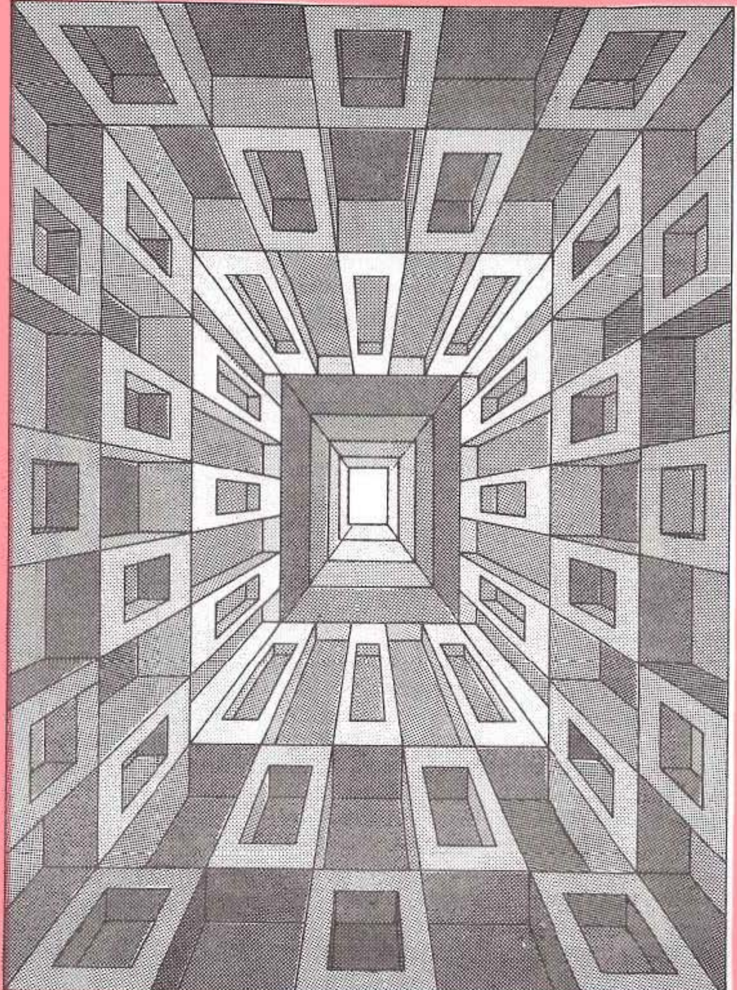
PROG/80

SEPTEMBER 1979
$3.00

Dedicated to the serious programmer



ST80D

Lance Micklus'
ST80-the Smart
Terminal Program-
***just got* SMARTER!!**

ST80D contains extensions for disk drive systems to exchange files with a timesharing computer or another TRS-80.

USING ST80D, your TRS-80 can do all this and more:

- Gather and pre-format data, store it on disk, then transmit it to a timesharing computer for processing.
- Processed data from the timesharing computer can then be sent back to the TRS-80.
- One TRS-80 can generate a data base and share it with another TRS-80 thousands of miles away by telephone.
- Users may customize their terminal program by redefining the translation tables. Conversion from one set of tables to another takes only seconds.
- Auto logon feature sends your account name, number and password upon request.
- ST80D can transmit any type of TRS-80 ASCII file, including BASIC programs stored in ASCII format, and most BASIC data files. Binary files can also be transmitted from one TRS-80 to another, allowing even machine language programs to be sent over the phone.

ST80D is a practical, full-feature terminal program that has been used on a variety of timesharing systems. These include IBM 370, Honeywell Sigma/6, Harris/7, DECSYSTEM 20, Dartmouth Timesharing, CDC Cyber and HP 2000.

If you're looking for a professional quality product, **not an amateur program**, then order ST80D today!

For 32K
disk systems -
$79.95

TSE TRS-80 Software Exchange

PROG/80

SEPTEMBER 1979
VOL. 1, NO. 4

FEATURE ARTICLES

WHAT'S NEW

PROG/80

STAFF

Editor and Publisher
Roger W. Robitaille, Sr.

Business Manager
Elizabeth Robitaille

Software Editor
Jack W. Moore

Design/Production
Sharon Demmerle

Creative Director
Elaine Cheever

Marketing Manager
Joseph A. Breton

Subscriptions Manager
Diana Bishop

Customer Service
Bette Keenan

Copy Editor
Freida Day

Accounting
Rita Ellis

Contributing Editors
George W. Blank
Lance Micklus

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 Watts 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 Watts Line will be open from:

9:00 a.m. to 9:00 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

Warning to Disk Users!

You should never turn your system ON or OFF with a diskette in the drive. Even turning the drive itself on or off can create an electronic pulse at the write head and spoil a diskette. To protect your diskettes, remove them before turning any part of the system off. When your turn on the computer, do it in this order:

1) Expansion Interface
2) Printer
3) Computer
4) Screen
5) Disk Drives

Only after everything is on should you load your diskettes and press the RESET button in order to load the Disk Operating System.

Radio Shack announced several months ago that diskettes were to be removed prior to shutting down the computer, but only recently have problems been verified when turning the system on. To be absolutely save, never turn any part of the system on or off with a diskette in the drive. (Note: It will not harm your disk drive to run empty on power up.)

CORRECTION!

In the July issue of PROG/80 there are a couple of errors (Cassette Controller) that may confuse the unwary. First, in two places in the text the author states that the meter is placed *in series* between the CPU and the CTR-41. Not so! And not shown as such in the schematics, which are properly shown with the meter *in parallel* across the line. The VU meter is a sensitive AC voltmeter, and is rectifying dc pulses for an average reading.

Note also that Figure 4 has a wiring error, although the text is correctly stated. The dpdt switch (lower), will effectively short the output of the CTR-41 when connected to the right side position as viewed. In addition, the "ground" shielded lead from the speaker which should go to the switch is "floating".

Outgoing Mail

In the TRS-80 world, the United States is made up of three sections: the east coast, the west coast, and Texas. The hardest place to get to seems to be Texas, especially Fort Worth. At times it seems like Radio Shack is in a vacuum.

Those of us who make our living in computer software have discovered certain needs. Yet, our cries for help do not seem to be heard in the great Texas vacuum.

To write software for a mass market, you need to write it for some sort of standard system, especially if you are using a disk system. Many disk system owners have never used a computer before. Getting them to do a simple task such as reading a directory is a major accomplishment.

We have learned, the hard way, that disk software must be complete. It should be ready to be put in the drive and run. Anyone who has tried to distribute software on a user-FORMATTED disk (no DOS) has found that many users get totally baffled just trying to copy the files to their own disks.

The fact of the matter is, first, that we software people need a good DOS we know everybody will have and can use. Second, we need an arrangement to include the DOS on our own disks. Sadly, we have neither.

In the first case, there is NEWDOS, TRSDOS 2.2, DOS 3.0, MICRODOS, TRSCPM, MMSFORTH, and PASCAL. Some of these DOS systems are unique to themselves. Others are variations of a basic theme. Most software people I know are planning to distribute on TRSDOS 2.2 because it is assumed that everybody owns a copy of it.

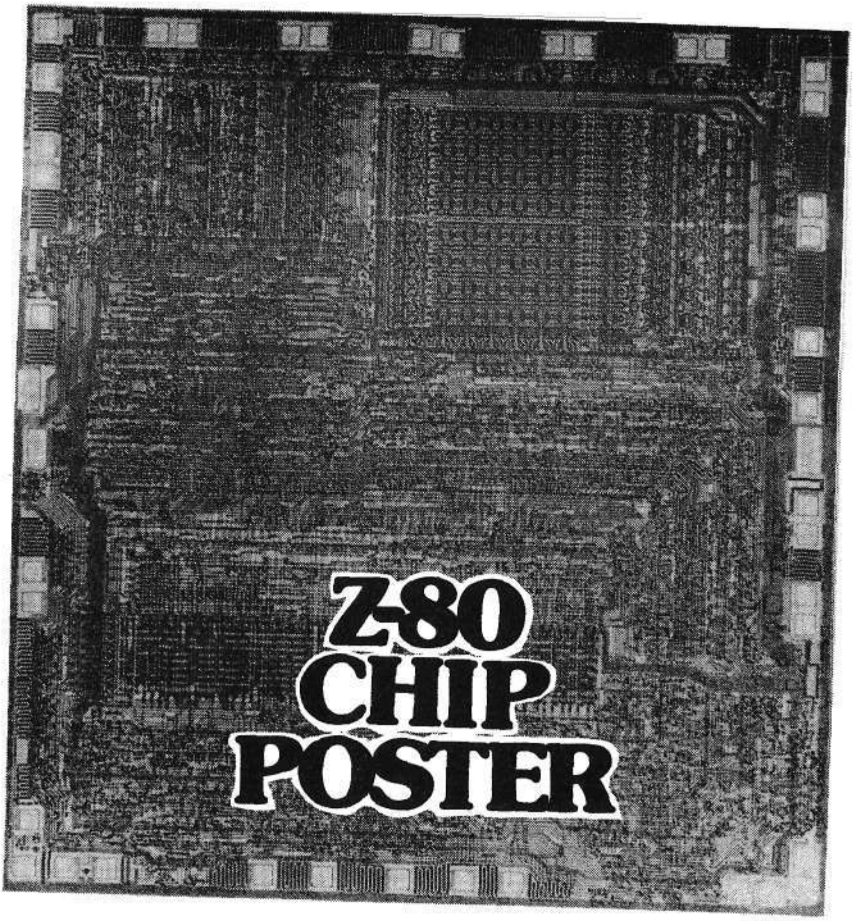
Obviously, this brings us to the second problem. I know of several software houses who have received notices from Radio Shack lawyers regarding the illegal use of TRSDOS 2.2. Specifically, they sold their products with TRSDOS 2.2 included on the disks, minus some of the utility programs.

Those of us who write software do not want to violate the rights of others. But we need to survive. If we could get a license arrangement with Radio Shack, I think we would all be glad to pay up and legally use the TRSDOS 2.2, rather than use it without Shack's permission and bank on the fact that copyright violations are hard to enforce.

Such an arrangement would make TRSDOS 2.2 the standard, solving our first problem. In addition, the extra money Radio Shack and Microsoft got could be used to further upgrade TRSDOS into the excellent operating system it was meant to be.

So, please Radio Shack, stop sending us legal notices. Make us a reasonable offer to use your DOS. We won't refuse.

GWB


Z-80
CHIP
POSTER

OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK

GROWING WEARY

Tracing Unstructured BASIC Programs

Introducing...

MMSFORTH

THE LATEST IN A HIGH-LEVEL LANGUAGE SOFTWARE DEVELOPMENT SYSTEM.

MMSFORTH is a new version of the powerful, fast, and structured **FORTH** language, specifically written for the TRS-80, and offers such features as:

- Double precision integer math
- Reverse polish notation
- Blinking cursor, auto-repeat keyboard
- Virtual Memory!
- Editing capabilities similar to Level II
- Supports Z-80 assembly blocks within the **FORTH** structure

MMSFORTH is available in two versions; a complete version for disk systems and a close approximation to that version which is designed for the Level II, 16K tape system. The disk version offers "virtual memory", supports one to four disk drives, does not require any other disk operating system, and also has both disk and tape input/output capabilities.

TAKE ADVANTAGE OF A POWERFUL STRUCTURED LANGUAGE WITH THE EXECUTION SPEED OF A COMPILER.

MMSFORTH is available now from TSE.

MMSFORTH

INTRODUCTORY OFFER

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

Routines For:

- String Handling
- Graphics
- File Sorting
- Screen Printing

Programs For:

- Game of Life
- Checkbook Balancing
- String Sort
- Number Guessing Game

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

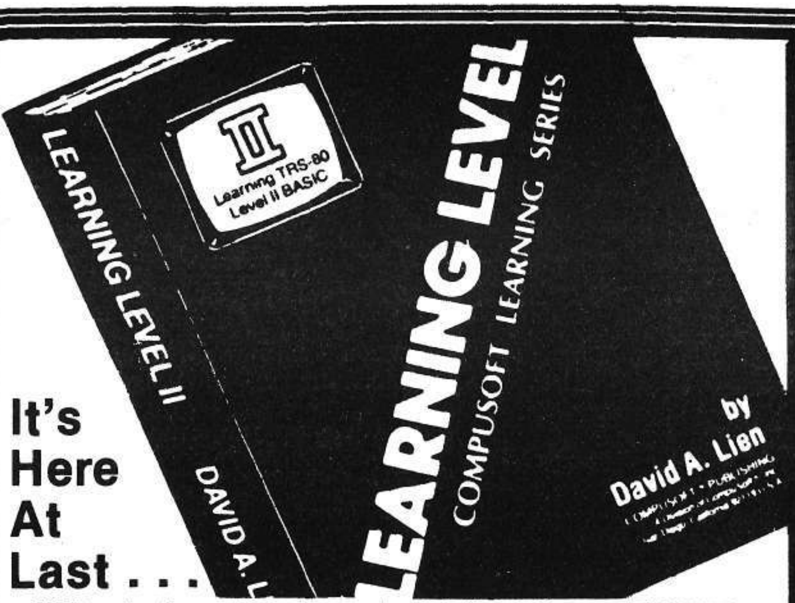
- floating-point package •
- assembler/cross compiler to provide • standard TRS-80 load modules
- large flexible mailing list system •
- generalized data base management system •
- word-processing package (**FORTHWRITE**) •

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

30-DAY INTRODUCTORY PRICE

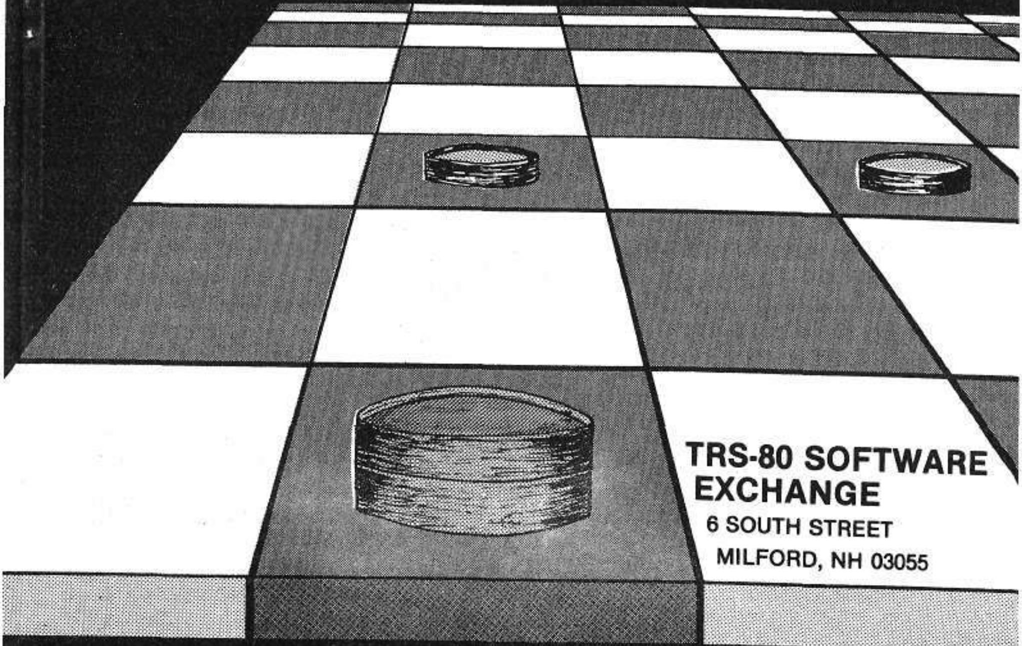
MMSFORTH cassette version, Level II, 16K	$34.95
MMSFORTH disk version, Level II, 16K	65.00
MicroFORTH primer	15.00

TRS-80 SOFTWARE EXCHANGE
6 SOUTH STREET MILFORD, NH 03055


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

THE MEAN CHECKERS MACHINE

by Lance Micklus



TRS-80 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

DOS

by Lance Micklus

An interesting thing happened on my way to TSE to buy some PERCOM disk drives. I think I found out why DOS has so much trouble with "LOST DATA" errors. Let me tell you a story.

Back in October 1978, when I got my first Radio Shack disk drive, I began to get LOST DATA errors. So, I thought I would play with the drive to see if I could improve the situation. I played. No more LOST DATA errors. I still had some of the other problems common to TRSDOS 2.1, but I always worked my way around them.

For example: after I used a disk for a while, I would copy all of the files to a clean DOS disk.

For the most part, I had very little trouble with TRSDOS 2.1 and always thought that the problems others had must be hardware related.

While I was waiting for my second Radio Shack drive, I used several other types given to me on loan, particularly four VISTA 80 drives which were terrible. I had to adjust all of them, especially the motor speed. Two of the VISTA 80 drives were in such bad condition (they were new, yet!) that I couldn't get them to run at all. Two others I babied along.

A couple of months ago, my second Radio Shack drive finally came. It, too, needed some adjustment. The head alignment was not set up correctly. And, in spite of all I did, it would not read disks created by its older brother.

I got a scope, and played with both drives for a couple of days. Finally, I got them to work fairly well. One thing I did discover was that my motor speed was set too fast. It turned out this was a major discovery, but I didn't know it at the time.

Now a strange thing happened. I started getting those darn LOST DATA errors. At this point, TRSDOS 2.2 was out, DOS 3.0 was on order, and I had NEWDOS+ sitting on a shelf. I decided to live with the problem a few more days until TRSDOS 2.2 or DOS 3.0 got here. I didn't want to change DOS systems until I had one I knew I would like.

TRSDOS 2.2 was a disappointment. It did not have the features in it I thought it was going to have. It also did not cure my LOST DATA errors.

I took a long overdue trip to the **TRS-80 Software Exchange** in Milford, New Hampshire and brought myRadio Shack drives with me. The folks at TSE have been bragging about those PERCOMS, and I needed a third drive for my system. After using the PERCOMS at TSE, I struck up a good deal, and traded my Shack drives for the PERCOMS. When I got home, I fired up the PERCOMS and just fell in love with them. But I was ***still*** getting those LOST DATA errors.

It was time to fix this problem once and for all. I made one small change to TRSDOS 2.2, and since then, I have not seen a single LOST DATA error.

Before I tell you what I did, let me first explain what a LOST DATA error is. When you are going to read a sector on a disk, you must first position the head over the correct track, then tell the disk controller which sector you want to read. Now you issue a read command to the disk control.

Next, you enter a loop. There are flags you can test to see how the read operation is going. One of the flags is a DRIVE BUSY signal. This means that the disk drive is still doing something and can not accept any new commands. The other flag is DATA REQUEST. When on, it means that there is one byte from the disk that is ready for the DOS to get and put into memory.

The game is to get bytes from the drive and put them in memory, as they become available, until the drive is no longer BUSY. Then you are done.

The DOS does not have a lot of time to fool around. Bytes from the drive are ready every 78 millionths of a second.

In the expansion interface is a clock which creates a special signal called an interrupt. The signal is sent 40 times a second. When the TRS-80 gets an interrupt signal, it stops whatever it is doing, and goes off to do the interrupt tasks. This includes updating the clock, changing the TRACE, the test for the BREAK key if DEBUG is on, etc. All of this takes a fair amount of time. When all of the interrupt tasks have been completed, the TRS-80 goes back to doing what it was doing before, as if nothing had happened.

What happens if an interrupt occurs while you are reading a sector from the disk? It takes so long for the TRS-80 to do the interrupt tasks that by the time it gets back to reading the disk again, bytes will have been missed. You've got to get them while they're there. When you miss bytes because the TRS-80 was busy doing something else, then the data you've read from the disk is in error because you have LOST some of the DATA.

The DOS handles a LOST DATA error in a very simple way. It just tries to read the sector again. Hopefully, the next time around, the interrupts won't interfere with the READ operation. In TRSDOS 2.2, the DOS will try 10 times to reread the disk, then it gives up and displays the error message LOST DATA.

You would think, Radio Shack thought, *after 10 tries, you should be able to get it. I mean 10 tries should be enough.*

WRONG!!!!!!! Radio Shack, you dummy, haven't you ever heard of MURPHY'S LAW?

It turns out that the speed of the disk drive motor is almost in perfect sync with the interrupts. If you get a LOST DATA error, more than likely the same thing will happen again next time until the interrupts *slowly* get out of step with the READ operations.

When I had my disk drive motors running at the wrong speed (too fast) the interrupts got out of sync with the READ operations very quickly - in less than 10 tries. But when I set the speed back to the correct RPM, the interrupts now were in sync; it took more than 10 tries before they got out of sync, and I got LOST DATA errors.

I guess one way to fix this problem is to set the speed of your disk drives up a little faster. But that's a very poor way to do things.

The second fix is much better. When a LOST DATA error occurs, just go back and read the sector again, and again, and again . . . forever . . . until you get it right. In practice, most reads will not be bothered by the interrupts. On those rare occasions when the interrupts do cause LOST DATA errors, they will work themselves out of sync within a few seconds.

The third solution is to disable the interrupts entirely. This is done by a CMD"T" in Disk BASIC. But then your clock, Debug, TRACE, etc., won't work -- if any of that is important to you.

As for myself, I use interrupt-supported features on my system. So, the second solution seemed to be the way to go. Here's the way to do it.

From BASIC POKE 18107, 202. You only have to do this once after you BOOT the system, i.e., load TRSDOS 2.2. Do **not** use this POKE on any other DOS except TRSDOS 2.2.

Let me warn you of something right now. After you make this POKE into your TRSDOS 2.2, for a while you are going to get scared. Occasionally, the drive will just sit there and spin. Sometimes, for as long as 5 seconds. Then, it will kick in and go on like nothing happened. Don't worry. Everything is fine. It's just MURPHY'S LAW proving to the world that it also operates in Fort Worth, Texas.

There are some other interesting POKES you might want to try out. Especially if you have PERCOM, VISTA 80, or some other non-Radio Shack drive. These POKES step up the access speed. (I'm running my PERCOMS at 10 millisecond access speed, and they work just fine. But instead of purring like a cat, they buzz like a saw when the head moves!) Experiment with the following values of X and Y. If you have a Radio Shack drive, try these POKES out anyway. I found that some Radio Shack drives will work fine at faster access speeds than they are specified for.

For 20 millisecond access, use the following values:

X = 18 and Y = 2

For 10 millisecond access, use these values of X and Y:

X = 17 and Y = 1

Now for the POKE addresses.

NEWDOS: POKE 18007,X and POKE 17484,Y
TRSDOS 2.2: POKE 18007,X and POKE 18122,Y
DOS 3.0: POKE 17746,X and POKE 17856,Y

I hope to have ready, very soon, a patch program to make these changes a permanent part of the DOS. It is more a matter of finding the time to sit down and work it out.

The biggest stumbling block right now has been DOS 3.0 by Randy Cook. If I am unable to get the patch worked out for his DOS, I will skip it and figure it out some other time.

In the meantime, try these changes out. I prefer you don't call me on the phone because I can't afford the time to talk to all of you. But, I would like a letter or post card (with pictures of beach girls) letting me know how these changes worked out for you. A few scribbled lines will do fine. Here's my address:

Lance Micklus
217 South Union Street
Burlington, Vermont 05401

WARNING TO DISK USERS!

You should never turn your system ON or OFF with a diskette in the drive. Even turning the disk drive on or off can create an electronic pulse at the write head and spoil a diskette. To protect your diskettes, remove them before turning any part of the system off. When you turn on the computer, do it in this order:

1) Expansion Interface 2) Printer 3) Computer 4) Screen 5) Disk Drives.

Only after everything is on should you load your diskettes and press the RESET button in order to load the Disk Operating System.

Radio Shack announced several months ago that diskettes were to be removed prior to shutting down the computer, but only recently have problems been verified when turning the system on. To be absolutely safe, never turn any part of the system on or off with a diskette in the drive. (Note: It will not harm your disk drive to run empty on power up.)

Computers are Kids' Stuff

by Michael Potts

If you're looking for a reason to buy a small computer - or you did it already and are feeling guilty - and you have school-age children, I'm here to tell you that you did the right thing. Educationally, computers are the biggest breakthrough since moveable type! Who else but a computer has the patience and precision to drill a kid on his math facts until he's got them cold? Where else can a child find an attentive respondent to help with spelling or geometry or physics? The challenge for you, the owner of a small system: there's very little decent teaching software. Your knowledge of what your child needs help with, your programming skill, and your micro can meet the challenge. Our motto is, "making better children is our most important work."

My job is teaching eight-year-olds in a Montessori school; my classroom helper is a Radio Shack TRS-80 purchased by a school fund-raising drive two years ago - and as far as I'm concerned, TRS-80 is the best help I could hire. I also work with the other teachers at Kinderhaus to develop programming which other children, as young as four, can use to help master acquired skills. Our experience shows that children approach the computer, after their first reluctance, with joy and unusual focus; they work through recess, into lunch periods, past the end of school despite the fact that our programs require them to learn constantly. I suspect that a child who has the good fortune of access to a computer *running good educational software* at home will get a rich education and maintain a healthy hunger for learning. The key, of course, is the software. I'd like to help you get started.

We've reviewed all the software that we could find that's intended for children, and find a general lack of a firm theoretical foundation, a tendency to talk down to the kids, and a confusion of new rules peculiar to each program which stand in the way of the child's learning the real meat of the intended lesson. It's no surprise to me that our use of Maria Montessori's principles makes the lessons in our programs easily accessible to children. Before I dive into the Montessori dogma, though, I want to make a big point: teaching kids is a pretty common-sense undertaking; if you know how to present a subject without contradicting yourself, you'll be OK, but if you can't understand why anybody would want to learn a given lesson, you're not ready to teach it. The first requirement of good education is clarity on the part of the teacher - a clear grasp of substance and goal. The main goal of a Montessori lesson is *independence* - giving the child the tools to solve a problem by himself.

This mainspring of the Montessori approach applies elegantly to the writing of educational software: as programmer and system operator, I want to be able to start a child on a program, and back off, allowing the child to enter into a dialogue with the machine, free from adult intervention. Here is where the machine works its magic - it is uncritical, patient, unflappable, and so the child expands to full potential without fear of disapproval. I'm left free to write the next program, or prepare the kind of original presentation that comes best from a person, or pursue my own education (so that I can stay a step ahead of the speedy little creatures). Or so I can just sit back and observe the child weaving through the intricate web I've woven, watch the wondrous process of learning, anticipate the surprises I've built in, and enjoy the operation of that wonderful creation, the computer.

THE FIRST TIME

How do you start a child on the right path? You assume that the child has never played with a computer, and take that as your first lesson. The best teacher would be another child with lots of computer experience - adults talk too much, explain the wrong things, are too quick to intervene, give right answers, push buttons. If there's no expert kid around, start yours out on something really easy, well within her ability, so that success with the material is assured, and she can use her mastery of the subject to explore the machine . . . what does it do if I give the wrong answer? What does it think the right answer is? Does it know everything? What does it do if I punch in nonsense? Can it teach me anything I don't already know? We use a simple counting routine for our youngest children; we can scale it down so that a child who can distinguish between 1 and 2 'beads' on the screen can start to play with the computer. With these youngsters (age 3 to 4) you must remember to keep the lesson short and the display very simple, because if they can't count past three, reading 'yes' and 'no' is probably about as far as you can take them. Even children as old as 10 enjoy this program if it has some score-keeping and perhaps a timing aspect that forces them to count fast or recognize patterns. Remember: If the main lesson is "know your computer", keep the subject matter simple.

A common false start turns a testing program (What's the capitol of North Dakota? Quick![1]) or an arbitrary game ("You have been vaporized by a laser cannon; Princess Leia weeps for you.") loose on an insecure kid and he is devastated by the machine's mastery. Although both kinds of programming have value, they do not belong anywhere near the beginning of a child's computer experience: build confidence first.

MOVING ON - A REAL TEACHING PROGRAM

Our child has time to build a good relationship with our machine. From here on we must always remember our chief goal, *independence for the child*, and our main priority, *the kid comes first*. The child's education and continued enjoyment of the process are crucial, while the machine is an elegant tool, a fast idiot. Lessons must be rigorous and accurate, so that we won't have anything we're sorry for later. And they must start out well within the realm of possibility for the child. It is very sad to see a child, eager to play the machine, lose enthusiasm and self-contentment when the machine produces problem after problem of unsolvable work. A careful scaling segment at the top of the program, even before the child is invited to the console, must tailor the problems precisely to the ability of the child - this requires sensitivity on the part of the operator.

When the child does come to the computer, let the machine interact and do the talking - *independence*, remember. Our machine starts out:

```
HELLO. THIS PROGRAM MAKES UP ADDITION
PROBLEM S. WHAT SHALL I CALL YOU?
```

and the child types in a name:

```
R2D2
```

(and delights in the interaction between fingers and display, and by the fact that the machine will call her anything she wants.)

```
OK, R2D2, DO YOU NEED INSTRUCTIONS?
```

---

[1] The capitol of North Dakota is Bismarck; Nevada's capitol is Carson City, and Delaware's is Dover. Aren't you glad you know?

Given a 'YES' the program leads the child through a step-by-step dialogue explaining exactly what will happen, what is expected of the child, and what help is available if the child gets stuck. Great care by the programmer in anticipating difficulties, and in writing at the child's level without talking down, will make the process much easier for the child . . . here is a place where education can happen. A fine balance between wordiness and comprehensiveness must be maintained. If the child doesn't understand, she will get "stumped" later, and have to resort to adult help; we don't want to compromise her relationship with the machine. We want to give her independence as early as we can (so she can support us in our old age).

The main program is usually the simplest part. It should be pretty and very, very clear. Computer output at its tamest is almost too much for kids; where concentration is needed the distractions should be kept to a dull roar. Our program presents only this (in large print):

```
R2D2    (something the child can relate to easily)

                    1 2 3
                  + 0 4 5
                  -------
             ??
```

After the child has concentrated, and provided the right answer, she deserves, and will enjoy, some glitter and flash:

```
YES YES YES YES YES YES YES YES YES YES YE
        S YES !!!     123 + 45 =168
 YOU HAVE GOTTEN 7 OUT OF 9 RIGHT, R2D2.
 HERE COMES ANOTHER ONE . . .
```

and make it go by so fast she has to learn to sort out the important information from the garbage she doesn't want to see - a little lesson in preparation for real life.

What if she gets the wrong answer? Here comes another little-used but most important program segment; the machine leaves the name, problem, and wrong answer on the screen and asks:

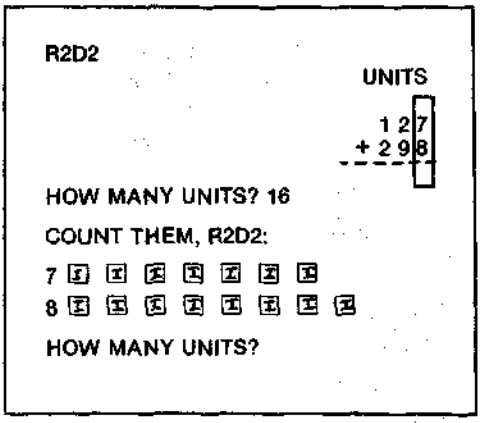
```
DO YOU WANT HELP?
```

Answered in the negative, the screen clears, the problem is restated exactly as before, and the machine awaits another attempt. Answered in the affirmative . . .

THE HELP SEGMENT

The fun begins. When you write the HELP segment, you must transport yourself back to childhood and remember all the mistakes *you* made, and then design a concise and compassionate procedure for getting through the problem without making them. How did you learn to carry? Was it the best way? Most important: How is your kid being taught at school? (If there's any doubt or misunderstanding, ask your child's teacher - before you try to help, be sure that you don't add to the confusion.)

In our Kinderhaus Addition Program we follow Maria Montessori's careful procedure, digit by digit, through the computational jungle. The child's name remains as a talisman of familiarity, and the problem is manipulated so that each consecutive process is isolated exactly as we want the child to do on paper or in her head. The program insists on the right answer at each step, and offers more and more careful help until the problem is found. For example, given the sub-problem 7 + 8 in the units column:



```
R2D2
                                   UNITS
                                    1 2|7|
                                  + 2 9|8|
                                  -----|-|

HOW MANY UNITS? 16

COUNT THEM, R2D2:

7 ▣ ▣ ▣ ▣ ▣ ▣ ▣
8 ▣ ▣ ▣ ▣ ▣ ▣ ▣ ▣

HOW MANY UNITS?
```

(sample display)

And then the routine goes through the same song-and-dance for the tens and hundreds (if any).

There are two points to this: first, make sure that the child knows the procedure precisely and exhaustively, so that mistakes will be minimized in future; second, make the process of resorting to the computer for help so picayune that it becomes much easier for the child to grab paper and pencil and have a go at the problem independently.

The simple expedient (to wallow for a moment in redundancy) of letting the kid get a problem wrong three times, then regale him with the answer, is good data processing, but it's lousy education. Who cares what the answer is: how do you get it? Many educators object to computer-education for precisely this (software based) reason: if the kids can always get the machine to do their work, they'll never learn anything on their own. If you've watched a kid with one of the Texas Instrument teaching calculators (or their equivalents) you've seen how quickly they get the trick to finding the answers without thought. A program without a HELP segment is just an expensive toy. Put it another way: without good instructions, good tailoring of the problems to the individual kid, and a genuinely helpful HELP segment, I don't think you have educational programming.

SCORING AND ENDGAME

Our programs expect a child to complete a set of problems - 15 works well on our 16-line display - before she can stop. This gives a reluctant learner enough repetition so that the concept sticks, and gives any child a nice sense of task-completion. (Our motivated kids love to zip through 5 pages or, better yet, send the machine's memory into overload.) At the end of each set, or page, the machine should offer some reward:

```
***   R2D2'S SCORE   ***

YOU HAVE DONE 15 PROBLEMS AND GOTTEN 14 RIGHT
FOR A SCORE OF 93%.

WELL DONE, R2D2!
```

(this only if the score's better than 90%)

```
DO YOU WANT MORE?
```

Answered affirmatively, the machine cranks out another set, perhaps slightly more difficult. If the child is tired, here is retreat with honor, and the machine's work is well done.

A good place to use the machine's memory is here at the end: a record of the problems the child has attacked, and her results, can help an adult spot areas where some explanation and drill can correct procedure. Our program generates a code during the HELP segment which reads out with the record of problems and pinpoints difficulties the child had in getting through the mechanics. This makes it much easier to spot simple confusions, like the sum of 7 + 8, which invalidate a complex calculation. A homebrew version of this program should include at least the score and review segments because they provide the kid with well-earned reward, and the adult with valuable insight into how to scale the program next time it's used.

And that's the best part: once you've got the program up and running, it can help a kid from elementary facts like 2 + 3 at age 5 to 5-digit mind-benders at age 15; the kid who survives will be fluent in addition.

MOVING RIGHT ALONG

The same care must go into other teaching programs. A multiplication program might drill facts (like 6 * 8 © ?) only, stress mastery by timing the response and giving a readout at completion of each problem, and offer help by reviewing the appropriate times-table upon request. Another multiplication program could lead the child through the jungle of dynamic double-digit multiplication. Similar programs would help in subtraction and division. We've adapted similar approaches to spelling and grammar. The fact is that any symbolic process which requires repetition and correction for mastery will benefit from computer instruction - the programmer's time and ingenuity are the only limitations; even a Level I Basic 4K memory TRS-80 can handle a lot of teaching.

Practical skills - estimation and measurement, time-telling, change-making, games based on simple computation and judgment of odds - build incredible facility in their devotees. My kids have a favorite called the 10-by-10 Graph Game, which allows the child to assemble block graphics on the screen by specifying a cell's coordinates (you remember you x's and y's). The machine fills it in (or given a negative, erases it.) Even six-year-olds grasp the concept and spend hours building patterns far more precise and orderly than their developing motor skills would permit on paper: the machine's absolute obedience and uncritical execution provide a perfect chance for what Maria Montessori called the "key experience" of ordering the universe. When the child needs coordinate geometry at some later date it will seem like (if you'll pardon the expression) child's play.

Logic games - tic-tac-toe against the computer, number-guessers, Frogs, even Chess at a tutorial level - because they are games, stimulate the child to play with the machine; they are valuable as education only if the game is non-trivial and has some didactic content. As soon as the concept is learned and mastered, the only value that remains is the child's delight at 'beating the machine' . . . which is short-lived. We find that our children, given a choice between games and 'teaching' programs, generally go for the latter.

Special use programs can be fun to write, and worth the programming overhead even if used only once, if they demonstrate a concept cleanly. The availability of cheap non-volatile storage (casettes) makes it possible to write a library of such programs and use them as the subject comes up for a succession of kids. We have a lovely triangle-classifying program which uses a decision table to separate right isosceles from equilateral by asking three yes-no questions. We fire it up 2 or 3 times a year to reinforce the names of the triangles, and once a year to illustrate a talk on what a decision tree looks like and how it works. We have a ball-thrower which graphically demonstrates the effects of gravity and 'muzzle velocity' on a projectile. The computer functions in this genre of program as an animated, tireless blackboard. In writing such a program, you get to polish your ability to play the computer like a piano; the kids get the benefit of a dynamic and precise demonstration of a concept.

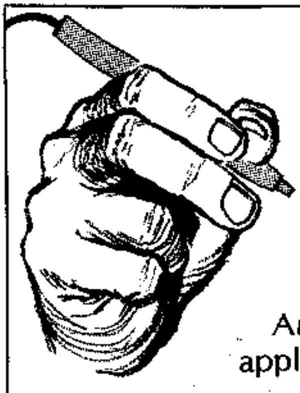
Testing programs are good education only if they contain a teaching component; otherwise their function is diagnostic and they don't belong, in our opinion, in a home or a school. A good example of a pure testing program is the oft-published State Game, wherein the student is supposed to supply, untutored, the capitols of Nevada and Delaware. If the program supplied corrections of bad guesses, and, say, a list of 5 good guesses to choose from, and required the child to type in the name, then we'd have some learning going on. An even better path might be to reverse the process and have the child find the state that matches the capitol city provided by the computer - it's much more likely that a knowledge of how to spell the names of the states will come in handy some day. Further refinement might group the states georaphically or in some other instructive way, and set up a hint system which provides useful facts in the process. The teaching content remains tangential to the game - the testing may motivate, but it shouldn't be taken very seriously. Spelling games are hard to handle without resorting to testing - which is probably not the most productive way to approach any but the most idiosyncratic words. Nevertheless, ingenuity (and lots of string manipulation) will undoubtedly win out. Testing is a sinsiter function, and should be approached with caution.

Programs that require the child-user to complete the program for their result offer a transitional step to full-bore programming, which is the logical conclusion of a Montessori approach to computers. We've played with a program that coaches the child to encode the line that defines the function she has already worked out, then invites her to enter the range over which the computer should calculate the results. Our 8-year-olds type in a function like z=3*r-2*c where r is the row and c the column, ask for r from 1 to 5 by ones, and the same for c, and the computer belches out a 5-by-5 matrix for the child to match against the one she has already calculated herself. The program coaches exactly what must be typed in - line number, expression, and re-entry code (GOTO400) - and runs flawlessly if *and only if* the child follows directions precisely. From this experience it's just a matter of time before the kid shoves you out of the way and starts composing her own programs.

THE COMPUTER GROWS UP

When all is said and done, the most important lesson that your kid will learn from the computer is the computer itself. As time goes on, and those who are now children mature, matriculate, and stumble into the labor market, computers and terminal-oriented work stations and a whole generation of smart machinery will rule the working world (or I miss my guess). Many older workers are facing their fears and learning to work in harmony with an electronic partner, and retraining for the skills that such work requires. Education has a tendency (you may have noticed . . .) to lag a dozen years or so behind the state of the art - and our children stand a good chance of falling right into the biggest gap ever, unless we do something about it. Not to put too fine a point on it: if our kids start investigating computers now, from the outside as program users, they'll be ready and eager for some heavy-duty programming before they're out of school, and then go on to work on the work-designing end of things, rather than being told what to do by the machines. We come full circle to our original mainspring goal: independence and self-sufficiency are, from first to last, the keys to growth and joy.

And who's going to complain if we can justify our own microcomputer to play with into the bargain?



Your Market
For New And Used
Microcomputer
Equipment

HARDSIDE™

NEW! TRS-80 Computers 10% off!

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

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

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

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

As an Introduction to Hardside™ we are offering two special packages:

1) TRS-80 expansion interface with our 16K RAM, single PERCOM disk drives with cable, and NEWDOS disk operating system. $830. Save $167.
2) TRS-80 Level II, with our 48K RAM, dual PERCOM disk drives with cable, Centronics 779 tractor printer with cable, and NEWDOS + disk operating system. $2999. Save $1144.

These two offers are good for 60 days only, and expire November 30, 1979.

We will soon be opening a retail location for our New England customers. Watch for details.

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

NEW			USED	
UNIT	RETAIL LIST PRICE	HARDSIDE DISCOUNT PRICE	WE'LL PAY	SELL
Level I 4K	$499	$449	$275	$375
Level II 4K	$619	$557	$370	$475
Level I 16K	$729	$656	$365	$475
Level II 16K	$849	$764	$500	$650
Level II 16K, No Keypad -		$669	$450	$600
EXPANSION INTERFACE				
. . . . 0K	$299	$269	$175	$240
. . . . 16K	$448	$369/$403	$225	$315
. . . . 32K	$597	$469/$537	$275	$390
DISK DRIVES				
Radio Shack #0	$499	$469	$275	$399
Radio Shack #1	$499	$469	$250	$375
Percom TFD-100		$399		
Percom TFD-200 dual density 197K		$675		
PRINTERS				
* Centronics 779 Tractor Feed	$1559	$1000	$650	$850
* Centronics P1	$499	$399	$250	$325
Quick Printer II	$219	$197	$125	$175
Line Printer II	$999	$899		
Line Printer III	$1999	$1799		
* Requires Cable R/S 26-1401	$39	$35	$20	$30
TRS-80 HARDWARE ACCESSORIES				
Telephone Interface II	$199	$179	$100	$150
16K Memory Kits	$149	$99		
RS232C Serial Interface	$99	$89	$50	$75
TRS-232 Interface		$49		
Data Dubber		$49		
Line Cord Suppressor/Filter		$32		

Prices do **not** include shipping

COD orders require
25% cash deposit

603-
673-5144

Your Market For New And Used Microcomputer Equipment

HARDSIDE™

6 South Street Milford, NH 03055

Shrayer Sues Vector Graphic

For Misappropriation of Electric Pencil, Seeks $1,094,000

Michael Shrayer Software, Inc. filed suit in Los Angeles Superior Court on August 8, 1979, alleging breach of contract, fraud and deceit against Vector Graphic, Inc. The suit seeks recovery in the sum of $1,094,000, and an injunction prohibiting Vector Graphic from distributing either the **Memorite II** or **Word Management System II.** The complaint further charges that the Vector Graphic **Memorite II** and **Word Management Systems II** incorporate, virtually byte for byte, the **Electric Pencil II** editor.

The suit further seeks recovery for quantities of the original version of the **Electric Pencil** which Shrayer charges were distributed by Vector without payment of the agreed license fees.

Michael Shrayer, when reached for comment, stated: "We intend to do whatever is necessary to protect our rights. Piracy of software, if left unchallenged, will destroy much of the incentive necessary for the creation of quality software, and will cause irrepairable damage to the industry."

TYPING TUTOR

by 80 US

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.

TRS-80 SOFTWARE EXCHANGE
6 SOUTH STREET   MILFORD, NH 03055

Basic Disk Cross Reference Utility

by L. C. Chesbro

Have you ever tried to modify someone else's program (or maybe one of yours that you haven't touched for six months) and wondered what the effect of changing a subroutine, deleting or inserting a line will be? Without spending a lot of time to understand fully the flow of the program, such changes can be a risky proposition. Here is a utility to take some of the guesswork out of it.

Those who work on larger computers take cross-reference listings for granted. If they want to know which lines reference a particular subroutine, they just look it up. However, the TRS-80 user hasn't had that luxury, until NOW!

The CROSS REFERENCE UTILITY will read the program you specify from the disk file and accumulate an ordered list of all program lines and the line(s) from which they were referenced. Upon completion, the cross-reference list will be printed on the line printer (see example output). There is no limit to the number of references for a particular line, but the total number of lines referenced by other lines is limited to 400. The file you cross-reference must be a BASIC compressed program (not in ASCII format).

Execution is simple: you enter the name of the disk file where the source program is stored. The hardware and software required is a 32K DOS system running TRSDOS 2.2. A line printer is recommended, but if you do not have one, either change the LPRINTs in lines 110-200 to PRINTs, or insert the following statement:

```
5   POKE 16422,PEEK(16414): POKE 16423,PEEK(16415)
```

The latter causes all output which would be printed to appear on the CRT screen.

```
10 'TRS-80 DISK BASIC LINE NUMBER CROSS REFERENCE UTILITY
20 'L. C. CHESBRO  07/08/79
30 CLS: DEFINT A-Z: DIM LN(400,5)
40 T=0: I1=5: I=0: J=0: K=0: T$="": M$=", #####": TL=0
50 LINEINPUT "FILESPEC: ";FS$: OPEN "R",1,FS$
60 FIELD 1,254 AS C$, 2 AS D$
70 FIELD 1,255 AS A$, 1 AS B$
80 GET 1,1: LL=CVI(MID$(A$,4,2)): N=1
90 IF LEFT$(A$,1)<>CHR$(&HFF) THEN PRINT "THE FILE IS NOT A COMP
RESSED BASIC PROGRAM": GOTO 250
100 GOSUB 260
110 IF T=0 THEN 140 ELSE IF T<141 THEN 100
120 IF T=159 OR T=149 OR T=145 OR T=141 OR T=202 THEN GOSUB 410
: GOTO 110
130 GOTO 100
140 LPRINT CHR$(12);"   CROSS-REFERENCE LISTING FOR ";FS$;"  ";T
IME$
150 LPRINT CHR$(138);" LINE #   REFERENCED IN LINE(S):"
160 LPRINT CHR$(138)
170 FOR I=0 TO TL
180 IF I<>0 THEN IF LN(I-1,0)=LN(I,0) THEN LPRINT TAB(9);USING "
#####";LN(I,1);: GOTO 200
190 LPRINT USING "  #####";LN(I,0);LN(I,1);
200 FOR J=2 TO 5
210 IF LN(I,J)<>0 THEN LPRINT USING M$;LN(I,J);: NEXT J
220 IF LN(I+1,0)<>LN(I,0) THEN 240
230 I=I+1: FOR J=1 TO 5: IF LN(I,J)<>0 THEN LPRINT USING M$;LN(I
,J);: NEXT J
240 LPRINT: NEXT I
250 CLOSE: END
260 I1=I1+1: IF I1>256 THEN GOSUB 400
270 IF I1=256 THEN T$=B$: GOTO 290
280 T$=MID$(A$,I1,1)
290 T=ASC(T$): IF T<>0 THEN RETURN
300 I1=I1+4: IF I1>257 THEN 350
310 IF CVI(MID$(A$,I1-3,2))=0 THEN RETURN
320 IF I1<=255 THEN 390
330 IF I1=256 THEN OL=LL: LL=CVI(D$): GOTO 260
340 H$=B$: GOSUB 400: OL=LL: LL=CVI(H$+LEFT$(A$,1)): GOTO 260
350 ON I1-257 GOTO 360, 370, 380
```

```
360 IF CVI(D$)=0 THEN RETURN ELSE GOSUB 400: GOTO 390
370 H$=B$: GOSUB 400: IF CVI(H$+LEFT$(A$,1))=0 THEN RETURN ELSE 390
380 GOSUB 400: IF CVI(LEFT$(A$,2))=0 THEN RETURN
390 OL=LL: LL=CVI(MID$(A$,I1-1,2)): GOTO 260
400 I1=I1-256: N=N+1: GET 1,N: RETURN
410 LN$="": OL=LL
420 GOSUB 260: IF T$=" " THEN 420
430 IF T$=>"0" AND T$<="9" THEN LN$=LN$+T$: GOTO 420
440 LN=VAL(LN$): IF LN=0 THEN RETURN
450 I=TL/2: IF LN(I,0)>LN THEN I=0
460 FOR I=I TO 400: IF LN(I,0)=LN THEN 530
470 IF LN(I,0)=0 THEN TL=I: GOTO 520
480 IF LN(I,0)<LN THEN NEXT I: PRINT "MORE THAN 400 LINES REFERENCED": GOTO 140
490 FOR J=TL TO I STEP -1
500 FOR K=0 TO 5: LN(J+1,K)=LN(J,K): NEXT K
510 NEXT J: FOR J=1 TO 5: LN(I,J)=0: NEXT J: TL=TL+1
520 LN(I,0)=LN
530 FOR J=1 TO 5: IF LN(I,J)<>0 THEN NEXT J: I=I+1: GOTO 460
540 LN(I,J)=OL: IF T$="," THEN 410 ELSE RETURN
```

PROGRAMMING HINT

Ever wish you could get a directory listing on your printer?

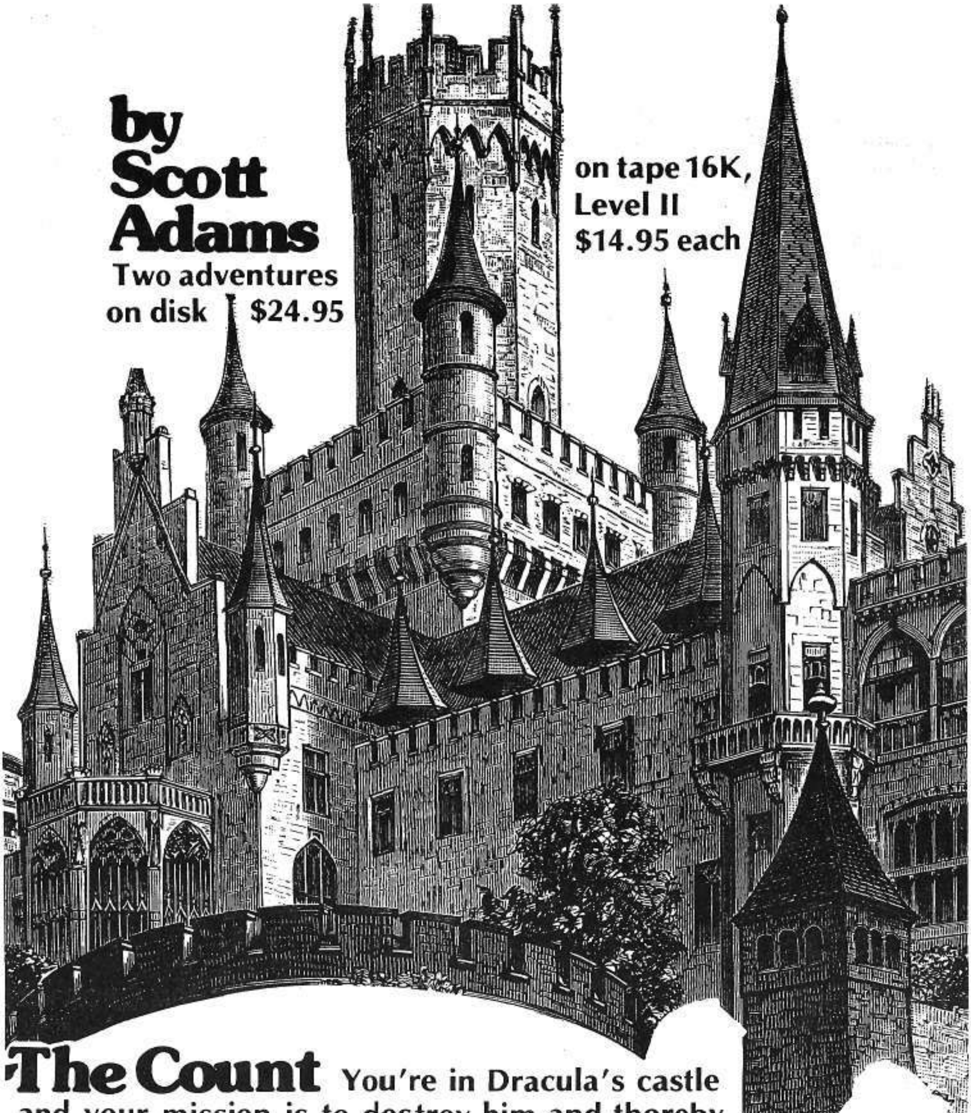
Here is a way!

In BASIC, enter:

```
POKE 16414,PEEK(16422): POKE 16415,PEEK(16423)
```

Then enter:

CMD"S" and issue your DIR command normally. Everything that would normally appear on the CRT, now goes to the printer (including what you type). To restore normal operation, press the RESET button and re-boot DOS.



Renumbering Program For TRSDOS

by Rocky Smolin

Yes, yes. I know what you're saying. A little late for a TRS-80 renumbering program, isn't it? Well, this one should be instructive from a programming standpoint, and if only for that reason, I hope you enjoy reading about it.

I tried to make it as simple and straightforward as possible keeping the bells and whistles to a minimum. The program executes in two passes. In the first pass each of the statements are read and an old and a new statement number are stored in equivalent positions in the array 'IA'. Then the program is read for a second time, this time replacing the old statement number with new ones and scanning the line for other statement numbers, replacing them with the new statement numbers.

Page number references below refer to ***TRSDOS and Disk Basic Reference Manual*** put out by Radio Shack. I highly recommend this excellently written and informative manual (they evenput in an index this time).

To get the renumbering program to work properly, you must first store the program to be renumbered in ASCII format, like this: SAVE "PROGRAM:1",A . It's the ,A that does it. Normally to save space on the disk, BASIC will take all the keywords like GOSUB and PRINT and convert them to a one-byte code. It also takes the statement numbers, converts them from string to integer and inserts them in the output file.

If you use the DISKDUMP program to look at a program you've saved on disk, it will be full of strange looking codes. When you load a program, the BASIC interpreter does the opposite operation - it translates the codes back into keywords again. The whole process is transparent to you, the user.

However, since this renumbering program isn't smart enough to deal with the condensed format, you must store programs to be renumbered in the ASCII, that is, uncondensed format. After you do this, look at the file on disk. You'll see all the keywords and statement numbers in the disk file just as they appear in your program. However, if you do a 'DIR (A)' which will give you a directory listing with program sizes (see page 4-16 for a complete explanation of the 'DIR' command), you'll see that the ASCII format programs take up significantly more space on your disk than the compressed format. See page 7-31 for more info on the ASCII format.

The program up to line 350 is pretty self explanatory: some housekeeping, some necessary inquiries, and file openings. The 'LINE INPUT' command in line 360 is required instead of a simple 'INPUT' because 'INPUT' alone will stop reading when it hits a comma, and leaves the rest of the statement behind (no good obviously, when reading a BASIC language program). 'LINE INPUT' on the other hand, reads until it sees a carriage return code, so the whole statement ends up in A$. Try it both ways and you'll see what I mean. More on 'LINE INPUT' can be found on page 7-42.

In the first pass all I'm doing is storing the new line number in row one of array IA, and the old line number in row two. Notice how easy it is to extract the line number from the statement. 'VAL (A$)' does it without having to scan each byte for a valid numeric character, and build the number byte by byte. Try some mixed alphabetic and numeric character strings as arguments for the VAL function and see how it pulls a number out of a string . . . very interesting, and handy, too, for an application like this. The ***Level II Basic Manual*** has the write up on page 5-8.

In the second pass, the program looks for the keywords. The subroutine at 920 has already read the keywords into the string array B$. This arrangement makes it easy to add keywords to the scanning process. Simply increase the size of B$, add the words to the DATA statement in 930, and raise the upper limits of the 'FOR' loops in statements 920 and 520.

The 'INSTR' function in line 530 returns a zero if the string being scanned for is not present. If it is present, 'INSTR' returns the position of the byte where the string starts. See page 7-15 for the complete story on 'INSTR'.

Statements 590-630 format the new line with the new statement number and write it to disk. Since the STR$ function puts a leading blank on the number which is converted to string format, the second statement in line 600 strips it off. Note the use of functions embedded within functions. The two statements could have been combined thusly:

```
N$=RIGHT$(STR$(IA,(1,N)),LEN(STR$(IA(1,N))))
```

but it would take much longer to execute (not to mention write and debug), because of the redundant use of the STR$ function. It's also more difficult to read.

The subroutine which starts at line 710 removes the old statement number reference and replaces it with the new one. 'K1' is the integer form of the old statement number, 'K2' is the length. The loop 760-780 searches the old array for the old statement number and pulls out the new statement number into KK$. Yes, I could have used a binary search routine, which would have been faster and more clever. If you can devise one which is fast, compact, and reliable, send it in. Maybe Mr. Robitallle will print it.

At line 850 is the subroutine which handles lists of statement numbers which occur in computed GOTO's and GOSUB's. It keys on commas and if it finds one it calls the change subroutine at line 710 with 'I' pointing to the next number to be changed.

And that's it, folks, The moral of the program is 'keep it simple'. You can easily outclever yourself in a program and end up with lots of debug for a little processing. Remember, the shortest distance between two points is still a straight line.

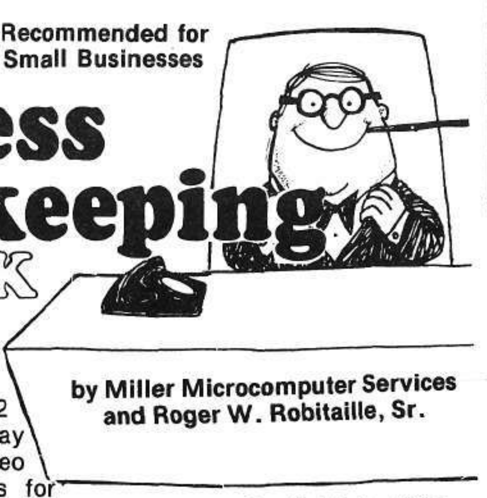
```
10 ' RENUMBERING PROGRAM FOR TRS-80 DISK SYSTEMS
20 ' WRITTEN BY ROCKY SMOLIN
30 '             7325 DRAPER
40 '             LA JOLLA, CA. 92037
50 '
60 ' DO SOME HOUSEKEEPING FUNCTIONS
70 '
80 ' FOR PROGRAMS WITH MORE THAN 200 STATEMENTS INCREASE THE
90 ' SECOND ARGUMENT IN IA FROM 200 TO THE LARGER NUMBER
100 '
110 CLEAR 2000:DEFINT I-N:DIM IA(2,200),B$(5):K=1:GOSUB920
120 '
130 ' ASK THE VITAL QUESTIONS
140 '
150 INPUT "WHAT IS THE OLD PROGRAM NAME";F1$
160 INPUT "WHAT DRIVE IS IT ON";D1$:F1$=F1$+":"+D1$
170 INPUT "WHAT IS NEW PROGRAM NAME";F2$
180 INPUT "WHAT DRIVE IS IT ON";D2$:F2$=F2$+":"+D2$
190 INPUT "WHAT IS STARTING STATEMENT NUMBER";IS
200 INPUT "WHAT IS INCREMENT";II
210 '
```

```
220 ' SET ERROR TRAP
230 '
240 ' ON ERROR GOTO 940
250 '
260 ' OPEN FILES
270 '
280 PRINT "OPENING ";F1$
290 OPEN "I",1,F1$:N=0
300 PRINT "OPENING ";F2$
310 OPEN "O",2,F2$
320 PRINT "FIRST PASS":PRINT
330 '
340 ' READ SOURCE FILE; STORE STATEMENT NUMBERS IN ARRAY
350 '
360 LINE INPUT #1,A$
370 PRINT@960,LEFT$(A$,60);
380 N=N+1
390 IA(1,N)=IS:IS=IS+II:IA(2,N)=VAL(A$)
400 IF EOF(1) GOTO 420
410 GOTO 360
420 PRINT:PRINT "FIRST PASS COMPLETE":
430 NS=N
440 '
450 ' SECOND PASS
460 '
470 CLOSE 1:PRINT "SECOND PASS":OPEN "I",1,F1$:PRINT:N=0
480 LINE INPUT#1,A$:N=N+1:K=LEN(A$)
490 '
500 ' SCAN FOR KEYWORDS
510 '
520 FOR J=1 TO 5:X=1
530 I=INSTR(X,A$,B$(J)):IF I=0 GOTO 550
540 I=I+LEN(B$(J)):GOSUB 710:GOSUB 850:X=I:GOTO 530
550 NEXT J
560 '
570 ' FORMAT OUTPUT LINE AND WRITE TO DISK
580 '
590 I=LEN(STR$(VAL(A$)))
600 N$=STR$(IA(1,N)):N$=RIGHT$(N$,LEN(N$)-1)
610 B$=N$+" "+RIGHT$(A$,(K-I))
```

```
620 PRINT#2,B$: PRINT B$
630 IF EOF(1) GOTO 670 ELSE GOTO 480
640 '
650 ' END OF PROGRAM
660 '
670 PRINT:PRINT "SECOND PASS COMPLETE":CLOSE1,2:END
680 '
690 ' CHANGE FROM OLD STATEMENT NUMBER TO NEW
700 '
710 IF I=K THEN RETURN
720 IF MID$(A$,I,1)=" " THEN I=I+1:GOTO 710
730 IF MID$(A$,I,1)<"0" OR MID$(A$,I,1)>"9" THEN RETURN
740 K1=VAL(MID$(A$,I,5))
750 K2=LEN(STR$(K1))-1
760 FOR K3=1 TO NS
770 IF IA(2,K3)=K1 THEN KK$=STR$(IA(1,K3)):GOTO 790
780 NEXT:PRINT "STATEMENT #";K1;" NOT FOUND":STOP
790 A$=LEFT$(A$,I-1)+RIGHT$(KK$,(LEN(KK$)-1))+RIGHT$(A$,(K-I-K2+
1))
800 I=I+LEN(KK$)-1
810 K=LEN(A$):RETURN
820 '
830 ' HANDLE STRINGS OF STATEMENT NUMBERS
840 '
850 IF I=K RETURN
860 IF MID$(A$,I,1)=" " THEN I=I+1:GOTO 850
870 IF MID$(A$,I,1)<>"," THEN RETURN
880 I=I+1:GOSUB 710:GOTO 850
890 '
900 ' LOAD KEYWORDS INTO STRING ARRAY
910 '
920 FOR I=1 TO 5:READ B$(I):NEXT I:RETURN
930 DATA "GOTO","GOSUB","THEN","ELSE","RESUME"
940 PRINT "PROGRAM ERROR=";ERR/2+1;"DISK ERROR=";ERR/2;"LINE NUM
BER=";ERL:STOP
```

Small Business Bookkeeping FOR DISK

Recommended for Small Businesses



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

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

Available for 32K
Disk Systems - $24.95

TRS-80 SOFTWARE EXCHANGE
P.O. BOX 68, MILFORD, NH 03055

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.

Level II, 16K $14.95 - Disk $19.95.

TRS-80 SOFTWARE EXCHANGE
6 SOUTH STREET MILFORD, NH 03055

INTRODUCTION TO TRS-80 GRAPHICS

by Don Inman

About the Book . . .

For some time now, graphic displays on microcomputers have been out of the reach of hobbyists because of their complexity and high cost. In this book, author Don Inman will show you how, with a minimal knowledge of the BASIC computer language and your TRS-80 computer, you can create graphic displays that only a few years ago were the exclusive turf of the big computer owners. The book begins with the basics and works from line drawings through geometrics and right on up to moving figure animation and more advanced operations. A great handbook on computer graphics for microcomputer owners at all levels of experience. **$7.95 + $1.00 shipping and handling**

TRS-80 SOFTWARE EXCHANGE
6 SOUTH STREET MILFORD, NH 03055

Squish-2

by Bill Everett

The following program is an improved version of SQUISH, which appeared in the May issue of PROG/80. The primary new feature that I added is the ability to remove remarks from a basic program.

The original features of SQUISH are listed below.

1) Remove all unnecessary blank spaces.
2) Write the squished program on the original file.
3) Inhibit the BREAK key.
4) Load the squished program at the end of the squish operation.

The modified SQUISH-2 has the following features:

1) It is written for a DISK based TRS-80.
2) Removes all unnecessary blank spaces from a basic program stored in the ASCII format on disk.
3) The user may select the file name that the squished program is output to. The original name may be used and the output will destroy the original or a different file name can be used which allows the retention of the original file.
4) The BREAK key is no longer inhibited. This function was only operational on some DOS systems and would destroy other DOS systems.
5) The user may optionally delete all remarks. If a remark is the only thing left on a line, the line number and a " ' " are left in case the program jumps to that line.
6) The program checks to see if the input file has been saved in the ASCII format, and if not, gives an error message.
7) The programmer may load the squished program at the end of SQUISH-2 if he chooses or he may go on to squish another program.
8) A full set of instructions has been included.
9) During operation the current line number being squished has been added to the display.
10) The display has been cleaned up.

```
10 'SQUISH2
20 'ORIGINAL PROGRAM FORM MAY 79 PROG 80
30 'DATE LAST MODIFIED 7-2-79
40 'MODIFIED FROM THE ORIGINAL BY:  BILL EVERETT
                                    14645 NE 34TH C-24
                                    BELLEVUE, WA.  98007
                                    (206) 883 8475
50 CLS:PRINT@23,"SQUISH-2
60 PRINT@128, "THIS PROGRAM REMOVES ALL EXTRA SPACES FROM A BASI
C
PROGRAM.  IT IS ALSO CAPABLE OF REMOVING ALL REMARKS FROM A
PROGRAM IF THAT OPTION IS SELECTED.  IF A LINE ONLY CONTAINS A
70 PRINT "REMARK IT WILL NOT DELETE THE LINE NUMBER IN CASE THER
E IS
A BRANCH TO THE REMARK LINE FROM SOME OTHER LINE IN THE PROGRAM
80 PRINT: PRINT "IF THE BREAK KEY IS DEPRESSED DURING EXICUTION
OF THIS
PROGRAM THE OUTPUT FILE WILL NOT BE COMPLETE NOR WILL THE OUTPUT
FILE BE CLOSED.
THIS PROGRAM TAKES A LONG LONG TIME TO EXECUTE.
90 INPUT "
DEPRESS ENTER TO START";A$
100 DEFINT A-Z
110 CLEAR MEM/2
120 CLS
130 PRINT@128,"THE INPUT PROGRAM MUST HAVE BEEN SAVED UNDER THE
ASCI OPTION
140 PRINT@256,"PROGRAM NAME TO BE SQUISHED";: INPUTFI$
150 PRINT@384,"THE SQUISHED PROGRAMS NAME MAY BE THE SAME AS THE
 INPUT
PROGRAMS NAME BUT IF IT IS THE INPUT PROGRAM IS LOST.  MAKE SURE
THERE IS ENOUGH FREE DISC SPACE FOR THE SQUISHED PROGRAM IF A
DIFFERENT NAME IS USED.
160 PRINT: INPUT"UNDER WHAT NAME DO YOU WANT TO SAVE THE PROGRAM
";FO$
170 INPUT "
DO YOU WANT TO DELETE ALL REMARKS";A$: IF LEFT$(A$,1)="Y" THEN K
K=1
180 CLS: PRINT CHR$(23): PRINT@142,"SQUISH
190 OPEN"I",1,FI$
```

```
200 OPEN"O",2,FO$
210 IFEOF(1)THEN460
220 LINEINPUT#1,A$
230 XX=VAL(A$)
240 IF XX<1 THEN CLOSE: CLS: PRINT@512,"YOU ARE READING A BAD PR
OGRAM": END
250 PRINT@448,"LINE NUMBER BEING READ";TAB(23);XX;
260 RC=RC+1:AA=AA+LEN(A$)
270 PRINT@384,"LINES READ";TAB(23);RC;
280 FORC=1TOLEN(A$)
290 PRINT@512,"SCANNING POS.";TAB(23);C;" ";
300 PRINT@576,"COMPRESS SWITCH";TAB(24);:IF SW=0PRINT"ON "; ELSE
 PRINT"OFF";
310 IFMID$(A$,C,1)=CHR$(34)ANDSW=0THENSW=1ELSEIFMID$(A$,C,1)=CHR
$(34)THENSW=0
320 PRINT@640,"CHARACTERS ELIMINATED";TAB(23);TSE;
330 IF KK=0 GOTO 370
340 IF SW=0 AND MID$(A$,C,1)="'" THEN GOTO 360
350 IF SW=0 AND MID$(A$,C,3)="REM" THEN 360 ELSE 370
360 TSE=LEN(A$)-C+TSE: IF C<8 THEN B$=B$+"'": TSE=TSE-1: GOTO 38
0: ELSE GOTO 380
370 IFSW=0ANDMID$(A$,C,1)=" "ORMID$(A$,C,1)=CHR$(10)THENTSE=TSE+
1:NEXTELSEB$=B$+MID$(A$,C,1):NEXT
380 IF RIGHT$(B$,1)=":"THEN B$=LEFT$(B$,LEN(B$)-1):TSE=TSE+1
390 PRINT#2,B$
400 WR=WR+1
410 PRINT@704,"LINES WRITTEN";TAB(23);WR;
420 PRINT@768,"BYTES SCANNED";TAB(23);AA;
430 B$=""
440 SW=0
450 GOTO210
460 CLOSE
470 PRINT@896,"DEPRESS THE 'L' KEY TO LOAD THE SQUISHED PROGRAM"
;:A$=INKEY$: IF A$="" THEN 470 ELSE IF A$="L" THEN CLS: LOAD FO$
: ELSE CLS: END
```

Get a LOAD of this!

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

SoftSide™

Your BASIC software magazine for TRS-80

On Cassettes

That's right! Now, all out monthly programs will be available on audio cassettes, for only about a dollar apiece - **SATISFACTION GUARANTEED.**

The price for 6 months of cassettes **plus** 6 months of SoftSide magazine is only $38.00. If you're already a subscriber to the magazine, you can enroll for the cassettes only, at a cost of $30.00 for 6 months.

So, if you want to combine the best programming available for the TRS-80 with the convenience of pre–recorded program cassettes, send your order to SoftSide today, and leave the coding to us!

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

ALL RIGHT! Enroll me for 6 months of SoftSide on cassette. My full payment is enclosed as indicated below;

☐New Subscriber - $38.00 ☐Subscriber - $30.00

☐Check ☐Money Order

☐Charge to Master Charge/VISA

Exp. date__________ Interbank#__________

Signature__________

NAME__________

ADDRESS__________

CITY, STATE__________ ZIP__________

Old Business

by Lance Micklus

I guess I didn't know what I was getting into when I published that upper-lower case mod back in the first issue of PROG 80. The letters and phone calls it generated were incredible.

One of the things we found out was that all keyboards are not alike. To do the modifications on some keyboards, you will have to add a pull up resistor of between 100 to 330 ohms from pin 13 to pin 16 of Z27. Try the modification without the resistor. If youget thunderbolts and Greek letters, put the resistor in.

This information is supplied with ST-80D, KVP, etc. I am now ready to leave well enough alone. But . . .

I got a call from John Gersbach here in Burlington. He wanted to do my modification but he did not like the way I OR'd the outputs of the chips. He thought he might be able to find a better way of doing this by adding an OR gate. I had to agree with him that his idea would be better. A few days later, he sent me his improved modification of my modification.

I haven't tried it out on my keyboard, but Mr. Gersbach does have it running on his keyboard. As near as we can figure, it should work on all keyboards the same way. (Gulp!) Assuming that's true, here is yet another upper-lower case, make-at-your-own-risk-then-tell-us-what-happened modification, that will void your Radio Shack warranty. (I think the lawyers are happy now.)

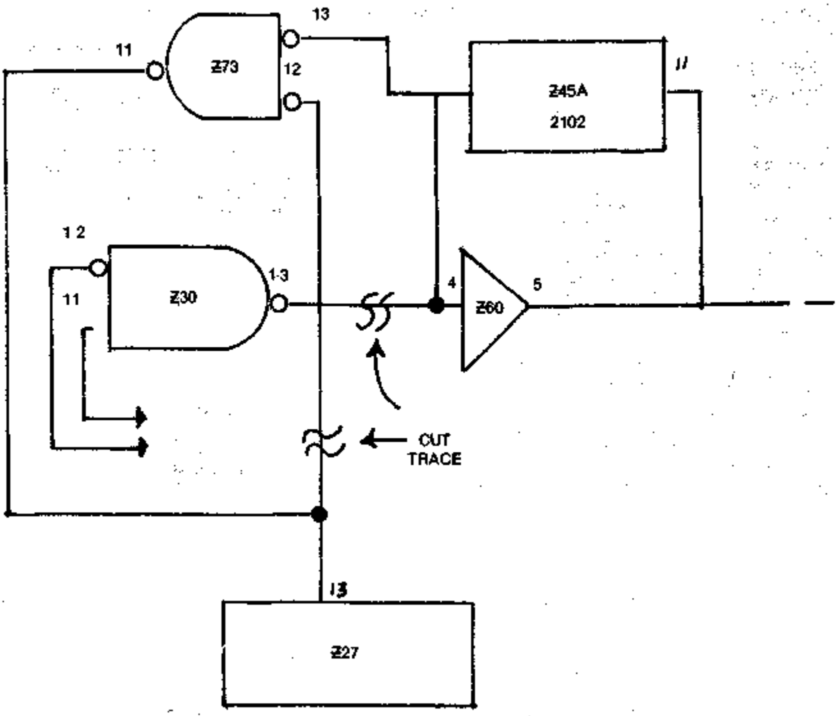
CUT 3 TRACES:

Z73 pin 9 to Z73 pin 12 on the trace side.
Z30 pin 13 to Z60 pin 4 on the component side.
Z60 pin 4 to Z27 pin 13 on the trace size near Z27.

ADD A PIGGY BACK 2102 over Z45 bending out pins 11 and 12. Note that only one 2102 is used in this modification.

ADD 5 WIRES:

Z45A pin 11 to Z60 pin 5
Z45A pin 12 to Z60 pin 4
Z45A pin 12 to Z73 pin 13
Z73 pin 12 to Z30 pin 13
Z73 pin 11 to Z27 pin 13



String Crasher

For 32K TRS-80

by Clayton E. Schneider

Perhaps you are already aware of the tendency of Disk Basic to overwrite the top 6 bytes of 48K RAM, destroying whatever string data may be located there. Perhaps you are also aware that "improperly" exiting a FOR-NEXT loop can cause an NF error later in your program. Well, get ready for the really bad news . . .

I came across this one while attempting to convert a Level I program to Level II (the program was from Fort Worth). The bug boils down to the following lines:

```
1  INPUT N$
2  Q=RND(9)
3  FOR I=1TO9
4  IF A(I)=0 THEN A(I)=Q
5  IF I=9 THEN 8
6  IF A(I)=Q THEN 2
7  NEXT I
8  PRINT N$
```

Try running this program with a 32K system (either Level II or Disk) without answering MEMORY SIZE or changing the normal CLEAR 50. (If you have a 48K system, use a memory size of 49153.) You will find that in the course of program execution, very curious things happen to N$ -- and no error message is displayed!

In addition, you will find that changing memory size *or* the size of the clear will make the problem "go away" -- it seems.

Yes, changing lines 3 and 7 to "E I=1" and "7 I=I+1: GOTO 4" makes it run perfectly, regardless of size, etc. And yes, there are more straightforward ways to do lines 2-7, and if R.S. didn't do it that way, I doubt I would ever have found the problem.

So it goes . . .

Date and Time Routine For TRS-DOS 2.2

by George W. Blank

If you are using a program that directly uses the TIME$ function in TRS-DOS 2.2, this will greatly simplify updating your clock. This subroutine is not copywritten, so feel free to use it in any program.

```
10 REM * SET DATE AND TIME *
20 REM * GEORGE BLANK * JULY 4, 1979 *
30 REM * FOR TRS-DOS 2.2 *
40 CLS : PRINT
50 PRINT "DATE IS SET TO "LEFT$(TIME$,8)
60 INPUT "IS THIS CORRECT";A$
70 IF LEFT$(A$,1)="Y" THEN 110
80 INPUT "MONTH (NUMBER)";A : POKE 16454,A
90 INPUT "DAY (NUMBER)";A : POKE 16453,A
100 INPUT "YEAR";A$ : POKE 16452,VAL(RIGHT$(A$,2))
110 PRINT : PRINT"TIME IS SET TO ";RIGHT$(TIME$,8)
120 INPUT "IS THIS CORRECT";A$
130 IF LEFT$(A$,1)="Y" THEN 180
140 INPUT "HOUR";A : POKE 16451,A
150 INPUT "MINUTE";A : POKE 16450,A
160 INPUT "SECOND";A : POKE 16449,A
170 GOTO 40
180 END
```

TSE Order Form

Special prices in effect 60 days from mailing

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

Check VISA Master Charge

Money Order

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

Not responsible for typographical errors

TRS-80 Software Exchange

6 SOUTH STREET MILFORD, NH 03055

Order toll free: 1-800-258-1790

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

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

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

Charge card account number

Signature..

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

Charge customers: Please fill in account information above and below

Name..

Address..

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

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

PROG/80
PO Box 68 Milford, NH 03055

U.S. POSTAGE
PAID
—BULK RATE—
PERMIT NO. 21
MILFORD, NH 03055