$,CN+1,1):
NEXT CN,CS
2050 PS=0:FOR P=0TO3:IF NP(P)>PSTHEN PS=
NP(P):S=P:NEXTELSENEXT
2060 N=S+1:IFN>3 THEN N=0
2070 GOSUB 23800:SN$="CLUBS   DIAMONDSHE
ARTS   SPADES   ":PRINT@335,CHR$(31);
2080 AT=793:P=S:GOSUB 22000
2090 FOR P=0TO3:IF P<>N AND P<>S  THEN W
=P:NEXT ELSE NEXT
2100 FOR P=0TO3:IF P=N OR P=S OR P=W  TH
EN NEXT ELSE E=P:NEXT
3000 R=0:PRINT@720,"BID : ";:AT=726:CV=0
:MP=6:SK=-1:AN=342:S$="C D H S NT":LS=5:
EA=330:WA=368:PRINT@949,TP(S);"POINTS";:
IF VL PRINT@832,"VULNERABLE";
3010 IF TC THEN AT=89:P=N:GOSUB 22000:AT
=726
3100 R=R+1:LV=CV:LK=SK:GOSUB 10000:IF PK
*CV THEN 4000 ELSE IF CV THEN IF D(SK)=5
 THEN D(SK)=S
```

Computer Tutor™



VIDEO INSTRUCTION TAPES

STEP BY STEP INSTRUCTIONS

VHS or BETA FORMAT

PICTURES ARE WORTH THOUSANDS OF WORDS AND SAVE HOURS OF FRUSTRATION

Use your VCR side by side with your computer to learn disk operating systems, how to program, and how to use programs. Your VCR along with your computer serve as your personal tutor. Pause your VCR to review and learn at your own pace.

SPECIAL OFFER

Lazy Writer word processor and video instruction tape complete with documentation and registration card for model 1, 3 or the model 4, (the model 4 has 80 character display).

ONLY $119.00

VERBATIM DATALIFE

Single side double density 10 in a hard plastic storage box.

ONLY $25.00

C.O.D. Orders Add $3.00
$3.00 Shipping Per Order
Illinois Residents Add 6% Sales Tax

Specify either VHS or BETA Tape Format

TAPES FOR THE TRS 80

CAT #	TOPIC	APPROX.	PRICE
DOS-1	TRSDOS 1.3 MOD 3	1 HR 30 MIN	$39.95
DOS-6	TRSDOS 6.0 MOD 4	1 HR 45 MIN	$39.95
DOS-3	NEWDOS 80 2.0	1 HR 45 MIN	$39.95
DOS-2	DOSPLUS 3.5	1 HR 45 MIN	$39.95
DOS-5	LDOS	1 HR 45 MIN	$39.95
DOS-7	CPM 2.2 MOD 4 MONTEZUMA	1 HR 45 MIN	$39.95
DOS-4	MULTIDOS	1 HR 45 MIN	$39.95
EW-1	VISICALC (TRS80)	1 HR 30 MIN	$39.95
WP-1	LAZY WRITER	1 HR 45 MIN	$39.95
WP-2	SUPERSCRIPSIT	1 HR 45 MIN	$39.95
WP-3	SCRIPSIT	1 HR 30 MIN	$39.95
WP-4	NEWSCRIPT	1 HR 45 MIN	$39.95
DB-1	PROFILE III PLUS	1 HR 45 MIN	$39.95
DB-2	PFS:FILE	1 HR 30 MIN	$39.95
DIO-3	TRSDOS 1.3 DISK I/O	1 HR 45 MIN	$39.95
DIO-4	TRSDOS 6.0 DISK I/O	1 HR 45 MIN	$39.95
BP-1	LEARNING MOD 3, 4 BASIC	3 HR	$49.95
BP-2	LEARNING MOD 1 BASIC	3 HR	$49.95
TC-1	MTERM	45 MIN	$29.95
UT-1	DOTWRITER	1 HR 15 MIN	$29.95

TAPE FOR THE TRS80 COLOR COMPUTER

BP-5	EXTENDED COLOR BASIC	3 HR	$49.95

TAPES FOR THE COMMODORE 64 AND VIC 20

BP-3	LEARNING C-64 BASIC	2 HR	$49.95
BP-4	LEARNING VIC-20 BASIC	2 HR	$49.95
DIO-1	COMMODORE 64 DISK I/O	1 HR 45 MIN	$49.95
DIO-2	VIC 20 DISK I/O	1 HR 45 MIN	$49.95

TAPES FOR THE IBM PC

EW-2	VISICALC (IBM PC)	1 HR 30 MIN	$39.95
TC-2	MTERM (IBM PC)	45 MIN	$29.95

LYNN COMPUTER SERVICE

6831 West 157th Street Tinley Park, Illinois 60477 (312) 429-1915

```
3110 IF R=1 THEN GOSUB 23000:IF CV=1ANDS
K=4 THEN MP=17 ELSE IF CV=1 THEN MP=14 E
LSE IF CV=2ANDSK=0 THEN MP=24 ELSE IF CV
>LV+1 OR (CV>LV AND SK>LK) THEN MP=MP+4
3120 IF R>1 THEN IF CV>LV+1 OR (CV>LV AN
D SK>LK) THEN MP=MP+3-3*(MP=6)
3130 GOSUB 20000:PRINT@EA+64*R,"PASS";
3140 IF SK=4  THEN IF CV=4 THEN GOSUB 22
200:CV=5:GOTO 3290 ELSE IF CV=5 THEN GOS
UB 22220:CV=6:GOTO 3290
3200 IF SK=4 THEN PN=NP(N) ELSE PN=TP(N)
:FOR CS=0TO3:L=LEN(H$(N,CS)):IF L<3 AND
CS=SK THEN PN=PN-2:NEXT ELSE NEXT
3210 IF TC THEN PRINT@300,MP;PN"POINTS";
3220 GOSUB 22100:IF R=1  THEN IF CV=2 AN
D SK=0 THEN 20020ELSE IF CV=1ANDSK=4 THE
N 20040ELSE IF R=2ANDST=1 THEN 20070
3230 IF MP+PN<33 THEN IF (CV>3 AND SK>1
) OR (CV=3 AND SK=4 ) OR CV>4 THEN 3280
3240 IF MP+PN<20 THEN 3280
3250 IF CV>2 AND MP+PN<26 THEN 3280 ELSE
 IF CV>5  THEN 3280
3260 IF CV=4 AND BS=4 THEN BS=B(0) ELSE
IF CV=1 AND BS=SK AND PN>12 THEN CV=CV+2
:GOTO3290 ELSE IF CV=0 THEN CV=1
3270 IF BS>SK THEN SK=BS:GOTO 3290 ELSE
SK=BS:CV=CV+1:GOTO 3290
3280 PRINT@AN,"PASS";:GOSUB 20010:AN=AN+
5:IF DB=2 THEN 3100 ELSE  4000
3290 PRINT@AN,STR$(CV);MID$(S$,SK*2+1,2)
;:AN=AN+5:GOSUB 20010
3300 IF D(SK)<>S THEN D(SK)=N:GOTO 3100
ELSE 3100
4000 TP(N)=PN:GOSUB 23500:AT=89:P=N:GOSU
B 22000:DA=718:WL$=CHR$(194)
4010 T$=CHR$(143):B$=CHR$(200):GOSUB 40
  :FOR I=320TO704STEP64:PRINT@I,BL$;:NEX
T:IF TC GOSUB 110
4020 IF SK<0 THEN N$="ALL PASS":GOTO 501
0 ELSE PRINT@832,"CONTRACT:"STR$(CV);MID
$(S$,SK*2+1,2);:TK=SK:IF VL THEN PRINT"
VL";
4030 IF DB=2 THEN PRINT" DBL"; ELSE IF D
B=4 THEN PRINT" RDBL";
4040 LP=E:DT=0:OT=0:GM=0:IF CV>4 THEN GM
=1 ELSE IF CV=4ANDSK>1 THEN GM=1 ELSE IF
 CV=3ANDSK=4 THEN GM=1
4100 LD=-1:PRINT@412,B$;:PRINT@412+128,B
$;:PRINT@473,B$;:TM=0
4110 P=LP:IF LP=S OR LP=N THEN GOSUB 130
00:LK=SK ELSE GOSUB 11000
4120 LD=0:LK=SK:GOSUB 23200:CM=C:WN=P:GO
SUB 23100
4130 IF P=S OR P=N THEN GOSUB 13000ELSE
GOSUB 12000
4140 GOSUB 23200:GOSUB 23100:IF P=S OR P
=N THEN GOSUB 13000ELSE GOSUB 12000
4150 GOSUB 23200:GOSUB 23100:IF P=S OR P
=N THEN GOSUB 13000ELSE GOSUB 12000
4160 GOSUB 23200:IF WN=S OR WN=N THEN DT
=DT+1:PRINT@DT*2+DA,T$; ELSE OT=OT+1:PRI
NT@OT*64+197,T$;
4170 DA=DA-(DT=6):GOSUB 40   :IF OT+DT<1
3 THEN LP=WN:GOTO 4100
5000 CV=CV+6:IF DT=>CV THEN GOSUB 24000E
LSE N$="DOWN "+STR$(CV-DT)+" !!":GOSUB 2
5000
5010 GOSUB 21000:B$=CHR$(192+LEN(N$)):FO
R I=1 TO 3:PRINT@899,N$;:GOSUB 20000:PRI
NT@899,B$;:GOSUB 20000:NEXT:GOSUB 40   :
IF BW>99 THEN UW=UW+BW:BW=0:GOTO 1010 EL
SE 1010
10000 PK=1:GOSUB 70   :IF K$="P" THEN PR
INT@AT,"PASS";:AT=AT+5:RETURN ELSE  IF K
$="R" AND DB=2THEN PRINT@AT,"REDOUBLE";:
DB=4:RETURN ELSE BN=VAL(K$):IF BN<CV THE
N 10030
10010 PRINT@AT+1,K$;:GOSUB 70   :FOR BK=
0TO4:IF K$=MID$(S$,BK*2+1,1) THEN 10020E
LSE NEXT:GOTO10030
10020 PK=0:IF BN>CV OR (BK>SK) THEN PRIN
T@AT,STR$(BN);MID$(S$,BK*2+1,2);:AT=AT+5
:CV=BN:SK=BK:RETURN ELSE 10030
10030 PRINT@AT,"  ";:GOTO10000
11000 IF P=E THEN PP=W ELSE PP=E
11010 MM=-99:FOR K=0TO3:M=0:L=LEN(H$(P,K
)):IF L=0 THEN 11050ELSE M=10*(K=TK)+5*(
LEN(H$(N,K))=0)+5*(LEN(H$(S,K))=0)-5*(LE
N(H$(PP,K))=0)-5*(LEN(H$(P,K))=1)-5*(LEF
T$(H$(P,K),2)="AK")
11020 IF (LEN(H$(N,K))>0) THEN H$=LEFT$(
H$(N,K),1):X=((H$="A")+(H$="K"))*3
11030 M=M+X*(P=E)-X*(P=W):IF (LEN(H$(PP,
K))>0) THEN M=M-5*(LEFT$(H$(PP,K),1)="A"
)-3*(LEFT$(H$(PP,K),1)="K")
11040 IF M>MM THEN MM=M:SK=K
11050 NEXT K:L=LEN(H$(P,SK)):IF MM>0 THE
N 11060ELSE C$=RIGHT$(H$(P,SK),1):H$(P,S
K)=LEFT$(H$(P,SK),L-1):GOTO 11070
11060 C$=LEFT$(H$(P,SK),1):H$(P,SK)=RIGH
T$(H$(P,SK),L-1)
11070 PRINT@473-6*(P=W),C$;MID$("CDHS",S
K+1,1);:RETURN
12000 AT=473:LW=0:SM=CM:SK=LK:L=LEN(H$(P
,SK)):IF P=E THEN PP=W ELSE PP=E
12010 PG=0:IF WN=PP AND (CM>9 OR TM>0) T
HEN PG=1
12020 IF P=W AND LP=N AND WN=E THEN PG=1
 ELSE IF P=E AND LP=N AND WN=W THEN PG=1
12030 IF P=E AND LP<>N THEN D$=LEFT$(H$(
N,LK),1):GOSUB 23300:IF D>SM THEN PG=0:S
M=D
12040 IF PG THEN IF L THEN LW=1:GOTO 120
70ELSE 12100
12050 IF L=0 THEN D$=LEFT$(H$(P,TK),1):G
OSUB 23300:IF D>TM THEN SM=TM:SK=TK:GOTO
 12060ELSE 12100
12060 L=LEN(H$(P,SK)):IF RND(10)>8ANDTK<
>SK AND SM<>13THEN LW=1
12070 N$=H$(P,SK):IF LW THEN C$=RIGHT$(N
$,1):H$(P,SK)=LEFT$(N$,L-1):GOTO 12120EL
SE 12080
12080 FOR I=LTO1STEP-1:D$=MID$(N$,I,1):G
OSUB 23300:IF D>SM THEN 12090ELSE NEXT:I
=L
12090 C$=MID$(N$,I,1):H$(P,SK)=LEFT$(N$,
I-1)+RIGHT$(N$,L-I):GOTO 12120
12100 L=0:FOR K=0TO3:IF LEN(H$(P,K)) > L
 THEN L=LEN(H$(P,K)):SK=K:NEXT K ELSE NE
XTK
12110 LW=1:GOTO 12070'==== DISCARD
12120 PRINT@AT-6*(P=W),C$;MID$("CDHS",SK
+1,1);:IF TC THEN GOSUB 110  :RETURN ELS
```

Figure 3

```
BRIDGE-80     1.1
GL FARNSWORTH (C)1983

CONTRACT: 6C  VL DBL
```

```
                2
CLUBS     : A9864
DIAMONDS  : 7
HEARTS    : A765
SPADES    : 9

    ?
TC
    QC
```

```
CLUBS     :
DIAMONDS  : AKT4
HEARTS    : 43
SPADES    : AKJ5
```

```
   WE      THEY
3,960      400

    0        0
```

CARD PLEASE

Figure 4

```
BRIDGE-80     1.1
GL FARNSWORTH (C)1983

CONTRACT: 6C  VL DBL
```

```
                2
CLUBS     : A986
DIAMONDS  :
HEARTS    : A765
SPADES    : 9

    4?
```

```
CLUBS     :
DIAMONDS  : KT4
HEARTS    : 43
SPADES    : AKJ5
```

```
   WE      THEY
3,960      400

    0        0
```

SUIT PLEASE

Figure 5

```
BRIDGE-80     1.1
GL FARNSWORTH (C)1983

CONTRACT: 6C  VL DBL
 240 POINTS BELOW 2740 ABOVE
```

```
                2
CLUBS     :
DIAMONDS  :
HEARTS    :
SPADES    :

    7H
QD     QS
    TD
```

```
CLUBS     :
DIAMONDS  :
HEARTS    :
```

```
   WE      THEY
3,960      400

    0        0
```

ANY KEY

```
E RETURN
13000 SK=LK:AT=412-128*(P=S):L=LEN(H$(P,
LK)):IF (NOT LD)AND L=1 THEN C$=H$(P,LK)
:SK=LK:GOTO 13030
13010 SU=0:FORI=0TO4:IF H$(P,I)>"" THEN
SU=SU+1:SK=I:NEXT ELSE NEXT:IF SU=1 THEN
 GOSUB 23400:L=LEN(H$(P,SK)):IF L>1 THEN
 GOSUB 90   :GOTO 13030ELSE C$=H$(P,SK):
GOTO 13030
13020 GOSUB 90   :IF LD OR L=0 THENGOSUB
 23600:IF K$<>"" THEN 13030ELSE GOSUB 10
0  :FORSK=0TO3:IFK$<>MID$("CDHS",SK+1,1)
 THEN NEXT:GOTO13020ELSE13030ELSE SK=LK
13030 L=LEN(H$(P,SK)):IF L=0 THEN 13020E
LSE X=0:N$="":FOR CD=1TOL:IF C$=MID$(H$(
P,SK),CD,1) THEN X=-1:NEXT ELSE N$=N$+MI
D$(H$(P,SK),CD,1):NEXT
13040 K$=MID$("CDHS",SK+1,1):IF X THEN H
$(P,SK)=N$:PRINT@412-128*(P=S),C$+K$; EL
SE 13020
13050 A=99-704*(P=S):PRINT@A+64*SK,N$;CH
R$(30);:RETURN
20000 FOR TT=0TO200:NEXT:RETURN' ===== T
IME DELAY
20010 GOSUB 20000:IF (CV>5 AND TP(E)+TP(
W)>10) OR (CV>3 AND TP(W)+TP(E)>16) THEN
 DB=2:PRINT@WA+64*R,"DOUBLE";:RETURN ELS
E DB=1:PRINT@WA+64*R,"PASS";:RETURN
20020 IF PN<3 THEN SK=1:CV=2 ELSE IF PN<
10 OR LEN(H$(N,BS))<4 OR BS<2THEN SK=BS:
CV=CV+1 ELSE SK=BS:' ===== RESPONSE TO 2C
LUBS
20030 GOTO 3290
20040 ' ===== RESPONSE TO 1NT =====
20050 IF PN>10 THEN CV=3:SK=BS ELSE IF P
N>7 THEN ST=1:CV=2:SK=0 ELSE IF BS>0 THE
N CV=2:SK=BS ELSE 3280
20060 GOTO3290
20070 '===== RESPONSE TO STAYMAN CONVENTI
ON
20080 IF SK=1ANDCV=2 THEN 3270 ELSE IFSK
>1ANDCV=2 THEN IF SK=BS THEN CV=4:GOTO 3
290 ELSE CV=3:GOTO 3290 ELSE 3230
21000 REM======= SCORE DISPLAY
21010 PRINT@52,"WE      THEY";:PRINT@177,
STRING$(15,140);:FOR I=56TO312STEP64:PRI
NT@I,CHR$(191);:NEXT:U$="##,###"
21020 PRINT@113,;:PRINTUSINGU$;UW;:PRINT
@121,;:PRINTUSINGU$;UT;:PRINT@241,;:PRIN
TUSINGU$;BW;:PRINT@249,;:PRINTUSINGU$;BT
;:RETURN
22000 FOR CS=0TO3:PRINT@AT+64*CS,MID$(SN
$,CS*8+1,8);": ";H$(P,CS);WL$;:NEXT:RETU
RN: REM == DISPLAY A HAND
22100 BS=B(0):DF=ABS(M(BS)-M(B(1))):IF M
(B(0))<10 AND R>1 THEN BS=4:RETURN ELSE
IF (LEN(H$(N,BS))<5) OR (DF<7ANDM(B(1))>
9) THEN B=B(0):B(0)=B(1):B(1)=B:RETURN E
LSE M(BS)=M(BS)-1:RETURN
22200 SK=0:FOR CS=0TO3: IF LEFT$(H$(N,CS
),1)="A" OR LEN(H$(N,CS))=0 THEN SK=SK+1
 :' ===== COUNT ACES & VOIDS
22210 NEXT:SK=-SK*(SK<4):RETURN
22220 SK=0:FOR CS=0TO3:IF LEFT$(H$(N,CS)
,1)="K" OR LEFT$(H$(N,CS),2)="AK" THEN S
K=SK+1
22230 NEXT:IF SK=4 THEN SK=0:RETURN ELSE
 RETURN
22800 TT=0:FOR I=K TO K+20:TT=TT+PEEK(I)
:NEXT:IF TT<>1354 THEN TP(RND(3))=1:RETU
RN ELSE RETURN
23000 FOR CS=0TO3:M=0:L=LEN(H$(N,CS)):IF
 L>0 THEN FOR K=1 TO L:M=M-(MID$(H$(N,CS
),K,1)>"9"):NEXT'===== SUIT RANKING
23010 IF CS=SK THEN M(CS)=M(CS)+3
23020 M(CS)=L*2+M-(CS>1):NEXT CS:FOR BS=
0TO3:M=-1:FOR CS=0TO3:IF M(CS)>M THEN M=
M(CS):B(BS)=CS:NEXT ELSE NEXT
23030 M(B(BS))=-M(B(BS)):NEXT BS:FOR I=0
TO3:M(I)=-M(I):NEXT:RETURN
23100 P=-E*(P=S)-N*(P=E)-W*(P=N)-S*(P=W)
:RETURN:'-- NEXT PLAYER-
23200 C=VAL(C$):IF C=0 THEN FOR C=1TO5:I
F C$=MID$("TJQKA",C,1) THEN C=C+9ELSE NE
XT:STOP
23210 IF (C>CM AND SK=LK) THEN CM=C:WN=P
 ELSE IF (SK=TK AND LK<>TK AND C>TM) THE
N WN=P:CM=15:TM=C
23220 RETURN' ===== WHO WINS TRICK
23300 IF D$="" THEN D=0:RETURN ELSE D=VA
L(D$):IF D>0 THEN RETURN ELSE FOR D=1TO5
:IF D$=MID$("TJQKA",D,1) THEN D=D+9:RETU
RN ELSE NEXT:STOP
23400 IF SK<>TK OR DT+OT>11THEN RETURN E
LSE IF H$(E,TK)=""ANDH$(W,TK)="" THEN PR
INT@AT-3,"ALL TRUMP";:DT=DT+LEN(H$(P,TK)
):GOTO 5000 ELSE RETURN
23500 IF SK=>0 THEN IF D(SK)=S THEN RETU
RN ELSE P=N:N=S:S=P:AT=793:FORI=89TO281S
TEP64:PRINT@I,CHR$(216);:PRINT@I+704,BL$
;:NEXT:GOSUB 22000:RETURN ELSE RETURN
23600 K$="":SU=0:FOR K=0TO3:LL=LEN(H$(P,
K)):IF LL>0 THEN FOR I=1TOLL:IF C$=MID$(
H$(P,K),I,1) THEN SU=SU+1:SK=K:NEXT ELSE
 NEXT
23700 NEXT K:IF SU=1 THEN K$=MID$("CDHS"
,SK+1,1):RETURN ELSE RETURN
23800 REM --- COUNT DISTRIBUTION POINTS
23900 FORP=0TO3:TP(P)=NP(P):FORCS=0TO3:L
=LEN(H$(P,CS)):TP(P)=TP(P)-3*(L=0)-2*(L=
1)-(L=2):NEXTCS,P:RETURN
24000 REM ===== SCORING =====
24100 IF DT=CV THEN N$="MADE IT !!" ELSE
 N$="MADE "+STR$(DT-CV)+" OVER !"
24200 PB=0:PU=0:IF TK<2 THEN P=20*DB ELS
E P=30*DB:IF TK=4 THEN PB=10*DB
24300 IF CV=12 THEN PU=DB*(750+250*VL) E
LSE IF CV=13 THEN PU=DB*(1000+500*VL)
24400 IF GM THEN PB=PB+P*(CV-6):VL=VL+1:
IF VL>1 THEN VL=0:PU=700+PU
24500 IF GM=0 THEN PB=PB+P*(CV-6):IF BW+
PB>99 THEN VL=VL+1:IF VL>1 THEN VL=0:PU=
PU+700
24600 IF DT>CV THEN PU=PU+P*(DT-CV)
24700 PRINT@960,PB"POINTS BELOW"PU"ABOVE
";BL$;:IF HX THEN RETURN ELSE BW=BW+PB:U
W=UW+PU:HX=1:HN=HN+1:RETURN
25000 IF VL AND DB>1 THEN PU=100*DB+150*
DB*(CV-DT-1) ELSE IFVL THEN PU=100*(CV-D
T) ELSE PU=50*(CV-DT)*DB
25010 PRINT@960,PU"PENALTY POINTS"BL$;:I
F HX THEN RETURN ELSE UT=UT+PU:HX=1:HN=H
N+1:RETURN
```

Figure 6 — Program Structure

Line(s)	Function
20-110	Frequently used I/O routines. (Placed here for speed.)
Main Line	
1000-1060	Displays board and shuffles.
2000-2100	Deals and counts points. Line 2050 selects South.
3000-3300	Bidding (helped by several subroutines).
4000-4040	Start of play.
4100-4170	Play of the hand (many subroutines).
500-5010	End of play. Score it and begin again.
Bidding and Play Subroutines	
10000-10030	Obtains and validates South's bid.
11000-11070	Selection of East-West leads.
12000-12120	East-West play (when not leading).
13000-13050	North-South play and detection of automatic plays not needing input.
Miscellaneous Service Routines	
21000-21020	Displays score at upper right.
22000	Displays hand P at location AT.
22100	Calculates whether North should rebid.
22200-22230	Responses to Blackwood.
23000-23030	Evaluates North's suits and ranks them.
24000-24700	Calculates score and bonus if contract is made.
25000-25010	Calculates penalties for "set" contracts.

Figure 7 — Key Variables

N,E,S,W	Player pointers (0-3)
H$(P,SK)	Hands. P is player, SK is suit key (0-4 for C-NT)
LK	Lead suit key (as above 0=Clubs, etc.)
TK	Trump suit key
CM	Highest card played in lead suit
TM	Highest trump played
WN	Winning player
CV	Contract value
DT	Declarer tricks
OT	Opponent's tricks
TP(P)	Total points for player P
NP(P)	High-card points for player P
NP	North's points
MP	North's estimate of your maximum points

Exploring VisiCalc

Date computations

Models I/II/III/4III/12/16II

Timothy K. Bowman, Contributing editor

If you are among the many VisiCalc users who have wondered how you can manipulate dates, and even use the results in other calculations on your spreadsheets, this month's column has the answer for you.

Dates are fundamental to virtually any calculation that includes time and money. On the surface, it appears that VisiCalc cannot handle date calculations like some of the "modern" spreadsheet programs. How can we overcome this apparent shortcoming?

Let's start by keying in and examining the listing of a VisiCalc template which can calculate the number of days between two dates. It is shown in Figure 1 and should be keyed in after setting the column width to 12 (/GC12). While keying in the spreadsheet template, you might want to turn off the automatic recalculation feature (/GRM). I recommend turning it back on after the template is keyed in.

In this template, the user keys-in the beginning and ending dates in cell positions A3 and A4. The program calculates and displays the number of days between the two dates in cell position D6. It doesn't make any difference whether the beginning date is greater or less than the ending date. While that explanation sounds simple enough, let's take a closer look at how the problem is solved.

Solving the Problem

The key to the solution is in how the date is keyed into the cell positions A3 and A4. The date must be entered in the form MM. DDYYYY, where MM stands for the month, decimal point, DD stands for the day (including a leading zero if necessary for a single-digit day) and YYYY stands for the year. For the date August 3, 1983, the entry 8.031983 is a permissable entry, while 8.31983 produces strange results. Try keying in the date in an erroneous format if you don't believe me. Unfortunately, there is no error checking, but the result in D6 will make it obvious that an error in input has occurred.

Once the dates are entered, the housekeeping section in lines 7 to 29 take over. Let's try to gain an understanding of how the template works by stepping through the logic of solving for values in the beginning date found in A3. First, the template determines the month by taking the integer value of A3. Secondly, in A11, it determines the day by subtracting the difference between the integer value of A3 and A3 itself, multiplies that result by 100 to place the year value on the right side of a decimal place and then calculates the integer value of the entire expression. In a similar manner to the preceding step, the year is determined in A12 by eliminating all of the digits preceeding the last four. Lastly, in C13, the Julian day is computed. For those of you not familiar with the Julian date, it is simply the consecutive number of a given date during the year. In other words, February 1 is day 32 (31 days in January, plus 1), March 1 is day 60, etc. This calculation involves the use of a Lookup table in which the template compares the value found in A10 with values found in the range A17 to A29 and based upon that comparison, selects the beginning Julian date for that month. It deducts one to obtain the preceding month's ending Julian number and then adds the day found in A11 to calculate the Julian date for the specific date in question. Similar calculations are performed for the ending date.

After the preceding calculations are complete, the value in D6 is computed. It is the number of years between the two dates, times 365 days, plus the ending Julian day, less the beginning Julian day. The absolute value of the preceding

calculation is completed to allow for a beginning date in A3 that is larger than A4. Please note that this final calculation is specifically placed in D6 and the re-calculation order is set by column because the D6 calculation depends upon the previously computed values in lines 10 to 14.

Why Didn't I Think of That?

By now, I hope you are thinking, "Gee, that wasn't so hard!" If you have the need for date calculation routines in your spreadsheets, the above technique should be of help. Note that leap years are not considered, but the extra day is generally immaterial for most interest calculations. Also, this routine can be placed anywhere on your spreadsheet as long as the re-calculation order is correct.

Finally, if you are performing date calculations involving dates within the same year, your entries in cell positions A3 and A4 can be shortened to the form MM.DD where MM is the month and DD is the day. This reduces the input needed and has no detrimental effect on the spreadsheet's results.

Printer Code Corrections

A number of readers contacted me concerning sending control codes to their printer (December, 1983 issue). As a result, a few corrections and clarifications are in order. For Model I users, if you are using TRSDOS, it appears that the technique will not work because TRSDOS will not recognize and transmit the control code (SHIFT @ in the article). Give me 30 lashes with a spreadsheet for building your hopes up. You'll still have to do it from BASIC.

If you are using another operating system, another key sequence will probably replace SHIFT @ and

Figure 1

```
This is the cell content of DATECALC/VC:1:                    Page 1

            A                   B                   C                   D

  1  "     Computa      "tion of Days      " Between Two      " Dates
  2
  3  10.031982          " Beginning D      "ate
  4  10.011982          " Ending Date
  5
  6  "Number of Da      "ys Between t      "he Two Dates      @ABS((((+C12-A12
                                                              )*365)+C14)-C13)
  7
  8  "Housekeeping      " Section:
  9  " Beginning                           "Ending
 10  @INT(A3            "  Month            @INT(A4)
 11  @INT(100*(+A3-@I   "  Day              @INT(100*(+A4-@I
     NT(A3)))                               NT(A4)))
 12  10000*((100*(+A3   "  Year             10000*((100*(+A4
     -@INT(A3)))-@INT                       -@INT(A4)))-@INT
     (100*(A3-@INT(A3                       (100*(A4-@INT(A4
     ))))                                   ))))
 13  "Julian Date       "- Beginning       @LOOKUP(A10,A17.
                                            ..A29)-1+A11
 14  "Julian Date       "- End             @LOOKUP(C10,A17.
                                            ..A29)-1+C11
 15  "   Lookup Ta      "ble
 16  "    Month         "       Days
 17  0                  +D13
 18  1                  1
 19  2                  32
 20  3                  60
 21  4                  91
 22  5                  121
 23  6                  152
 24  7                  182
 25  8                  213
 26  9                  244
 27  10                 274
 28  11                 305
 29  12                 335
```

These are the DATECALC/VC:1 Global commands for its main window.

/GOC = The calculation order is by COLUMN.

/GRA = This sheet is set to recalculate with each value entry.

GFG = The General Format is in effect for this window. Values are displayed in decimal or scientific notation for the largest number of significant digits. Labels begin in the leftmost space.

/GC12 = Column width is set at 12.

A3 = Position at which the cursor is located.

generate a control code. In LDOS, for example, holding down the keys shift downarrow generates the control code. In order to obtain compressed print on an Epson printer from VisiCalc, at the prompt SETUP string or enter, hold down the shift downarrow and O (capital O) and press enter. Be sure to perform this operation smoothly because of possible keyboard repeat problems. If you are running an operating system other than TRSDOS or LDOS, consult your documentation as to what keystrokes create control codes.

TRSDOS 1.3 (Models III and 4III) users need to correct the instructions from the December column. The H character was missing following the SHIFT@ instructions. On page 72, next to last paragraph, the command SHIFT@HOF should be SHIFT@H0F. The command SHIFT@H1B SHIFT@14 should be SHIFT@H1B SHIFT@H14. On page 73, second paragraph, the command 1BSHIFT@0E should be 1BSHIFT@H0E. On TRSDOS 6.x, Model 4, VisiCalc is quite different. Setting printer codes requires filtering and translation tables. That subject will be covered later.

For Model II users, the technique is slightly different because your computer has escape and CTRL keys that are functional in VisiCalc. For example, to print in compressed print on a LPVIII or DMP 200, type the following key sequences with your cursor at the top left of the material to be printed (spaces have been added for clarity, but should not be typed): /PP *press ESC CTRLT enter* and move the cursor to the bottom of the text to be printed and press enter. The CTRL key and T should be pressed together. To escape compressed print, type: /PP *press ESC CTRLS enter*. It is important to note that while pressing the ESC and CTRL T or S keys, there will be *no* evidence on the screen of any key entry. Be careful.

Third, you have probably noticed that I switch between using a CTRLT sequence and SHIFT@H14. They are equivalent instructions.

Finally, for all users of either DWII, LPVII, or Epson MX80 or 100, if you have found that the compressed code sequence works and would like a fairly complete list of printer control sequences, send a stamped, self-addressed envelope to me for more information in care of *Basic Computing*. Examples abound such as printing unidirectional on the Epson, double-strike printing, italics, etc. If you have a different printer that you would like to have control code sequences developed for, write to me for an estimate to develop those for you. Keep in mind that I use Model IIIs and 4s that are totally "stock" Radio Shack machines with no modifications or alternate operating systems.

Update on Liaison

Following the review of Liaison (a BASIC program which converts ASCII files into VisiCalc DIF files (and vice versa) and allows for sorting of both types of files) in the September, 1983 issue of *Basic Computing*, I received upgraded versions of the programs which incorporated virtually all of the suggestions that I made and a few more. A particularly helpful one is that at the appropriate times, the operator is given the name of the current file in memory and has the option of sending it out to a different drive than the one from which the file was loaded without rekeying in the filename. The author, David Kjell, is including (at no extra charge) on the current distribution diskettes, another nifty program (VF/DTK) he wrote which reads a VisiCalc VC file and allows the user to leave it in the current order or to sort it and then either display it on the screen or send it to your printer. With the screen display, debugging large spreadsheets could probably be made much easier. VF/DTK is also available for separate purchase. For further information, contact David T. Kjell, P.E., 200 Timberland Trail, Euless, Texas 76039.

That's all for this month. If you have a spreadsheet question or comment, please write to me in care of *Basic Computing*, enclosing a stamped, self-addressed envelope if you desire a prompt, personal reply. Matters of a general interest may become the subject of a future column.

VisiCalc is a registered trademark of VisiCorp, Inc.

MC-10 utilities

Program renumber and screen dump

MC-10

Dan Keen and Dave Dischert

Those readers who sit down at their computer and develop programs off the top of their head know that the finished product usually winds up with erratic line numbering. If the program is solely for your own use, who cares? But if you are going to give the program to someone else or, perhaps, submit it to a magazine for publication, your program needs to have a nice sequential pattern.

For those reasons, we decided to write a short renumber utility for the MC-10. The program is designed as a simple routine which can be part of any program, without sacrificing a lot of memory.

Our renumber program modifies only the first through the last lines, but does not change the linenumber associated with such commands as IF. . .THEN, GOTO, and GOSUB. This could be done but it would require a lot more instructions to do it, and the machine does not have enough memory to handle it. Such a program would have to temporarily store all the old linenumbers and the new linenumbers. It would have to then search through all GOTO and GOSUB statements.

Even though our MC-10 renumber routine does not have the ability to modify line references, it is an exceptionally short, powerful, and easy routine to use. We suggest you type in the routine, Listing 1, and save it on tape. Before you begin to write a program, load in this routine. That way it will be present at the completion of your work when you need it to clean up your linenumbers. As we mentioned, you will have to retype any lines which refer to other lines.

How Renumber Works

We discovered that the first address of any BASIC program on the MC-10 begins at address 17222. This, in conjunction with the following byte (17223), stores the memory location of the next instruction. Bytes 17224 and 17225 store the BASIC linenumber.

Now that we know these addresses, we can easily find the address of each BASIC line that follows. Perhaps it is best stated as: The beginning of the last line always points to the address of the next instruction. In our routine, line 2010 establishes Q as 17222, the first memory address.

Execution then jumps to a subroutine which returns the address of the next line in variable Q. The current BASIC linenumber is LN, arrived at by Q+2 and Q+3. We used the variable LN to represent the current LineNumber and NL for the New Line. Also calculated are the most and least significant bytes of the new line you want.

LN is checked to see if it equals 2000. Normally, the end of a BASIC program could be detected by checking for a zero as the reference to the next linenumber. Here we check for linenumber 2000 to prevent the subroutine from renumbering itself. This would destroy the routine since it contains GOSUB references which would not be changed.

Obviously you must keep all of your program's linenumbers below 2000. This can be modified by changing the linenumbers in our renumber program as well as the value of LN in line 2030.

The most significant byte (MSB) of the new linenumber is poked into Q+2 and the least significant byte (LSB) is poked into Q+3. The new linenumber is incremented by ten. The program is set to start counting with a linenumber of ten and to increment by ten. You can easily set it to any other values.

With each pass of the subroutine (lines 2100 through 2140), Q is advanced to point to the next line. The subroutine is executed as many times as necessary until all lines have been changed.

Screen Dump and Memory Dump

One of the first things a hobbyist may want to do is explore his new MC-10's memory. Since not much material has been released regarding its internal workings, many hours of wandering through the machine's RAM and ROM are at the top of the list for new MC-10 owners.

In order to make peeking around in memory a simple task, we created the memory dump program in Listing 1. Using this program is much easier than typing PEEK for every address you wish to examine. Also, the printout is formatted to fit nicely on the screen for quick and clear viewing.

The screen format is laid out in three columns of data. Each column consists of three values. The left-most number is the decimal value; the middle is the hexadecimal code; the right character is the ASCII equivalent.

Non-printable characters, such as carriage returns (ASCII 13) are suppressed and only the decimal and hex numbers are shown.

The program first prompts you with "ENTER STARTING LOCATION", at which point you type the decimal value of the address you wish to begin examining. If you choose an odd memory address value, the program will subtract one from it to keep the starting address on an even location. We did this because, generally, references on the 6803 are on even addresses.

Each screen shows 32 bytes of memory at a time, making a nice symmetrical video layout. Three keys command all functions. Hit the enter key to continue incrementing through the memory and see the next 32 bytes. The "S" key restarts the entire program, returning you to the "ENTER STARTING LOCATION" prompt. If you have a printer, hitting "X" will give a screen dump of the video.

The screen dump option is of particular value and you may wish to incorporate that section into your own programs. Creating a screen-to-printer dump is not as easy as it sounds on the MC-10. Lprinting CHR$ values that are obtained by peeking the video memory will result in mostly garbage on the paper. This is due to the fact that the MC-10 apparently does not store all characters in their usual ASCII code format. Normally you can print or lprint using CHR$ and there is no problem since the BASIC interpreter takes care of restoring the values to their proper ASCII equivalent characters.

Lines 1100 to 1190 comprise the screen dump subroutine in which we perform the necessary mathematics to enable a peeked value from the video memory to have its ASCII equivalent character printed on a printer. These lines could be added to any BASIC program where a screen dump is necessary. It is a totally stand-alone routine called by a GOSUB command.

Figure 1 shows a sample of a screen dump taken during use of this program when an examination of the block of memory from 32 to 64 (decimal) was in progress.

Finally, we would like to point out to inexperienced programmers that the routine in lines 250 and 260 do the actual hexadecimal conversion and you may be interested in the technique we employed. In line 10 we establish N$ as being equal to every value which is possible under base 16. By doing a little math and some MID$ manipulating, a hexadecimal "string" value can be obtained from a known decimal number. In line 250, the variable M represents the most significant nibble and L the least.

For the MID$ trick to work, we had to add one to the most and least values. If L was zero, an illegal function call would result when the MID$ of position zero was attempted, since zero is not a valid position number.

Line 270 insures that a four digit string value will always be produced. By doing so it maintains straight columns on the screen when printing hex numbers. There is no PRINT USING command on the MC-10 so we opted for this trick.

Armed with this program, you can begin your search through the internal mysteries of the MC-10. Happy hunting!

Figure 1 Sample Screen Dump

```
MEMORY DUMP 32 TO 64
 32 20       33 21       34 22 "
 35 23 #     36 24 $     37 25 %
 38 26 &     39 27 '     40 28 (
 41 29 )     42 2A *     43 2B +
 44 2C ,     45 2D -     46 2E .
 47 2F /     48 30 0     49 31 1
 50 32 2     51 33 3     52 34 4
 53 35 5     54 36 6     55 37 7
 56 38 8     57 39 9     58 3A :
 59 3B ;     60 3C <     61 3D =
 62 3E >     63 3F ?     64 40 @

   HIT ENTER TO CONTINUE

 S TO RESTART X TO PRINT VID
```

Listing 1 MC-10 Renumber Utility

```
2000 REM * * RENUMBER * *
2010 Q=17222:NL=10
2020 GOSUB2100:PRINTN,LN
2030 IFLN=2000THENEND
2040 POKEQ+2,M:POKEQ+3,L
2050 NL=NL+10:Q=N:GOTO2020
2100 N=256*PEEK(Q)+PEEK(Q+1)
2110 LN=256*PEEK(Q+2)+PEEK(Q+3)
2130 M=INT(NL/256):L=NL-256*M
```

```
2140 RETURN

3000 FORA=65TO70:CSAVECHR$(A)
:NEXT:END
```

Listing 2 — MC-10 Memory and Screen Dump Routine

```
5 CLS
10 CLEAR1000:N$="0123456789ABCDEF"
20 PRINT "MEMORY PEEK IN HEX & ASCII"
30 PRINT"  WRITTEN BY DISCHERT/KEEN"
40 PRINT"            05/27/83"
50 PRINT:INPUT"  ENTER START LOCATION ";
LO
200 IF INT(LO/2)<>LO/2THENLO=LO-1
210 CLS3:PRINT"MEMORY DUMP";LO;" TO";LO+
32
220 PRINT:FORX=LO TO LO+32
240 I=PEEK(X)
250 M=INT(I/16):L=I-16*M+1:M=M+1
260 H$=MID$(N$,M,1)+MID$(N$,L,1)
270 I$="    "+STR$(I):I$=RIGHT$(I$,4)
300 IFSW=0THENPRINTI$;" ";H$;:IFI<>13THE
NPRINT" ";CHR$(I);
310 IFSW=1THENPRINTTAB(10)I$;" ";H$;:IFI
<>1313THENPRINT" ";CHR$(I);
320 IFSW=2THENPRINTTAB(22)I$;" ";H$;:IFI
<>13THENPRINT" ";CHR$(I):GOTO340
330 IFSW=2THENPRINT
340 SW=SW+1:IFSW=>3THENSW=0
350 NEXT
400 PRINT"     HIT ENTER TO CONTINUE"
410 PRINT" S TO RESTART X TO PRINT VID"
420 GOSUB 2000:IFIK$="S"THENRUN
430 IFIK$="X"THENGOSUB1100:GOTO400
440 IF ASC(IK$)=13THEN LO=LO+32:GOTO210
500 GOTO 420
999 END
1000 FORA=65TO70:CSAVE"HEXDUMP"+CHR$(A):
NEXT
1010 END
1100 FOR A=16384TO16895STEP32
1110 FORB=0TO31
1120 S=PEEK(A+B):W=S-64:IFW>31ANDW<96THE
NLPRINTCHR$(W);:GOTO1140
1125 IFS>31ANDS<96THENLPRINTCHR$(S);:GOT
O1140
1130 LPRINT" ";
1140 NEXT
1190 LPRINT:NEXT:RETURN
2000 IK$=INKEY$:IFIK$=""THEN2000
2010 RETURN
```

Basically BASIC

Reading your data and some short but useful programs

For all models

Karen Matthews

Three key BASIC statements (READ, DATA, and RESTORE) combine to allow you to store data to be used by your program within your program. This can be helpful for situations where you don't wish to read the data from an external device like a disk drive or cassette recorder. You can read your program and the data into the computer at one time.

The READ statement is of the format:
READ *variable (, variable, . . . variable)*
It causes the program to read the value currently pointed to in the DATA statement and store the value in the variable specified. The DATA statement has the format:
DATA *value (, value, . . . value)*
Each value must be separated by a comma except the last. DATA statements may be placed anywhere within the program as they are ignored by the program until needed. The RESTORE statement is of the form:
RESTORE
There are no parameters to the RESTORE statement. It is used to reset the DATA pointer to the beginning and may be executed many times within a program.

Let's look at a simple program:

```
10  FOR X = 1 TO 10
20 READ Y
30 PRINT Y;
40 NEXT X
50 DATA 2, 4, 6, 8, 10, 12, 14, 16, 18, 20
```

Lines 10 and 40 contain a loop that is executed ten times. Line 20 READS a value from the DATA statement and line 30 outputs the value. Notice that the program skips over line 50 and that line could be placed anywhere within the program. The output should be:
2 4 6 8 10 12 14 16 18 20
If we had forgotten a value in line 50, we would have received an "out of data" error from the computer.

Now, let's use the RESTORE statement.

```
10 FOR X = 1 TO 10
20 READ Y
30 PRINT Y;
35 RESTORE
40 NEXT X
50 DATA 2, 4, 6, 8, 10, 12, 14, 16, 18, 20
```

The output is:
2 2 2 2 2 2 2 2 2 2
The RESTORE statement always moves the DATA pointer back to the beginning value in the first DATA statement.

Let's look at another program:

```
10 DATA 1,2,3,4,5
20 READ A
30 READ B
40 READ C, D, E
50 PRINT A;B;C;D;E
```

The output is:
1 2 3 4 5
This shows that the DATA statement may be placed anywhere and that the READ statement may be executed many times as the computer keeps track of which DATA item to READ.

We're not limited to only integer data, we can also use string or character data. We might do something like this:

```
10 FOR X = 1 TO 4
20 READ A$
30 PRINT A$
40 NEXT X
50 DATA I , LIKE , MY , COMPUTER!
```

This will produce:
I LIKE MY COMPUTER!

Notice that in the DATA statement, there's a blank after each word before the comma. If we didn't include the blank, we'd obtain ILIKEMYCOMPUTER! Quotes must surround any string data that contains a colon, comma or leading blanks.

Of course, data types can be mixed within a READ or DATA statement. You could keep your entire Valentine's Day card list in DATA statements.

```
10 FOR X = 1 TO 3
20 READ NA$, AD$, ZIP
30 PRINT NA$, AD$, ZIP
40 NEXT ZIP
50 DATA "BASIC COMPUTING","TACOMA, WA",
98409
60 DATA "BLOW, JOE","ANYTOWN, MT",10101
70 DATA "ROACH, JOHN", FT WORTH, TX", 76101
```

This would generate an output of:

BASIC COMPUTING	TACOMA, WA	98409
BLOW, JOE	ANYTOWN, MT	10101
ROACH, JOHN	FT WORTH, TX	76101

It would be interesting to create a short program that could print labels and keep track of who sent cards back. Note that the commas enclosed in the quotation marks are part of the data to be read and not part of the DATA statement syntax.

Since you will probably not know exactly how many items are in your data file at one time, the last item is usually marked with some ending flag. This allows you to "look" through all the items in a data list without worrying about an "out of data" error. This end flag is an absolute must when using a FOR loop to match up an item in a data list. The code is something like this:

```
10 INPUT A$
20 IF A$ = "QUIT" THEN STOP
30 READ B$
40 IF B$ = "END" THEN 60
50 IF A$ = B$ THEN 80 ELSE 30
60 PRINT "SORRY, ";A$;" IS NOT IN THE LIST."
70 GOTO 90
80 PRINT A$; " IS IN THE LIST."
90 RESTORE
100 GOTO 10
110 DATA MONTANA, WASHINGTON, NEW YORK,
OREGON, ALABAMA, END
```

A typical session might be:

```
?MONTANA
MONTANA IS IN THE LIST.
?NEW YORK
NEW YORK IS IN THE LIST.
?ARIZONA
SORRY, ARIZONA IS NOT IN THE LIST.
?QUIT
Break in 20
READY
>
```

Together, DATA, READ and RESTORE add great flexibility to your programs. The uses for internally-stored data are almost limitless. For our last program, let's combine all the principles we've covered into a conversion program. The conversions are stored in the DATA statements at the end of the program. As written, the program will convert from feet to inches, or inches to feet. It will be very easy to add conversions by simply adding DATA statements.

```
5 CLS
10 PRINT : INPUT "CONVERT FROM, HOW
MANY":A$,Z
15 IF A$ = "QUIT" THEN STOP
20 INPUT "CONVERT TO";B$
30 READ C$, XF, D$
40 IF C$ <> "END" THEN 70
50 PRINT A$;" TO ";B$;" CONVERSION NOT
FOUND"
60 RESTORE : GOTO 10
70 IF (A$=C$) AND (B$=D$) THEN 90
80 GOTO 30
90 PRINT Z;A$;" EQUALS";Z*XF;B$
100 RESTORE : GOTO 10
110 DATA INCHES,.08333333,FEET
120 DATA FEET,12,INCHES
999 DATA END,0,END
```

I would enjoy receiving your hints or questions, write to me at *Basic Computing* and include a self-addressed, stamped envelope. That's it for this month.

Tandy topics

New Model 2000 software and a Model 100 telephone dialing program

For all readers, Model 100 program

Ed Juge, Director of Merchandising, Business Computer Products
1500 One Tandy Center, Ft. Worth, TX 76102

The Tandy 2000 introduction at Comdex in November was a success. We held it on Wednesday, November 30, at Caesar's Palace in Las Vegas, Nevada. At 9:00 a.m., we had a press conference, followed by demonstrations of some of the major software. Attendees asked questions of us and the software vendors who helped us get the software ready. Then at 11:00 a.m., a group of financial people came in for just about the same show. That night, we hosted a gathering for the Tandy Business User's Group and our Third Party Software vendors who were in attendance. The T-2000's were "unveiled" at noon in our booth on the convention floor, and drew quite a crowd. Several people told us the T-2000 was without a doubt **the** introduction of the Comdex show!

As expected, one of the first questions was, "Why not 100 percent IBM-PC compatibility?" As I indicated in January, it made no sense to design something less than the best that our engineers, and current technology, would permit. Why maintain compatibility with a two-year old machine that could be up- or out-dated at any time by its maker? The Tandy 2000 runs any MS-DOS software which respects the MS-DOS operating system, but it also benchmarks out at almost three times the speed of the IBM-PC, with two to four times the disk storage space, more memory, twice the graphics resolution in twice as many colors, and at a lower cost.

Another question we were asked, and one that will be of interest to present TRS-80 owners, is "How do we expect the Tandy 2000 to affect our other computers?" Well, we know it will have some effect. How much, we don't know, but we don't expect it to be major. The T-2000 is a top-of-the-line "personal" or "professional" computer. The Model 12/16 family, for example, are more data-processing oriented. A person who wants a computer that can grow to multi-user or networking, will still choose (we think) the 12/16 family. Those who have been buying the Model 12 as a "personal" computer, will probably migrate to the T-2000. Of course, the Model 4/4P is a much lower-cost family, which should keep right on going, except for those who find the T-2000 irresistible enough to move up. So, we don't expect any of the above to go away because of the T-2000.

John Dvorak (gossip columnist for *Infoworld*) speculated that the reason we weren't 100 percent compatible with the PC was because eating too much Texas chili fogs the brain. The general reception of the Tandy 2000 was so good, in fact, that I probably should send John a can of Texas chili — it appears it may be a *lack* of chili that fogs the brain!

Ovation Software

I mentioned last month that we were looking at some really breathtaking integrated software for the T-2000. We announced a product called "Ovation," from Ovation Technologies. Is it something! You really have to see it to appreciate it.

Where most other integrated (spreadsheet, word processing, database, graphics and maybe communications) packages work in what could be described as a "spreadsheet environment," Ovation works in more of a word processing environment, but it is completely "mode-less." There is no switching from one mode to another. There's a menu of just over 30 plain-English commands, which appears down the left-hand side of the screen when it is called. As you step down the word list with your cursor, an extended one-line explanation of the selected word overlays the screen.

Let's say you've typed a letter, and decided to graph some sales projections you included in your text. Select GRAPH, and you'll be asked what kind of graph you want. Select PIE, and a section of the screen is blanked, with a circle on

one side, with a question asking what numbers you want to graph. You can use the cursor to point to them in the letter (which is still totally active, useable, and scrollable in the other part of the screen). As you point to each number, the graph is drawn in real time. When you tell Ovation you're finished, it gives you back the full screen word processing display, and asks where you want the chart. You point to opposite corners of the area, and Ovation instantly puts the chart in, and reformats your text around it! Should you change one of the numbers in your text (on which the chart is based), the chart is instantly re-drawn. This isn't done by exchanging data between windows. There are no windows. It's all integrated in the most elegant sense of the word! The chart can be in color, right there in your letter! That's something the IBM-PC version won't be able to do, due to the PC's graphics limitations. That's just a sample of what Ovation is. You have to see it to believe it! Oh yes, it is planned to read in Lotus 1-2-3 (and other) files and formulas.

The Tandy 2000 and IBM versions (our package will contain both versions) should be available by early summer, although the internal target dates are sooner.

Non-Radio Shack Software Available Through Stores

Yes, that's right. You'll soon be able to go into one of our stores and order non-Radio Shack software, for shipment to the store within 24 hours! They won't carry a Radio Shack logo at all. We will stock the software in a Fort Worth warehouse. I've talked to a number of software vendors, all of whom have expressed a strong interest in participating.

Now, there are two things about the program that must be made clear. First, we won't stock just any program. We have to watch inventories just like any other retailer. We intend to carry proven, popular items only, from reliable, top-notch companies. Secondly, after-the-sale support will be *only* from the vendors, not from Radio Shack. There will be instructions in each package, telling you if you have a problem, where to call for help. Our customer service people will **not** know anything about these packages or how they work. Obviously some of our people will have verified that they do work, and any product we get repeated complaints about will be pulled from the program. These packages will be listed in our future catalogs and you'll see some of them in our advertising. It's a great departure for us, and we're excited about it! As of this writing, I've still got a lot of logistics to iron out, but the program should be in operation within the next couple of months.

Model 100 Telephone Dialer

At the risk of boring you non-Model 100 owners, I'm going to subject you to another one of my non-professional programs. Those of you who use the TELCOM program and have a switchboard to contend with at your place of business, know that there's a problem! If you want to be able to dial Dow Jones from home or work, you have to 1) modify your phone number to include or exclude the "9" it takes to get an outside line through the switchboard, or 2) have two separate listings in your ADRS.DO file. If you have an extensive phone number file and travel, there's one more problem; the "8" access code required by most hotels, for a long distance number. Well, here's one I call PBXCHG.BA, which handles the problem rather well. To set it up, you need to:

1) In line 20, set FW$ equal to "1-" plus your area code (FW$="1-817-" for ours).

2) In line 20, if you're on a telephone system like Centrex, which allows you to dial in-house numbers with only the last four digits, set PF$ equal to your first three in-house telephone digits (our numbers are all 390-nnnn, so for me, PF$="390-"). If you're not, set PF$="000-", or some unused prefix.

3) Your ADRS.DO file must contain phone numbers immediately preceded, and followed, by colons. Don't use spaces before or after the number! The listing should look like:

Topics

Tandy Corporation :1-817-390-3700: followed by whatever you like.

4) Your telephone listings must contain dashes as shown above.

Other information, such as any listing without a phone number, shouldn't have colons at all. For example, if you're using EMAILR. EJ from CompuServe's Model 100 SIG to send automatic EMAIL, an EMAIL address only listing would be:

Ed Juge [70007,1365

No colons; PBXCHG will bypass it entirely.

5) And finally, PBXCHG uses ASD.DO as a scratch file, so you shouldn't have one by that name.

Using the program is as simple as running it. It prompts you for "Access Code Desired," which you answer one of 3 ways:

1) Press *enter* if you just want the straight numbers dialed. All numbers are dialed without prefixes, except for the ones you've entered to be long distance ("1-xxx-").

2) Enter "9" to dial through your switchboard. All numbers are dialed as above except for a "9-" prefix, and *if* you've set it up, only the last four digits of in-house numbers are dialed.

3) Enter "8" if you're in an out-of-town hotel. In this case, all your hometown numbers are prefixed by "8-1-area code-."

Sorry, I didn't make provisions to handle local (when you travel) numbers normally in your ADRS.DO file with "1-area code-", but then I'm sure those of you who need to can refine the program to suit your specific needs. The program actually modifies your ADRS.DO file each time you run it. Mine has about 2.4K of addresses and it takes about 12 seconds to go through the whole thing. The program is given in Listing 1.

A parting word. The editor told me a while back that this column rated on a reader survey as "most often read." I'm amazed, and I thank all you readers (and voters), very sincerely. If there are any subjects you'd like to see discussed or cussed in "Topics," please drop me a line, and just say . . ."How about discussing xxxx in Topics sometime?"I get no requests now, and would like very much to provide the kind of information you want. Please forgive me if I don't answer your letter until you see it here. Thanks again. See you next month.

Listing 1

```
10 CLS:CLEAR1000:MAXFILES=
2
20 PF$="390-":FW$="1-817-"
30 PRINT@169,"Access Code
Desired ";:INPUTP$:IFP$=""
THEN50
40 IFINSTR("89",P$)THENP$=
P$+"-"ELSE10
50 OPEN"adrs.DO"FORINPUTAS
1
60 OPEN"ASD.DO"FOROUTPUTAS
2
70 IFP$=""THEN170
80      REM ADD PREFIX
90 GOSUB230
100 GOSUB250:IFC=0THEN160
110 GOSUB260:P1$=P$
120 IFMID$(IN$,C+1,1)<>"1"
ANDP$="8-"THENP1$=P1$+FW$
130 IFMID$(IN$,C+5,1)<>":"
THEN150
140 IFP$="9-"THENP1$=""ELS
EP1$=P1$+PF$
150 IN$=L$+P1$+MID$(IN$,C+
1)
160 PRINT#2,IN$:GOTO90
170     REM REMOVE PREFIX
180 GOSUB230
190 GOSUB250:IFC=0THEN220
200 GOSUB260
210 IFMID$(IN$,C+5,1)=":"T
HENIN$=L$+PF$+MID$(IN$,C+1
)
220 PRINT#2,IN$:GOTO170
230 IFEOF(1)THEN300
240 C=0:LINEINPUT#1,IN$:RE
TURN
250 C=INSTR(1,IN$,":"):L$=
LEFT$(IN$,C):RETURN
260   IFMID$(IN$,C+2,1)="-"
ANDMID$(IN$,C+1,1)>"1" THE
NIN$=L$+MID$(IN$,C+3)
270 IFMID$(IN$,C+1,6)=FW$T
HENIN$=L$+MID$(IN$,C+7)
280 IFMID$(IN$,C+1,4)=PF$T
HENIN$=L$+MID$(IN$,C+5)
290 RETURN
300 CLOSE:KILL"ADRS.DO
310 NAME"ASD.DO"AS"ADRS.DO
"
320 MENU
```

UTILITIES
BILLING PROGRAMS

Gas, Water
Sewage, Garbage

Goodman's Computerized Business Services

Paul G. Goodman
Box 203
Miami, TX 79059
(806) 868-5661

#35

BARCLAY WHYTE ASSOCIATES

WORDPROC I-III VERSION 3-0

Basic plus machine language Word Processor

TRS-80* Model I/III 48K Disc

Special Version Now you can embed control codes in your text for

Superscript Subscript Underline Italics More

Supports all MX-80** typefonts

Still only $14·95 each, with disc
$10·95 each, documentation only

BARCLAY WHYTE ASSOCIATES
P.O. Box 948,
New Westminster
B.C. Canada. V3L 5C3.
Add $1·00 for postage
B.C. Residents add tax at 6%

*Trade mark of Tandy Corporation
**Trade mark of Epson America

#36

IMAGINE!

An Income Tax Program for $24.44

Trademark of Tandy Corp — TRS-80™ Mod. I & III

That's the pro-rata cost of our excellent tax program when you buy BAP$ SUPER-9 package for $21.95. Simply enter earnings, deductions, capital gains. The program computes tax due or refund. The other 8 programs include 2 great games—OSCAR DRAFT (flight simulation with navigation) and WIZARD'S CITY (adventure) — and Income/ Deductions, Financial Statement, Mail List, Mileage, Income/Expense and Stock Charting.

9 PROGRAMS! $175 VALUE!

BAP$ SUPER

$21.95 DISK

Price includes disk
Add $1.50 shipping

BAP$ Software

6011 San Felipe
Houston, TX 77057
713/783-3433

#37

FREE

business software directory

- Radio Shack's Model I, 2, 3 & 16
- CPM: Xerox, Alto . . .
- IBM PC & compatibles

Data base manager, integrated accounting package, inventory, word processing, and advanced mailing list.

#38

Micro Architect Inc.
6 Great Pine Ave.
Burlington, MA 01803
671-273-5658

$$ MONEY TRAK $$

MODEL I/III MULTIDOS USERS - ONLY!

Now! A practical program, at a sensible price. This easy to use program keeps track of all your expenses on any number of files, including tax deductible items.

Sorts by date and saves automatically. Add, delete, correct & print-out functions. Self-prompting. Only $15.00 includes S & H. EXTRA! Free surprise gambling program with every disk.

Send check or money order to:

VICTORIA PRODUCTIONS
4344 LAUREL CANYON BLVD.
STUDIO CITY, CA 91604

#39

SUPER – P/R PAYROLL

Power and flexibility for all your payroll needs from 5 employees to an accounting firm with 40 payrolls

SAMPLING OF FEATURES

- Machine language sorts
- Multiple state tax withholding
- Departmental payroll journals
- Report generator for W-2's, 941a's and non-standard reports
- 14 User Defined Earnings/Deduction Categories With Employee Base Rates
- Pay deducted from check option for tips and meals
- Works with all DOSES

Complete System	$225
Manual With Demo System	$45*
Manual Only	$25*

Add $2.50 for S & H
CHECKS VISA UPS COD

*Full credit allowed when purchase complete system

Users say SUPER-P/R is simply the best.

If you don't agree after trying SUPER-P/R for 60 days, return the complete system for a full refund

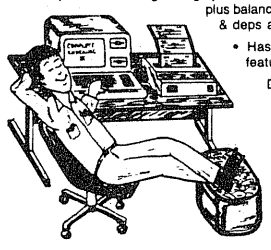
MICROCOMPUTER APPLICATIONS
3485 Mock Orange Court South
Salem, Oregon 97302
#40 (503) 364-1090

Write or call for free brochure or additional details
Specify Model I, Model III, or Model 4

FOR TRS-80 MODEL I/III COMPUTERS

Complete Checking II

- A complete CHECK WRITING and accounting system for business and personal use
- You can input your check info Manually or Automatically and deposits can be inputted semi-automatically, giving you a very easy job of maintaining your account
- The Automatic mode allows you to enter a 1 or 2 digit code to generate re-occurring checks such as: monthly utilities, rent, mortgage, etc.
- Specify where you want the appropriate info on each check only the first time you use it and it will print out and/or display the correct format automatically every time
- No more checkbook balancing worries. This program has a handy reconciliation feature that does it for you. The computer keeps all records and will print or display on request, reports such as: complete acct. listing, listing by dates, amts, desc., tax deduct plus balance, unreconciled cks & deps and more
- Has powerful editing features



Disk version (48K)
$39.95
Cass version (32K)
$34.95
plus $1.00 shipping

Sample Printouts available upon request
Also Free catalog

#41 212-454-9833

Coleman Computer Services
P.O. Box 588 St. Albans, N.Y. 11412

TRS-80 III / IV

★ BOOKKEEPER ★

Perfect For Small Business
TIME DOME Has Preloaded File

★ SCRIPSIT-MAILER ★

LETTERS - ENVELOPES - LABELS
made by Scripsit

★ SERVICE-BILLER ★

Bill By Rate Chart
From-To Date

★ BOOKKEEPING MACHINE ★

Printed LEDGERS - STATEMENTS
1400 Accounts Receivable

$34.50 EACH PROGRAM

(714) 774-9383
#42 H. A. K. WORKSHOP
9791 ORANGE AVE.
ANAHEIM, CA 92804

2- C

STOP GAMBLING -

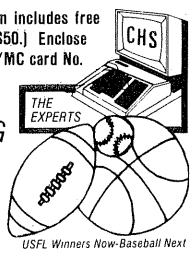
Use your personal computer for expert handicapping like the experts do - - -

SEND FOR FREE DATA ON USFL SYSTEM

- **Predicts game scores**
- **Situation analysis program**
- **Trend analysis program**
- **Unique handicapping guides**
- **Simple operating manual**
- **Use with Radio Shack, Apple Commodore 64 and others**
- **Order today-Win tomorrow**

START WINNING WITH

Send $225. for USFL System includes free money mgt. program (worth $50.) Enclose check, money order or Visa/MC card No.

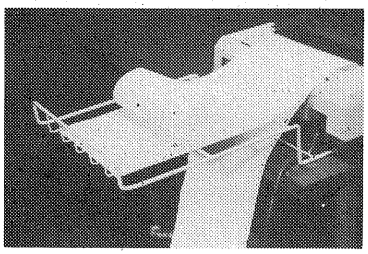


USFL Winners Now-Baseball Next

COMPUTER HANDICAPPING SYSTEMS
P.O. Box 33034
Cleveland Ohio
44133-0034

#43

MICRO PRINTER BASKET



- Use with any table
- Quiets printer
- Special power cord and signal cable retainer to keep paper path clear
- Sturdy steel construction
- Durable epoxy finish
- Beige color blends with equipment
- 12" for 80 column printers $22.50
- 18" for 132 column printers $24.50
(this size can also be used with 80 column printers)

Add $3.00 shipping and handling

SEE, INC.
P.O. Box 40215
Indianapolis, IN 46240
317-844-8817
Dealer Inquiries Welcome

#44

MasterCard VISA

SYSTEM LARGE CAPACITY SYSTEM LARGE CAPACITY SY

LARGE CAPACITY ACCOUNTING PROGRAMS

Model 1, 3/4, LNW, LOBO

ACCOUNTS RECEIVABLE

5000+ ACCOUNTS
30-60-90 DAYS AGED STATEMENTS SHOW DATE/INV#/DESCRIP/AMT(WITH AGEING). SELECTIVE FINANCE CHARGES & RATES, FAST ENTRY, POSTING W/AUDIT REPORT, SUB-ACCTS, % OF CREDIT LIMIT, DATE OF LAST PAYMENT, SALES ANALYSIS.

GENERAL LEDGER

400+ ACCOUNTS
+REPORT FLEXIBILITY/CAPACITY/DETAIL
+DEPARTMENT P & L (UP TO 9)
+UP TO 100+ SUB-TOTALS
+STATEMENT OF CHANGES (ASSETS ONLY)
+PERCENT P & L — MONTH vs YEAR

149.95 Each ★ Both for 199.95

add $50.00 ea. for HARD DISK
Super P/R payroll $225.00

#45

H. D. P.
2059 West Lincoln
Oroville, CA 95965
916 - 533-5992

★ Manual & Demo Disk - $20.00 ea.

3.00 S&H

VISA - MC or COD

APACITY SYSTEM LARGE CAPACITY SYSTEM LARGE CAP

WRITE FOR OUR FREE CATALOG

QUALITY SELECTION OF:

ARCADE GAMES
ADVENTURE GAMES
STRATEGY GAMES
WAR GAMES

Below are just three of many games available for the TRS-80 Model 1,3 and 4. Some games also available for the TRS-80 Color and TI 99/4A.

★**CENTURION**
As commander and gunner of an Israeli Centurion tank you must seek out and destroy PLO terrorists hiding in Lebanon. The enemy is displayed with an excellent use of graphics, and combat sound effects. Move, aim, and fire to destroy PLO troops, tanks, houses, buildings, and aircraft. You must act fast because the enemy's accuracy becomes better with time.

★**F-15 DEATH PILOT**
Turn your computer into the cockpit of the World's most advanced fighter aircraft. Prowl the skies on a search and destroy mission of Societ MIG's. This arcade game will allow you to enjoy tactical fighter combat in graphics and sound. The computer pilots the enemy MIG-25 fighters. No matter how good you think you are, you will eventually be killed.

★**B-52 STRATEGIC BOMBER**
Pilot your huge B-52 Bomber over enemy territory bombing and destroying bridges, buildings, factories, tanks, houses, bunkers, refineries and more. You must constantly maneuver around enemy Surface-to-Air missiles. Then return to base and refuel and rearm for the next mission.

Each game above is $10.95 on tape and $12.95 on disk. Our selection of games range from $8.95 to $12.95.

We have generous quantity discounts, up to 20% off when you buy five or more games.

Add 85¢ shipping or $2.85 C.O.D. #46

COMTRONIC SYSTEMS
31620 121st Ave. SE, Auburn, WA 98002

The House-Ware Genies

★ **The Mail Genie**
★ **The Pantry Genie**
★ **The Recipe Genie**
★ **The Calendar Genie**
★ **The Insurance Genie**

At last, the Turn-Key Programs that allow Dad to influence Mom, that a TRS ™-80 Model III and Color Computer is a modern day necessity.

Introductory Price $29.95 each

#47 Developed By
SOUTHERN CENTER FOR RESEARCH & INNOVATION
P.O. Box 1713
Hattiesburg, MS 39403
TELEPHONE 601-545-1680
TELEX 585400 SCRI US HATI
CABLE SCRI US

Dealer Inquiries Invited

...FAST...
...FAST...
...RELIEF...

From Basic
Announcing —
Assembly Language Business Programs For Model III

If you bought your computer with the anticipation of doing your business quickly, why wait for the Basic programs to do your work? All you need to wait is 1 or 2 seconds for your data, when the programs are written in Assembly Language.

We have.

General Ledger	**$99.95**
Accounts Payable	**$74.95**
Accounts Receivable	**$74.95**
Check Writer	**$39.95**
Church Giving	**$49.95**
Church Directory	**$39.95**
Mailing List/Labels	**$39.95**
Inventory Control	**$74.95**

For more information contact:
HOFFMAN ASSOCIATES

5 Sussex Ave.
Cherry Hill, N.J.
08003

#48 For More Info. Call **(609) 795-9467**

FlashSpell

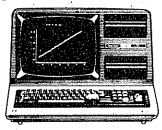
Spelling word flashes on screen, flash time doubles until spelled correctly.

NN Disk storage.
Your child's lists stored for nightly, cumulative review. No retyping.

NN Printed scoresheet.
Cram sheet for test day school bus review, shows which words are hard.

#49

Mod 3,4 data disk $14.99 + .85 post, incl. free "Insult". Chilly Disk Software, Code BC, 200 N. Cushman, Suite 201, Fairbanks, AK. 99701

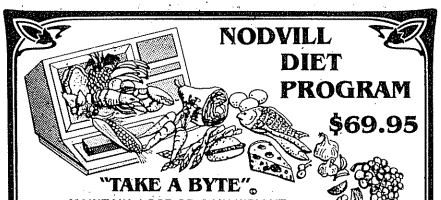


Best Prices On TRS-80 Computers

Our 6th year of discounts
Ed or Joe McManus
Fgt. Prepaid. Save Tax.
Toll Free 800-231-3680
Marymac Industries, Inc.
22511 Katy Fwy., Katy
(Houston) Tx 77450
1-713-392-0747
Telex 774132

See us in the Wall Street Journal every Tues., Wed., Thurs.

#50



NODVILL DIET PROGRAM

$69.95

"TAKE A BYTE"©
MAINTAIN, LOSE OR GAIN WEIGHT
With your TRS-80 Double Density Disk Model I/III/4 and The NODVILL DIET PROGRAM "Take a Byte"

you can quickly and accurately:
★ **Calculate** caloric and nutritive food intake
★ **Evaluate** nutritive value of your diet
★ **Compare** daily diet to individual RDA Chart
★ **Create** personalized daily meals and menus
★ **Plan** varied daily menus based on sound nutrition
★ **Save** records of daily meals and menus for future planning
★ **Print** nutrition charts, food, meal, menu and grocery lists

"Take a Byte" is a Modular **BASIC** Program

MAIN Program Menu
(1) Recommended Daily Dietary Allowance (RDA) CHART
(2) 733 Expandable Random Access FOOD LIST Data File
(3) 28 Nutritional MEAL LIST Data File Examples
(4) 7 Balanced DAILY MENU LIST Data File Examples
(5) GROCERY LIST Program Module
(6) 25 Page USER'S MANUAL including Charts and Tables

ALL Data Files can be EXPANDED and MODIFIED systematically and flexibly to reflect your personal diet, your choice of diet book, or your doctor's suggested diet.

For more information: Call: (203) 431-6449
To Order: Mail a check or money order for $69.95 to:

#51

NODVILL Software
24 Nod Road
Ridgefield, Conn. 06877

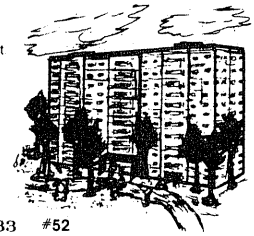
TRS-80 is a Trademark of the Tandy Corporation

FOR TRS-80 MODEL I/III COMPUTERS

Apartment Building Management System

FOR UP TO 30 BUILDINGS
FOR BUILDING OWNERS AND REAL ESTATE MANAGEMENT

- Takes care of all necessary accounting transactions for owners and tenants such as: RENT PAYMENTS • LATE CHRGS • MORTGAGE PMTS • UTILITY PMTS • TAXES • INSURANCE • CAPITAL EXPENDITURES • Plus 40 USER DEFINED INCOME AND EXPENSE CATEGORIES
- Features System Maintenance Development and Report Modes. Some Reports are: AUDIT TRAIL • LATE RENTS • PERCENT OF RTN ON INV • G/L PRINTOUTS • INCOMES • EXPENSES • INVESTMENTS • RCVD RENTS • LATE CHRGS • DEDUCTIBLES • UTILITIES • TAXES • INSUR • USER EXPENSE & INCOME CATS • VACANT UNITS • PERCENT OCCUPANCY • PROFIT/LOSS STMT and more!
- Minimum System: 32K 1 DISK. Also supports Hard Disk Drive. Can accommodate up to 30 buildings with up to 1000 units each with 48K memory.
- Sample Printouts are available upon request
- Free catalog too



- COST ONLY **$79.95** PLUS $1.00 SHIPPING

212-454-9833 #52

Coleman Computer Services
P.O. Box 588 St. Albans N.Y. 11412

Reviews

Guardian
32K Color Computer
Quasar Animations
1520 Pacific Beach Drive
San Diego, CA 92109
(619) 274-2202
$27.95 tape, $29.95 disk

Although the market is swamped with "Defender" clones, Color Computer gamers have been in desperate need for a really good version of it for some time. In comes "Guardian," by Quasar Animations, and takes many by surprise. With its unique graphic explosions and implosions, along with very well done sound effects and an attractive title page, Guardian lures people to play it. The pace is fast and wild, and gets much faster as it goes along.

Your planet has been invaded by Landers who aim to make off with your energy pods. If successful, they transform themselves into all-powerful Mutants, which can, and will, fire upon you. To aid them in their assault, there are the Pods who, if hit, release four deadly Swarmers. Swarmers move diagonally and are extremely hard to hit. There are also the Munchies, which fly around randomly and are a general nuisance. There are the Pulsers, who blink on and off, and can only be destroyed if hit when on. Finally, there are the killer Baiters. If you get a few of these on the screen, a smart bomb is your only safe way out.

Guardian utilizes the right joystick controller's up/down axis for the up and down movements of the ship. It also uses the left/right control of the joystick for left/right thrusting, and serves as a reverser. The spacebar will release a deadly Smart-Bomb, capable of destroying any alien life on the screen. The game also contains a useful long-range scanner, which comes in handy when fighting Swarmers, Mutants and Baiters.

This game also has some drawbacks. Even though you may have up to five laser shots on the screen at any given time, only the one which was fired last can kill. Though it is not obvious, the game is not written using true high-res graphics, but utilizes one of the semi-graphic modes, similar to low-res graphics. This makes the Landers over-sized, but accounts for the ease of handling the brilliant, multi-colored explosions. The instructions do not provide loading instructions for the novice programmer.

Guardian is addictive. I can't stop playing it. It contains nice sound effects and explosions. The plusses outweigh the minuses by a large margin.

Steve Skryniarz

TRS-80 for Kids from 8 to 80
By Dr. Michael Zabinski
For all models
Howard W. Sams & Co.
4300 North 62nd
Indianapolis, IN 46268

(800) 428-3696
$9.95

If you have a child who has just been dying to learn BASIC, then *TRS-80 for Kids from 8 to 80* is the book for you. It is written by Dr. Michael Zabinski, the founder and director of the National Computer Camps.

Each chapter is embossed with cartoons of smiling computers, kids and happy puppies. The book assumes that the user has had no prior experience with computers. It explains the most fundamental things, such as the video monitor, keyboard, and cassette recorder. The chapters are kept interesting by the use of checkpoints, experiments and reviews. Each chapter ends with a set of exercises that cover all the aspects of BASIC learned in that chapter. The answers to these exercises are in the back of the book. There are even Fun Time activities sprinkled throughout, such as word searches and crossword puzzles, that make use of the BASIC statements learned in the chapter.

Dr. Zabinski has done an excellent job of making learning BASIC fun. My son is already three-fourths of the way through the book, with no prompting from me. Those of you with children will realize what a miracle this is. Children hate to read books, especially in the summer. This book would make an excellent text for teachers considering the challenge of teaching BASIC to children.

One warning before you go out and buy this book. If you think you can lock your child and this book in the closet for a week and have an instant programmer, you will be sadly disappointed. Although the book is aimed at children, some of the examples and exercises may be a little tricky to understand without some adult interpretation. If you're willing to spend a few moments helping your children through the rough spots, you may have to buy them a computer so that you can use yours once in awhile.

Charles A. Quante

(Volume II, written by the same author, has just been released. —Ed.)

Grafyx Solution
Model III
Micro Labs, Inc.
902 Pinecrest
Richardson, TX 75080
(241) 235-0915
$299.95
Bizgraph Program $98
Draw Program $39.95
Surface Plot Program $39.95
Mouse Interface $99.95

The Grafyx Solution, a high-resolution graphics board for the Model III, from Micro-Labs, Inc., includes the add-on module, two manuals, two disks of software, and it sells for $299.95. Also available are the Joy-mouse Interface for $99.95 and, from any Radio Shack dealer, the Color Computer Mouse for $49.95.

Since I've never looked inside a computer before, or tinkered with one, I was the perfect guinea pig for the installation phase.

The module arrived in plastic and attached to a foil-covered styrofoam bottom support. First lesson: be very careful when you lift the module from its packing. The stuff clings for dear life. If you're not careful, you'll wind up picking little pieces out from between the tracings and wire ends, like I did. This delay isn't fun while anxiously looking forward to 512 x 192 dots worth of resolution.

Next, I tackled the "integrated circuits," each with fourteen fragile pins. The trick was to get all fourteen plugged simultaneously into some very snug receptacles. What if the pins bent? What if they snapped? Grizzly thought, huh? To avoid this, I recommend using an IC puller.

A light would be handy to check that all pins on the module are lined up and plugged into the corresponding holes on the main computer board. You never know what might boil if pins get crossed or plugged where they shouldn't be plugged. The solderless "micro-clips" which must be attached at various locations to complete installation went on with little trouble.

Dutifully noting Micro-Labs' caution about CRT neck implosion, I replaced the Model III cover and was ready for the moment of truth. I plugged the computer in and powered up. Not a puff of smoke. No alien sounds. Not even a marching band! TRSDOS booted up like I'd never tampered with the machine!

I inserted the "Grafyx Solution Software" disk containing BLD files to run a modified BASIC, 80-column video drivers, and a copious assortment of high-resolution data files demonstrating the capabilities of the graphics package.

On a 48K Model III, running TRSDOS 1.3 the BLD file didn't work. Having come so far, I certainly wasn't going to quit, so I entered BASIC, setting memory size to 60000, and typed from command mode: DEFUSR1 = &HF4A0: CMD "L", "GBASIC48/CMD": A = USR1(0). This got the software running in fine style.

The demonstration pictures were quite impressive but, in GBASIC, one three-dimensional function took over three hours to plot. I wondered if it would take as long for my Mouse Interface.

As with most drawing, one of the major issues is control. Two versions of Micro-Labs' "Draw" program attempt to provide this flexibility and control. One is limited to cursor-controlled sketching and the other utilizes the mouse interface. The mouse interface uses the Model III I/O port and allows you to attach joysticks or the Color Computer mouse. One nice feature with the mouse interface is the additional I/O port provided so that the user could conceivably operate a hard disk and mouse simultaneously.

I found that the mouse was a definite improvement over the cursor keys, especially when drawing curves. Yet, even with the mouse, precision control is difficult. The next development will undoubtedly be a graphics tablet or light pen.

The "Draw" program provides one curve function, several line functions, a brush command with four-point maximum width, screen load and save functions, variable velocity cursor, as well as erase, skip, or set functions. Despite all this, the more I drew, the more I felt it needed a graphics tablet. My creative endeavors were, in a word, tedious.

Micro-Labs, Inc. provides additional

support such as Bizgraph, which interfaces the Grafyx Board with VisiCalc, the Draw and Surface programs, and provides for 80-column video display. The Grafyx Solution does not interface directly with the Scripsit, VisiCalc, or Vidtex programs. The package offers everything Micro-Labs says it does, and at quite a bargain.

Greg Sheppard

Ace of Space
Models I/III/4III
Soft Systems and Consulting
P.O. Box 60031B
Santa Barbara, CA 93160
Tape $26.95, Disk $29.95

Get ready, space game aficionados. The Ace of Space is here and it's hot. This game is addicting, extremely challenging, and just plain fun. Up to three players can battle the alien saucers or each other (not one at a time, as in most games). We're talking about each player piloting a separate ship in real time — simultaneously. Granted, the keyboard gets a little cramped with three sets of hands competing for space; but, that just makes it more challenging. The space ship's graphics are crude, but the explosions are very realistic. The debris spreads rapidly in all directions and it can cause other ships in the vicinity to disintegrate.

What really makes this game great is the incredible number of playing options. You can control the number of ships, the number of alien saucers, the speed of the saucers, type of weapons aboard each ship (missiles or lasers), number of weapons aboard each ship, individual fuel supply, the length of your laser beams, the speed of the meteors, and the force of gravity. You may play with, or without, meteors, flying saucers, objects, space mines, or black holes. A unique option is the way an object is treated when it reaches the edge of the screen. It can be made to bounce off the edge, or to wrap-around, like a word processor does. A ship that traveled to the very top of the screen will then appear at the bottom of the screen in the "wrap" mode.

There are two scoring goals: cooperative or competitive. In the cooperative mode, one, two or three players attempt to destroy all of the enemy saucers before any one of the players gets eliminated. If you choose the competitive goal, you are pitted against other players, and the object is to kill or be killed. Numerous combinations of scoring options are allowed. Each player controls his ship with five keys on the keyboard. You have a variable thrust control (10 speeds), a hyperspace control for quick exits, right and left rotation, a 180-degree flip control, a right or left barrel toggle and, of course, a weapon-firing button. The control arrangement takes some getting used to; but, once learned, it is very fast, efficient, and seems quite natural.

The black hole is optional and its gravity can be varied. It acts very realistically to pull in all unsuspecting ships (not the enemy ones, though). If you are unfortunate enough to be pulled into the black hole, you are instantly scored as dead and the game ends. The high score for the session is saved, and a new game can be resumed by simply pressing one key if no option changes are wanted. Actually, the options are very easy to alter. The author is obviously quite a game player, as he has eliminated all of the annoying features of the past ones. All in all, a fantastic game. The disk is well protected, so don't expect to see this one passed around. Buy it!

Jim Klaproth

LSI Help System
Model 4
Logical Systems, Inc.
8970 N. 55th Street
Milwaukee, WI 53223
(414) 355-5454
LS-LDOS Help $29
LS-Technical Help $39
LS-Help Generator $59
plus $3 shipping/handling

The LSI Help System is a series of stand-alone programs for the Model 4 that provide on-line help for programmers and users of TRSDOS 6.x and BASIC. The system works in two modes. First you can call a help file from DOS by typing "HELP." You will be prompted for what sort of help you want (each type of help has its own filename). A list of keywords for that help file will be listed on one or more screens with a prompt at the bottom to enter the keyword. Let's say you need help with the BASIC keyword "LSET." You would enter HELP BASIC. A screen of keywords from BASIC would appear on the screen. If the word you want isn't there, hit enter and another screen of keywords appears. Type the word you want to know about and hit enter. A text file full of information will appear on the screen, or can also be sent to the printer.

I hear a voice in the background saying, "Hey, that's all pretty neat, but by the time I exit BASIC, load this program, look at the help file, exit the program and enter BASIC again, I'm going to forget what I was doing in the first place!" Wrong. This is where HELP gets good. There is another part of the program that's called HELPRES. This allows you to load the HELP drive into high memory where it waits to be called when needed. If you still need help with "LSET," and you are in BASIC, just hit clear shift H. You will be prompted for the keyword as before and the information will be shown as before. But when you are done reading about "LSET" and his break, you will be put right back where you were, in BASIC, with the screen just as it was when you left.

HELPRES also works at DOS level and with some assembly language programs, although LSI will not guarantee compatibility because of the thousands of programs out there and the programming practices of some authors. I did, however, enable HELPRES while using LSFEDII, the LSI disk-zapping utility, and it worked without problem.

LS-LDOS HELP contains two help files. The first gives information about LDOS (TRSDOS 6.0) disk commands. It explains how to use each command and how to use some of the special features such as the key-click filter and COMM, the communications program. The other help file covers all the BASIC keywords and gives brief examples of their use.

LS-Technical HELP works in much the same way, but the information contained is for the advanced programmer. There are four HELP files on the Technical HELP disk. Tech1 covers device control blocks, file control blocks, granule allocation tables, hash index tables, or (to put it another way) the housekeeping chores of the operating system.

Tech2 gives all the SVC calls and how the registers must be loaded when calling each SVC. The use of SVC's and this help file will save hours of time for the assembly language programmer. Also, for the assembly language programmer, the other two HELP files contain Z-80 mnemonics, their descriptions, flag information and the op-code generated by each.

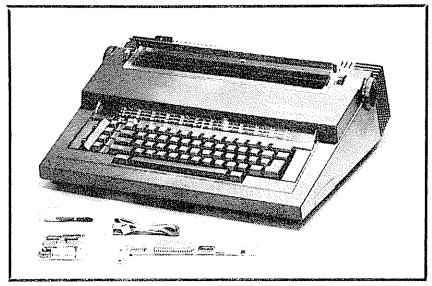
The LSI Help System really covers the bases. LS-LDOS HELP is for the person just getting started with LDOS or TRSDOS 6.0. LS-Technical HELP is for the person ready to write applications programs that use the operating system to its fullest potential.

Wouldn't it be nice to have such help files for Scripsit, or maybe a BASIC program that may really need such on-line help, but you just can't spare the room? Well, I wouldn't ask the question if I didn't have an answer. LS-HELP Generator is a sort-of compiler that takes text prepared on a word processor and makes it readable by HELP/CMD. You can imagine how good your program would look if clear shift H brought a screen full of explanations to a person trying to learn to use it for the first time.

These three programs prove to me that the decision by Radio Shack to do what they do best, build and sell computers, and let Logical Systems, Inc. do what they do best, design and support operating systems, means that the end user is going to get a better product in the long run. The HELP programs are available for LDOS 6.0 (TRSDOS 6.0) and LDOS 5.1. They work as advertised and are sold at a fair price. I recommend them for anyone who needs information quickly and doesn't want to scan through page after page of a user's manual to find it.

Al Mashburn

Bytewriter Praxis 40 retails for $645.



Bytewriter Praxis 30
Bytewriter
125 Northview Road
Ithaca, NY 14850
(607) 272-1132
$495

The Bytewriter daisy-wheel printer, an Olivetti Praxis 30 electric typewriter with parallel interface, sells for $495, not including cables which sell for $39 apiece. Replacement ribbons and various print-wheels (about twenty fonts are available) can be purchased directly from Olivetti, or indirectly through

Figure 1

```
!"#$%&'()*+,-./0123456789:; = ?@ABCDEFG
HIJKLMNOPQRSTUVWXYZ . _ abcdefghi jklmn
opqrstuvwxyz½,¼
```

Figure 2

```
!$[↑()ç;-.|0123456789:; é ?£ABCDEFG
HIJKLMNOPQRSTUVWXYZ ¿ _ abcdefghi jklmn
opqrstuvwxyzñ¡Ñ
```

Figure 3

```
This dual purpose daisy wheel printer/typewriter can
underline,                    tab and print in three pitches.

Program and circuit board design copyright (c) 1982
by Williams Laboratories.

BYTEWRITER
125 Northview Rd.
Ithaca, N.Y. 14850
```

Montgomery Ward. The Bytewriter we received arrived with cables for the Models I, II, III and Model 100 Radio Shack computers, as well as one Esteem Pica printwheel.

Installation was a simple matter of plugging the proper cable to the computer and printer and running paper through the roll-type 8-inch carriage. We had the machine printing in a few minutes.

The first thing we noticed was, though moderately slow (8 to 12 characters per second) and definitely noisy, the Bytewriter's print quality was excellent. We assumed, as with most other parallel printers, character selection would be ASCII-based. At this point, we looked through the manuals and discovered the ASCII codes weren't listed. With the keyboard manually selected to normal, LPRINTing CHR$(161) to CHR$(253) produced what is shown in Figure 1. With the keyboard switched to the alternate character set, we obtained Figure 2. Using CHR$(148) generated the message in Figure 3.

It's possible, though it isn't explained in the documentation, to do carriage returns, linefeeds, tabs and underlining using ASCII codes. Keyboard, margin and pitch selection must be done manually. If you try to print past the margin, which is automatically set to 30 on the left and 90 on the right at power-up, the printer pleasantly beeps, stops and waits.

Bytewriter does make available their interface kits for those who have purchased the Praxis 30, 35 or 40 typewriters elsewhere. The kit retails for $165.

Overall, the Bytewriter is simple to use and to move around. For those who tend to forget, Olivetti thoughtfully provided a red indicator light which glows when the machine is on. If you want, or need, quality type but aren't ready to spend your life savings, the Bytewriter is a low-cost, but productive, solution.

Greg Sheppard

Hexman Disk Management System
Models I/III/4III
Hexagon Systems
P.O. Box 397, Station A
Vancouver, BC, Canada V6C 2N2
(604) 682-7646
$49.95

The first thing one notices when he examines the Hexman disk management system is four sheets of 33 labels each. It is obvious at once that this is intended to be a very comprehensive directory management system. Put simply, it is a collection of compiled BASIC programs which is designed to take over the responsibilities of locating, placing and backing up all the files in your disk library.

I tried for several hours to initialize a set of disks according to the Hexman instructions. During this process, two /JCL files are executed. After following the instructions, the Hexman system should be ready to use. I found this not to be so.

According to the documentation, the system is supposed to catalog the Hexman files as its first task. In my case, after three tries, all I could get was "Error: too many files. DIR/SYS:7." This I received despite the fact that I don't have a drive 7! On the fourth attempt, the cataloging of Hexman's own files was skipped and a menu presented. After adding some disks to the library, Hexman told me to reinsert my system disk and immediately gave me an "illegal function call error at address 9D38." No additional indication as to what caused this error was given.

The documentation is sometimes inaccurate. I was not able to get past the above-mentioned errors to a satisfactory system. One gets the impression that it would be faster to catalog files manually rather than fuss with Hexman. Perhaps this is so, but the program wouldn't work properly on either a Model III or a MAX-80 computer.

After receiving a new pair of disks from Hexagon systems, I tried again. After all, it would only be fair to verify that the problem wasn't a defective diskette. It seemed to go better this time. I actually got some files into the filestore! Unfortunately, I had to do some things that were not outlined in the documentation. After following the initialization procedure exactly, which involves little more than executing a couple of /JCL files and typing SYSTEM(SYSGEN) when told to, I found that Hexman expected four of its overlay files to be on the system, but they were not. I had to copy them over manually in order to avoid "Program not found" errors. The documentation shows this error message in the back, but says

that it will place all its needed files on the disk in the front.

After getting the system going, I tried several of the options on the menu. Most performed as the documentation stated, and I could see that Hexman was intended to be a very comprehensive file manager with some very good screen displays and a number of good ideas. Unfortunately, the program aborted to LDOS Ready when I asked for a search of the filename "PASCAL" from option 1 of the review menu. The error given was a message from the compiler runtime package: "Illegal Function Call at address 95C8." I tried again, asking for the file "PASCAL/CMD." This time, there was nothing but the same four choices printed again on the screen. This happened very fast, without any disk access.

Perhaps I did something wrong, but there was no hint as to what it was from the error message. The filenames "PASCAL/CMD" and "PASCALB/CMD" did exist, both in the filestore and on the system, at the time. Whatever was wrong, it caused differing results because of the different spellings of these filespecs.

The documentation appears to be a real culprit in these problems. It is poorly prepared. On page 2.11, there is part of a line that was printed, but manually crossed out. There are numerous spelling errors throughout the manual. I think this is peculiar since Hexagon Systems sells an excellent spelling checker program called HEXSPELL. They should have taken the time to use it. There were also a few sentence fragments as well as poor grammar, but these were not glaring; they were just indications that the documentation was poorly prepared. I think that Hexman has great potential as a complete file management system, but the authors should spend more time leading the purchaser through the first few operations of the program. There really is no excuse in a commercial software package for the instructions to be either inaccurate or incomplete. It's a shame, because this type of file manager really needs to be written.

On the positive side, the program has some excellent screen displays. The menus presented, though not self-documenting, are uncluttered and easy to read. Efficient use of graphics leads you to the selections offered with ease. Quite a bit of effort must have been spent in designing the screens. The "ADD NEW FILES" option allows inclusion of user data about each file. The screen presents a good deal of useful information about a file to the user. The activity history of a file is shown for the month, together with the date it was added to the filestore, and the average number of loads. You are allowed to edit the description given to a file whenever you wish as well as change its security level (the number of backup copies you want to maintain). The size of the file and the disk number it is on is also shown.

With the Hexman system comes a tutorial disk. This is an excellent tutorial and ran perfectly the first time. Unfortunately, one cannot interrupt the Hexman program to run the tutorial, or it might serve to partially replace the poor documentation. I must applaud Hexagon Systems for this well-implemented idea.

Repeatedly, the instructions say something like, "Load the review program." You have to read carefully to find out what is meant by "review program" and how to load it. Experienced computer users will be able to do this, but newcomers will have trouble.

On one occasion, I left a drive door open. The error message was "Disk Error on File: DIR/SYS:2 File No. 9999 Too Many Files." Just how you can have too many files on an empty drive totally eludes me. The problem here may be dependent on the error code returned by LDOS. If it is possible for an error code to have multiple meanings, the error message should reflect this.

The LDOS for the MAX-80 is, as far as the Microsoft BASIC Compiler is concerned, identical to the Model III LDOS. The TRS-80 ROM is duplicated in the RAM in the MAX with only some low-level changes to keep from copyright infringement. All of the documented system calls in the back of the LDOS manual are identical and work the same way. All programs that I have previously compiled with this compiler, even those done on a Model I over a year ago, ran perfectly and without error. It would appear that the author is using undocumented system memory and disregarding good programming practice. Hexagon Systems, however, when asked if this was what they were doing, responded with "No" and a statement that only BASIC code which would run with the interpreter had been used. If this is true, it certainly rules out the MAX-80 as the source of the problem.

I called Hexagon to see what they could do. I was greeted with an answering machine at 3:00 in the afternoon their time. The message appeared oriented toward order taking. My call was returned the next day, and I was assured that these problems had not been reported by other users. I was told that one MAX-80 owner had reported a different problem some time ago, but he had apparently solved it because they had not heard from him again. Although my problem was not solved, the person was kind and sympathetic. He was, I found out later, the program's author, Bernard Hughes.

My best advice would be to wait awhile before purchasing this program. There has been a good deal of effort put into it and it has the potential to be fantastic. It just looks like it might have been released prematurely. Perhaps the folks at Hexagon Systems will improve the product. If you do decide to buy it, protect yourself by negotiating a return agreement in case of difficulty.

Charles P. Knight

Subterranean Encounter
Models I/III/4III
Toucan Software
4024 Canonero Court
Fair Oaks, CA 95626
$22.95 tape or disk

There are many computer enthusiasts who enjoy challenging adventures as well as graphic programs. Now there is a new program that has emerged from the active minds at Toucan Software: Subterranean Encounter! This graphic adventure game combines all of the elements that encompass both categories into a masterful blend of enticing programming enjoyment.

Subterranean Encounter is a 49-room graphic adventure that requires 48K of memory for the Model I, III or 4III computer. It is available on disk as well as tape (a nice benefit for those who have 48K, but no disk).

The game centers around and inside (for those clever enough to enter) an old castle full of surprises and deadly traps. It takes place during the days of medieval combat and intrigue, B.C. (before computers).

To complete this adventure, you must conquer many problems inside as well as outside of the insidious stone structure. A few of the simpler obstacles include an angry hermit, a cold and deep moat surrounding the castle, sword-carrying suits of rusty armor, confusing mazes, secret passageways and compartments — in other words, Subterranean Encounter features *all* of the elements that are found in most classic (or destined-to-become classic) adventures.

The graphics take up most of the screen. Commands are shown and given at the bottom of the screen. The graphic depictions are, indeed, realistic. For example, when you are in the forest, it honestly looks like a forest. Obviously, the authors took great pains to insure the accuracy and detail of their graphic renditions.

Subterranean Encounter is the first of many new graphic adventures. If all future adventures are as fun and as challenging as this quality program, the gaming community will be overjoyed.

Bob Krotts

For immediate release

Software Directory

The first volume of the Microcomputer Software Directory is now available. The directory emphasizes the major, original and specialized packages, including detailed descriptions of over 3,600 packages and 1,000 suppliers. Designed to provide the user with enough information on each package to make a well-founded buying decision, the directory's listings include product description, the machines it will run on, matching operating systems, memory and peripherals required, number of users and when the package was first operational. The directory is available at $35 (plus shipping and handling) on a 15-day money-back trial, and can be ordered from Computing Publications, Inc., Princeton-Forrestal Center, 101 College Road East, Princeton, NJ 08540, (609) 452-8090.

Under-$100 Printer

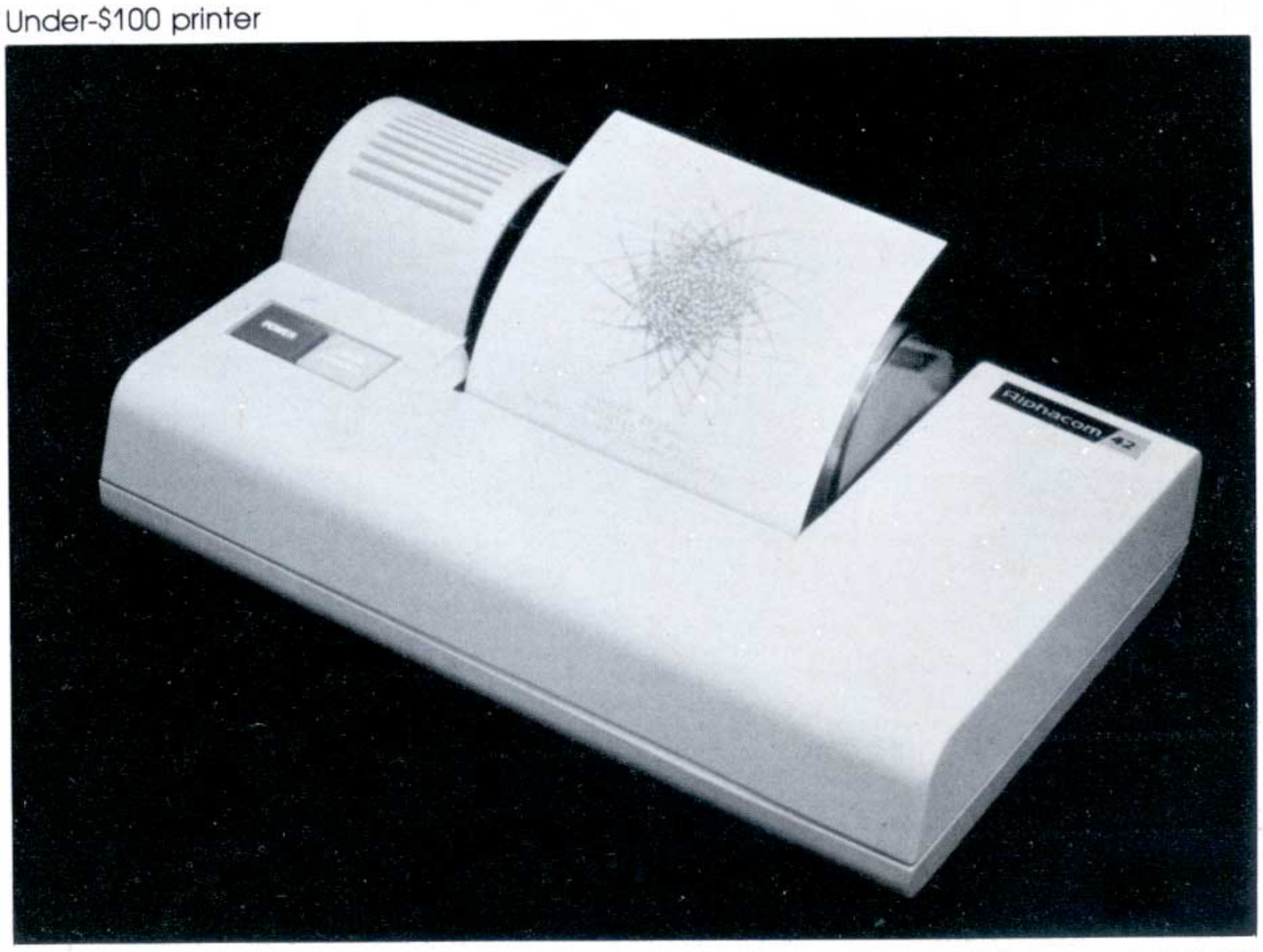
Alphacom, Inc. announced a price cut on its 40-column Alphacom 42 "universal" printer, lowering its suggested retail price to $99.95 including an interface cable. The printer may be linked to popular computers by plugging the appropriate interface cable into the printer's cartridge-like slot. The printer may be purchased separately at a suggested retail price of $79.95. Interface cables are available separately at $20 each (except the cable for the TI 99 4/A, which is $40).

The Alphacom 42 combines a single-chip microprocessor and an Olivetti print mechanism using thermal technology. The unit operates at two lines per second and features bit-mapped graphics. Other features include upper- and lower-case letters, wraparound for lines longer than 40 characters, and it recognizes standard ASCII control or "action" codes for changing the printing mode. Codes include carriage return, linefeed, right justification, form feed, graphics control, and multi-line feed. Contact the Consumer Sales Department, Alphacom, Inc., 2323 South Bascom Ave., Campbell, CA 95008, (408) 559-8000.

Portable Computer Case

The Chip-Tote is a carrying case for the TRS-80 Model 100, NEC PC-8201 and Epson HX-20 portable computers that doubles as a desk. The case features a slim, fully foam-padded design that opens up into a one-piece work station. The computer never leaves the bag. The stand-up top holds papers and inner pockets keep notebooks and notepads handy. A zippered pouch

Under-$100 printer



holds an AC adapter, acoustic coupler, modem cord or extra batteries. The Chip-Tote is made of DuPont Cordura nylon in black or smoke gray for $59.95. It's available from Kangaroo Video Products, Inc., 9190 Manor Drive, La Mesa, CA 92041, (619) 698-0230.

Trav-L-Case

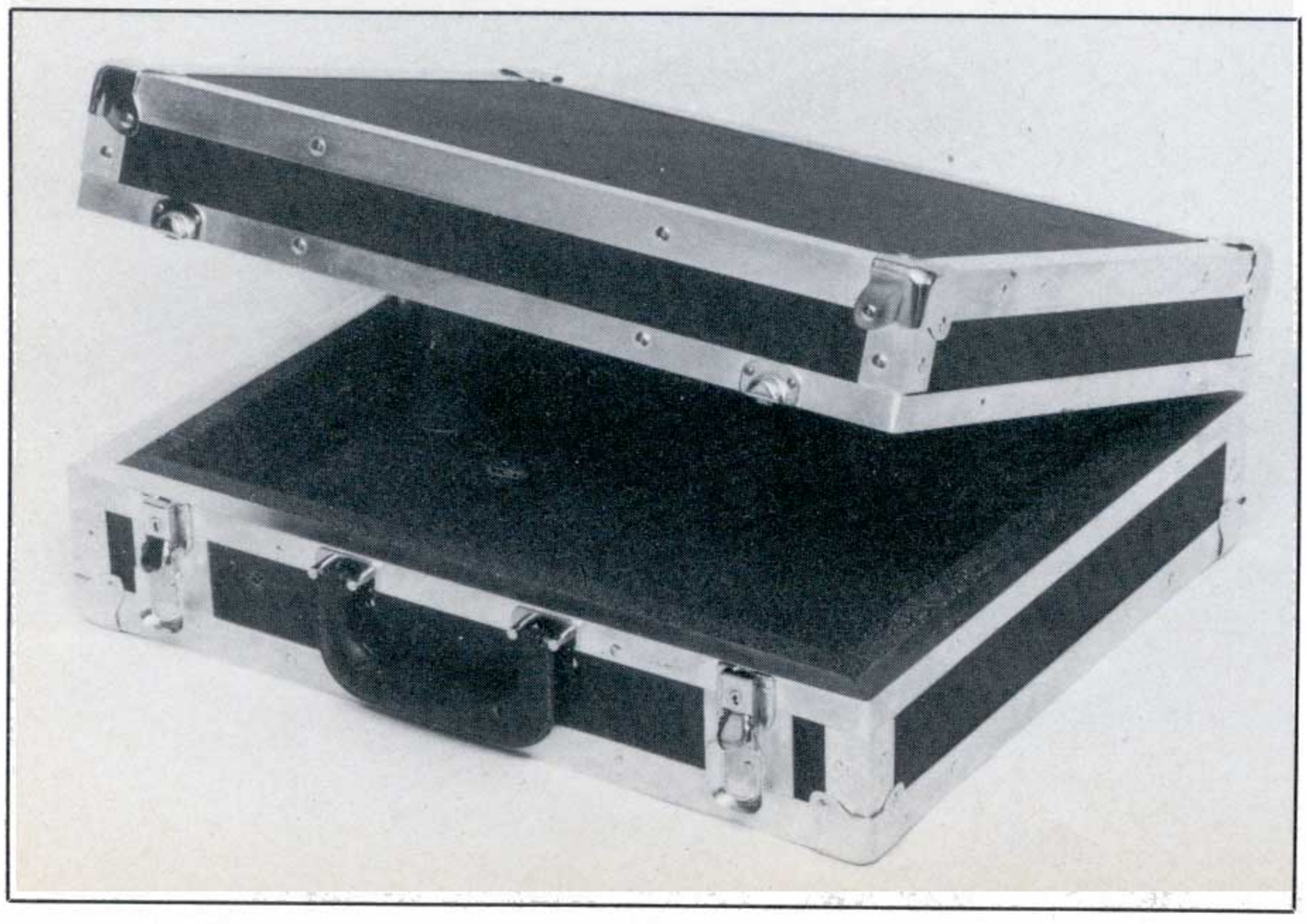
The Computer Case Company has introduced a complete line of heavy-duty cases for use by the frequent long-distance traveler. These cases, called Trav-L-Case, are made of plywood sides covered with scuff-resistant vinyl on the outside, metal protection on the edges and corners, key draw bolt locks, padded handle and a minimum of one-inch foam padding on all sides for protection of the equipment. The cases have been sized to hold the equipment that can be handled by one person and will easily fit inside the trunks of most standard-size cars. The cases can be obtained through most computer stores or direct from the Computer Case Company, 5650 Indian Mound Court, Columbus, OH 43213, (614) 868-9464 or (800) 848-7548.

Color Computer Interfacing Book

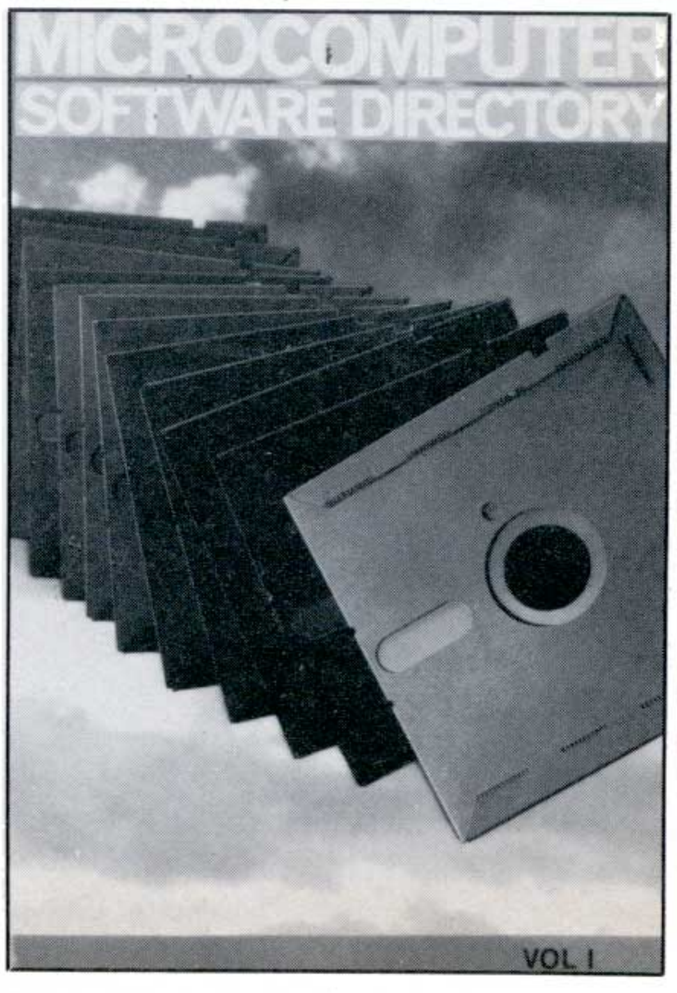
TRS-80 Color Computer Interfacing, With Experiments, a 203-page book by Andrew C. Staugaard, Jr., provides the reader with a basic understanding of the inner structure and operation of the TRS-80 Color Computer, its major large-scale integrated (LSI) circuit components, including its 6809E central processing unit and its expansion capabilities. The author shows how the three fundamental interfacing functions (decoding, three-state buffering and latching) can be performed using digital logic design techniques. He describes the use of a programmable interface adapter to develop a simple parallel input/output interface for the expansion connector. Digital-to-analog and analog-to-digital converters are covered and instructions given for connecting them to the Color Computer to monitor and control external events. Six practical experiments illustrate the material presented. The book is 5½ x 8½ inches, softbound, book number 21893, $14.95 plus $1 shipping. Contact Group Technology, Ltd.,

Portable computer case

Trav-L-Case



Software directory

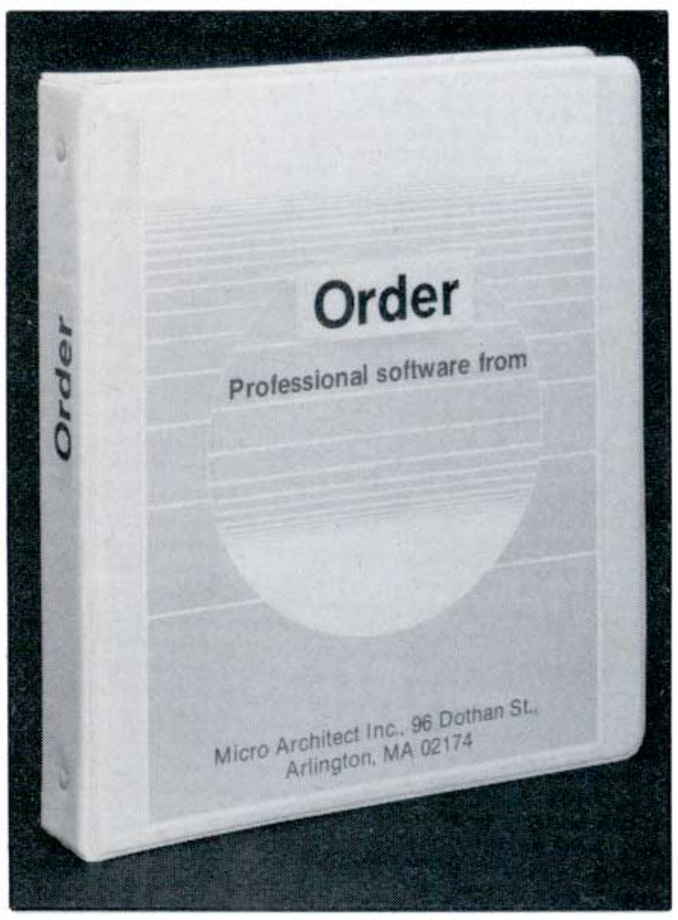


P.O. Box 87, Check, VA 24072, (703) 651-3153.

Model 100 Software

Magnificent 7 is a collection of Model 100 programs for entertainment and serious use. Tape One includes programs entitled Perpetual Calendar, Bank Finance, Bio-Rhythm, Sliding Squares Game, Casion Slot Machine, Casino Blackjack, and Draw Poker. Tape Two contains music, finance, and graphics programs. Price per tape is $21.95, including shipping. Contact Complete Computer Services, 8188 Heather Dr., Newburgh, IN 47630, (812) 853-5140.

Order entry system



Pedestal computer table

Crayon Deluxe

Crayon Deluxe is a complete bit-image graphic/text word processor for the Models I, III, and 4III computers. It comes with an 18K full-screen editor, a font design utility, three complete character fonts, and 130 page manual. Besides normal word processing tasks such as justification, programmable tab lines, user defined keys, it includes specialty fonts, sub and super scripting, a screen format that is the same as the printout, and more. The package sells for $150, and supports all Epson MX and FX printers, Gemini, Prowriter, and NEC printers. Print samples and a complete brochure are available on request. Contact Pioneer Software, Inc., 1746 N.W. 55th Ave., #204, Lauderhill, FL 33313, (305) 739-2071.

Order Entry System

Micro Architect's Order Entry System can integrate with their Inventory Control System, INV-X, and their Accounts Receivable System, AR. It can print invoices and update inventory information automatically. Multiple inventory items may be entered. Taxes and shipping charges are automatically computed. The Order Entry System sells for $99 and the complete Order, INV-X, and AR package is $479. All programs are designed for the TRS-80 Models II, 12, and 16II. Contact Micro Architect, Inc., 6 Great Plains Ave., Burlington, MA 01803, (617) 273-5658.

Free CoCo and MC-10 Software

Commterm is a new communications terminal program for the Color Computer and MC-10. With the correct hardware (RS-232 and a modem), the program will allow access to bulletin boards, timesharing services and remote computers. Star-Kits Software Systems Corportation will provide the program free to anyone who sends them a blank cassette and a stamped self-addressed envelope. In return, they ask for your evaluation of the program and a "fair contribution" to them to encourage further development of inexpensive software. Contact Star-Kits Software Systems Corporation, P.O. Box 209, Mt. Kisco, NY 10549, (914) 241-0287.

Color Computer Compiler

Speed up your BASIC programs an average of 40 times faster. The Color Computer Compiler from Computerware supports 46 commands; most are a subset of Extended Color BASIC. Generated code is position independent and can reside anywhere in memory, including ROM packs. The package sells for $39.95 and requires 32K, one disk drive. Contact Computerware, P.O. Box 668, 4403 Manchester Ave., Suite 103, Encinitas, CA 92024, (619) 436-3512.

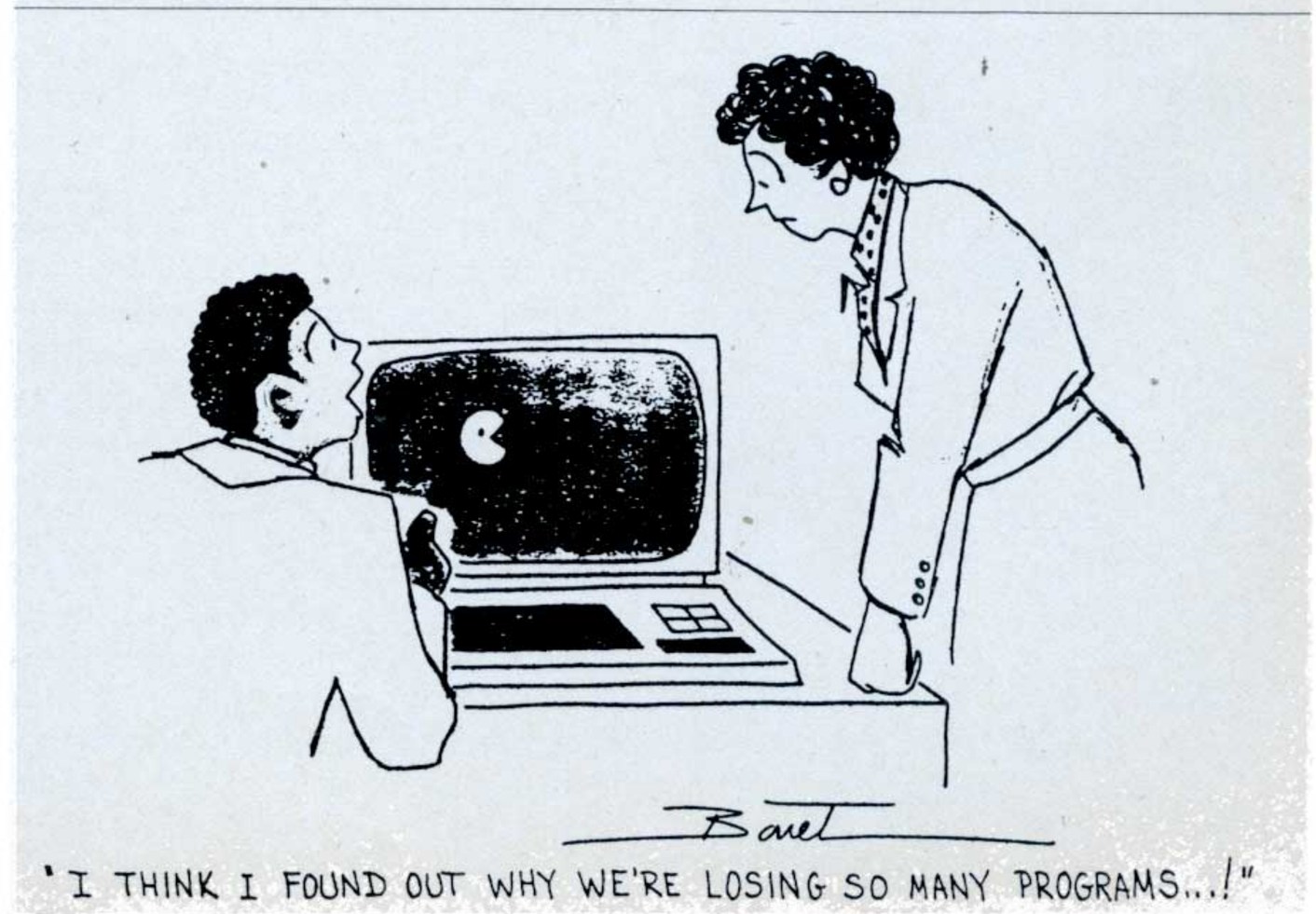
SeekEasy

SeekEasy is an easy to use filing system that accepts vague, incomplete, misspelled, or only partially correct inputs, yet still finds and displays the information you want. Most likely items are at the top of the displayed list. Even if you forgot "good old Steve's" phone number and last name, SeekEasy will find it. There are no new commands or input order to learn; just type what you want to find and press enter. Storing new data is just as simple. The program uses the CP/M 2.0 or 2.2 operating system. Model I TRS-80's modified with Omikron, or Model III's modified with Memory Merchant are supported. A demonstration disk and manual are available for $15, the complete program sells for $235. Contact Correlation Systems, 81 Rockinghorse Road, Rancho Palos Verdes, CA 90274, (213) 833-3462.

Pedestal Computer Table

The award winning pedestal computer table is available from Taylor Woodcraft, Inc. Its features include locking casters, bumper guard, cord organizer, low-glare finish, and automatic height adjustment. The keyboard ledge adjusts from 22 to 28 inches and slides under the top for storage when the computer is not in use. The top pivots a full 360 degrees for shared reference to the CRT by more than one operator. Suggested retail price is $260. Contact Taylor Woodcraft, Inc., P.O. Box 245, South River Road, Malta, OH 43758, (614) 962-3741.

Advertiser index



programmer s

READ THIS...

* WHO TO SELL YOUR PROGRAMS TO
* THOUSANDS OF COMPANY NAMES AND ADDRESSES, WITH DETAILED LISTINGS SHOWING:
  (1) WHAT PROGRAMS PUBLISHERS ARE LOOKING FOR
  (2) HOW THEY WANT YOU TO SUBMIT YOUR PROGRAM
  (3) HOW MUCH THEY PAY — AND WHEN!
* 100 CATEGORIES — FROM "ACCOUNTS RECEIVABLE" TO "GAMES" TO "VIDEO CONTROL" PROGRAMS
* HOW TO WRITE CLEAR DOCUMENTATION
* DEBUGGING TECHNIQUES

Enclose check or money order for $19.95 (No C.O.D.'s) to:

#63 **IPF Publications**
146 Country Club Lane
Pomona, NY 10970
(914) 354-5585

Order Your Copy Today!

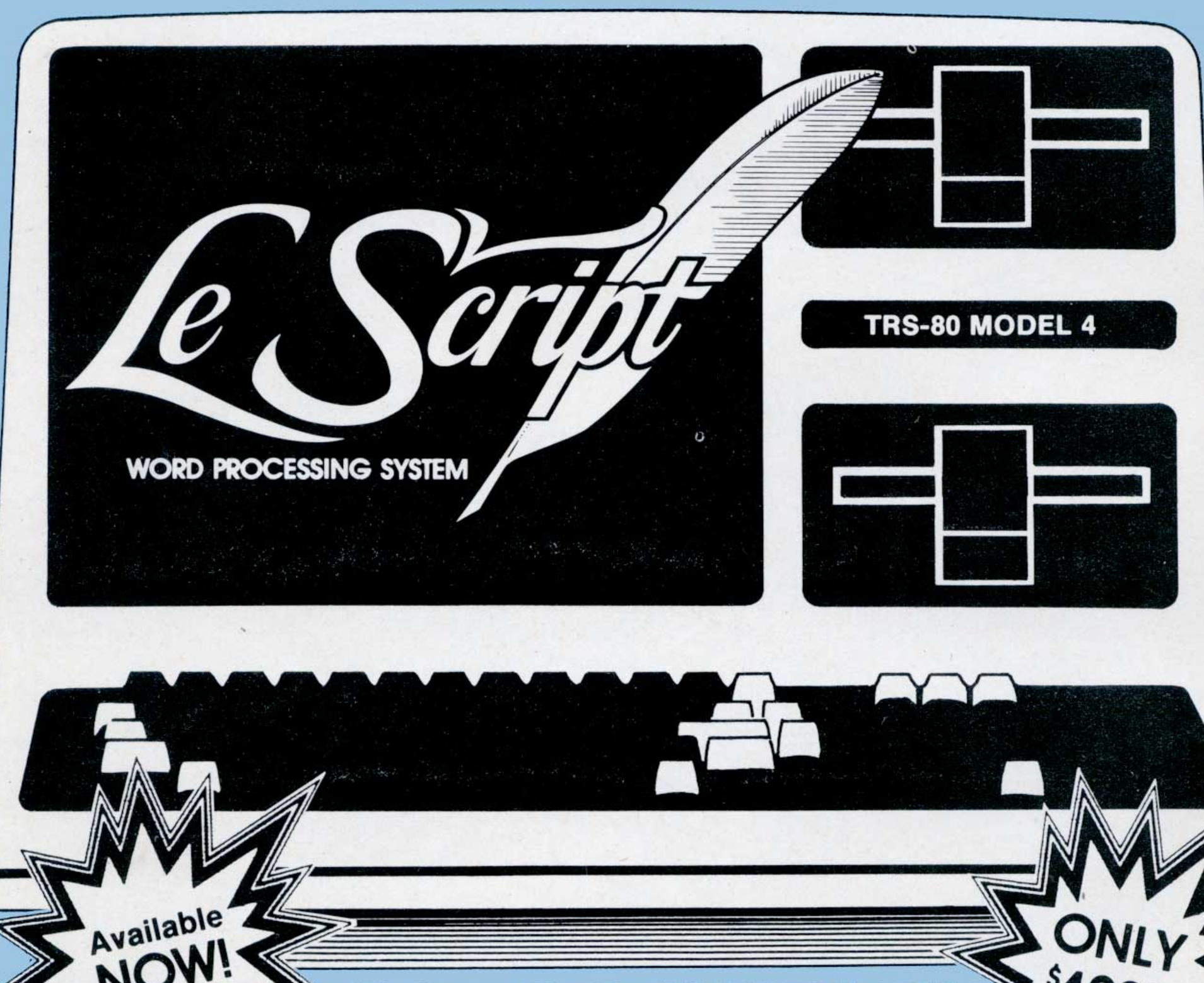
Name .....

Address .....

City..... **State**..... **Zip**.....

See us at
SOFTCON
La Superdome
New Orleans
Tuesday-Thursday
February 21-23, 1984
BOOTH P-2063

TRS-80 MODEL 4 Word Processing Program



Available NOW!

ONLY $129.95

80x24 DISPLAY

- Works with **TRSDOS 6** or any **Model III DOS.**
- **90K TEXT BUFFER** when expanded memory is installed.
- Built-in **FORM LETTER** and **MAIL-MERGE** capabilities.
- Performs **PROPORTIONAL-SPACE** right-margin justification on over 60 different printers (all drivers included **FREE**).
- Same powerful editing features as **ZORLOF II,** with dozens more features added.
- Versions also available to run on **MODEL I, MODEL III, LNW-80, PMC-80,** and **MAX-80** (80x24 display).

FREE

- 30-Day Support to Registered Owners
- Subscription to LeScript Newsletter

LeScript costs $200 less than competing products matching its capabilities.

VISA MasterCard

CALL 1-305-259-9397

ANITEK SOFTWARE PRODUCTS
P.O. BOX 361136
MELBOURNE, FL 32936 #64

FREE SHIPPING WITHIN THE U.S.; OUTSIDE THE U.S. ADD $10.00 FOR SHIPPING; FLORIDA RESIDENTS ADD 5% SALES TAX. ALL ORDERS PREPAID BY CHECK, MONEY ORDER, CREDIT CARD OR C.O.D.

SWIG©

SOFTWARE WRITERS INTERNATIONAL GUILD

THE LARGEST PAID MEMBERSHIP PROGRAMMERS GUILD - OVER 5,000 MEMBERS WORLDWIDE!!

SCHEDULED SWIG ACTIVITIES & MEMBERSHIP BENEFITS

(1) $10,000 PROGRAMMING CONTEST (Members only)

(2) NATIONAL COMPUTER WEEK (May 11- May 20, 1984)

(3) ANNUAL CONFERENCE AND SOFTWARE AWARDS CEREMONY (During National Computer Week)

(4) CONSULTANT REGISTRY (With computer store referral system for customized software)

(5) JOB PLACEMENT SERVICE (Free to individual members, fixed maximum fee to companies)

(6) FREE SEMINARS & MEETINGS LOCALLY

(7) SOFTWARE LIBRARY LENDING & EXCHANGE SERVICE (Professional quality assemblers, utilities, games, etc.)

(8) SOFTWARE LOCATION SERVICE (For companies & individuals-if it exists, **SWIG** will find it. If not, see #9)

(9) SOFTWARE DEVELOPMENT SERVICE (From novice to scientist, **SWIG** members can work on any project-from applications to games to R&D)

(10) LEGAL SERVICE

(11) AGENT (**SWIG** can represent you in sales to software publishers)

(12) 24 HOUR - 7 DAY BULLETIN BOARD SYSTEM (BBS) ACCESSIBLE BY COMPUTER FREE

(13) AND MORE!!!!

MEMBERSHIP APPLICATION FOR SOFTWARE WRITERS INTERNATIONAL GUILD

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

PHONE # ( ) _____

- CLASSIFICATION:

  ☐ NOVICE ☐ BEGINNER TO ADVANCED

  ☐ ADVANCED WITH ON THE JOB EXPERIENCE ☐ RESEARCH/SCIENTIST

- WHAT EQUIPMENT DO YOU HAVE EXPERIENCE WITH &/OR ACCESS TO &/OR PLAN TO BUY?

  ☐ MAINFRAME ☐ MINI ☐ MICRO ☐ DESIGN/R&D

  BRAND NAME(S): ☐ IBM ☐ XEROX ☐ APPLE ☐ TI

  ☐ COMMODORE ☐ RADIO SHACK ☐ ATARI ☐ OSBORNE

  ☐ TIMEX/SINCLAIR ☐ NORTH STAR ☐ HEWLETT PACKARD

  ☐ OTHER _____

- AREAS OF INTEREST:

  ☐ DATA PROCESSING ☐ BUSINESS APPLICATIONS ☐ GRAPHICS

  ☐ LEGAL ☐ VOICE ☐ MEDICAL ☐ APPLIANCE (HOME) CONTROL

  ☐ ROBOTICS ☐ GAMES ☐ MUSIC ☐ R&D ☐ OTHER _____

- MEMBERSHIP ACTIVITIES AND SERVICES OF INTEREST:

  READ THE LIST ON THE LEFT AND CIRCLE THE NUMBERS BELOW THAT APPLY.

  1 2 3 4 5 6 7 8 9 10 11 12

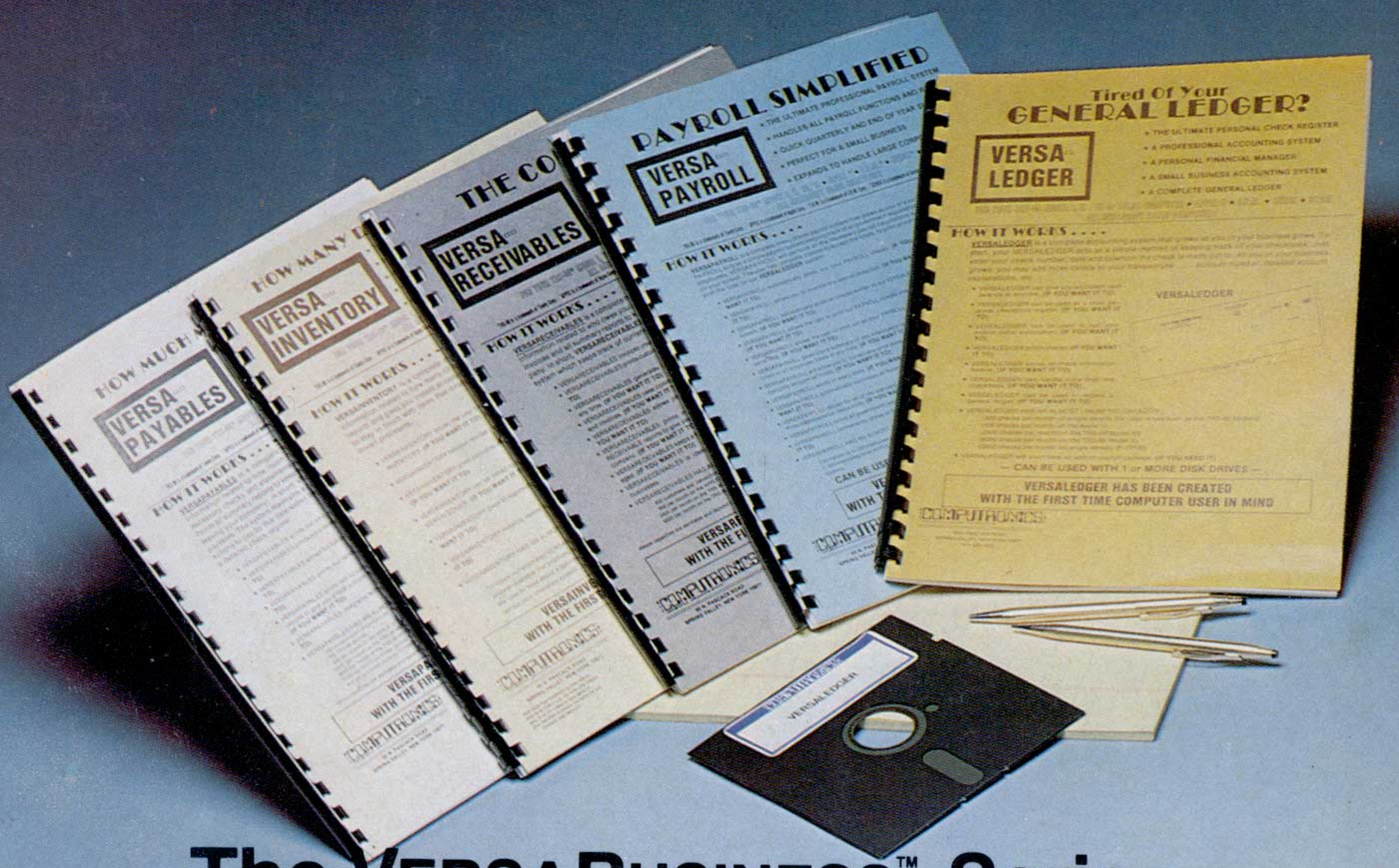
☐ I HAVE ENCLOSED $20 ANNUAL MEMBERSHIP FEE ☐ CK ☐ MO
(MAKE CHECK PAYABLE TO: **SWIG**)

RETURN TO: **SWIG** #65
P.O. BOX 87
STONY POINT, NEW YORK 10980
(914) 354-5585

SWIG© SOFTWARE WRITERS INTERNATIONAL GUILD

Introducing the Most Powerful Business Software Ever!

TRS-80™ (Model I, II, III, or 16) • APPLE™ • IBM™ • OSBORNE™ • CP/M™ • KAYPRO™



The VERSABUSINESS™ Series

Each **VERSABUSINESS** module can be purchased and used independently, or can be linked in any combination to form a complete, coordinated business system.

VERSARECEIVABLES™ $99.95

VERSARECEIVABLES™ is a complete menu-driven accounts receivable, invoicing, and monthly statement-generating system. It keeps track of all information related to who owes you or your company money, and can provide automatic billing for past due accounts. VERSARECEIVABLES™ prints all necessary statements, invoices, and summary reports and can be linked with VERSALEDGER II™ and VERSAINVENTORY™.

VERSAPAYABLES™ $99.95

VERSAPAYABLES™ is designed to keep track of current and aged payables, keeping you in touch with all information regarding how much money your company owes, and to whom. VERSAPAYABLES™ maintains a complete record on each vendor, prints checks, check registers, vouchers, transaction reports, aged payables reports, vendor reports, and more. With VERSAPAYABLES™, you can even let your computer automatically select which vouchers are to be paid.

VERSAPAYROLL™ $99.95

VERSAPAYROLL™ is a powerful and sophisticated, but easy to use payroll system that keeps track of all government-required payroll information. Complete employee records are maintained, and all necessary payroll calculations are performed automatically, with totals displayed on screen for operator approval. A payroll can be run totally, automatically, or the operator can intervene to prevent a check from being printed, or to alter information on it. If desired, totals may be posted to the VERSALEDGER II™ system.

VERSAINVENTORY™ $99.95

VERSAINVENTORY™ is a complete inventory control system that gives you instant access to data on any item. VERSAINVENTORY™ keeps track of all information related to what items are in stock, out of stock, on backorder, etc., stores sales and pricing data, alerts you when an item falls below a preset reorder point, and allows you to enter and print invoices directly or to link with the VERSARECEIVABLES™ system. VERSAINVENTORY™ prints all needed inventory listings, reports of items below reorder point, inventory value reports, period and year-to-date sales reports, price lists, inventory checklists, etc.

VERSALEDGER II™ $149.95

VERSALEDGER II™ is a complete accounting system that grows as your business grows. VERSALEDGER II™ can be used as a simple personal checkbook register, expanded to a small business bookkeeping system or developed into a large corporate general ledger system **without any additional software.**

- VERSALEDGER II™ gives you almost unlimited storage capacity (300 to 10,000 entries per month, depending on the system),
- stores all check and general ledger information forever,
- prints tractor-feed checks,
- handles multiple checkbooks and general ledgers,
- prints 17 customized accounting reports including check registers, balance sheets, income statements, transaction reports, account listings, etc.

VERSALEDGER II™ comes with a professionally-written 160 page manual designed for first-time users. The VERSALEDGER II™ manual will help you become quickly familiar with VERSALEDGER II™, using complete sample data files supplied on diskette and more than 50 pages of sample printouts.

SATISFACTION GUARANTEED!

Every **VERSABUSINESS™** module is guaranteed to outperform all other competitive systems, and at a fraction of their cost. If you are not satisfied with any **VERSABUSINESS™** module, you may return it within 30 days for a refund. Manuals for any **VERSABUSINESS™** module may be purchased for $25 each, credited toward a later purchase of that module.
All CP/M-based Computers must be equipped with Microsoft BASIC (MBASIC or BASIC-80)

To Order:
Write or call Toll-free (800) 431-2818
(N.Y.S. residents call 914-425-1535)

* add $3 for shipping in UPS areas
* add $4 for C.O.D. or non-UPS areas
* add $5 to CANADA or MEXICO
* add proper postage elsewhere

VISA · AMERICAN EXPRESS · MasterCard

DEALER INQUIRIES WELCOME

All prices and specifications subject to change / Delivery subject to availability.

COMPUTRONICS INC

50 N. PASCACK ROAD, SPRING VALLEY, N.Y. 10977