Vol. 1, No. 6

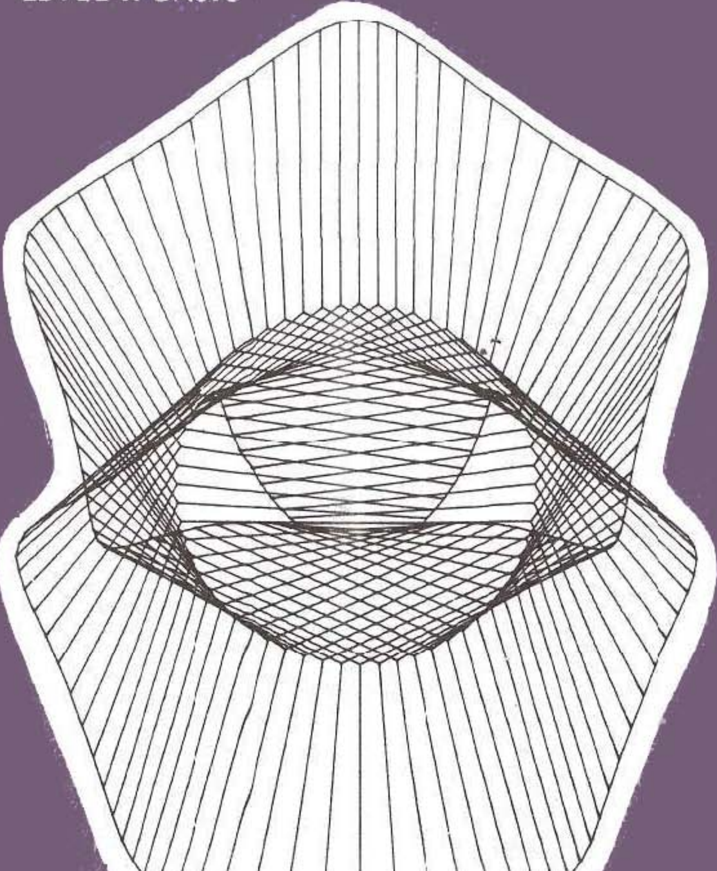
PROG/80

DEDICATED TO THE SERIOUS PROGRAMMER

FEBRUARY 1980 $3.00

Timesharing with The Source

LEVEL II BASIC



PROG/80

FEATURE ARTICLES

MICROSOFT

PROG/80

STAFF

Publisher
Roger W. Robitaille, Sr.

Business Manager
Elizabeth J. Robitaille

Managing Editor
George W. Blank

Editor
Lance Micklus

Software Editor
James Garon

Timesharing Editor
Richard Taylor

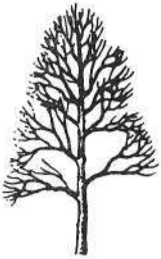
Production
Alison Willis

Subscriptions Manager
Diana Bishop

Customer Service
Bette Keenan

Copy Editor
Freida Day

Accounting
Rita Ellis



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

PLEASE NOTE:

Our new WATS line number is:

1-800-258-1790

It is ONLY to be used for ordering software.

Programmers will not be available on this line.

The WATS line will be open from:

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

ALSO...

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

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

673-5144

New **HOTLINE** Hours:

Tuesday Nights
7-10 p.m.*

673-5144

Your Cooperation Will Be Most Appreciated

---

*Eastern Standard Time

Outgoing Mail

by Lance Micklus

By now, most of you have seen the most awaited movie of the decade, just gone by — **Star Trek / The Motion Picture.** And, if you're really into science fiction, you've also seen **Black Hole**. Plenty has been written elsewhere about the merits of each of these films. So, I won't bother to give you an evaluation here. Rather, I would like to contrast them with other films, as far back as the 1950's, to point out some interesting things.

Generally, science fiction has always reflected the average person's view of technology and the type of future that will result from the existence of that technology.

In the 1950's, the technology of the atom bomb was foremost on the mind of the average person. In many science fictions films, we saw plots based on events that might happen as a result of the bomb. Typically, this was in the form of giant creatures, like ants and spiders. They were so close to the absolute truth. Weren't they really trying to point out that nuclear technology could severely damage to the environment?

Other science fiction movies turned their attention toward invaders from outer space. These movies were the other side of the coin. Can all of our weapons, including the A-bomb and H-bomb, really defend us from all perils?

Noticeably missing were stories about computers. That's odd in a way, because computers existed right after World War II.

The first mention I ever remember of a computer was on the **Superman** TV show. This thing, called a thinking machine, told the crooks at exactly what time to make their getaway after robbing a bank, so all of the traffic lights would be green. I remember asking my mother, later, if such machines really existed. This thinking machine, with blinking 75 watt light bulbs and a belt driven CPU, did make an interesting statement. It said that a machine could solve a problem too complicated for humans (in this case, figuring out the perfect travel route which avoids all red lights).

Computers really didn't show up much until the 1960's. The space program made us very aware of the technology, including computers which were able to solve the complicated problems of space travel.

Most science fiction from that period came from TV shows such as **Lost In Space** and **Star Trek.** Both shows had space ships which used computers for navigational aids.

The extremely rapid advances in computer technology raised some new questions by the end of the 1960's. Specifically, could computers become smarter than the people who had made them? In the movies **2001** and **Colossus — The Forbin Project,** they did. And because of the powers given to them, they became gods; for they had, and used, the power of life and death. This fear wasn't totally unfounded. After all, by the late 1960's computers were computing and printing our pay checks and sending us our bills.

Now, we come to the end of the 1970's. For the first time we have micro-computers. Not only do they do what computers do, they can hear (Voxbox), speak (Voice Synthesizer), see (slow scan television), and even move (The Turtle). But where are they in today's science fiction films?

Some of their names are household words - R2D2, C3PO **(Star Wars),** and Vincent **(Black Hole).** They are the micro-computers of the future. And you know what else I think? Their great, great, great grandpa was the Radio Shack TRS-80.

CROSS REFERENCE UTILITIES

from
Lance Micklus

•BASIC Cross Reference
•XREF•RX ('REX')•RENUMX

Displays all line number references and variables in a Level II or Disk Basic Program. Four options are available: display all line references and variables, line numbers only, variables only, or print all references to a single line number or variable.

Versions available:

RX (Disk Version) 32K,1 Disk Systems, on Disk	$24.95
XREF (Level II Version) Level II, 16K, on cassette	$19.95
RENUMX (XREF with renumber and rescue after reset) Level II, 16K, on cassette,	$24.95
RENUMBER (separately)	$7.95

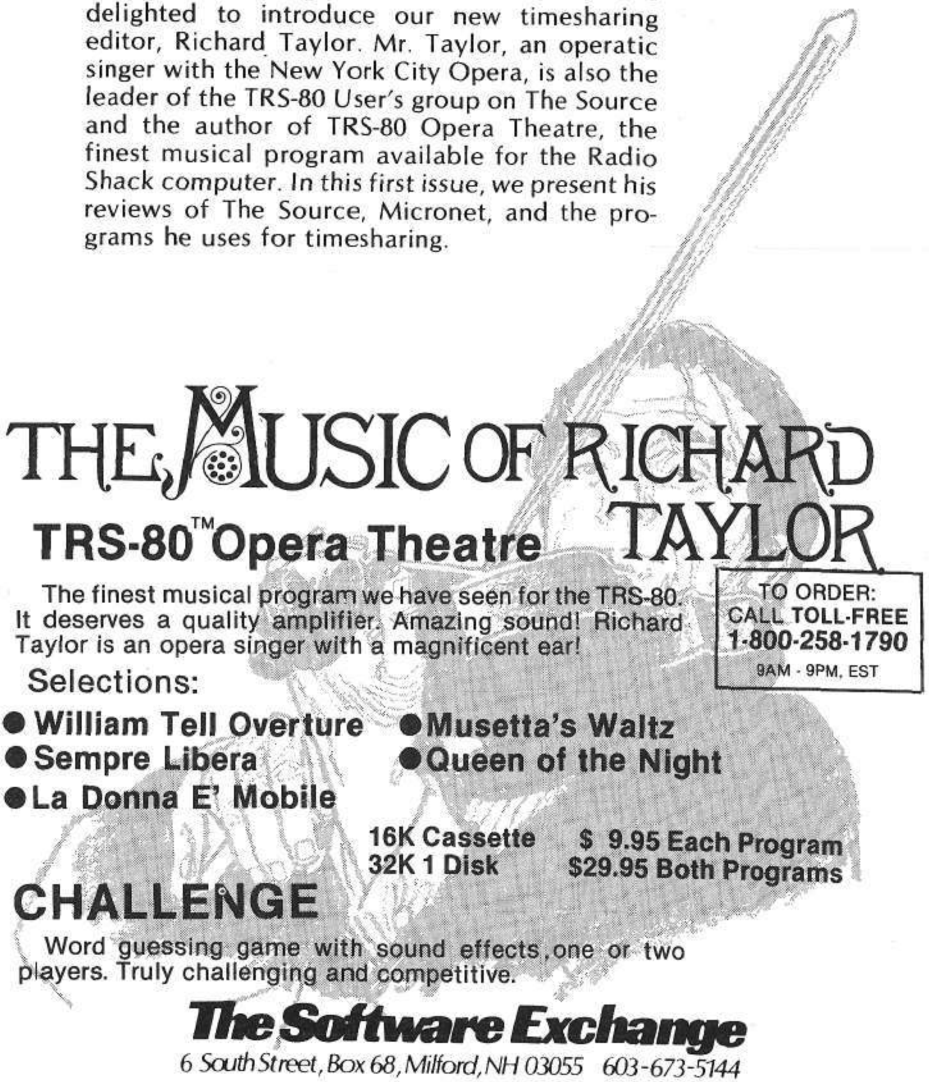
TSE

TO ORDER:
CALL TOLL-FREE
1-800-258-1790
9AM - 9PM, EST

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

TIME SHARING SECTION

The next few pages are the beginning of a regular timesharing column in PROG/80. We are delighted to introduce our new timesharing editor, Richard Taylor. Mr. Taylor, an operatic singer with the New York City Opera, is also the leader of the TRS-80 User's group on The Source and the author of TRS-80 Opera Theatre, the finest musical program available for the Radio Shack computer. *In this first issue, we present his* reviews of The Source, Micronet, and the programs he uses for timesharing.



MAY THE SOURCE BE WITH YOU. .. OR WILL IT BE MICRONET?

by Richard Taylor and Mark Kirmayer

*A trademark of Lance Micklus, Inc.

If you are thinking of purchasing a MODEM or if you own one already, you undoubtedly have been reading the ads for MICRONET and THE SOURCE. They are offering us "Micros" a chance to timeshare on one of the "Big Guys". So the question is: Do you need THE SOURCE or do you need MICRONET? I have used both and perhaps what I have learned will be able to help you answer that question.

If THE SOURCE's one-time fee of $100 to open your account seems excessive (as it did at first to me), you may be interested to know that it is clear to me, after just a few weeks on THE SOURCE that the difference in the hourly rate ($2.75 vs $5.00) and the greater abundance of local phone numbers, makes THE SOURCE's price the better buy in the not too long run. And if you've already calculated the differences and are wondering how I was able to use forty hours so quickly, take this as a testament to the many excellent features on the system. Accessing the different data bases, sending 'electronic mail' to subscribers across the country and receiving your 'mail' when you log-on, and 'chatting' directly with fascinating people from coast to coast, (just a few of the possibilities which will be explored later) are but a few of the addictive and rewarding features of the system, and at $2.75 an hour it is indeed a bargain.

MICRONET lists only 25 phone numbers for the entire U.S., and only if one of these is in your local calling area can you avail yourself of MICRONET at the published pripe of $5.00 an hour. If you do not happen to live in one of those 22 cities you have two options. You may call the nearest long distance number or use the TYMNET Telephone Network (about 200 locations) to access MICRONET. Using TYMNET will add $4 an hour to your charge. (The SOURCE includes both TYMNET and TELENET at no extra charge). I live in a highly-populated region of the New York Metropolitan area, and while THE SOURCE had a number within my local calling area, MICRONET did not. I was able to find a local number which required the use of TYMNET (for a $4/hour surcharge), and a comparison of the toll-charges versus the surcharge proved that it was less expensive to call out of my area, but still the cheapest way I could use the MICRONET system was at a cost of $8.06 an hour, versus the advertised $5.

One thing is immediately obvious: time sharing on one of these systems is to the idea of personal computing what the jet engine was to air travel. Both inventions open up new and untold worlds of exploration, learning, and just plain having a good time.

Each system has a slightly different log-on procedure, but both systems have an update feature to let you know what has been added to the system since you last used it. As a first-time user you will find yourself going through file after file of information that you will need to know to use the system effectively. The SOURCE's file of system command and other function updates, takes 55 minutes to print out. This can be done in installments.

For just this purpose, and for reasons to follow, a printer is an invaluable tool for the timesharer, for at the very least, it leaves you with permanent hardcopy when you sign off. Perhaps more important is a good terminal program. I use ST8OD, which allows me to open a memory buffer and save out to disk any amount of information I want. The ideal setup would include a "Smart" terminal program and a fast printer.

For those not owning a copy of a terminal program which allows for the transferring of disk files, MICRONET downloads free of charge to its subscribers its own terminal program (MICRONET EXECUTIVE), which some people use to access THE SOURCE. And in the discussion of MICRONET's features, it became clear again why a printer enhances the value of the systems to you.

While THE SOURCE's best feature (and it is a great one at that) is the ability to access different data bases, such as the New York Times Consumer Data Base, the transmissions of U.P.I., lectures on Astrology and ESP, and impressively, the ability to book confirmed airline, hotel and car-rental reservations, MICRONET's strength lies in another area altogether. Through MICRONET you can purchase software for the TRS-80. The system will sell you the programs you desire, and while they charge you the price of the program, they do not, in a sense of fair play, charge you for the transmission time. Thus you get software sent right to your terminal, with no-fuss shopping, no wait for the mails, and no extra service charge. MICRONET will also download entire manuals to your terminal, should you so desire, quoting before you begin the "connect time charges" for the transmission, versus the cost of having the manuals or documentation mailed to you.

To get into a discussion of each of the different services offered on each system would take no end of time, for while MICRONET, for example, sells TRS-80 software to you, THE SOURCE sells just about anything else to you, through an electronic shop-at-home feature, offering excellent prices on cameras, stereos, etc. From your terminal you make your purchases, which is charged to a major credit card, and then sent to you.

It should be obvious by now that with either system you will spend many hours at your terminal. Response times vary according to the time of day. I have had waits of 6 minutes of THE SOURCE at 6:15 PM, and instant response at 1 AM. I have also been disconnected in the middle of entering files, and I had one instance each where both systems were too busy to accept my log-on. System crashes do occur and in my few weeks of experience, both systems have gone "down" for several hours. THE SOURCE has a toll-free number for complaints, and you do get results.

Both systems offer a number of similar features, such as games, like 'Adventure', 'Startrek', and and both systems offer a large more, selection of programming languages. The Source also offers (or will be offering) business programs, (General Ledger, Accounts Receivable, etc.), and educational programs, from grade school level up to and including the Dartmouth College library of courses to take right at your own terminal.

One last feature, which is available on THE SOURCE, is an elaborate MAIL system. Subscribers are asked to list themselves and their interests in the USER DIRECTORY, so that other users can find and communicate with them. I wound up forming a TRS-80 users group through THE SOURCE's electronic 'mail'. Using a simple command (DISEARCH) I was able to get a list of all users who included TRS-80 as an interest. I took these different account numbers and entered them into a file called "Group". Then, using the command MAIL (GROUP), I was able to write the invitation only once and have it sent automatically to everyone listed in the file GROUP. We have a constant exchange of information, and it has been possible to send entire BASIC programs through the 'mail' to the group. The abilities of this system are astounding and would take too long to describe. Trust me, it's addictive.

Chatting on THE SOURCE is just that. Any two users can "chat" directly while on the system, the important point being that this cross-country "chatting" and the exchange of huge amounts of data via the "Mail System" is at the rate of $2.75 an hour, cheaper than the rates of the phone company, and not only does MICRONET have nothing to match this feature, if it did it would be at a rate of at least $5.00 an hour.

It undoubtedly seems that I have a strong preference for THE SOURCE, something I didn't realize myself until I undertook to write this comparison. I pondered for a while that it might be that I like THE SOURCE more because I have spent more time on it, which I must admit. However, I think the reason I've spent more time on it is the many features it has which MICRONET does not (at least at present).

As a subscriber to both systems, I think if I were forced at solder-gunpoint to suggest one or the other it would be THE SOURCE. But, if the one time charge of $100.00 makes you hesitant, then by all means send in your $9.00 to MICRONET, and start timesharing. You pay only for the time you use, and I'm willing to bet that once you step up to such a system, you'll become involved to the point where you see your way clear to subscribe to THE SOURCE also. Bear in mind that both companies are constantly adding features and improving the features already there.

One piece of advice though, needs no qualification whatsoever -- TIMESHARE!! (I don't think you'll disagree with me, but bear in mind that if it turns out that you do, we can battle it out some evening on a one-to-one basis, in the wonderful world of timesharing.)

	THE SOURCE	**MICRONET**
1 time fee to open account	$100.00	$9.00
hourly charge	2.75 (6PM to 7AM +weekends) 15.00 (other times)	5.00 (6PM to 5AM) +weekends) (not available)
disk storage	3.3 cents a day per 2048 char	64K FREE
Data bases	Many and varied (UPI, TRAVEL, NY TIMES)	NO DATA BASES
Min. monthly charge	5.00	1.00
Local phone #?	Very good chance	WATCH OUT!!

'THE TIME SHARER'S PARADISE' —ST80D + ELECTRIC PENCIL + VTOS 3.0

By Richard Taylor

I think ST80D is the best single terminal program available for the TRS-80. If you have this program, and you have VTOS 3.0 and the ELCTRIC PENCIL, you can combine them to make a very powerful communications package.

All by itself, ST80D lets you control the opening and closing of a 'memory buffer' for the saving or sending of data. ST80D also lets you control your printer, zero your-system clock, on/off auto line feeds, reset your RS232 switches, send lower case, display lowercase, send out an auto logon message, define 10 programmable keys, and more!

With the addition of ELECTRIC PENCIL you can write messages at your leisure and load them into ST80D when you're ready to transmit them. PENCIL saves its data in ASCII format and is perfectly compatible with ST80D. Just be sure they are both on the same vintage DOS. Sorry but NEWDOS will not work. **(Ed. note: ST80D version 1.4 will work with NEWDOS.)**

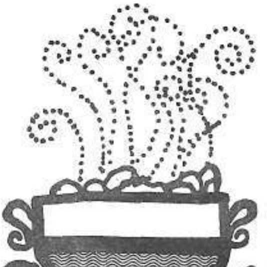
If PENCIL creates files usable with ST80D then it would seem that ST80D must create files readable by PENCIL. But no. It seems PENCIL is a little more fussy about what it loads than is ST80D. PENCIL will not read in the files created by ST80D even if you give them the extension /PCL. Lance Micklus has given us NUMBER/BAS which reads in ST80D files and puts line numbers on them so that we can edit out any mistakes or garbage that might have crept in before we closed that memory buffer.

Alas, NUMBER/BAS takes quite a bit of time to use and needs a lot of room, and if the program or data line exceeds about 240 bytes the whole thing will crash. There is an easier way, a much easier way, which has turned my use of the ST80D memory buffer into a constant occurrence.

Many of you have a copy of VTOS 3.0 even though you may not use it. Here's your chance to justify that $39 or $49 purchase. Using XFER (if you have one drive) put Electric Pencil on your 3.0. Then type PATCH PENCIL PENCIL. 3.0 will tell you what it is doing in its semi-obscure fashion and you will be left with a Patched version of ELECTRIC PENCIL. This is a one time operation which you don't have to repeat. Now type XFER ST80D/CMD (or COPY if you have 2 drives) and get ST80D on your 3.0 disk. You are now ready. When you call for a directory in PENCIL it will give you a complete directory of the disk, not just PENCIL files. Anything saved in ASCII can now be read in PENCIL. Try those documentation files on 3.0. See, they were PENCIL Files!

Note that if you are using lowercase PENCIL your Control Key will not function. Use Shift Down Arrow instead.

Now create a file with ST80D from your memory buffer by, for example, calling up your local FORUM 80 and letting those messages pour into your buffer. When you save the file be sure to give it an extension!! PENCIL files need extensions or PENCIL will look for a /PCL extension. I use SOR for THE SOURCE and FOR for FORUM 80 etc. This is surely the best of all possible worlds for us MODEM people. The only problem I have ever had occurred with a BASIC program that had a lot of "string packing". The transmission was fine but it would not load into anything. No answer yet for this problem.



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

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

Buy the

Z80 Software Gourmet Guide & Cookbook

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

New from SCELBI Publications

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

$14.95 plus $1.00 shipping

MORE BOOKS FROM TSE

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

TRS-80 DISK AND OTHER MYSTERIES

by Harvard C. Pennington

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

LITTLE BOOK OF BASIC STYLE

John M. Nevison

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

$5.95 plus $1.00 shipping and handling

TRS-80 ASSEMBLY LANGUAGE PROGRAMMING

William Barden, Jr.

Covers Z-80, an Introduction to Assembly Language, the Radio Shack Shack Editor/Assembler and T-Bug, and debugging methods; also explains how to move data, the use of arithmetic, compare, logic, and bit operations, shifts, strings, tables, input and output, and 12 commonly used subroutines. Well indexed and illustrated.

$3.95 plus $1.00 shipping and handling

TRS-80 INTERFACING

by Jonathan A. Titus.

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

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

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

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

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

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

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

TRS-80* FORTRAN SOFTWARE PACKAGES

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

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

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

Plus $5.00 shipping
(separately $2.50 each)

TSE

The Software Exchange

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

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

INPUT

A COLUMN OF YOUR LETTERS

Gentlemen,

As an addendum to Lance Micklus' article "DOS" in your Sept. issue where he describes a method of speeding up access time to the disk drive by poking changes into Ram; permanent changes can be effected by modifying the DOS itself as follows:

Using the Newdos + program, "Superzap", make these changes in Sys0:

NEWDOS Track 0 Sector 7
Change relative byte 0F from 0B to 02
Track 0 Sector 9
Change relative byte 0F from 1B to 12

TRS-DOS 2.2 Track 0 Sector 7
Change relative byte 52 from 1F to 12
Change relative byte C5 from 0B to 02.

These changes speed access time to 20 millesecond and my Radio Shack drives work perfectly. For those who have Pertec drives and want to try a 10 millisecond access time (my drives won't function at that speed), use 01 and 11 respectively in place of 02 and 12 in the above changes.

Sincerely yours,
N.W. Duncan
Stone Mountain, GA.

Editor:

In his software review "Blaiseing along in Pascal", Robert P. Johnson says that CIE has left off of People's Pascal II the big compilers that should be on side two, and instead given a second copy of the small compiler.

Not so. We have left nothing out. We have combined what was supposed to have been side one and two on one side, and given a backup copy on the other. Five programs/five programs, instead of three/two.

I'm going to have to examine our promotion, because while we do supply 16 and 32K compilers, we do not have 48K People's Pascal. UCSD does!

While I must take exception to the "missing side", I wish to thank PROG/80 for presenting an important review fairly, and in an interesting way.

Thanks ever so much,
Bill McLaughlin
Computer Information Exchange
San Luis Rey, CA.

The Best Has Gotten Better!

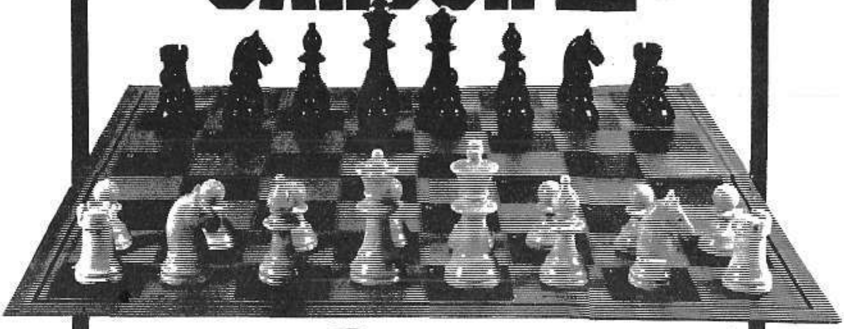
Sargon, the program that came in first in the Creative Computing Microcomputer Chess Tournament, has become Sargon II - **$29.95**

- A vastly improved game
- Faster response time
- New level 0 for beginners
- Easier to pre-set board
- Hint mode - What does the computer suggest

Sargon II is the program that took on the maxi-computers in the West Coast tournament, and finished in the money! More thinking power than you ever expected in a TRS-80.

Now in stock for immediate delivery!

SARGON II



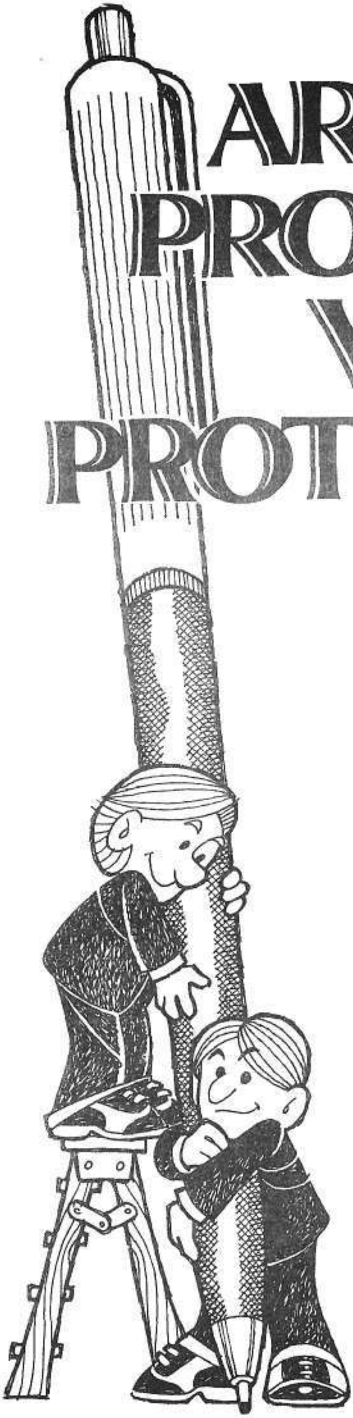
TSE

TO ORDER:
CALL **TOLL-FREE**
1-800-258-1790
9AM - 9PM, EST

The Software Exchange

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

ARE YOUR PROGRAMS WRITE-PROTECTED?



by Frederick B. Hagar

You've just completed a tedious week of writing the ultimate game on your TRS-80 and now it's time for the big test. The wife has been trying to make the couple next door more receptive with coffee and home made doughnuts. Now you've filled your 16K with the latest amusement or entertainment. The moment of truth arrives and the computer displays:

```
Level of Difficulty—Please enter
          1, 2, 3, or 4
```

After a victorious first game, your neighbor, looking for a little harder version, presses 4 (ENTER).

OOPS — how do you tell him that the program wasn't ready to play again and that he's just deleted line number 4? You can load from tape or disk again (and admit that the TRS-80 actually possesses human frailities), or you could have write-protected your program. The procedure is very simple, requiring only the addition of two lines.

To help you understand how to write-protect any program, you should know that the TRS-80 stores a line number as two bytes, the low order byte followed by the high order byte of the line number. For example the line number 1 is stored as (01) (00). If you poke 255 (FF HEX) into the high order byte and list the program, you'll find a new first line number; 65281. You'll also find that it's impossible to delete, edit, or list individual lines by the conventional method. This is because the TRS-80 reads line 65281 and expects the next line number to be the larger. As there are no larger line numbers, all lower numbered lines are now treated as sub—statements, separated by colons.

The only problem created here stems from the fact that the TRS-80 will not branch to a colon. For this reason, GOTO, GOSUB, and THEN statements will not execute without one more change. Just before the first branch statement of your program, poke the high order byte of the first line number with 0. This will unlock the program and allow execution of all branch statements. Write protection should be reactivated after the last branch statement.

```
1 REM THIS LINE # WILL BE 65281 AFTER FIRST RUN
2 POKE 27175,0 :REM UNLOCKS PROGRAM **
3 '****************************************
4 '          THIS
5 '           IS
6 '          YOUR
7 '         PROGRAM
8 '****************************************
9 POKE 27175,255:REM THIS LINE LOCKS THE PROGRAM
10 END
```

The addresses of your first line number will vary according to memory size, whether or not you have a disk, what DOS you're using, etc.

If you have a disk system you can debug to locate line number one. But, if you haven't added your disk yet, the following program will locate the high order byte of your first line number.

```
10  A$="HELLO":CLS
20  L=PEEK(VARPTR(A$)+1):M=PEEK(VARPTR(A$)+2)
30  X(1)=FIX(M/16):X(2)=M-(X(1)*16)
40  X(3)=FIX(L/16):X(4)=L-(X(3)*16)
50  X=(X(1)*4096)+(X(2)*256)+(X(3)*16)+(X(4))-5
60  PRINTX
```

Delete the -5 in line 50 and this program will find any string in memory. I'm sure you can find other uses for this.

So, have fun and happy computing!

WHAT'S IN A NAME?

by James Garon

Microsoft, the authors of Level II Basic, have thoughtfully (or accidentally?) provided us with the word "NAME" — without having given it any meaning. We can give the word NAME any meaning we wish, and in so doing, we will be creating a second (Level II) or eleventh (DISK Basic) USR subroutine.

As an illustration, let us teach the computer that NAME means:

> "Reverse all graphics on the screen, but leave all non-graphic information intact."

First we write a machine-language program to do the job:

```
HEX       MNEMONIC       COMMENTS
D9        EXX            Save whatever BASIC was doing
21 FF 3F  LD HL,3FFF     Pointer to last byte of screen
06 3C     LD B,3C        Used to see if done
7E        LD A,(HL)<--   Get byte from screen
FE 80     CP 80          Is it a graphics byte?
38 04     JR NC --       Jump if not
2F        CPL            Change all bits
F6 80     OR 80          Turn graphic bit back on
77        LD (HL),A      Send reversed character back to screen
2B        DEC HL<--      Point to next screen location
7C        LD A,H
B8        CP B           See if we're still on the screen
30 F2     JR NC --       If so, continue
D9        EXX            Restore BASIC's stuff
C9        RET            Back to BASIC
```

Keep in mind when writing your own routines, that all jumps must be relative, and no bytes of the routine may be zero (00).

Next, we write a DATAstatement which will contain the decimal values of the 22 bytes of our routine:

```
10 DATA 217,33,255,63,6,60,126,245,128,56,4,47,246,128,119,43,124,
     184,48,242,217,201
```

Then a "dummy" string to receive the bytes:

```
20 A$="ANY 22 CHARACTERS ////"
```

And a routine to place the DATA bytes into A$:

```
30 I=VARPTR(A$):J=PEEK(I+1)+256*PEEK(I+2)
40 FOR K=J TO J+21:READ X: POKE K,X:NEXT
```

Now we tell NAME where to find the routine in A$:

```
50 POKE 16783,PEEK(I+1):POKE 16784,PEEK(I+2)
```

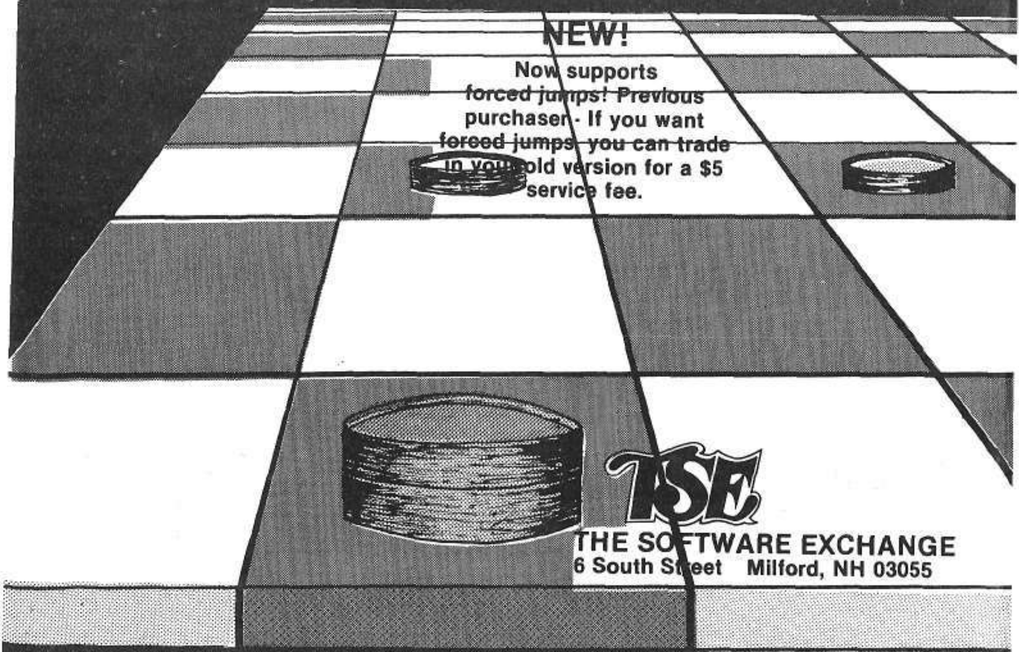
Let's add a couple of lines to test our new NAME:

```
60 CLS:FOR I=1 TO 5
70 PRINT@137=I,CHR$(128)CHR$(140)CHR$(160)CHR$(144)CHR$(140)CHR$(128)
               CHR$(26)STRING$(6,24)
               CHR$(128)CHR$(143)STRING$(2,140)CHR$(143)CHR$(128):NEXT
   80 FOR I=1 TO 200:NEXT:NAME:GOTO 70
```

Once the program is working, you may DELETE lines 10, 30, 40, 60, 70 and 80. Now whenever this program is executed (lines 20 & 50), the word NAME will reverse all graphics.

THE MEAN CHECKERS MACHINE

by Lance Micklus



NEW!

Now supports forced jumps! Previous purchasers - If you want forced jumps, you can trade in your old version for a $5 service fee.

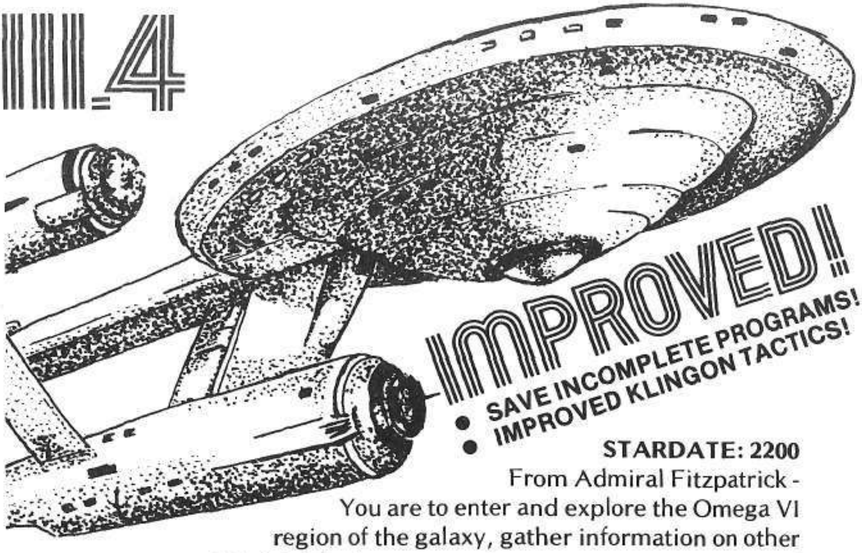
THE SOFTWARE EXCHANGE
6 South Street Milford, NH 03055

Our resident wizard has done it again! Designed in FORTRAN, run as machine language, this program turns your TRS-80 into an unbelievably wicked checker player! Four levels of play; at the most difficult, the machine may take ten minutes per move, as it attempts to assess all possibilities. MEAN CHECKERS MACHINE is to checkers what SARGON is to chess. (Level 4 is an exercise in humility!)

Level II, 16K	$19.95
Disk	$24.95

STAR TREK

III.4



IMPROVED!!

- SAVE INCOMPLETE PROGRAMS!
- IMPROVED KLINGON TACTICS!

STARDATE: 2200

From Admiral Fitzpatrick -

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

It is also patrolled by Klingon battle cruisers, so look before you leap.

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

TO ORDER:
CALL **TOLL-FREE**
1-800-258-1790
9AM - 9PM, EST

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

Specs: Star Trek III.4

Play Board: 8 by 8 by 3 quadrants

Weapons Systems: Phasers and Photon Torpedoes

Power Systems: Warp and Impulse

Computer Systems: Science and Ship's computer

Sensors: Long and Short Range

Reports: Damage Control and Status

Play Elements: 20 Klingon battle cruisers, 100 stars and planets, black holes, pulsars

Review
by Teri Li

SUPERZAP

from APPARAT INC.

SUPERZAP! Its very name promises results. This disk oriented BASIC program allows you to assault your diskettes, in one sector chunks, with all the normal protection systems off. You can examine and modify your diskette in any way your heart desires (well, within the constraints of reality). Don't like the way TRS-DOS appears at the top of the screen everytime you do a DIR command? Then change it to read MY DOS! Or did you just kill a file that you need now desperately? Then go into the directory and restore it (providing you haven't written another file on top of it). And finally, have you ever wanted to see just what it was that was actually written on the diskette? Then check it out!

Yes, Superzap can do all of these things, and much more.

Superzap was written for the TRS-80 by Apparat, the people who brought you NEWDOS, the R/S DOS that worked!

Superzap provides the serious disk programmer with a fast and reliable method of handling the information stored on a diskette. In its data dump mode the information is displayed in one sector (256 byte) chunks. Sixteen hexadecimal numbers, and their ASCII equivalents, are displayed on each line of the video. The leftmost column on the video, a set of six digits, is actually the relative address of the information in that sector.

The first digit is the relative number of the drive that the sector you are looking at came from, the next two digits are the track number (in hex), while the fourth digit is the relative number of the sector on that track. The last two digits in that column are the relative position (hex again) of the first byte in each 16 byte row of information being displayed on the video.

You can easily alter the informatin on the video and put it back onto the disk in its new form, or even just terminate this mode without affecting the diskette.

Even finding your disk sector is simple. You just specify the relative Disk Drive, track, and sector numbers wanted. In the newer editions of Superzap you can also use the filespecs of the file in which you're interested.

But this is not all that Superzap does. It can

Zero Disk Sectors — and oh boy, does it. Don't play around with this one unless you really mean business.

Copy Disk Sectors — as few or as many as you want.

Copy Disk Data — this will allow you to copy a string of data up to 65536 bytes long from any position to any other postion on the diskette.

Verify Disk Sectors — very handy if you think part of your diskette may have been 'cooked'.

Disk Backup — this one alone would be worth the price of the program. It is very slow, taking anywhere from 5 minutes on up to complete its functions (as opposed to R/S's 1 and 1/4 minute BACKUP). But it makes up for this with its tenacity. In fact it never gives up unless you do.

The first thing to do when entering this command is to go make yourself a cup of coffee, as you're going to be waiting a while. As soon as Superzap encounters a problem, it really goes to work. You can actually hear the drive going over and over that sector. After trying for a few dozen reads, it will tell you what the cause of its problem is, usually a PARITY ERROR, and ask you if it should; a) go back to work; b) skip this sector; or c) give up and return to command mode. As long as you're willing to sit there and punch the retry key, it'll keep trying.

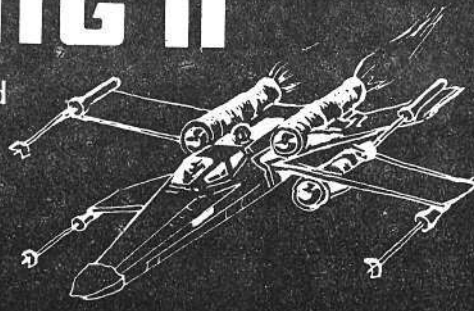
As you can see, I found Superzap to be a very useful program, second only to the Electric Pencil.

This is not to say that Superzap cannot be improved. It can. For example, I would like to see the BACKUP command modified to display the information that it has managed to recover when it encounters a bad sector and give me the option of copying that sector onto the new diskette (even though it is flawed, maybe I can fix it using the rest of Superzap). As it stands now, I have to skip that entire sector and try to rebuild it completely. A very depressing prospect.

Overall, I would strongly suggest that any serious disk user purchase this program for his/her library. After all, if you can recover even just one bad diskette using Superzap, then it will have paid for itself.

X-WING II

by Chris Freund



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

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

Your Mission, Destroy the Death Star!

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

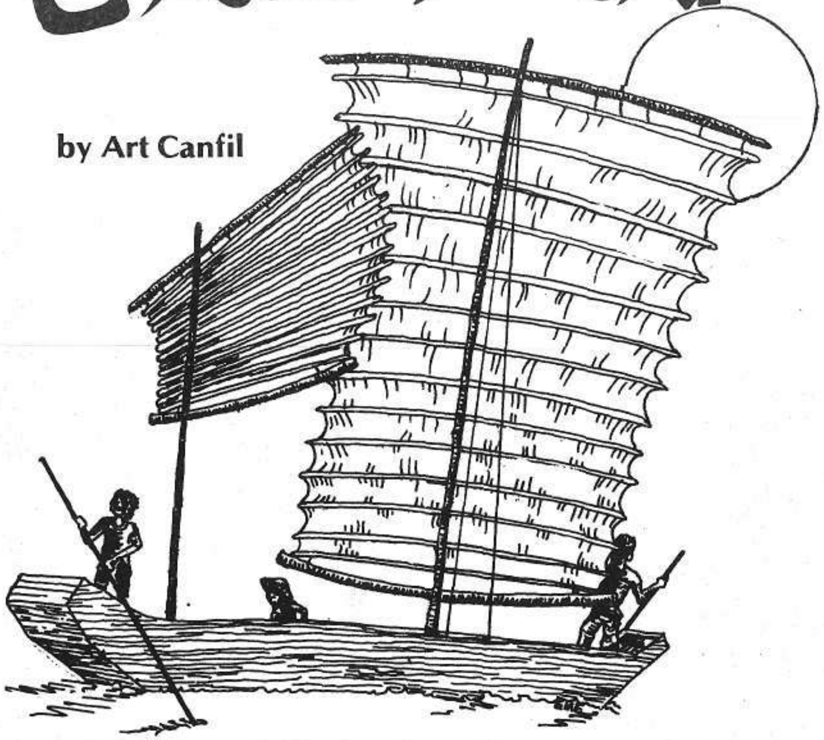
Level II, 16K — $9.95

The Software Exchange

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

TAI PAN

by Art Canfil



A Single-player Game
Based on the China Trade of the 1800's

Ply arms and opium in exotic oriental ports, dodging pirates, cutthroats, usurious moneylenders and tropical storms. Should you arm your ship to stand and fight, or keep it light to run swiftly? Will the price of opium still be high in Hong Kong when you return laden with the stuff from Batavia? **Warning:** If you get a polite message from Brother Wu, the moneylender or Li Yuen, the pirate, you'd better pay attention! If you outwit your enemies, you win by piling up a million dollars in profits.

Level II, 16K — $9.95

TSE

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

ATTENTION ALL WAR GAMERS

The war games you have come to love have come to the TRS-80!

- River ●
- Hexagonal Grid ●
- Terrain Obstacles ●
- Armor, Infantry, and Engineers ●
- Computer determined combat results ●

This Two Player game is an elementary war game (Kriegspiel is German for "war game") giving a battle between two countries located in opposite corners of the screen. The object is to capture the enemy capital. During play, each piece flashes in turn, displaying the number of moves left. You can move it, wait until you have moved other pieces, skip it, or even go directly to combat after moving only some of your pieces. You must stop and do battle whenever you are adjacent to an enemy piece. In battle, you can be eliminated, the other player can be eliminated, an exchange can take place, or a retreat may be required. You do the brainwork, General; the computer carries out your orders. Strategy and fate determine the battle!

$7.95 Level II Cassette

Kriegspiel

TSE THE SOFTWARE EXCHANGE
6 South Street
Milford, NH
03055

OLD BUSINESS

by
Lance Micklus

In the September issue of PROG 80, I showed you yet another way to make your TRS-80 display upper and lower case letters. At the end of the article was jokingly added that if anybody tries this modification, I'd like to hear how it worked out, on the back of a post card with pictures of beach girls on the front.

Well, I did get a reply. The gentleman actually sent a letter, but included three post cards of beach girls **(TOPLESS!)**. I'm sure he died of laughter when he got my letter back. Nowhere did I ever mention the beach girls. But I sure hinted at it.

Also, in that issue, I gave you some POKES you could use to speed up the access time of your disk drives. Some interesting problems developed from this.

Ever since I began running Advanced Personal Finance, I noticed that when the Budget program was verifying the output files, it would sometimes read a sector, home the head, then read the sector again, and go on. This was the reason I put the display of the track and sector in the program. It turned out that the disk was going to the wrong track, would home the head, and then come out to the correct track. It didn't hurt anything, so I never really tried to find out why.

But, when I tried running APF with fast access speeds, it not only did all of the above, the **verify** then failed. I quickly discovered that BASIC was reading the data back with one bad byte whenever this happened. The data files were, indeed, O.K.

A short time later I tried APF under VTOS 3.0. Now something new happened. BASIC would get stuck reading the same sector over and over again . . . forever. It turned out other people were running into this problem using VTOS 3.0 and it was driving Randy Cook crazy. I sent Randy a copy of the file so he could reproduce the error, and by the end of November he had it all tracked down.

Disk BASIC looks ahead one byte when it reads a sequential file. When it doesn't find what it's looking for, it back spaces one byte. The problem occurs when the end of a record crosses a sector boundary. Disk BASIC does not backspace correctly. All of these weird effects result from this error.

I do not have a patch for Disk BASIC to cure this problem, but I thought I should pass this information along to you. Until this bug can be fixed, beware that under TRSDOS 2.2 and 2.3, Disk BASIC may read bad bytes from sequential files at access speeds greater than 40 milliseconds.

BEE WARY

(More Magic from Leo Christopherson)

This fast paced real time action game is a contest between a Bee operated by the player and a Spider operated by the computer. The Bee tries to sting the Spider in a vulnerable spot while the Spider tries to jump up and swallow the Bee. Fast paced animation, sound, and real time action. Machine language subroutines, but loads as Level II for easy operation. $14.95

The Software Exchange

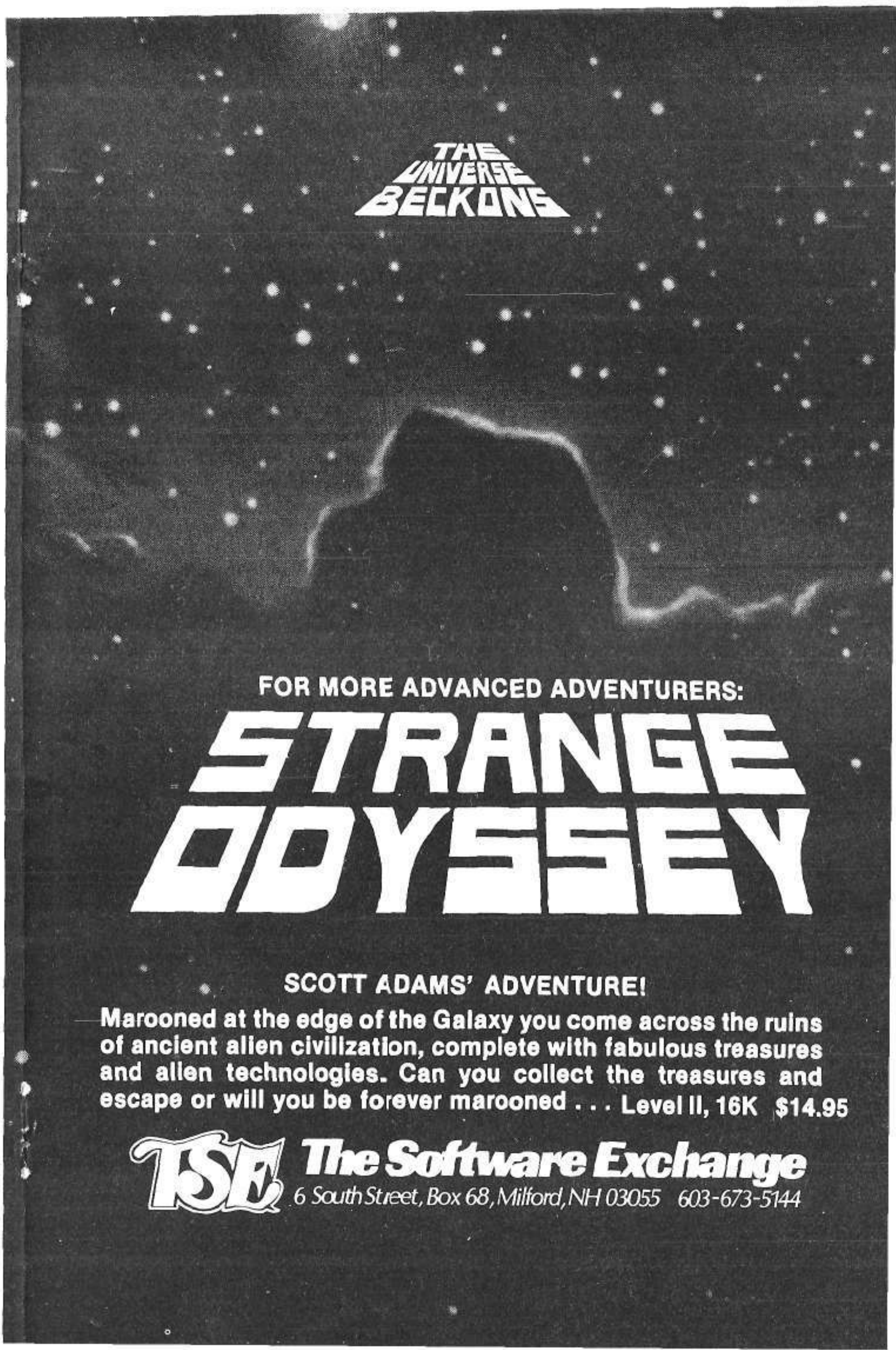
6 South Street, Box 68, Milford, NH 03055 603-673-5144

PR Dogfight

by David Bohlke

A challenging game of aerial battle - you climb, dive, fire at the enemy forces, and with every hit you gain more points toward becoming an ace. But if you get too close, they'll shoot you down! Between raids, you land to replenish your fuel supply at friendly airstrips (but don't crash into the tree). You have seven missions to score 1,000 points. Total point accumulation chart, bar graph showing progress of each player, intelligence report of entire Battle Zone. 1-5 players. Excellent graphics.

Level II, 16K $7.95


THE UNIVERSE BECKONS

STAD

Symbolic Trace and Debug

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

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

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

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

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

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

HexMem

by John T. Phillipp, M.D.

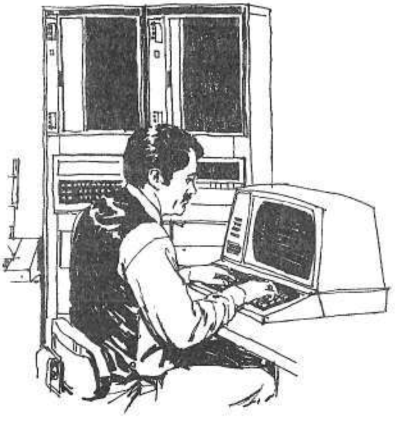
It's bound to happen sooner or later. You work on a program for hours and, while you're not watching, the TRS-80 memory changes one or two bits. You CSAVE your program and get a perfect CLOAD? After all, what is being read from the tape is **exactly** what is in memory. But later, when you try to CLOAD, a LISTing produces garbage on the screen. You CLOAD again with a different volume setting and get the same thing. After a few more tries, you realize with a sinking feeling that the problem is not in loading the tape, but rather with what is on the tape. And you're going to have to re-do the whole program . . .

Don't despair . . . yet. If you know how BASIC stores programs, and if you have a monitor program that will let you inspect and modify memory you may be able to save your program, or at least part of it. And if you can save 8K bytes of a 10K program that's 80% you don't have to retype!

HEXMEM is a BASIC program which duplicates some of the functions of machine language monitors like TRS-DOS DEBUG and the RSM-2 monitor by Small Systems Software. Although it doesn't support all of the functions of these sophisticated monitors (it does not access the Z-80 registers, for example), it does enable the user to convert hexadecimal numbers, display memory in hex or ASCII in a format similar to the machine language monitors, modify memory, enter machine language programs directly, load other BASIC programs into memory with HEXMEM and save HEXMEM and other BASIC programs on tape.

The code for HEXMEM takes 2,399 bytes, leaving 13,173 bytes for the other programs in a Level II, 16K TRS-80. It is densely packed into line numbers 1-28 without REM statements. This makes the program logic hard to follow, but was necessary since any program loaded into memory with HEXMEM must have line numbers greater than HEXMEM itself. HEXMEM can reside in memory with any BASIC program whose line numbers begin with line 30 or higher.

Commands — all commands are single letters. When HEXMEM asks COMMAND? type the command letter (ENTER). HEXMEM will return to COMMAND? after the command has been executed.



SUMMARY OF COMMANDS:

H —	Hexadecimal	- Converts hex to decimal value
D —	Decimal	- Converts decimal to hexadecimal value
G —	Graphics	- Converts any hex or decimal value to ASCII character
M —	Memory Dump	- Displays block of memory as hex values
A —	ASCII dump	- Displays block of memory as ASCII characters
E —	Edit	- Displays and edits the contents of a memory location
O —	Object code	- Enters machine language programs
L —	Load	- CLOADs a BASIC program into memory with HEXMEM
C —	*Combine*	*- MERGES a BASIC program on tape with one in memory*
S —	Save	- CSAVEs a BASIC program and HEXMEM on tape

EXPLANATION OF COMMANDS:

H — Hexadecimal

This routine will convert any hexadecimal number up to 10 digits to its decimal equivalent, although very large numbers will be displayed in decimal by exponential notation. Leading zeroes need not be entered 0AF2 and AF2 will be converted the same way. Type the hexadecimal number (ENTER).

D — Decimal

This routine will convert any decimal number from 0 to 65,535 to its hexadecimal equivalent. Larger numbers will not be converted as the routine will not produce a hexadecimal number more than 4 digits in length. Type the decimal number (ENTER).

Addition and subtraction of hexadecimal numbers may be done using the H and D commands providing the difference is not less than 0 nor the sum greater than FFFF hex (65,535 decimal). Use the H command to convert both numbers to decimal, perform the desired operation, then use the D command to convert the result to hexadecimal.

G — Graphics

This routine will convert a hexadecimal number in the range 0 to FF or a decimal number in the range 0 to 255 to its ASCII character, graphics character, or space compression code representation.

Hexadecimal numbers need only be typed, then (ENTER) but decimal numbers **must** be followed by an X (enter as 127X, for example). If the X is omitted, HEXMEM will consider the number to be hexadecimal and an erroneous conversion will result.

M — Memory dump (HEX)
see also A — ASCII dump

This routine will display the contents of any block of memory in hexadecimal

HEXMEM asks for ADDRESS (HEX)#1? Type any hexadecimal address from 0 to the top of memory (4FFF for a 4K machine, 7FFF for a 16K machine) then (ENTER).

Use the D command to convert decimal addresses into hex for the M command.

HEXMEM then asks for ADDRESS (HEX)#2? Type a hexadecimal address which is larger than ADDRESS (HEX)#1. If no ADDRESS (HEX)#2 is typed, and only (ENTER) is pressed, HEXMEM will display from ADDRESS (HEX)#1 to the top of memory.

NOTE:

HEXMEM is programmed for a 16K machine. If you have more or less memory line 8 in the program must be changed:

```
If H2$="" THEN D6=32767(16K)
                  49151(32K)
                  65535(48K)
                  20479(4K )
```

HEXMEM will display memory in lines of 16 bytes each. The address of each line is on the far left of the screen. For example, if the starting address is 4FB3 HEX the screen will display:

```
4FBO: XX XX XX XX XX XX XX XX
      XX XX XX XX XX XX XX XX
4FCO: XX XX XX XX XX XX XX XX
      XX XX XX XX XX XX XX XX
```

and so on, where XX is any hexadecimal number from 00 to FF. The address of the first byte of each line will always end in 0

To stop the dump at any point, press any key on the keyboard. To continue the dump, press any key except K. Pressing K will return to COMMAND?

A — ASCII Dump

This routine will display the contents of any block of memory as its ASCII equivalents. Control characters and graphics characters are displayed as periods (.). (The ASCII equivalent of any HEX or decimal digit (00-255) can be displayed using the G command).

The ADDRESS (HEX)? prompts should be answered in the same manner as for the M command, and the memory display is in the same 16 byte line format.

The A command is particularly useful for searching the memory for a BASIC program. Try displaying addresses 1600 to 1700 HEX using this command. This is the area where the Level II BASIC ROM stores its command table. HEXMEM itself resides in addresses 42EA to 4C48 HEX (17130 to 19528 decimal) and can be displayed by the A command.

E — Edit memory
see also 0-object code enter

This routine is used to modify a single memory location by replacing its current value with a new one.

HEXMEM asks ADDRESS TO CHANGE? Type in any HEX or decimal address from 0 to the top of memory, then (ENTER). Decimal numbers must be followed by an x, or HEXMEM will consider them to be hexadecimal and errors will result.

The current contents of that memory location will be displayed in HEX and decimal.

HEXMEM will then ask for the NEW VALUE? Type in the desired value from 0 to FF HEX or O-25 decimal. Again, decimal numbers must be followed by an X. Then press (ENTER). The contents of the memory location will be erased, the new value will be entered and HEXMEM will return to COMMAND?

Pressing (ENTER) after the NEW VALUE? prompt leaves the contents unchanged and returns to COMMAND? This may be used to PEEK at one memory location rather than using the M command. especially if you want to know the address contents in decimal.

Addresses from 0 to 3000 HEX (0 to 1288 decimal) comprise the Level II ROM and cannot be changed by the E command. Addresses from 3001 to 42E8 HEX (12290 to 17128 decimal) are RAM and may be changed although they are used by Level II BASIC for housekeeping and Input and Output.

Changing them may cause the system to crash. Try it though — there is no danger to the TRS-80 hardware from the keyboard. At worst, you may have to press RESET and re-load HEXMEM.

O — Object code enter

This routine is similar to the E command and is used for modifying a consecutive series of memory locations as when entering a machine language (object code) program.

After the starting address is entered and its contents changed with a new HEX (or decimal plus X) value, HEXMEM advances to the next memory location, displays its value in HEX and decimal and accepts the new value. Pressing (ENTER) with no value leaves the memory location unchanged and advances to the next.

Typing K for the value leaves the memory location unchanged and returns to COMMAND?

L — Load BASIC program

This routine allows a BASIC program to be CLOADed into memory co-resident with the HEXMEM monitor.

In order to CLOAD a BASIC program without erasing the program in memory (HEXMEM), some PEEKing and POKEing is necessary. HEXMEM does most of the work. However, after the CLOAD is complete, two more memory locations must be POKEd. These values — POKE 16548, 233 : POKE 16549, 66 — should be typed in and (ENTER) pressed. The combined program (HEXMEM + BASIC program) may then be LISTed, RUN, CSAVEd (with the S command), EDITed, etc.

The line numbers of the BASIC program must be higher than those of HEXMEM, or HEXMEM will be erased during the CLOAD. HEXMEM uses line numbers 1-28 so the BASIC program should start at line 30 or higher.

Try this command with the 10 line demonstration program on the cassette just after HEXMEM. Its line numbers are 100-200.

C — Combine BASIC programs

This routine will MERGE a BASIC program on tape with one in memory. As with the L command, the program on tape must have higher line numbers than the one in memory, or the program in memory will be erased during the CLOAD.

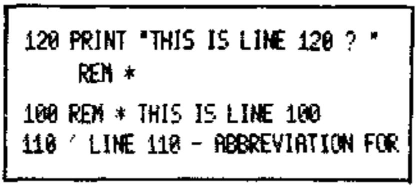
The same two memory locations must be POKEd after the CLOAD as with the L command. Follow the prompts on the screen.

The C command will enable frequently used sub-routines to be stored on tape and then added to new main programs as needed, saving the trouble of re-typing them.

After the programs are combined in memory, type DELETE 1-28 (ENTER) to erase HEXMEM. The combined program may then be LISTed, RUN or CSAVEd on tape.

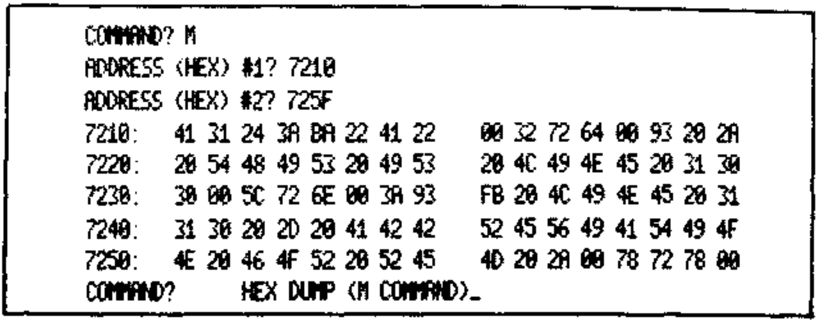
S — Save

This routine will CSAVE HEXMEM and any other BASIC program in memory on cassette tape.

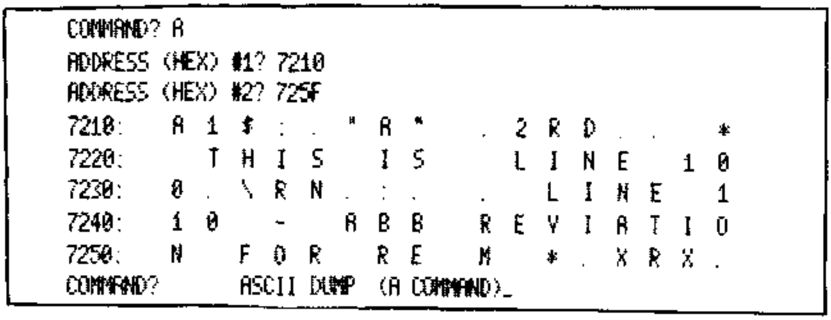


```
120 PRINT "THIS IS LINE 120 ? "
    REM *
100 REM * THIS IS LINE 100
110 ' LINE 110 - ABBREVIATION FOR
```

This is an example of how BASIC stores program test.
The ASCII dump and the HEX dump are looking at the same block
of memory—address,



```
COMMAND? M
ADDRESS (HEX) #1? 7210
ADDRESS (HEX) #2? 725F
7210:   41 31 24 3A BA 22 41 22   00 32 72 64 00 93 20 2A
7220:   20 54 48 49 53 20 49 53   20 4C 49 4E 45 20 31 30
7230:   30 00 5C 72 6E 00 3A 93   FB 20 4C 49 4E 45 20 31
7240:   31 30 20 2D 20 41 42 42   52 45 56 49 41 54 49 4F
7250:   4E 20 46 4F 52 20 52 45   4D 20 2A 00 78 72 78 00
COMMAND?      HEX DUMP (M COMMAND)_
```



```
COMMAND? A
ADDRESS (HEX) #1? 7210
ADDRESS (HEX) #2? 725F
7210:   A 1 $ : . " A *   . 2 R D . .   *
7220:     T H I S   I S     L I N E   1 0
7230:   0 . \ R N . : .   .   L I N E   1
7240:   1 0   -   A B B   R E V I A T I 0
7250:   N   F O R   R E   M   * . X R X .
COMMAND?      ASCII DUMP  (A COMMAND)_
```

These addresses contain the last 8 bytes of HEXMEM and the beginning lines of the demonstration program. One HEX number of 2 digits OO—FF HEX can store the numbers 0-255 decimal, the same as 1 8-bit binary byte.

By looking at the LISTing, and comparing it to the HEX and ASCII dump one can learn how BASIC stores the program lines.

OO is used by BASIC as the terminator of every line. Following 00 are 4 bytes (4 HEX digits) which are housekeeping for BASIC. The first 2 bytes are the memory address of the start of the next line, given with the least significant byte first, most significant byte last.

At location 7219, the address bytes are 7232 meaning that the next program line (line 110) starts at memory address 7232. Looking at the HEX dump, it is seen that the terminator of line 100 (00) is at address 7231 and 7232 is the beginning of the next line.

The next two bytes 64 00 in addresses 721B-721C are the BASIC line number of the line given in HEX least significant byte first --00 64 HEX equals 100 decimal (use the H command). This is why BASIC line numbers cannot exceed 65,535. 2 bytes can only store up to FFFF HEX.

Lines 110 (006E HEX), 120 (0078 HEX), 130 (0082 HEX) up to line 200 (00C8 HEX) of the demonstration program may be identified the same way as line 100 was.

If the byte following the line terminator 00 is 0 (meaning address of the start of the next line is 0) BASIC assumes it has reached the end of the program. This is as far as the program can be LISTed on the screen, even though the complete program remains in memory and can be accessed by the A of M commands.

After the four housekeeping bytes, BASIC starts the text of the line. To conserve memory space as 1 byte abbreviations:

80 HEX = END
81 HEX = FOR
8A HEX = DIM
87 HEX = NEXT
8D HEX = GOTO
91 HEX = GOSUB
92 HEX = RETURN
93 HEX = REM
D5 HEX = = (Equals sign)

and so on.

The complete list of BASIC TOKENS AS STORED IN MEMORY may be found in the May, 1979 issue of *Prog 80*, page 34.

The rest of the line is stored byte as the HEX equivalent of the decimal ASCII character codes. The complete list (in decimal which can be converted to HEX by the D command) may be found on page C/2 of the LEVEL II BASIC REFERENCE MANUAL.

08 — 1F are cursor control codes

20 — 7F are ASCII character codes

80 — BF are graphic codes

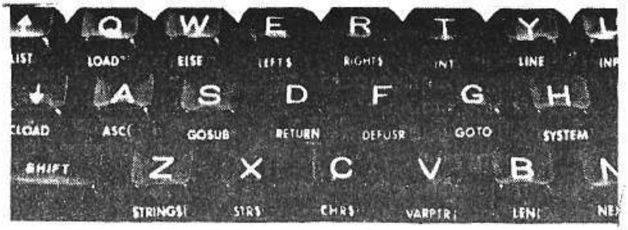
CO — FF are space compression codes (tab 0-63 spaces)

This method of storage and housekeeping explains the 'garbage' seen on the screen after a bad CLOAD is LISTed.

If any byte of the code in memory has been changed to 00, BASIC assumes the next 2 bytes are the starting address of the next line, and the 2 bytes after that are the line number. Following bytes are LISTed as the full printing of the command abbreviations (see BASIC TOKENS). This mess is complicated by the fact that HEX digits 08-1F are moving the cursor all over the screen and causing the commands to be printed anywhere at random.

```
1 CLEAR100:CLS:DIMH(16),H$(16),X$(16):X$="0123456789ABCDEF":DATA
"0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F":
V$="28":U$="HIGHEST LINE IN MEMORY":D$="":T$="TO ERASE HEXMEM, T
YPE:  DELETE 1-28  <ENTER>":'
* HEXMEM VER 1.1 *
2 P=0:O=0:INPUT"COMMAND";C$:IFC$="H"THEN3ELSEIFC$="D"THEN4ELSEIF
C$="G"THEN5ELSEIFC$="M"THEN8ELSEIFC$="A"THENP=1:GOTO8ELSEIFC$="E
"THEN15ELSEIFC$="O"THENO=1:GOTO15ELSEIFC$="L"THEN24ELSEIFC$="C"T
HENV$=U$:D$=T$:GOTO24ELSEIFC$="S"THEN28ELSEPRINT"+INVALID+":GOTO
2
3 INPUT"HEXADECIMAL";A$:A$="0"+A$:GOSUB19:PRINT"  DECIMAL =";D:G
OTO2
4 INPUT"DECIMAL";D:GOSUB21:PRINT"  HEXADECIMAL = ";H$(1)+H$(2)+H
$(3)+H$(4):GOTO2
5 INPUT"HEX OR DEC (TYPE X AFTER DECIMAL)";M$:IFRIGHT$(M$,1)<>"X
"THENA$=M$:GOSUB19:MM=DELSEMM=VAL(M$)
6 IFMM<32THENG$="CONTROL CHARACTER"ELSEIFMM=32ORMM=128THENG$="SP
ACE"ELSEIFMM<192THENG$=CHR$(MM)ELSEIFMM<=255THENG$="TAB FOR"+STR
$(MM-192)+" SPACES"
7 PRINT"  ASCII CHARACTER = ";G$:GOTO2
8 H1$="":H2$="":INPUT"ADDRESS (HEX) #1";H1$:A$=H1$:GOSUB19:GOSUB
27:D5=D:INPUT"ADDRESS (HEX) #2";H2$:A$=H2$:GOSUB19:D6=D:LF=1:W=D
5:PRINTH1$":    ";:IFH2$=""THEND6=32767
9 FORM=D5TOD6:G$=INKEY$:IFG$<>""GOSUB26
```

```
10 IFLF=17THENLF=1:W=W+16:PRINTCHR$(29):D=W:GOSUB21:PRINTH$(1);H
$(2);H$(3);H$(4);":   ";
11 M1=M:IFM1>32767THENM1=M1-65535
12 D=PEEK(M1):IFP=1THENGOSUB23ELSEGOSUB22
13 PRINTH$(3);H$(4);" ";:LF=LF+1:IFLF=9PRINT"   ";
14 NEXTM:PRINT:GOTO2
15 INPUT"ADDRESS TO CHANGE (HEX OR DEC - TYPE X AFTER DEC)";M$:I
FM$="K"THENGOTO2ELSEIFRIGHT$(M$,1)<>"X"THENA$=M$:GOSUB19:MM=DELS
EMM=VAL(M$)
16 PRINT"CURRENT CONTENTS: ";:IFMM>32767THENMM=MM-65535
17 CC=PEEK(MM):D=CC:GOSUB22:PRINTH$(3)+H$(4);" HEX  (";CC;"DECIM
AL)":M$="":PRINT:INPUT"NEW VALUE (HEX OR DEC - TYPE X AFTER DEC)
";M$:IFM$=""THENNN=CCELSEIFM$="K"THENGOTO2ELSEIFRIGHT$(M$,1)<>"X
"THENA$=M$:GOSUB19:NN=DELSENN=VAL(M$)
18 POKEMM,NN:IFO=0THENGOTO15ELSEPRINT"NEXT ADDRESS - ";:MM=MM+1:
GOTO16
19 D=0:K=1:FORJ=1TOLEN(A$)-1:K=K*16:NEXTJ:K=INT(K+.01):FORI=1TOL
EN(A$):FORJ=1TO16:IFMID$(A$,I,1)=MID$(X$,J,1)THENGOTO20ELSENEXTJ
20 D=D+K*(J-1):K=K/16:NEXTI:RETURN
21 H(1)=INT(D/4096):D1=D-H(1)*4096:H(2)=INT(D1/256):D2=D1-H(2)*2
56:H(3)=INT(D2/16):D3=D2-H(3)*16:H(4)=D3:FORX=1TO4:RESTORE:FORZ=
0TOH(X):READH$(X):NEXTZ:NEXTX:RETURN
22 H(3)=INT(D/16):D1=D-H(3)*16:H(4)=D1:FORX=3TO4:RESTORE:FORZ=0T
OH(X):READH$(X):NEXTZ:NEXTX:RETURN
23 H$(3)=" ":IFD<32THENH$(4)=".":RETURNELSEIFD>127THENH$(4)=".":
RETURNELSEH$(4)=CHR$(D):RETURN
24 PRINT"BE SURE LINE NUMBERS ARE GREATER THAN ";V$;".":PRINT"WH
EN TAPE RECORDER STOPS, TYPE:":PRINT"POKE 16548,233: POKE 16549,
66  <ENTER>":PRINTD$:PRINT:PRINT"PRESS <ENTER> WHEN READY TO   *
 CLOAD *":INPUTA1$
25 IFPEEK(16633)>=2THENPOKE16548,PEEK(16633)-2:POKE16549,PEEK(16
634):CLOADELSEPOKE16548,PEEK(16633)+254:POKE16549,PEEK(16634)-1:
CLOAD
26 G$=INKEY$:IFG$=""THENGOTO26ELSEIFG$="K"THENPRINT:GOTO2ELSERET
URN
27 IF(D<16)OR(H1$="")THEND=0:H1$="0000":RETURNELSEIFD/16=INT(D/1
6)THENRETURNELSED=D-(D-INT(D/16)*16):GOSUB21:H1$=H$(1)+H$(2)+H$(
3)+H$(4):RETURN
28 CLS:PRINT"PRESS <ENTER> WHEN READY TO   * CSAVE *":INPUTA1$:C
SAVE"A"
```



T-SHORT™

SHORTHAND for LEVEL II and DISK BASIC

- ✱ TSHORT™ lets you type LEVEL II and/or DISK BASIC more quickly and accurately than ever before. Save up to 90% programming time and achieve 100% accuracy.
- ✱ 31 programming statement keys.
- ✱ KUSTOM™ key, up to 64 characters - changeable anytime.
- ✱ 42 key decals (see picture). 10 area different for DISK users.
- ✱ A single, shifted-key entry types entire statement on screen.
- ✱ Fast, efficient machine language.
- ✱ Uses 580 bytes of LOW memory, i.e.: No MEM SIZE req'd.
- ✱ Comes on cassettes, one side LEVEL II, the other for DOS.
- ✱ Compatible with DOS 2.1, 2.2, 2.3, NEWDOS, KBFIX, etc.
- ✱ DOS version loads to and executes from disk via TAPEDISK.
- ✱ Features self-entering commands: CONT; GOTO10: KUSTOM™ (Self-enter optional)
- ✱ TSHORT™ W/4 page instruction manual **$9.95**

TSE

TO ORDER:
CALL **TOLL-FREE**
1-800-258-1790
9AM - 9PM, EST

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

HARDSIDE™

NO SUBSTITUTIONS ON PACKAGE PRICES

THESE OFFERS EXPIRE MARCH 28, 1980

DISK EXPANSION PACKAGE

This package includes everything necessary to add disk capabilities to your TRS-80. To buy everything in this package at the local Radio Shack store would cost $1100.00.

PACKAGE INCLUDES:

- **16K RAM Expansion Interface**
- **PERCOM Data Separator, Installed**
- **PERCOM TFD-100, 40-track disk drive**
- **Dual disk drive cable**
- **NEWDOS Disk operating system**
- **10 BASF diskettes**

HARDSIDE
SPECIAL PRICE **$899.95**

SMALL BUSINESS PACKAGE

SAVE $579.00

An ideal starter package for the small business. Includes a TRS-80 with 16K RAM and Level II BASIC, an expansion interface with an additional 16K RAM Installed, two Percom disk drives with cable and data separator, NEWDOS disk operating system (40 track version) and the Centronics 730 line printer.

$2799.95

HIGH SPEED STRING HANDLING IN BASIC

by Marty Zwilling

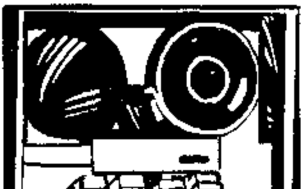
If you have a Level II BASIC program which manipulates a lot of text or character data, you have probably noticed that it will intermittently "hesitate" or run slowly during execution for no apparent reason. These pauses are due to a required reorganization of the available string space pool, the size of which is defined on the **CLEAR** command. The time required for this reorganization varies in proportion to the number of string variables and string array elements defined and in use, and may range from trivial to many minutes with several thousand string elements in use. This article will discuss techniques for assessing and minimizing this delay.

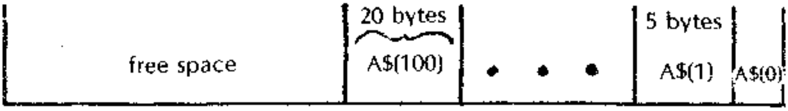
As indicated in the Level II BASIC Reference Manual, space for character string variables and character arrays is dynamically allocated at run time from a single pool of free string space defined by the **CLEAR xxx** command. For example, consider the following program segment:

```
10 CLEAR 1000
20 DIM A$(100)
30 FOR I=1TO100
40 INPUT A$(I)
50 NEXT I
```

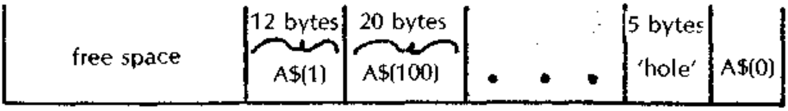
These statements might be used to define space in memory for 100 product names and prompt the user for these names. More precisely, the **CLEAR** command reserves 1000 bytes of memory which is available only to string variable text (the variable name, length, and a pointer to the text are stored elsewhere). String variables must end in a $ or be defined via the DEFSTR command. In this case, the 100 elements of A$ must not average more than 10 characters each, or an *OUT OF STRING SPACE* error will occur.

As string elements are assigned, space is sequentially used from the high addresses of the string pool down to the low boundary. All available free bytes are contiguous:





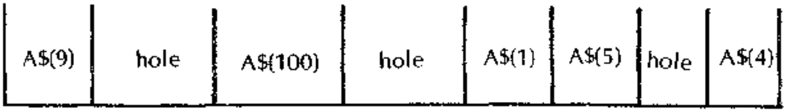
A problem begins to occur, however, when a given element or variable is reassigned a new value. For example, if A$(1) has the value RADIO, and later in the program is assigned the value A$(1)= REFRIGERATOR, the five bytes for RADIO are no longer used and the new value is assigned in the free space area:



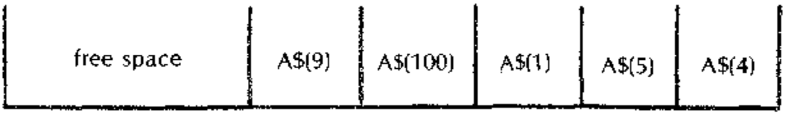
When string assignments and reassignments have exhausted the free space area, a "reorganization" of the entire pool is required to consolidate all the holes into useable free space. This reorganization involves moving the data from all string variables in the pool to contiguous addresses at the high end of the area. The time required to do this is the "hesitation" referenced earlier.

Notice that the five bytes released cannot immediately be used for new assignments, since they are not contiguous with other available space. These "holes" in the string pool are always created by the reassignment of a variable value, even if the new value is shorter than the old. Previous holes are also not used for new variable assignments.

Before Reorganization



After Reorganization



The reorganization is an automatic function of BASIC, and will occur as necessary in the midst of statement execution. The BREAK key is ignored until the completion of the reorganization, so the CPU appears to lock up during this process. The frequency of invocation is a function of the amount of *reassignment of string variables* being done by the program, and the amount of unassigned space in the pool. The duration of the reorganization function is determined primarily by the number of string variables with values assigned as follows:

Number of strings	**Elapsed time to reorganize**
10	trivial
100	3 seconds
500	45 seconds
1000	3 minutes
2000	12 minutes

Note that the elapsed time to reorganize goes up exponentially as the number of assigned strings increases. The time required to reorganize even 500 strings is normally intolerable, but 500 strings may be needed to hold even 100 names and addresses in memory (DIM N$(100),A1$(100),A2$(100),A3$ (100),A4$(100)).

To measure the reorganization time in your own program, simply interrupt the execution at various points with the (BREAK) key and use ?FRE(A$) to display the free pool space available. The FRE command will always cause a reorganization of the pool to get an accurate assessment of space available, and you can time the response to this command. If the response is not immediate, you need to implement one or more of the recommendations given below.

Here are some ways that you can minimize the reorganization problem through careful programming:

1. Always store numeric data in numeric variables and arrays. Multiple conversions from characters to numerics is much faster than string reorganization with many strings.

   Combine strings within a single array to minimize the number of unique strings stored. For example, in the name and address case above, use a single array NA$(100) of 100 elements, each containing a name and four address lines concatenated into a single string with imbedded line returns. Strings may be combined to a maximum length of 255 characters.

3. Use tape and disk to store character data, rather than keeping it in memory.
4. Define your string pool as large as possible with the CLEAR command to minimize the number of trips through reorganization.
5. Use simple string assignments directly in your program (A$="SAMPLE") or use READ and DATA statements, since string pool space will not be used by these assignments.
6. Minimize the use of substring assignments (RIGHT$, LEFT$, MID$) and concatenations (A$+B$), since these require additional space in the string pool for intermediate work areas.
7. Do your own string pool management with assembler language subroutines.

8. Store character data in numeric arrays through manipulation of variable and string pool pointers as shown below.

I have developed routines in Level II BASIC to store, retrieve, and manipulate the text for over 500 checks in memory for a personal finance program, with negligible string pool reorganization delays. This was accomplished by using double precision numeric arrays to store the string data.

Each double precision number is eight bytes in length and can thus hold eight characters or letters. Arrays of these numbers are always allocated contiguously, allowing a 24-byte string to be contained in three contiguous elements of the array. Since strings are not normally all the same length, I stored the exact length of each string as the first byte of the string in the numeric array.

For this approach to work well, the strings must be similar in length so that little space is wasted when each string is started at a new element in the array. In my case, the strings were the payee names from checks which I limited to 23 characters in length, which always occupied 3 elements of the array. This convention always left room for the exact length count (3*8=1+23).

A subroutine was required to move the data from a standard string variable A$ to the proper three numeric elements of the double precision array P#(500*3). Note that 1500 numeric elements are needed to hold the 500 strings anticipated.

```
1000 REM * SUBROUTINE TO MOVE A$ TO P#(Z,Z+1,Z+2) *
1010 Y=VARPTR(P#(3*Z-2)): POKE Y,LEN(A$)
1020 XX=PEEK(16598):YY=PEEK(16599):X=Y+LEN(A$)
1030 POKE 16599,INT(X/256):POKE 16598,X-INT(X/256)*256
1040 ZZ=PEEK(16545): POKE 16545,0: A$=A$+""
1050 POKE 16545,ZZ: POKE 16598,XX: POKE 16599,YY
1060 RETURN
```

Line 1010 gets the address of the beginning of the desired numeric element and sets the actual string length as the first byte of the first element. Line 1020 saves the current address of the end of the available string space pool and calculates an equivalent address in the numeric array (beginning of element + length of string). Line 1030 temporarily sets this new address in 16598-16599 as the high end of the string space pool. Line 1040 saves the low end of the pool address MSB and zeroes it out to prevent a reorganization attempt. The statement A$ = A$ + ""will cause the string to be moved into the numeric array behind the length byte from its current location in the string pool. Line 1050 resets all original pointers for the return. The value of A$ has now been quickly and safely stored in the numeric array. A simple calling sequence to this subroutine might be as follows:

```
100 FOR Z=1TO500
110 INPUT#-1,A$: GOSUB 1000
120 NEXT Z
```

A second subroutine is required when referencing the data already stored in the numeric array. This routine requires no movement of the data, but simply alters the string pointer portion of A$ to address the string within the numeric array.

```
2000 REM * SUBROUTINE TO POINT A$= P#(I,I+1,I+2) *
2010 YY=VARPTR(P#(3*I-2))+1: ZZ=VARPTR(A$)
2020 POKE ZZ,PEEK(YY-1): IF YY<0 THEN YY=YY+65536
2030 POKE ZZ+2,INT(YY/256):POKE ZZ+1,YY-INT(YY/256)*256
2040 RETURN
```

Line 2010 loads YY with the address of the desired text string, and ZZ with the address of the A$ length and text pointer. Line 2020 sets A$ with the new length, and adjusts the text address if above 32K. Line 2030 sets the text address in A$ to point to the text in P#, and the operation is complete. A typical calling sequence to display the strings would be as follows:

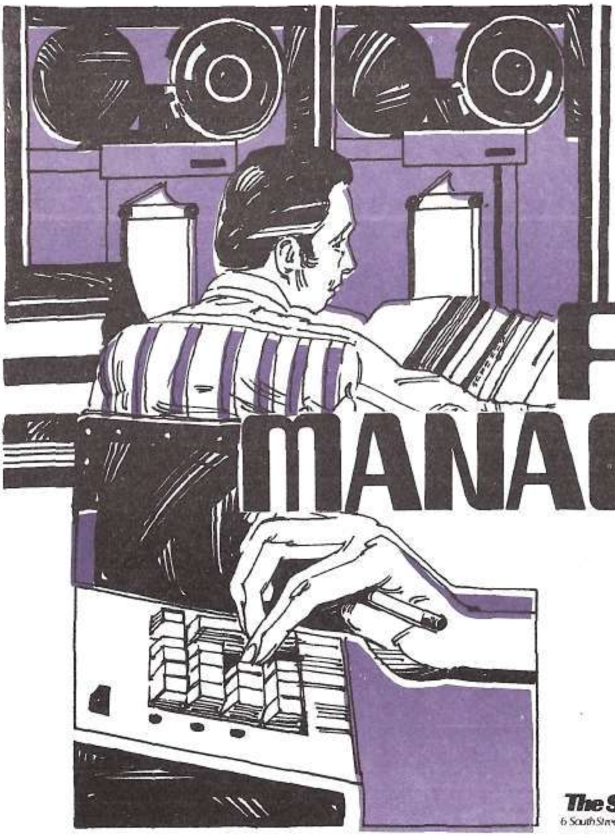
```
200 FOR I=1TO500
210 GOSUB2000:PRINT A$
220 NEXT I
```

A few precautions are required when using the above approach:

1. Never define a new variable between the use of VAPTR to get the address of a variable and the use of that address. The old variable may be moved to make room for a new variable. In my example, XX, YY, ZZ, X, Y, and Z must be set to zero at program initialization.
2. Make sure that the character strings will fit in the allotted space in the numeric array before moving them, to prevent overlay of other data.
3. Be careful when swapping two values in the array, such as during a sort, since simple string assignments (B$ = A$) may not cause data movement. Use a dummy concatenation such as B$ = A$ + "".

In summary, you can dramatically affect the speed and useability of your BASIC program by your programming techniques relative to character strings. The techniques offered here range from simple to complex—all you have to do is match the best approach to your requirement.

NEW! utility package from NEPENTHE PROGRAMS



FILE MANAGER 80

TSE

The Software Exchange

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

Programming utility for professional programming.

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

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

Disk Payroll

Features:

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

SAMPLE OUTPUT:

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

--------------------------------------------------------------------------------
          Employee                  Gross    Fed.     FICA    State    City    Other
--------------------------------------------------------------------------------
David E. Smith     456-78-9999      500.00   123.81   30.00   20.00    10.00   0.00
Cindy D. Taylor    444-55-6666      500.00   123.81   30.00   20.00    10.00   0.00
Philip G. Jones    454-54-5454      280.00    27.78   16.00   11.20     5.60   0.00
Jane Doe           555-56-5555      240.00    23.81   14.40    9.60     4.80   0.00
J.T. Johnson Jr.   511-11-5111      400.00    34.12   24.00   16.00     8.00   0.00
Mary M. Marks      400-00-0004      280.00    34.12   16.80   11.20     5.60   0.00

           CHECK INFORMATION LISTING FOR PAYROLL PAY 2    (NOVEMBER 14, 1978)
--------------------------------------------------------------------------------
  Name              Soc Sec No.   Gross   Fed.     FICA   State   City    Other   Net
--------------------------------------------------------------------------------
David E. Smith      456-78-9999   500.00   12.81   30.00  20.00   10.00   0.00    316.19
Cindy D. Taylor     444-55-6666   500.00  123.81   30.00  20.00   10.00   0.00    316.19
Philip G. Jones     454-54-5454   280.00   27.78   16.80  11.20    5.60   0.00    218.62
Jane Doe            555-55-5555   240.00   23.18   14.40   9.60    4.80   0.00    188.02
J.T. Johnson Jr.    511-51-5111   400.00   56.11   24.00  16.00    8.00   0.00    295.89
Mary M. Marks       400-00-0004   280.00   34.12   16.80  11.20    5.60   0.00    212.28
--------------------------------------------------------------------------------
  Company                         2200.00 388.81  132.00  88.00   44.00   0.00    1547.19
--------------------------------------------------------------------------------

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

Level II, 16K $39.95

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

TYPING TUTOR

by
Roy Groth

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

Level II, 16K — $19.95

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

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

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

TSE

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

Programming Problem? Question about a TSE Program?

Try our Hotline...

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

(603) 673-5144

CTR-41 MODIFICATIONS

by Bob Cahill

Would you like to add these features to your CTR-41?

- Automatic level playback
- Ground loop correction
- Switched motor
- Monitor (Playback only) or Monitor (both Playback and Record)

One way of accomplishing these things is to buy a box which connects to your recorder for around $50. The other way is to do it yourself with about a dollars worth of parts. (Let's see – where did you put that soldering iron?)

The tools you will need are:

- Soldering iron and solder (of course)
- Phillips screwdriver
- Wire cutter/stripper

Parts needed are:

- Wire (10 feet)
- 1/4w 68 resistor

The first step is disassembly. Remove all plugs and cables from the recorder. Place recorder top down with controls facing you (being careful not to scratch the top). Remove three phillips screws from the back case; remove battery cover and batteries; remove two phillips screws from the battery compartment; and carefully remove back cover and handle. The PC board is now in view. Using Fig 1. as a guide, make the modifications as detailed below.

MOD 1 -- AUTOMATIC LEVEL PLAYBACK

Improvements:

Improves tape loading considerably

Allows Level I and Level II conversions without resetting

Volume loads all programs, Level I or II, at one setting: approximately 7½ (you will notice that volume setting is no longer critical).

Modification

Locate RP switch (switch with 18 pins in a row – see Fig. 1).

Short pins 2 and 3 (notice pin 1 is not used – there is no foil leading to it).

MOD 2 -- GROUND LOOP

Improvements:

Improves tape loading

Removes most tape hum

Modification

Cut large foil path between REMOTE and MIC jack (see cut 2 in Fig. 1).

Solder a small wire from point B to A.

MOD 3 -- MOTOR SWITCH

Eliminates pulling REMOTE plug to FAST FORWARD or REWIND: just flip tone switch (Note: tone always set to high).

Modification

Cut two PC foils going to tone switch, pins 2 and 3.

Solder a wire from switch pin 1. to point F.

Solder a wire from switch pin 2. to point A.

MODS 4 and 5 -- MONITOR SOUND (Choose One)

To monitor sound from recorder PLAYBACK only:

Solder a ¼w 68 resistor from points D to C.

To monitor sound for both PLAYBACK and RECORD:

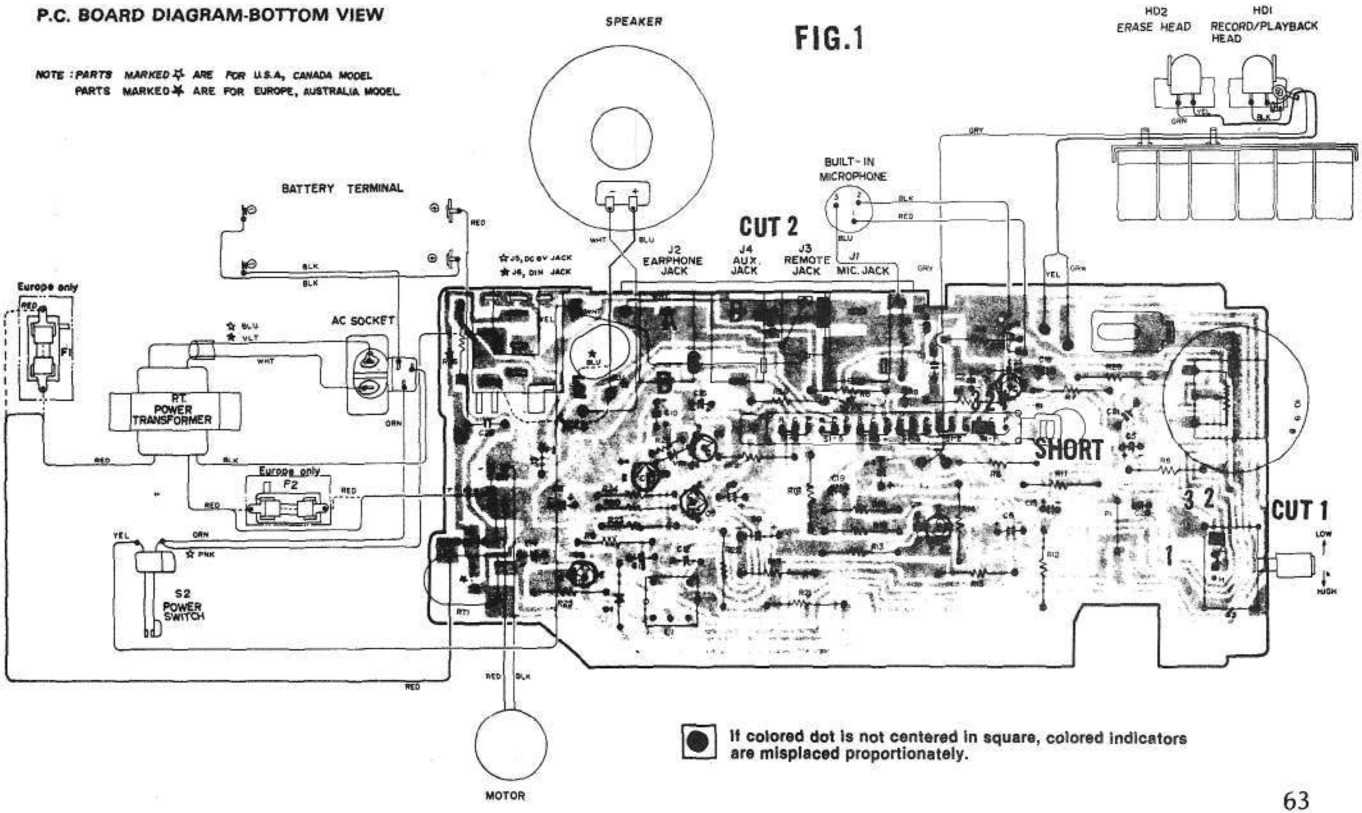
Solder a ¼w 68 resistor from points D to E.

Now reassemble and test! A schematic with mods installed is shown in Figure 2.

P.C. BOARD DIAGRAM-BOTTOM VIEW

NOTE : PARTS MARKED ☆ ARE FOR U.S.A, CANADA MODEL
PARTS MARKED ★ ARE FOR EUROPE, AUSTRALIA MODEL

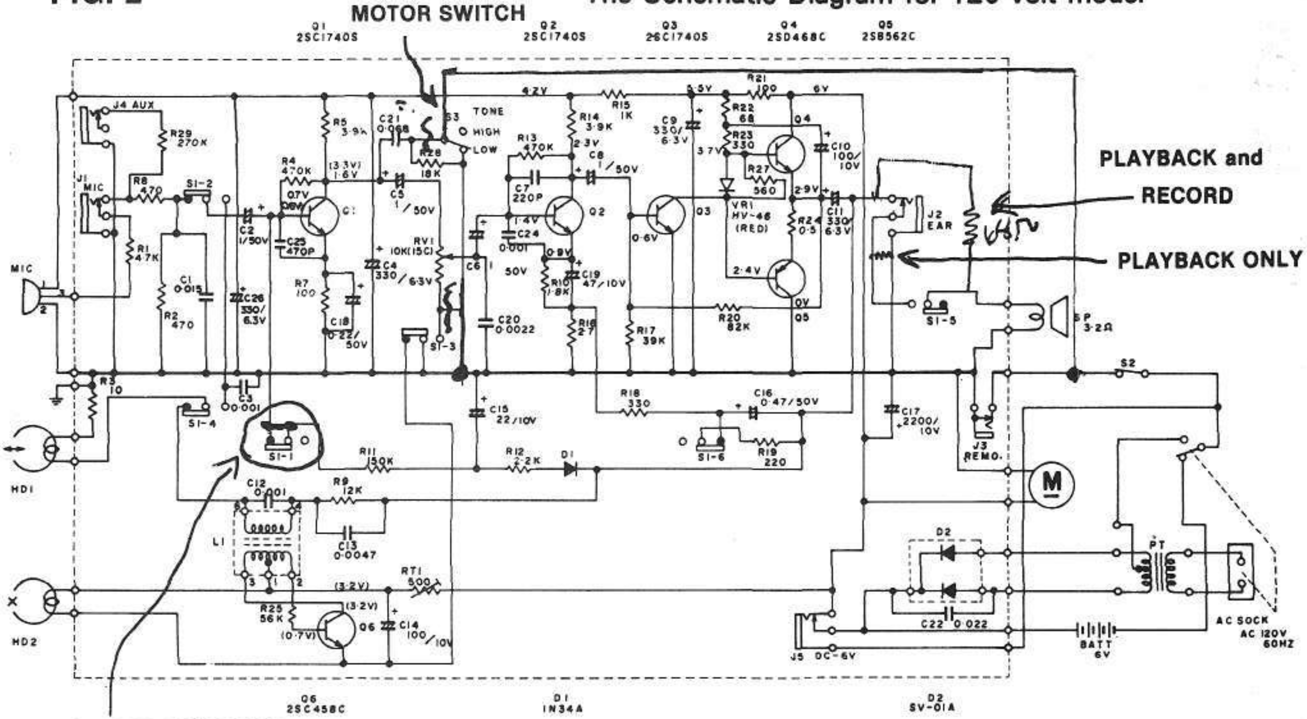
FIG.1



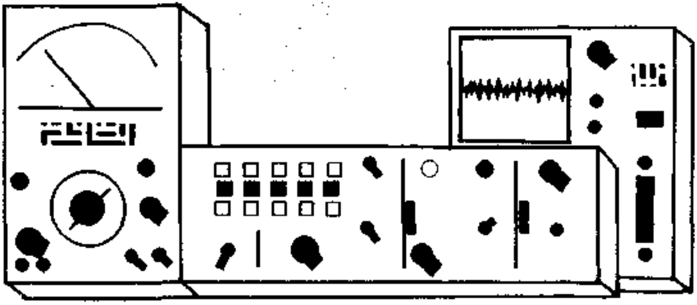
If colored dot is not centered in square, colored indicators are misplaced proportionately.

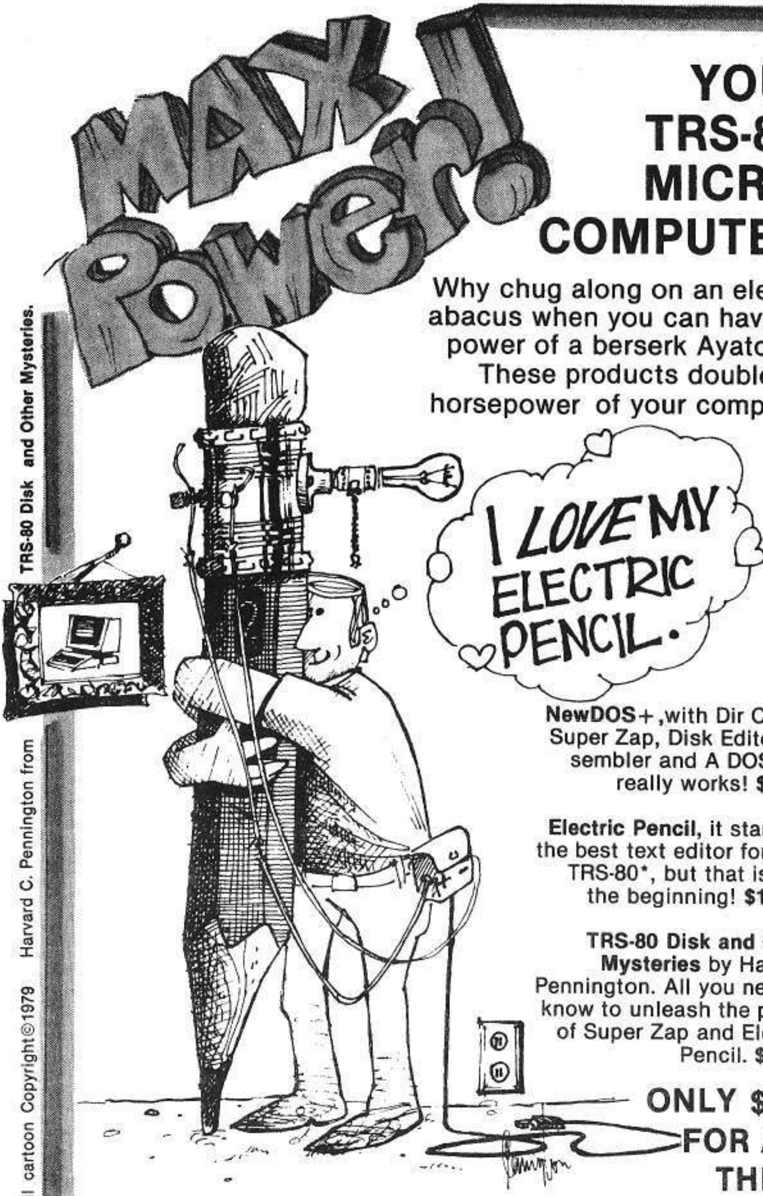
FIG. 2

The Schematic Diagram for 120 volt model



- ALL RESISTANCE VALUES IN OHMS, K=1000
- ALL CAPACITANCE VALUES IN μF EXCEPT "P"
  P=μμF=PF
- SPECIFICATIONS AND SCHEMATIC DIAGRAM ARE SUBJECT TO CHANGE WITHOUT NOTICE.




MAX Power!
I LOVE MY ELECTRIC PENCIL.

PERIODICAL CROSS REFERENCE

by Dave Stambaugh

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

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

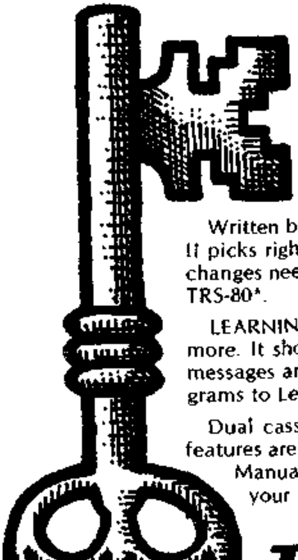
Cassette version: Level II, 16K	**14.95**
Disk Version: Single disk, 32K	**19.95**

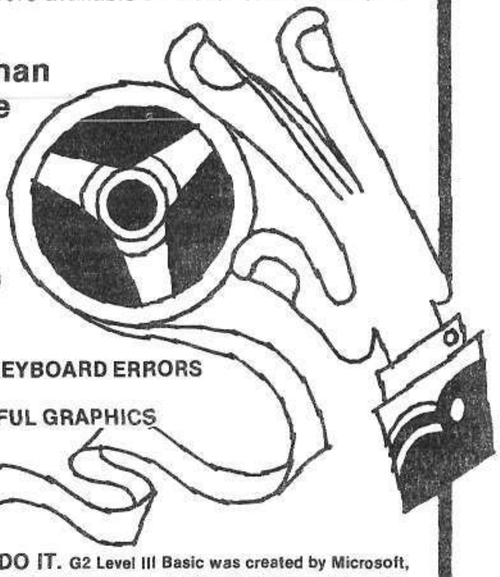
(Minimum configurations)

TSE

The Software Exchange

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





TSE Order Form

Special prices in effect 60 days from mailing

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

DO NOT USE FOR HARDWARE

☐ Check ☐ VISA ☐ Master Charge

☐ Money Order

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

Not responsible for typographical errors

THE SOFTWARE EXCHANGE

6 SOUTH STREET MILFORD, NH 03055

Order toll free: 1-800-258-1790

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

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

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

Charge card account number

Signature..

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

Charge customers: Please fill in account information above and below

Name..

Address..

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

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

SoftSide™

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

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

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

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

Credit Card

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

Signature________

Name________

Address________

City________ State________ Zip________

Please remit in US funds ONLY

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

SUBSCRIPTION RATES - 6 issues per year

USA		
Bulk mail - $15.00	Canada	$21.00
First Class Mail - $21.00	Mexico	
Overseas airmail - $27.00	APO/FPO	
	Overseas surface mail	

PROG/80

☐ **Check/Money Order enclosed** ☐ **Master Charge** ☐ **VISA**

SIGNATURE________

ACCOUNT #________

EXP. DATE________ INTER. #________

Subscription Order Form

NAME________

ADDRESS________

CITY________ STATE________ ZIP________

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

PO BOX 68, MILFORD, NH 03055

PROG/80
A SoftSide Publication
P.O. Box 68, Milford, NH 03055

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