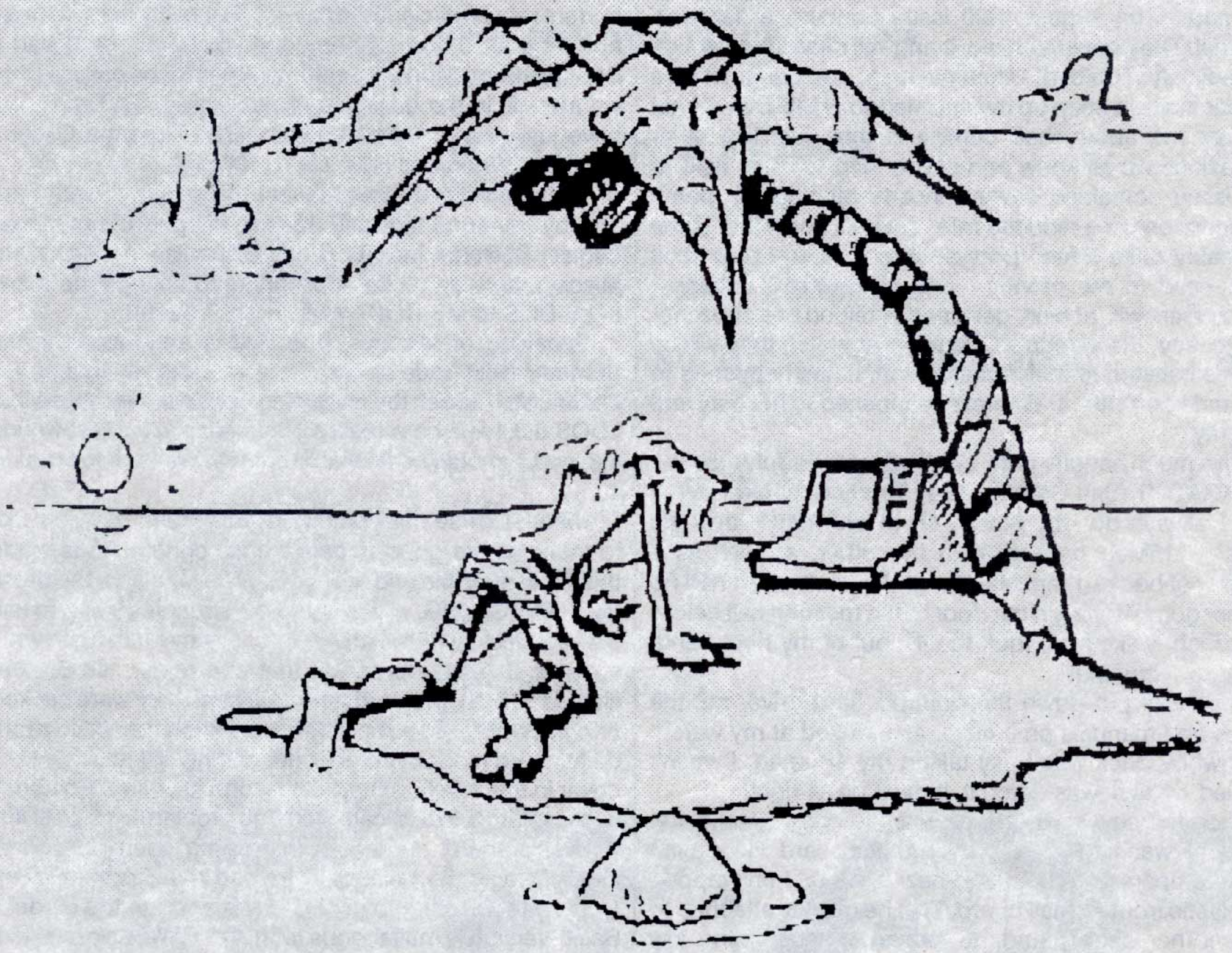
TRSTimes

Volume 3. No. 5. - Sep/Oct 1990 - $4.00



LUCKY MAN

LITTLE ORPHAN EIGHTY

Though it really has nothing to do with the TRS-80, I thought you might want to know the latest on the computer magazine front. Right before 80 Micro deserted us, and subsequently folded, the people at 80 Elm Street in Peterborough, NH spun off a sister magazine geared for the PC-world. It was called PC Resource, the contents being a mix of old 80 Micro philosophy for the PC and the boring regular PC mags. The august 1990 issue contains a 'farewell' editorial. They couldn't make it and will cease publication immediately. Too bad! I liked the mag, and bought it on a regular basis to keep up my knowledge of MS-DOS. What bothers me about their demise is that, just like other magazines we all knew and subscribed to, their next to last issue contained several inserts offering 12 month subscriptions at a reduced rate. You just know that some (probably quite a few) poor soul took them up on their offer, send in the money - and is now real unhappy. Subscribers will, at best, get a partial refund or a subscription to, say, Fisherman's Quarterly! At worst, they will be out the balance of their subscription. It has happened to me, and I don't like it. Whatever happened to honesty and integrity?

On a much happier note, I walked in to the July meeting of SAGATUG (San Gabriel Valley Tandy User Group) the way I always do: my Model 4P in one hand, and my portable 15Meg hard-drive in the other, as well as a number of books, manuals, etc. under each arm. I had no sooner gotten through the door to the meeting hall before Jack Eich walked up, took the 4P out of my right hand, and said "Time me!"

I carefully put down the portable hard drive, put the books and manuals on a table, and looked at my watch. Meanwhile, Jack was busy taking my 4P apart. I wasn't worried - if Jack was doing it, it must be all-right!

Soon the cover was off, as was the back metal plate, and Jack was fiddling with the mother-board. He moved some jumpers and, before I knew it, he had unwrapped the plastic from a small board. This he quickly attached to the mother-board, and he began reassembling my machine. When the last screw was in place, he asked: "Well, how long did it take? It had taken him right around 12 minutes to install a Radio Shack hi-res board in my 4P. I was duly impressed, as were the crowd that had gathered around to watch Jack perform his magic. 12 minutes!! - had I had to do that myself, it would have taken all day, and I would probably have blown up the machine somewhere along the way. Thanks, Jack. You are, indeed, a good friend (and a fast one).

Needless to say, I was equally impressed when we turned the machine on and ran some hi-res programs. Allen Jacobs took over and began demonstrating Frank Slinkman's SLOTS4, VIDPOKR4 AND GIF4MOD4. Gee, was that really my Mod 4P displaying those great graphics? I'm hooked, and I recommend all you Mod 4 owners out there without a graphics board, to get one as soon as possible. It adds a whole new dimension to the machine.

Now on to business. We are still receiving mail addressed to the Canoga Park address. Please remember that we have moved. Sending anything to our old address will cause a significant delay. As a matter of fact, with the current rate of efficiency in the U.S. Postal System, we may not get it all. If you, or someone you know, have had a letter or submission to us returned, please be assured that we are still in the business of publishing TRSTimes. We have just moved our offices to: 5721 Topanga Canyon Blvd. #4. Woodland Hills, CA 91367.

The other day a thought went through my head (I am told by my sons that this doesn't happen often). Since Logical Systems Inc. (LSI) has relinquished LS-DOS to Misosys, wouldn't it be appropriate to change the name of LS-DOS to MS-DOS? Well, just a thought!

Speaking of Misosys, Roy Soltoff announces in the Summer 1990 issue of The Misosys Quarterly that he will continue to publish the magazine. He also announces that LDOS 5.3.1 will be a reality. He is hard at work rewriting 5.3, and it should be available shortly. We look forward to it.

Before I close this column for this month, I would like to thank all the good people whose contributions made this issue possible and timely. George Madison, the president of SAGATUG, writes about his struggles trying to use his new Deskjet Plus with AllWrite. Danny Mullen presents a clever database program that can read Little Brother files. Jack Eich, the southland's ultimate hardware hacker, brings us more timely tips. Delmer Hinrichs provides a tour de force on random number generating. What he doesn't cover in this article, simply isn't worth knowing. Roy Beck writes about the duplicate filename problem on Model 4's. Sam McFarland shares an interesting shorty program, GALAXY. For the heck of it, I typed the program listing (slightly modified) into my PC-clone and ran the Model 4 Basic version simultaneous with the GW-Basic version. Model 4 Basic ran the program much faster. Not too bad, considering that the PC is running at 10 mhz vs Model 4's 4 mhz. Rick DesMarteau provides a translation of a fun C64 program for Models I, III & 4. This one will help you get through the month safely.

Thanks to all, and now welcome to..........TRSTimes 3.5

TRSTimes magazine

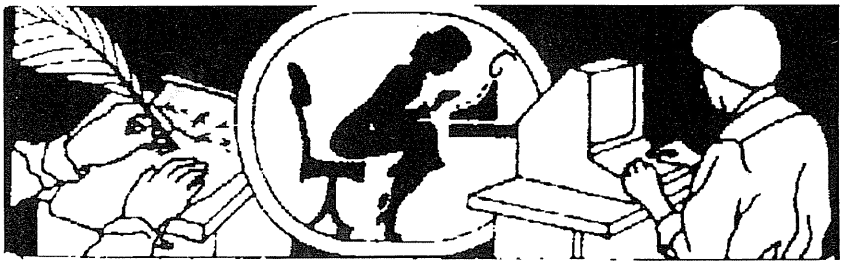
Volume 3. No. 5. - Sep/Oct 1990

PUBLISHER/EDITOR
Lance Wolstrup
CONTRIBUTING EDITORS
Roy Beck
Dr. Allen Jacobs
TECHNICAL ASSISTANCE
Members of:
San Gabriel Tandy Users Group
Valley TRS-80 Users Group
Valley Hackers' TRS-80 Users Group

TRSTimes magazine is published bi-monthly by TRSTimes Publications. 5721 Topanga Canyon Blvd, Suite 4. Woodland Hills, CA. 91364. (818) 716-7154.
Publication months are January, March, May, July, September and November.

1990 subscription rates (6 issues):
UNITED STATES & CANADA:
$18.00 (U.S. currency)
ALL OTHER COUNTRIES:
$23.00 for surface mail, or
$29.00 for air mail.
(U.S. currency only)

Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible, a disk with the material saved in ASCII format. Any disk format is acceptable, but please note on label which format is used. Also, please make sure that your name and address is written legibly on both hardcopy and disk label.



THE MAIL ROOM

PATCHING TRSDOS/LS-DOS 6 AND LDOS

I have learned to how to patch TRSDOS 1.3. However, on my machine, LDOS 5.3 and LS-DOS 6.3 do not PATCH as advertised. The most common message that I see is "Find line mismatch". I type them in from the DOS command line, being very careful not to make any typos. This pertains to D-type patches, when I later use FED/CMD to check the existing code there is really no mismatch! Roy Soltoff made two mandatory patches to LS-DOS 6.3 to improve disk I/O. Only FED let me make those patches (I used LS-DOS in drive one as a data disk , as FED needs LDOS 5.3 as an operating system). But I am at a total loss as to how to apply an L-type patch to SYS7/SYS of LDOS 5.3. It is an extension to a library Partioned Data Set. The use of /FIX files is so far beyond my understanding, but /BLD files under TRSDOS 1.3 are a snap.

What really bothers me is that the MAKE621/JCL, the upgrade from TRSDOS 6.2 to 6.2.1, went without a hitch. It is the most extensive alteration of a DOS that I have seen being done on my computer. What is going on??? Do you know it, or do you have a GURU who could explain it? I am not overly concerned with applications (even though they are the ultimate reason for a computer); but more intersting to me is the operation, or lack of it, at the chip and bit level. Any light you could shed on this failure to patch properly these two DOSes would be very much appreciated. By the way, I am not the only reader or yours who has this problem.

I am enclosing copies of my original LDOS and LS-DOS masters, using QFB for LDOS and DISKCOPY for LS-DOS, to keep the original interleave correct. While a poor PATCH utility is not a fatal flaw in a DOS, it detracts from the quality of it. Then again, the latest patch to LS-DOS is to compensate for a hardware problem. It is my firm belief that 4MHZ is "pushing the envelope" of RS equipment. Just read about the speed-up mods that had to be yanked out because of reliability problems. My preferred DOS is LDOS, its display is easier on my eyes, and it has done the job on a "farkled" LS-DOS disk that LS-DOS itself couldn't do.

Willi Wald
Hamilton, Ontario, Canada

Sending the disks and a screen dump of the input and the subsequent error messages was most useful. I now see what the problem is.

You typed the following patch (LS-DOS 6.3.0):

```
PATCH BOOT/SYS.LSIDOS:0
(D0C,B4=FD7E03F5C5: F0C,B4=F5C5FD7E03)
```

This gave you a 'FIND line mismatch' error. The manual doesn't tell you specifically, but you must put spaces between the data items. Thus, when I typed the patch as follows, it worked beautifully.

```
PATCH BOOT/SYS.LSIDOS:0 (D0C,B4=FD 7E 03 F5
C5:F0C,B4=F5 C5 FD 7E 03)
```

The same thing holds true in LDOS 5.3.0. You must use spaces to separate the data items. You typed:

```
PATCH SYS8/SYS.SYSTEM (D00,F5=F53A6747F1:
F00,F5=2E0100000000)
```

This also got you the 'FIND line mismatch' error. However, when you insert spaces between the data items, you run into a slight problem: there is not room enough on the line for the entire patch. To solve this, split the patch into two patches, as follows:

```
PATCH SYS8/SYS.SYSTEM (D00,F5=F5 3A 67:
F00,F5=2E 01 00)
```

and then:

```
PATCH SYS8/SYS.SYSTEM (D00,F8=47 F1:
F00,F8=00 00)
```

This will install the patch. Incidentally, the LDOS manually does mention this. Somehow it was omitted from the TRSDOS 6 manual.

Hope this solves your problem, and restores your faith in the PATCH utility.

Ed.

MORE CLAN

I too use CLAN and have located the bugs mentioned by Jim Savage. Unfortunately I have omitted to document them as the same bug appears more than once, but it is easy to fix. The program files are in Basic, and one must list the data entry files. Look for number of the year that your version will not accept and change it for a higher value. Different versions I have fixed have different cutoff years. As I recall, about 3 fixes are required. As mentioned in the instructions of CLAN, different printers may require changes in printer instruction codes to get correct readout.

Owen E. Edmonds
French's Forest, N.S.W. Australia

SAY TRS

I have a Model 4. I've seen the ads for TRSLINK. I'm new at all of this. What is a TRS-80 BBS? The other source is to "download it from 8/n/1 #4, etc". To do this I presume

I need a modem. This sounds like something I should have. What do I need - exactly - to hook up a modem - and which modem? Does RS sell these, or is there a better source?

One other question I have is about the pronounciation of 'TRS". My brother-in-law says everyone pronounces this TRASH. I've been pronouncing this "TRIS". He doesn't own one, but is he right?

Kathleen Steed
Bowie, MD

Let's take your second question first. The correct pronounciation of TRS is "TRIS", as in TRISDOS, TRISLINK, TRISTIMES, etc. The term 'TRASH-80' was coined in the late seventies by jealous Apple owners describing the very fragile Model I. We have been haunted by this term ever since. Does your brother-in-law own an Apple?

Secondly, in order to access a BBS, you do need a Modem. Radio Shack sells these, but usually are somewhat more expensive than other outlets. Wherever you choose to buy one, make sure it is at least 1200 baud (2400 baud is even better). Also make sure you get a connector cord that will fit into your Model 4. The next thing you need is a communications program. Two good ones are XT4 and FASTERM.

XT4 is included on disk 4 of the Valley TRS-80 Hackers User Group's public domain series. Get in touch with them at: Box 9747. North Hollywood, CA 91609.

FASTERM can be obtained from the author: Mel Patrick, 13699 70A Ave, Surrey, British Columbia, Canada V3W 2J8.

A TRS-80 BBS is a Bulletin Board System that caters to the TRS-80 machines. You no doubt have many BBS's in your area, but most will probably cater to the IBM crowd. Calling an IBM board will not do you a lot of good, as you cannot use any of the files you might download. If there is a TRS-80 BBS in your area, call them and download any or all issues of TRSLINK. If not, you can call long distance to Philadelphia and access the TRS-80 BBS called 8/n/1 #4 run by Luis Garcia-Barrio. They will certainly have all issues of TRSLINK available (they are the ones publishing it), as well as other good stuff. Good luck with BBSing.

Ed.

CP/M

I was caught up by the Aerocomp bargain prices for certain software like Twist & Shout. The only problem is that I cannot get hold of any CP/M program to run it at a decent price. But I recently got hold of Supercross/Xt (v2.0) that can read CP/M disks. I did transfer my files to the TRS-80 system. A bit complicated, but it cannot do a thing for machine language programs written in CP/M. The question is, can anybody help me out there. I mean, if you can get Twist & Shout at $6 an run it eventually, why pay close to $40 for a TRSDOS version of it.

PS. The above is no real pancea, as the manual still has CP/M commands, hence.....freebies are hard to come by! And I am now weary of transfer programs, to say the least.

R. Yves Breton
P.O. Box 95, Stn Place d'Armes
Montreal, Quebec, Canada H2Y 3E9

If I understand your letter correctly, you purchased a copy of Twist & Shout written for CP/M. In order to use this program you need the CP/M oprating system. It is available from Montezuma Micro, PO Box 763009, Dallas, TX 75376-3009. If you do not have the CP/M operating system and do not intend to buy it, why did you buy the program in the first place?

Using a transfer program, such as Supercross, will certainly transfer the program from CP/M to TRSDOS, or from TRSDOS to CPM, but there is no guarantee that the program will work on the new media. That is not the problem of the transfer program - it has done its job. Once transferred, it becomes your job to rewrite the file, if necessary. Data files will usually work as is. Basic program files will require a little modification, but machine language programs - forget it - unless you have plenty experience with both 8080 and Z80 machine code.

Your best bet, if you insist on running Twist & Shout, is to buy CP/M from Montezuma, or spring for the TRS-80 version, which I believe is now handled by Computer News 80, PO Box 680, Casper, WY. 82602-0680.

Ed.

MULTIDOS

For the convenience of the TRSTimes readers, let me point out that Alphabit Communications, which handles Multidos (and Lazy Writer) is no longer at 13349 Michigan Av., Dearborn, MI. Their new addres is: Alphabit Communications, PO Box 20067, Ferndale, MI 48220-0067.

Jim King
Topanga, CA

Thanks for the info.

Ed.

SPEED-UP BOARD

Many thanks for your help with my speed-up board problem. I now have 8 of the connections, with the help of Bryan Mumford's contribution. All I need now are the other six (connections F,G,I,J,K & M). I may be able to persuade a friend of mine, who can think in 'Boolean', to work out the rest, as I have a good idea which are the I/C's involved on the main board. Thanks again.

E.C. Kilpatrick
Sudbury, Suffolk, England

HUNTING FOR BURIED TREASURE

Peeking and Poking the Model 4 Character Sets (and other stuff)

By Lance Wolstrup

This installment of HUNTING FOR BURIED TREASURE came about because a member of a user group, which I attend on a regular basis, recently asked the following question:

"Using CHR$(21) and CHRS$(22), you can select space compression, special, or alternate characters for values 192-255. I can't find how (and I've looked through all of the manuals) to find out what the current character mode is. I have seen programs that print something, and then ask if the cursor moved (to find out if space compression is on). Then they print something and ask if it was a heart (to find out if special or alternate characters are set).

There has to be a better way. How can I do it with SVCs? Using CHR$() only allows toggling of modes, without knowing which mode is currently set. I want to be able to see which mode is set, and be able to select any of the modes from assembly or C."

The answers to these questions were partly covered in HUNTING FOR BURIED TREASURE from the very first issue of TRSTimes (Jan/Feb 1988). However, since that article dealt strictly with Basic code (and I omitted the information about the alternate character set), let's go over the details again, but first let me give a little background by quoting from appendix C of the TRSDOS 6 user manual:

"Codes 192 through 255, when output to the video display, represent either space compression codes or special or alternate characters, as determined by software. Toggling between these modes is done via codes 21 and 22.

Code 21 toggles the video driver between space compression codes and the special/alternate character set. Code 22 toggles the video driver between the special character set and the alternate character set. The setting of the toggle controlled by code 21 determines if the code 22 toggle will have any effect on what is subsequently displayed.

The following chart illustrates the power-up and first toggle states for codes 21 and 22:

	CODE 21	CODE 22
POWER-UP STATE	space compression characters	special characters
FIRST TOGGLE STATE	special/alternate characters	alternate characters

At power-up, codes in the range 192 to 255 will produce one or more spaces (space compression mode). From this point, you can enter the special character set by outputting a code 21 to the display. You can then enter the alternate character set by outputting a code 22 to the display. To switch back to the special set, output another code 22. To switch back to space compression codes from either the special or alternate character set, output a code 21.

When you are in space compression mode, outputting a code 22 still toggles between special and alternate character sets, even though it does not affect the characters subsequently displayed. Any characters in the range 192-255 that are already on the display will toggle between special and alternate character sets each time a code 22 is received."

Having now brushed up on the manual, we see that the problem is outputting codes 21 and 22 simply toggles the character sets to their alternate states.

What is asked for is a method to determine which character mode is active, so that corrective codes can be sent to make a desired mode active. This is possible, of course, but it is of no actual importance!

What is important, is that you can FORCE the any of the character sets to be active, no matter what the current mode is. I will cover the undocumented ways to do this from Model 4 Basic, as well as from assembly language.

Memory location &HB94 (2964 decimal) controls a couple of goodies. The particular goodie we are looking for is controlled by bit 3. If it is reset (off), space compression is active; if bit 3 is set (on), the special character set is active. Thus, you can force the special character set to be active with the following Basic code:

```
POKE &HB94,PEEK(&HB94) OR 8
```

On the other hand, should you wish the space compression codes to be active, you issue this Basic code:

```
POKE &HB94,PEEK(&HB94) AND 247
```

Type in this short Basic program, RUN it, and see happens:

```
10 POKE &HB94,PEEK(&HB94) OR 8
20 PRINT CHR$(143);CHR$(244);CHR$(245);CHR$(246)
```

RUN the program as many times as you wish. Each and every time the program is RUN, the special character set is activated. It doesn't toggle.

Now replace line 10 with this:

```
10 POKE &HB94,PEEK(&HB94) AND 247
```

RUN the program again, as many times as you wish. Space compression is active each and every time. It doesn't toggle.

The assembly language version to force the special character set is:

```
LD    HL,0B94H      ;point to 0B94H
LD    A,(HL)        ;get value
OR    8             ;set bit 3 - force special chrs
LD    (HL),A        ;copy new value back to 0B94H
```

To force space compression you do this:

```
LD    HL,0B94H      ;point to 0B94H
LD    A,(HL)        ;get value
AND   247           ;reset bit 3 - force space comprs
LD    (HL),A        ;copy new value back to 0B94H
```

Now we come to the alternate character set, which is slightly more difficult. This character set is controlled by bit 3 in memory location &HB94 in conjunction with bit 3 in memory location &H76. To force the alternate characters, bit 3 of both &HB94 and &H76 must be set. However, simply setting bit 3 of &H76 doesn't do the job. The new value of &H76 must be sent to port &HEC before the change takes effect.

Thus, to force the alternate character set:

```
10 POKE &HB94,PEEK(&H94) OR 8
20 POKE &H76,PEEK(&H76) OR 8
30 OUT &HEC,PEEK(&H76)
```

To turn off the alternate character set:

```
10 POKE &HB94,PEEK(&HB94) OR 8
20 POKE &H76,PEEK(&H76) AND 247
30 OUT &HEC,PEEK(&H76)
```

The assembly version to force the alternate character set is as follows:

```
LD    HL,0B94H      ;point to 0B94H
LD    A,(HL)        ;get value
OR    8             ;set bit 3 - force special chrs
LD    (HL),A        ;copy new value back to 0B94H
LD    HL,76H        ;point to 76H
LD    A,(HL)        ;get value
OR    8             ;set bit 3 and
LD    (HL),A        ;copy new value back to 76H
OUT   (0ECH),A      ;alternate characters now active
```

To remove the alternate character set -

```
LD    HL,0B94H      ;point to 0B94H
LD    A,(HL)        ;get value
OR    8             ;set bit 3 (AND 247 to reset bit 3)
LD    (HL),A        ;copy new value back to 0B94H
LD    HL,76H        ;point to 76H
LD    A,(HL)        ;get value
AND   247           ;reset bit 3 and
LD    (HL),A        ;copy new value back to 76H
OUT   (0ECH),A      ;alternate characters now off
```

That is all there is to it. You don't need to know which one of the character sets is active, you simply activate the one you want.

A related matter is CHR$(23). Issuing a code 23 switches the screen to 40 character per line mode.

The problem with 40 character mode is that it uses CHR$(28) to return to 80 character mode. Since the CLS command is essentially CHR$(28) followed by CHR$(31), it becomes obvious that it is impossible to CLS the screen and STAY in 40 character mode. Well, by using a little ingenuity, the impossible can quite often become possible. It is certainly so in this case. Let us start at the beginning.

Memory location &H76 is the key to many things. We just saw that bit 3 controlled whether the special or alternate character set was active. Now we will deal with bit 2. It controls the video mode; that is, it determines whether screen is in 80 chararcter mode or 40 character mode. If bit 2 of 76H is set (on), the video mode is set to 40 characters. If bit 2 of 76H is reset (off), the video state returns to normal 80 character mode.

Switch to 40 character mode:

```
10 POKE &H76,PEEK(&H76) OR 4:
OUT &HEC,PEEK(&H76)
```

Switch to 80 character mode:

```
10 POKE (&H76),PEEK(&H76) AND 251:
OUT &HEC,PEEK(&H76)
```

OK, getting the value of &H76, ORing it with 4, POKEing this back to &H76 and sending this new value to port &HEC does nothing more than if you had typed: PRINT CHR$(23). I am presenting this way of entering 40 character mode to show the similarity with returning to 80 character mode. Still, when in 40 character mode, typing CLS, PRINT CHR$(28), or even NEW, returns you to 80 character mode, whether you want to be there or not.

The fix to this resides in memory location 0C0CH. This location is used by CHR$(28), CLS and NEW to mask out bit 2 of memory location 76H. The normal value in 0C0CH is 251. If you fiddle with binary, you'll notice that all bits EXCEPT bit 2 are set. This value is the ANDed with the value in 76H, in essence forcing bit 2 off. Thus, CHR$(28), CLS and NEW always return us to normal 80 character mode. What to do about this - simply POKE 0C0CH with 255. This will keep bit 2 of 76H on if it was on in the first place.

Allow 40 column mode to PRINT CHR$(28), CLS and NEW without returning to 80 column mode:

```
POKE &H0C0CH,255
```

Set 40 column mode back to normal (PRINT CHR$(28), CLS and NEW brings back 80 column mode.)

```
POKE &H0C0C,251
```

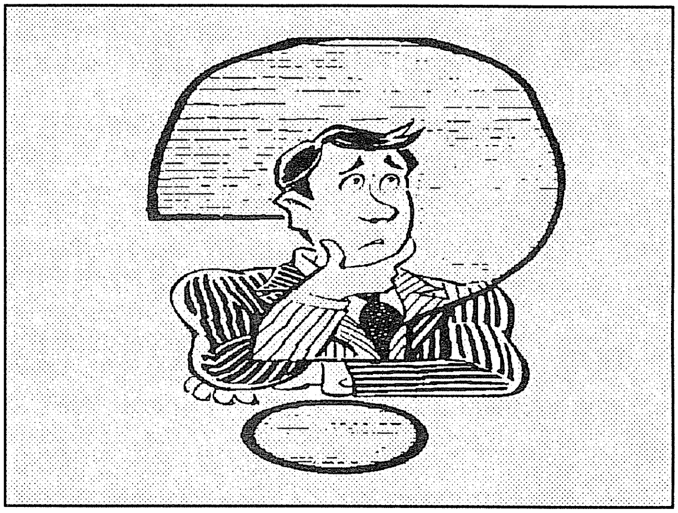
The following program, VID40/BAS, is a short demonstration of the things we have discussed in this installment.

VID40/BAS

```
        Initialize and jump over subroutines
10 SW=80:PRINT CHR$(15):goto 100
        Subroutines - Print routines
20 H=0:GOTO 23
21 H=INT((SW-(LEN(A$))*2)/2):
IF H/2<>INT(H/2) THEN H=H-1:GOTO 23
ELSE 23
22 H=79-LEN(A$)
23 PRINT@SW*V+H,A$;:RETURN
        40 column mode
30 POKE &HC0C,255:POKE &H76,PEEK(&H76) OR 4:
OUT &HEC,PEEK(&H76):RETURN
        80 column mode
40 POKE &HC0C,251:
POKE &H76,PEEK(&H76) AND 251:
OUT &HEC,PEEK(&H76):RETURN
        force special characters
50 POKE &HB94,PEEK(&HB94) OR 8:RETURN
        force space compression codes
60 POKE &HB94,PEEK(&HB94) AND 247:RETURN
        force alternate characters
70 GOSUB 50:POKE &H76,PEEK(&H76) OR 8:
OUT &HEC,PEEK(&H76):RETURN
        alternate characters off
80 GOSUB 50:POKE &H76,PEEK(&H76) AND 247:
OUT &HEC,PEEK(&H76):RETURN
        program begins here - 40 column mode,
        erase screen and display some text
100 GOSUB 30:CLS
110 V=1:A$="TRSTimes presents:":GOSUB 21
120 V=2:A$="VID40 DEMO":GOSUB 21
130 V=3:A$="(c) 1990 by Lance Wolstrup":GOSUB 21
        switch between 80 column mode and
        40 column mode until a key is pressed
140 FOR Y=1 TO 150:NEXT:GOSUB 40:
FOR Y=1 TO 150:NEXT:GOSUB 30
150 IF INKEY$="" THEN 140
        draw border, using chr$(28) to home cursor
160 PRINT CHR$(28);CHR$(156);STRING$(38,140);
CHR$(172);
170 FOR V=1 TO 21:A$=CHR$(149):GOSUB 20:
A$=CHR$(170):GOSUB 22:NEXT
180 A$=CHR$(141)+STRING$(38,140)+CHR$(142):
GOSUB 20
190 V=4:H=2:A$=STRING$(38,140):GOSUB 23
        force special characters and display them
210 GOSUB 70
220 X=192
230 FOR V=9 TO 16
240 FOR H=18 TO 64 STEP 6
250 A$=CHR$(X):GOSUB 23
260 X=X+1
270 NEXT:NEXT
        switch between 80 character mode and
        40 character    mode until a key is pressed
280 FOR Y=1 TO 150:NEXT:GOSUB 40:
FOR Y=1 TO 150:NEXT:GOSUB 30
290 IF INKEY$="" THEN 280
        switch between special and
        alternate characters until a key is pressed
300 FOR Y=1 TO 150:NEXT:GOSUB 80:
FOR Y=1 TO 150:NEXT:GOSUB 70
310 IF INKEY$="" THEN 300
        reset to 80 character mode,
        activate space compression codes,
        erase the screen and end program
320 GOSUB 40:GOSUB 80:GOSUB 60:CLS:END
```

TWO FOR THE PRICE OF ONE?

Roy T. Beck



THE PROBLEM

Have you ever had two (or more) copies of a file turn up on a floppy disk? I personally have never had it occur, but a few unfortunate souls have. If this happened to you, how would you know which file is the latest, and what would you do about it?

BACKGROUND

I have been reading The MISOSYS Quarterly, and I was intrigued by the latest in a series of articles on this problem. So much so, that I dug out all the issues that contained references to the problem.

While there were earlier references to the problem, nothing significant was published until the Summer 1988 issue, which contained a letter from Charles Ainsworth and a reply by Roy Soltoff (MISOSYS). Ainsworth runs a business involving heavy use of Allwrite on Model 4's and 4D's, and the problem was driving him up the wall.

Apparently the problem is very rare, and at least initially, was quite elusive. Ainsworth had noticed a previous comment by someone who also had the problem, and the other writer thought maybe the problem was due to a floppy drive not reaching full speed soon enough, and therefore failing to report a file already existed. If no file existed, apparently Allwrite would then create a new file to save to. The other user set the DELAY parameter to allow 1 second for the floppies to accelerate. No further comment from the other party. Ainsworth tried this also, but it didn't solve the problem for him.

Ainsworth and Soltoff wrote and talked back and forth over a period of about a year, and neither could pin down the problem, much less the solution. Initially Soltoff thought the DOS was at fault, mainly because the problem did not occur under LDOS under the same circumstances. Ainsworth thought the problem was in the XLR8er or its software. (He uses XLR8ers to increase his typist's throughput). Since the original software for the XLR8er was a trifle flaky, his suspicions were probably reasonable. He first noticed the problem when he began using Model 4D's equipped with XLR8er's. Apparently he bought XLR8ers for the 4D's at the time he bought the machines. He had previously used Mod 4's without trouble.

RESEARCH

Soltoff devoted quite a bit of time to the problem, reviewing all the DOS code which has anything to do with creating files. Later, he set up a test loop and let a machine "cook" overnight. He also invited the members of the LDOS SIG on Compuserve to join the hunt, running a test routine until failures showed. As a result of these efforts, he was able to accumulate some statistics on the errors, and the finger of suspicion pointed to the gate array machines. The PAL machines would run for 3000 or more cycles without error, but the gate array machines would sometimes bomb in less than 300 cycles. With this clue, the hunt was on!

THE MACHINES

At this point, I will list the various machines so you will know the players in this game:

Model 4	Model 4D	Model 4P	FDC chip	FDC Support
		PAL machines:		
26-1069	-------	26-1080	WD 1793	74LS123
		Gate Array machines:		
26-1069A	26-1070	26-1080A	WD 1773	IC 4.4

What are the differences between the machines which may have a bearing on the doubling of files? The PAL machines use the Western Digital 1793 floppy disk controller (FDC) chip, and the gate array machines use the WD 1773 FDC. Further, there are some support chips which work with the FDC, and some of these are unique to Radio Shack! Aha! Could WD be the villain? How about RS? or MISOSYS? Or any combination of the above?

ANALYSIS

Soltoff tracked the problem as far as he could and concluded the problem was apparently a hardware defect caused by the gate array machine occasionally failing to detect the passing of the index hole in the disk drive, which therefore logically appeared like an empty drive (or one with the door open). The DOS would therefore not detect an existing file, and would proceed to open a new one. However, the logic flow to open a new file ran a different algorithm to decide which drive to use, and under these circumstances, the drive which previously seemed devoid of the file would be accessed again, usually successfully. Presto, a new copy of the same file could appear on the drive. (Ainsworth reported up to FIVE copies of the same file on one disk)!

At this point, Soltoff asked for help. Frank Durda IV to the rescue! Frank is another excellent programmer. (He is the author of the ROM in the 4P, and also the recently announced autoboot program for the 4 and 4P). He continued the search, building upon what Soltoff had discovered. Durda, in his analysis, determined that Radio Shack used to use (in the Mods I, III and PAL version of the 4's) a hardware timer (a chip, a resistor, and a capacitor) to turn on the floppy motors and keep them running for a length of time. In the Mod 4's, RS replaced this assembly with a new, custom support chip, containing the motor timing and some other logic, all in one package. The schematics I have for the 4P identify this as IC 4.4.

Durda then utilized a Z-80 emulator which allowed him to execute the code step by step, analyzing as he went along. He discovered the problem really wasn't a failure to detect the index hole transition, but in effect, a failure to create the hole transition. The new FDC support chip is supposed to turn on the floppy drive motors when port F4 is written to. Once in a while, on some chips, writing to port F4 fails to start the timing sequence and the motors do not start! Since the motors do not start, no index pulse can be generated, and from there on Soltoff's analysis was valid. Since the failure to trigger is random, a second OUT to port F4 usually works as intended.

If a motor is not running when the DOS assumes it should be, then a failure to detect an index pulse is accepted as evidence there is no disk in the drive. The DOS tries the other drives, and eventually concludes the requested file does not exist. It then proceeds to create a new file, same filespec, via a different software path, which contains additional OUTs to port F4. The drive now starts, a new file of the same name is opened, and multiple files result.

Why does LDOS not exhibit similar trouble? The LDOS code apparently issues multiple OUTs to port F4, and even if one trigger event fails, another works, and the trouble is not evident.

THE FIX

Having determined the FDC-support chip is the villain, what could be done about it? Trying to get Radio Shack to even admit the problem, much less fix it at this late date is probably an exercise in futility. So Durda began looking for a software solution. Assuming the misbehavior was rare and random, was there any way to issue additional OUTs to port F4? Durda is also no slouch at software, and he began examining the code. How to squeeze in an extra two byte instruction into two places in existing code? Well, he did it. Both patches are in BOOT/SYS; In one case he was able to alter the sequence of three existing instructions which accomplished the additional OUT to port F4. In the other case, he was able to replace a four byte opcode with a two byte opcode, and thus gained space where he could insert the required extra OUT instruction. Pretty darn clever, says I!

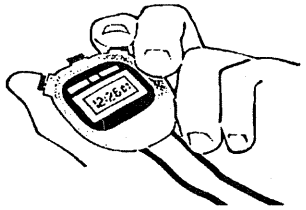
Soltoff listed Durda's patch for TRSDOS 6.3.0 in the article, which I guess constitutes a blessing on it. But what about V 6.3.1? I checked my copy, and found Soltoff has incorporated Durda's patch verbatim into 6.3.1. If you are using 6.3.1 as you should be, you need not be concerned. If you are experiencing the problem, then buy 6.3.1, as you should have done. If you are really a cheapskate and insist on running 6.3.0 or earlier, then the patch for 6.3.0 is on page 17 of the Spring 1990 MISOSYS Quarterly. Watch out, there is a one byte typo in the patch. See if you can find it! You will spot it easily if you read the boxed-in code carefully. While Soltoff did not say so, my guess is that patch also applies to 6.2.X. Use your analytic ability in this case.

IN CONCLUSION

The problem is a rare failure in a Western Digital chip, which was custom-made for Radio Shack. Naturally, no responsible vendor wants bad chips in the market place, and I believe this bug was simply due to slightly inadequate testing on WD's part. The rarity of the problem shows how few bad chips got by WD's QC. If you want to test your own gate array machine, Soltoff's article includes a BASIC routine which puts a counter on the screen. If the bug bites, the BASIC routine stops on an error with the loop count displayed. If you find you have the bug, the hardware fix is to replace the FDC-support chip. But actually, the patch by Durda solves the problem as far as LS-DOS is concerned, and LDOS does not fail even if the bug does bite. If you are using some other DOS, you are on your own. But at least you now know where to start searching!

Since all the circumstances have to be just right for the problem even to appear, it is really not surprising that it took so long to analyze and solve. It is certainly a credit to Roy Soltoff that he stuck with the problem until a solution was reached. Frank Durda IV also deserves a great big Thank You! from all of us, as also do the unsung members of the SIG who spent some time and money helping to track down the bug when it was completely obscure.

-Roy-



DANNY'S LITTLE BROTHER

TRSDOS/LS-DOS 6 & TRSDOS 1.3

By Danny Mullen

I use the 'Little Brother' data base (from Misosys) to keep an article index of TRSTimes. I have compiled a large data file, which is very handy when I want to reference a particular piece of information. Little Brother is a very powerful program, and I urge the readers to get it from Misosys.

However, realizing that not everybody will own this program, I have written a Basic program that will let people who don't have LB read and search through the data file for wanted information. As written, my program does not allow any data changes - only read & search. It wouldn't be hard to add that feature, but I since I don't need it, I just did not include it. The code could probably be optimized also.

There are two versions of the program. The listing named TRSTLB4/BAS is for Model 4 TRSDOS/LS-DOS 6. The listing named TRSTLB3/BAS is for Model 3 TRSDOS 1.3. Both are provided to illustrate a method to extract information from an LB data file for use with a Basic program.

(Editor's note: Danny also provides the data file and the screen/index/print modules for LB for anyone who has LB wanting to use it as is. To use the Model 4 Basic program as written, the file named 'TRSTIMES/DEF' MUST BE ON THE DISK. It reads this file to get the record count. This file, along with the above mentioned ad-ons will be included on a special extra TRSTimes-on-Disk.

The Model 3 TRSDOS 1.3 version needs only the datafile. It will be included in a special TRSDOS 1.3 extra TRSTimes-on-Disk.)

Let me briefly mention a few things about the data file itself. Some of the comments in the COMMENTS section may be hard to read, but I wanted to include as many keywords/subjects as possible in the related article - thus, spaces were at a premium: I may edit this in the future. Also, I haven't gotten around to putting many debug references in yet: again, in the future.

The format I used was the same as (*sigh*) 80 Micro used, and it did look to be about appropriate:

TITLE, AUTHOR, VOLUME, CATEGORY, MODEL,
DEBUG, COMMENTS.

Danny Mullen can be reached at:
6641-B Tracey Pl. Fort Polk, LA 71459

There may be some question on the categories I listed just an arbitrary choice in some cases, guesses in others..

I intend to keep my personal copies updated; If any of the readers would want updates, I could supply them for a small fee, say $4.00 if I supply the disk and mailer. If I am supplied with disk and mailer, the cost would be $2.50. This offer is not good until about September 1990, as the US Army wants me training out in the woods most of the summer!

TRSTLB4/BAS
Model 4 - TRSDOS/LS-DOS 6

```
1 'TRSTIMES/BAS (c) Danny C. Mullen 6641-B Tracey
Place Fort Polk, LA 71459    *** PUBLIC DOMAIN ***
2 'Reads/displays/searches my TRSTIMES/LB data file
which contains TRSTIMES articles & comments/index
reference for Vol 1 No 1 thru current month.
3 'There is no method to change the data as written - I
use the LITTLE BROTHER database program to do the
normal update/additions - this is just a little help for
those who don't have LB.
4 'This is the LS-DOS version. *** to use this, you
MUST have TRSTIMES/DEF on your disk ***
5 SYSTEM "SYSTEM (BREAK=NO)":
PRINT CHR$(15):CLS
6 GOSUB 6000
7 PRINT TAB(24);CHR$(16);" TRSTIMES ARTICLE
INDEX ";CHR$(17):PRINT
8 PRINT@(2,0),CHR$(16);"RECORD # ";CHR$(17);:
GOSUB 3000
10 OPEN "D",1,"TRSTIMES/LB":E=LOF(1)-1
20 FIELD 1,30 AS T$,20 AS A$,10 AS V$,20 AS C$,5 AS
M$,10 AS D$,161 AS C1$
30 GOSUB 2000
35 IF REC%<1 THEN REC%=1 ELSE IF REC%>E1
THEN REC%=E1
36 PRINT@(2,10),USING "###";REC%;:
PRINT " OF";E1;" ACTIVE RECORDS";
40 GET #1, REC%
50 GOSUB 4000:A=INSTR(T$,CHR$(0)):
IF A=0 THEN PRINT T$;
ELSE PRINT LEFT$(T$,A-1);:A=0
55 GOSUB 4005:A=INSTR(A$,CHR$(0)):
IF A=0 THEN PRINT A$;
ELSE PRINT LEFT$(A$,A-1);:A=0
60 GOSUB 4010:A=INSTR(V$,CHR$(0)):
IF A=0 THEN PRINT V$;
ELSE PRINT LEFT$(V$,A-1);:A=0
65 GOSUB 4015:A=INSTR(C$,CHR$(0)):
IF A=0 THEN PRINT C$;
ELSE PRINT LEFT$(C$,A-1);:A=0
70 GOSUB 4020:A=INSTR(M$,CHR$(0)):
IF A=0 THEN PRINT M$;
ELSE PRINT LEFT$(M$,A-1);:A=0
```

```
75 GOSUB 4025:A=INSTR(D$,CHR$(0)):
IF A=0 THEN PRINT D$;
ELSE PRINT LEFT$(D$,A-1);:A=0
80 GOSUB 4030:A=INSTR(C1$,CHR$(0)):
IF A=0 THEN PRINT C1$;
ELSE PRINT LEFT$(C1$,A-1);:A=0
999 GOTO 30
1000 CLOSE 1:CLS
1010 PRINT "TRSTIMES/LB FILE CLOSED....."
1020 SYSTEM "SYSTEM (BREAK=YES)"
1999 END
2000 PRINT@(22,0),CHR$(16);"<<< Use L/R arrows
to dec/inc or SHIFT UP to quit or ENTER to GOTO
record # >>>";CHR$(17);
2005 PRINT@(23,16),CHR$(16);"<<< or S for search
(starts at next record # >>>";CHR$(17);
2010 Z$=INKEY$:IF Z$="" THEN 2010
2015 IF Z$=CHR$(27) THEN GOTO 1000
2017 IF Z$=CHR$(13) THEN GOSUB 5000:
PRINT@(2,10),STRING$(60," ");:RETURN
2020 IF Z$=CHR$(9) THEN REC%=REC%+1:
IF REC%>E THEN REC%=E:RETURN:ELSE RETURN
2030 IF Z$=CHR$(8) THEN REC%=REC%-1:
IF REC%<1 THEN REC%=1:RETURN:ELSE RETURN
2035 IF Z$="S" THEN GOSUB 7000:
PRINT@(23,0),"                    ";:RETURN
2040 GOTO 2010
3000 PRINT@320, "TITLE   : "
3005 PRINT@400, "AUTHOR  : "
3010 PRINT@480, "VOLUME  : "
3015 PRINT@560, "CATEGORY: "
3020 PRINT@640, "MODEL   : "
3025 PRINT@720, "DEBUG   : "
3030 PRINT@880, "COMMENTS: "
3035 RETURN
4000 PRINT@330,STRING$(30," ");:PRINT@330,;:
RETURN
4005 PRINT@410,STRING$(20," ");:PRINT@410,;:
RETURN
4010 PRINT@490,STRING$(10," ");:PRINT@490,;:
RETURN
4015 PRINT@570,STRING$(20," ");:PRINT@570,;:
RETURN
4020 PRINT@650,STRING$(5," ");:PRINT@650,;:
RETURN
4025 PRINT@730,STRING$(10," ");:PRINT@730,;:
RETURN
4030 PRINT@960,STRING$(161," ");:PRINT@960,;:
RETURN
5000 PRINT@(2,10),;:
INPUT " What RECORD do you want";REC%:
IF REC%<1 OR REC%>E THEN GOTO 5000:
RETURN:ELSE RETURN
6000 OPEN "D",1,"TRSTIMES/DEF":FIELD 1,26 AS A$,2
AS B$,2 AS C$,226 AS X$
6010 GET #1,1:E1=CVI(C$):
' E1 = # ACTIVE RECORDS
6020 CLOSE:RETURN
6998 '
6999 '****** SEARCH ROUTINES FOLLOW ******
7000 '
7005 PRINT@(22,0),STRING$(159," ");:PRINT@(22,0),;:
INPUT"<A>scending or <D>escending search";S1$
7006 IF S1$="A" THEN GOTO 7010 ELSE GOTO 8000
7007 '
7008 '** ASCENDING SEARCH ROUTINE FOLLOWS **
7009 '
7010 PRINT@(22,0),STRING$(80," ");:
PRINT@(22,0),"Search routine..";:
INPUT "Enter the data to search for";S$
7020 FOR K%=REC%+1 TO E1:GET #1,K%
7025 PRINT@(23,0),"ASCENDING  ==> Searching
record #";K%;
7030 IF INSTR(T$,S$) OR INSTR(A$,S$)
OR INSTR(V$,S$) OR INSTR(C$,S$) OR INSTR(M$,S$)
OR INSTR(D$,S$) OR INSTR(C1$,S$)
THEN REC%=K%:PRINT@(23,0),STRING$(60," "),;:
RETURN:ELSE NEXT K%
7035 PRINT@(23,0),"<<< NOT FOUND >>>
.....any key to continue...";:SOUND 7,2
7037 Z$=INKEY$:IF Z$="" THEN 7037
7040 PRINT@(23,0),STRING$(79," ");:RETURN
7997 '
7998 '* DESCENDING SEARCH ROUTINE FOLLOWS *
7999 '
8000 PRINT@(22,0),STRING$(80," ");:
PRINT@(22,0),"Search routine..";:
INPUT "Enter the data to search for";S$
8020 FOR K%=REC%-1 TO 1 STEP -1:GET #1,K%
8025 PRINT@(23,0),"DESCENDING  ==> Searching
record #";K%;
8030 IF INSTR(T$,S$) OR INSTR(A$,S$)
OR INSTR(V$,S$) OR INSTR(C$,S$) OR INSTR(M$,S$)
OR INSTR(D$,S$) OR INSTR(C1$,S$)
THEN REC%=K%:PRINT@(23,0),STRING$(60," "),;:
RETURN:ELSE NEXT K%
8035 PRINT@(23,0),"<<< NOT FOUND
>>>.....any key to continue...";:SOUND 7,2
8037 Z$=INKEY$:IF Z$="" THEN 8037
8040 PRINT@(23,0),STRING$(79," ");:RETURN
```

TRSTLB3/BAS
Model III - TRSDOS 1.3

```
1 'TRSTIMES/BAS (c) 1990 Danny C. Mullen 6641-B
Tracey Place, Fort Polk, LA 71459 * PUBLIC DOMAIN *
2 'Reads/displays/searches my TRSTIMES/LB data file
which    contains TRSTIMES articles & comments /in-
dex reference for Vol 1 No 1 thru current month.
3 'There is no method to change the data as written - I
use the LITTLE BROTHER database program to do the
normal update & additions - this is just a little help for
those who don't have LB.
4 'This is the TRSDOS 1.3 version.
```

```
5 CMD"B","OFF":CLS
6 CLEAR 256
7 PRINT@17,"TRSTIMES ARTICLE INDEX";
8 PRINT@128,"RECORD # ";:GOSUB 3000
10 OPEN "R",1,"TRSTIMES/LB":E=LOF(1)-1
20 FIELD 1,30 AS T$,20 AS A$,10 AS V$,20 AS C$,5 AS
M$,10 AS D$,161 AS C1$
30 GOSUB 2000
35 IF REC%<1 THEN REC%=1
ELSE IF REC%>E THEN REC%=E
37 PRINT@138,USING "###";REC%;:PRINT" OF";E;
40 GET #1, REC%
50 GOSUB 4000:A=INSTR(T$,CHR$(0)):
IF A=0 THEN PRINT T$;
ELSE PRINT LEFT$(T$,A-1);:A=0
55 GOSUB 4005:A=INSTR(A$,CHR$(0)):
IF A=0 THEN PRINT A$;
ELSE PRINT LEFT$(A$,A-1);:A=0
60 GOSUB 4010:A=INSTR(V$,CHR$(0)):
IF A=0 THEN PRINT V$;
ELSE PRINT LEFT$(V$,A-1);:A=0
65 GOSUB 4015:A=INSTR(C$,CHR$(0)):
IF A=0 THEN PRINT C$;
ELSE PRINT LEFT$(C$,A-1);:A=0
70 GOSUB 4020:A=INSTR(M$,CHR$(0)):
IF A=0 THEN PRINT M$;
ELSE PRINT LEFT$(M$,A-1);:A=0
75 GOSUB 4025:A=INSTR(D$,CHR$(0)):
IF A=0 THEN PRINT D$;
ELSE PRINT LEFT$(D$,A-1);:A=0
80 GOSUB 4030:A=INSTR(C1$,CHR$(0)):
IF A=0 THEN PRINT C1$;
ELSE PRINT LEFT$(C1$,A-1);:A=0
999 GOTO 30
1000 CLOSE 1:CLS
1010 PRINT "TRSTIMES/LB FILE CLOSED....."
1020 CMD"B","ON"
1999 END
2000 PRINT@896,"<<< L/R arrows to dec/inc, SHIFT
UP to quit, ENTER to GOTO # >>>";
2005 PRINT@980,"<<< or S for search >>>";
2010 Z$=INKEY$:IF Z$="" THEN 2010
2015 IF Z$=CHR$(27) THEN 1000
2017 IF Z$=CHR$(13) THEN GOSUB 5000:
PRINT@138,STRING$(53," ");:RETURN
2020 IF Z$=CHR$(9) THEN REC%=REC%+1:
IF REC%>E THEN REC%=E:RETURN:ELSE RETURN
2030 IF Z$=CHR$(8) THEN REC%=REC%-1:
IF REC%<1 THEN REC%=1:RETURN:ELSE RETURN
2035 IF Z$="S" THEN GOSUB 7000:
PRINT@960,STRING$(63," ");:RETURN
2040 GOTO 2010
3000 PRINT@256,"TITLE   : "
3005 PRINT@320,"AUTHOR  : "
3010 PRINT@384,"VOLUME  : "
3015 PRINT@448,"CATEGORY: "
3020 PRINT@512,"MODEL   : "
3025 PRINT@576,"DEBUG   : "
```

```
3030 PRINT@704,"COMMENTS: "
3035 RETURN
4000 PRINT@266,STRING$(30," ");:PRINT@266,;:
RETURN
4005 PRINT@330,STRING$(20," ");:PRINT@330,;:
RETURN
4010 PRINT@394,STRING$(10," ");:PRINT@394,;:
RETURN
4015 PRINT@458,STRING$(20," ");:PRINT@458,;:
RETURN
4020 PRINT@522,STRING$(5," ");:PRINT@522,;:
RETURN
4025 PRINT@586,STRING$(10," ");:PRINT@586,;:
RETURN
4030 PRINT@714,STRING$(161," ");:PRINT@714,;:
RETURN
5000 PRINT@138,;:
INPUT"What RECORD do you want";REC%:
IF REC%<0 OR REC%>E THEN 5000:RETURN:
ELSE RETURN
7000 '
7005 PRINT@896,STRING$(127," ");:PRINT@896,;:
INPUT"<A>scending or <D>escending search";S1$
7006 IF S1$="A" THEN GOTO 7010 ELSE GOTO 8000
7007 '
7008 '***** ASCENDING SEARCH ROUTINE *****
7009 '
7010 PRINT@896,STRING$(63," ");:
PRINT@896,"Search routine..";:
INPUT "Enter the data to search for";S$
7020 FOR K%=REC%+1 TO E:GET #1,K%
7025 PRINT@960,"ASCENDING ==> Searching
record #";K%;
7030 IF INSTR(T$,S$)ORINSTR(A$,S$)
ORINSTR(V$,S$)ORINSTR(C$,S$)ORINSTR(M$,S$)
ORINSTR(D$,S$)ORINSTR(C1$,S$)
THEN REC%=K%:PRINT@960,STRING$(63," ");:
RETURN:ELSE NEXT K%
7035 PRINT@960,"<<< NOT FOUND >>>.....any
key to continue...";
7037 Z$=INKEY$:IF Z$="" THEN 7037
7040 PRINT@960,STRING$(63," ");:RETURN
8000 PRINT@896,STRING$(63," ");:
PRINT@896,"Search routine..";:INPUT "Enter the data
to search for";S$
8020 FOR K%=(REC%-1) TO 1 STEP -1:
IF K%<1 THEN 8035: ELSE GET #1,K%
8025 PRINT@960,"DESCENDING ==> Searching
record #";K%;
8030 IF INSTR(T$,S$)ORINSTR(A$,S$)ORINSTR(V$,S$)
ORINSTR(C$,S$)ORINSTR(M$,S$)ORINSTR(D$,S$)
ORINSTR(C1$,S$)THEN REC%=K%:
PRINT@960,STRING$(63," ");:RETURN:ELSE NEXT K%
8035 PRINT@960,"<<< NOT FOUND >>>.....any
key to continue...";
8037 Z$=INKEY$:IF Z$="" THEN 8037
8040 PRINT@960,STRING$(63," ");:RETURN
```

HINTS & TIPS

THE RADIO SHACK DRUNK

Model III - TRSDOS 1.3

By Lance Wolstrup

Henry Herrdegen recently wrote a letter asking me if I knew who the 'RUMMY BUZZARD' is? He went on to explain that sector 2 of HERZ50/BLD (TRSDOS 1.3) contains a hidden message for this person.

I had no idea, as a matter of fact, I had never even run across it. However, Henry's info fascinated me, so I began to look for it. Super Utility4 would not display record 2 of HERZ50/BLD in 'file mode', so I searched for the message on a sector by sector basis. I finally found it on track 5, sector 11. That is, I found it on one disk, dated Wed Jul 1, 1981, but not on another which was also dated Wed. Jul 1, 1981. Strange - what the heck was RS doing to disks of the same series? The obvious conclusion is that someone at RS found it and decided that it wasn't funny and, rather than making any change to the date or version number, the message was discretely removed. I'll bet some heads rolled over that one!

For those of you tackling Assembly language, here is a short listing that will display whatever data is on track 5, sector 11 on the TRSDOS 1.3 disk in drive :0.

```
00050 ; RUMMY/SRC
00060 ; FOR TRSDOS 1.3
00070 ; DISPLAYS CONTENTS OF TRACK 5,
00080 ; SECTOR 11 ON THE SCREEN (DRIVE :0)
00090 ;
00100               ORG    7000H
00110 START         CALL   1C9H          ;cls
00120               LD     C,0           ;C is drive #
00130               LD     DE,050BH      ;D is track #,
                                         ;E is sector #
00140               LD     HL,BUFFER     ;HL=buffer
00150               CALL   4675H         ;pos. head &
                                         ;read sector
00160               LD     DE,15360      ;point de to
                                         ;top of scrn
00170               LD     BC,256        ;transfer buffer
00180               LDIR                 ;to screen
00190               LD     (4020H),HL    ;pos. cursor
                                         ;below text
00200               RET                  ;back to trsdos
00210 BUFFER        DEFS   256           ;storage area
                                         ;for sector data
00220               END    START
```

For the record, the routine spanning lines 120 to 150 can be used to read any sector, simply change the value in register C to the desired drive number; change the value in register D to the track number of your choice, and the value in register E to the new sector number.

I still do not know who this mysterious 'RUMMY BUZZARD' is, but if you have the right version of TRSDOS 1.3, at least you can now see the message intended for him. Have fun.

GALAXY

Model I/III & 4

By Sam McFarland

Who says that a program has to be long to be interesting? GALAXY/BAS is short and sweet. Type it in, RUN it, and experience a voyage through outer space.

(Editor's note: As Sam's program was submitted for Model 4 only, we made a couple of minor changes so our Model I & III people could join us on the trip.)

```
1 'GALAXY/BAS
2 'TRS-80 Model I/III & 4
3 'by Sam McFarland
4 '
10 PRINT CHR$(15)
20 A$="*":B$="+":C$=".":D$=" "
30 IF PEEK(42)=64 THEN C=RND(24)+39
ELSE C=RND(24)+55
40 A=RND(27):B=RND(24)+28:X=RND(0)
50 IF X<.15 THEN TP$=A$:A$=B$:B$=TP$:
GOTO 110
60 IF X<.3 THEN TP$=B$:B$=C$:C$=TP$:
GOTO 110
70 IF X<.45 THEN TP$=A$:A$=C$:C$=TP$:
GOTO 110
80 IF X<.6 THEN TP$=A$:A$=D$:D$=TP$:
GOTO 110
90 IF X<.75 THEN TP$=B$:B$=D$:D$=TP$:
GOTO 110
100 IF X<.9 THEN TP$=C$:C$=D$:D$=TP$
110 PRINT TAB(A);A$;TAB(B);B$;TAB(C);C$
120 IF INKEY$="" THEN 30
ELSE PRINT CHR$(14):END
```

BIORHYTHM

Model I/III & 4 - printer

By Rick DesMarteau

BIO/BAS is a translation of a Commodore 64 program. I found it in my collection of Public Domain disks and thought it might make a nice programming challenge for me. Well, it took me quite a bit longer than I expected, but here is the TRS-80 version of BIO/BAS.

(Editor's note: With Rick's permission, we added the code to allow the program to also run on Model I & III. Other minor additions were made as well.)

```
1 'BIO/BAS
2 'adapted for the TRS-80 Model I/III & 4
3 'by Rick DesMarteau
4 '
5 IF PEEK(42)=64 THEN CLEAR 2000:SW=64
ELSE SW=80
10 L$=STRING$(42,32):W$="sumotuwethfrsa":M1$=
"JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC":
C1=2*3.141593#:PRINT CHR$(15);:GOTO 100
21 H=INT((SW-LEN(A$))/2)
23 PRINT@V*SW+H,CHR$(30);A$;:RETURN
100 CLS:V=0:A$="BIORHYTHM":GOSUB 21:V=1:
A$="Adapted for TRS-80 by Rick DesMarteau":
GOSUB 21:V=2:H=0:A$=STRING$(SW,140):
GOSUB 23
110 V=5:H=22:
A$="Please enter your name: "+CHR$(14):
GOSUB 23:LINE INPUT N$:PRINT CHR$(15);:
IF N$="" THEN 110
120 V=6:H=0:
A$="Please enter your date of birth (mm/dd/yyyy): "
 +CHR$(14):GOSUB 23:LINE INPUT M$:
PRINT CHR$(15);
130 IF M$="" OR MID$(M$,3,1)<>"/"
OR MID$(M$,6,1)<>"/" THEN 120
140 IF VAL(MID$(M$,1,2))<1
OR VAL(MID$(M$,1,2))>12 THEN 120
ELSE M=VAL(LEFT$(M$,2))
150 IF VAL(MID$(M$,4,2))<1
OR VAL(MID$(M$,4,2))>31 THEN 120
ELSE D=VAL(MID$(M$,4,2))
160 IF LEN(M$)<>10 THEN 120
170 FL=0:FOR X=7 TO 10:
IF ASC(MID$(M$,X,1))<48 OR ASC(MID$(M$,X,1))>57
THEN FL=1
180 NEXT:
IF FL THEN 120 ELSE Y=VAL(MID$(M$,7,4))
190 IF M=2 AND Y/4=INT(Y/4) AND D<30
THEN 200 ELSE IF M=2 AND D>28 THEN 120
200 GOSUB 730:M=M4:D=1:Y=Y4:
GOSUB 610:S1=J:GOSUB 730:L1=31
210 V=8:H=1:
A$="Enter date for biorhythm calendar (mm/yyyy): "
 +CHR$(14):GOSUB 23:LINE INPUT M4$:
PRINT CHR$(15);
220 IF MID$(M$,3,1)<>"/" THEN 210
ELSE IF VAL(LEFT$(M4$,2))<1
OR VAL(LEFT$(M4$,2))>12 THEN 210
ELSE M4=VAL(LEFT$(M4$,2))
230 FL=0:FOR X=4 TO 7:
IF ASC(MID$(M4$,X,1))<48
OR ASC(MID$(M4$,X,1))>57 THEN FL=1
240 NEXT:IF FL THEN 210
ELSE Y4=VAL(MID$(M4$,4,4))
250 GOSUB 730:M=M4:D=1:Y=Y4:
GOSUB 610:S1=J
260 GOSUB 730:L1=31
270 IF M4=12 THEN 280
ELSE GOSUB 630:S3=N3:M=M4+1:
GOSUB 630:L1=N3-S3
280 B=J-S1+1:E=B+L1-1
290 V=11:A$="Your biorhythm will now print. Press
<ENTER> when your printer is ready "+CHR$(14):
GOSUB 21
300 I$=INKEY$:IF I$<>CHR$(13) THEN 300
ELSE H=0:A$="":GOSUB 23:
A$="Printing....":GOSUB 21
310 LPRINT:
LPRINT TAB(7)"BIORHYTHM INDEX FOR ";N$
320 LPRINT
330 LPRINT TAB(23)MID$(M1$,(M4-1)*3+1,3);Y
340 LPRINT TAB(9)"-....................0.................... +"
350 V=0:FOR I=B TO E:V=V+1:J3=I-1:
K1=J3/23:K3=J3/28:K5=J3/33:K2=K1-INT(K1):
K4=K3-INT(K3):K6=K5-INT(K5)
360 P2=SIN(C1*K2):E2=SIN(C1*K4):I2=SIN(C1*K6)
370 O=P2+E2+I2:O=INT(16666*(O+3))+1:
P=INT(21.5+20*P2):Q=INT(21.5+20*E2):
R=INT(21.5+20*I2)
380 MID$(L$,21,1)="."
390 IF P>41 THEN 400 ELSE MID$(L$,P,1)="p"
400 IF Q>41 THEN 410 ELSE MID$(L$,Q,1)="e"
410 IF R>41 THEN 420 ELSE MID$(L$,R,1)="i"
420 LPRINT O;TAB(10);L$;
430 LPRINT TAB(54);V;MID$(W$,(N2-1)*2+1,2)
440 L$=STRING$(41,32):N2=N2+1
450 IF N2<8 THEN 460 ELSE N2=1
460 NEXT I
470 LPRINT TAB(9)"-....................0.................... +"
480 LPRINT:LPRINT TAB(10)"I = Intelligence"
490 LPRINT TAB(10)"P = Physical Abilities"
500 LPRINT TAB(10)"E = Emotions"
510 LPRINT TAB(10)"A curve to the right of the central
dotted"
520 LPRINT TAB(10)"line indicates a good time for that
trait,"
530 LPRINT TAB(10)"while a curve to the left indicates a
bad time."
540 LPRINT TAB(10)"A number on the left around
50000 indicates a"
550 LPRINT TAB(10)"critical time; you should be on
your guard,"
560 LPRINT TAB(10)"especially with a trait that is
tending to the"
570 LPRINT TAB(10)"left at that time."
580 LPRINT CHR$(12)
590 V=11:A$="Would you like another month (Y/N) "
 +CHR$(14):GOSUB 21
600 I$=INKEY$:
IF I$="Y" OR I$="y" THEN PRINT CHR$(15);:
```

```
V=8:H=0:A$=CHR$(31):GOSUB 23:GOTO 210
ELSE IF I$="N" OR I$="n" THEN CLS:END ELSE 600
610 IF M<3 THEN M1=M+10:Y1=Y-1
ELSE M1=M-2:Y1=Y
620 C=INT(Y1/100):D1=Y1-(C*100):
N4=INT((13*M1-1)/5)+D+D1+INT(D1/4):
N=N4+INT(C/4)-2*C+77:N1=INT(N/7):
N2=N-N1*7+1:RETURN
630 Y2=INT(Y/4):Y3=Y-Y2*4
640 IF Y3=0 THEN Y2=INT(Y/100):Y3=Y-Y2*100
ELSE 680
650 IF Y3=0 THEN Y2=INT(Y/400):Y3=Y-Y2*400
ELSE 670
660 IF Y=0 THEN 670 ELSE 680
670 L1=1:GOTO 690
680 L1=0
690 N1=INT((3055*(M+2))/100)-91:L=0
700 IF M<3 THEN 720
710 IF L1=0 THEN L=2 ELSE L1=0
720 N3=N1+D-L:RETURN
730 IF M<3 THEN M1=M+9:Y1=Y-1
ELSE M1=M+9:Y1=Y-1
740 C=INT(Y1/100):D1=Y1-C*100
750 N=INT((146097!*C)/4)+D+INT((1461*D1)/4)
760 J=N+1.72112E+06+INT((153*M1+2)/5):
RETURN
```

TRSDOS 6.3.1 TIPS

Model 4

By Lance Wolstrup

The character used to separate the filename from the extension has always been the / (slash). This is still true with LS-DOS 6.3.1, however, Roy Soltoff has given us the option of using the . (period) with the DIR, CAT and BACKUP commands in conjunction with the 'wild-card' feature.

For example: **DIR C.:0** (or **DIR C/:0**)
should give you a directory of CLICK/FLT, COM/DVR, COMM/CMD and CONV/CMD (and any other visible file you might have on drive :0 starting with the letter C).

CAT C.C:0 (I) (or CAT C/C:0)
should display COM/DVR, COMM/CMD and CONV/CMD to the screen, as well as any other visible or invisible files you might have on drive :0 starting with the letter C and having C as the first letter of the extension.

BACKUP .CMD:0 :1 (or **BACKUP /CMD:0 :1**)
should copy all visible files having the extension /CMD from drive :0 to drive :1.

I checked my manuals and the . (period) ability is not documented. While checking, however, I did come across a couple of mighty fine features which, though documented, I never knew existed.

You can limit a multidrive DIR with the - (hyphen) character.

Normally, when you type DIR and omit the drive number, TRSDOS/LS-DOS 6 (and LDOS) will display the directory of all the drives in your system, one drive at a time.

The - parameter acts as a delimiter; for example, if you have a four drive system and wish to display the directories of JUST drive 1 and 2, you can issue the following command:

DIR 1-2 <ENTER> (or DIR :1-2 <ENTER>)

You can also use the hyphen to indicate default parameters. For example, if you wish to display the directory of all drives from 0 to 2, you can do this:

DIR -2 <ENTER>

Alternatively, if you want to display the directory of the drives from drive 2 to the last drive, type:

DIR 2- <ENTER>

The CAT command follows the same rules.

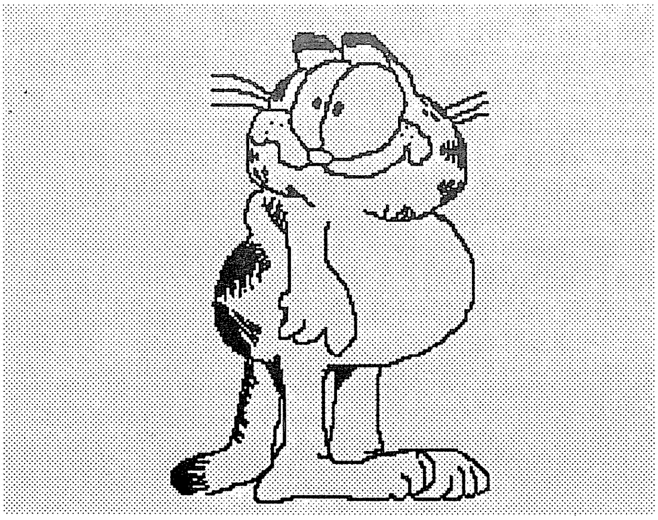
The other command that surprised me with an, until now, unknown capability, is LIST. I have LISTed files for years, always reading ASCII. Well, by adding the H parameter, LIST will display the file in hex mode; that is, it will display the file just like a ZAP program. For example, if you wish to see what SYS0/SYS looks like, you issue this command:

LIST SYS0/SYS.SYSTEM6 (H) <ENTER>

This display is most useful, especially if you are about to enter patches. Now you can see if the F(ind) bytes are where they are supposed to be. Incidentally, the LDOS LIST command works identically.

The DIR, CAT and LIST options are very adequately covered in the manual, I just never bothered to read that portion. I mean - *really* - who needs to read about DIR, CAT and LIST? Just goes to show you!!

Now, if somebody will tell me how to backup invisible and system files using the wildcard method, I'd be most interested.



ATTENTION TRSDOS 1.3. USERS!

ANNOUNCING "SYSTEM 1.5.", THE MOST COMPREHENSIVE 1.3. UPGRADE EVER OFFERED!

MORE SPEED!! MORE POWER!! MORE PUNCH!!

While maintaining 100% compatibility to TRSDOS 1.3., this DOS upgrade advances TRSDOS 1.3. into the 90's!
SYSTEM 1.5. supports 16k-32k bank data storage and 4MGHZ clock speed (4/4P/4D).

DOUBLE SIDED DRIVES ARE NOW 100% UTILIZED! (all models).

CONFIG=Y/N	CREATES CONFIG BOOT UP FILE	DATE=Y/N	DATE BOOT UP PROMPT ON or OFF
TIME=Y/N	TIME BOOT UP PROMPT ON or OFF	CURSOR='XX'	DEFINE BOOT UP CURSOR CHAR
BLINK=Y/N	SET CURSOR BOOT UP DEFAULT	CAPS=Y/N	SET KEY CAPS BOOT UP DEFAULT
LINE='XX'	SET *PR LINES BOOT UP DEFAULT	WP=d,Y/N (WP)	WRITE PROTECT ANY or ALL DRIVES
ALIVE=Y/N	GRAPHIC MONITOR ON or OFF	TRACE=Y/N	TURN SP MONITOR ON or OFF
TRON=Y/N	ADD an IMPROVED TRON	MEMORY=Y/N	BASIC FREE MEMORY DISPLAY MONITOR
TYPE=B/H/Y/N	HIGH/BANK TYPE AHEAD ON or OFF	FAST	4 MGHZ SPEED (MODEL 4'S)
SLOW	2 MGHZ SPEED (MODEL III'S)	BASIC2	ENTER ROM BASIC (NON-DISK)
CPY (parm,parm)	COPY/LIST/CAT LDOS TYPE DISKS	SYSRES=H/B/'XX'	MOVE/SYS OVERLAY(s) TO HI/BANK MEM
SYSRES=Y/N	DISABLE/ENABLE SYSRES OPTION	MACRO	DEFINE ANY KEY TO MACRO
SPOOL=H/B,SIZE	SPOOL is HIGH or BANK MEMORY	SPOOL=D,SIZE='XX'	LINK MEM SPOOLING TO DISK FILE
SPOOL=N	TEMPORARILY DISABLE SPOOLER	SPOOL=Y	REACTIVATE DISABLED SPOOLER
SPOOL=RESET	RESET (NIL) SPOOL BUFFER	SPOOL=OPEN	OPENS, REACTIVATES DISK SPOOLING
SPOOL=CLOSE	CLOSES SPOOL DISK FILE	FILTER *PR.ADLF=Y/N	ADD LINE FEEDS BEFORE PRINTING 0DH
FILTER *PR.IGLF	IGNORES 'EXTRA' LINE FEEDS	FILTER *PR.HARD=Y/N	SEND 0CH to PRINTER (FASTEST TOF)
FILTER *PR.FILTER	ADDS 256 BYTE PRINTER FILTER	FILTER *PR.ORIG	TRANSLATE PRINTER BYTE TO CHNG
FILTER *PR.FIND	TRANSLATE PRINTER BYTE TO CHNG	FILTER *PR.RESET	RESET PRINTER FILTER TABLE
FILTER *PR.LINES	DEFINE NUMBER LINES PER PAGE	FILTER *PR.WIDTH	DEFINE PRINTER LINE WIDTH
FILTER *PR.TMARG	ADDS TOP MARGIN to PRINTOUTS	FILTER *PR.BMARG	ADDS BOTTOM MARGIN to PRINTOUT
FILTER *PR.PAGE	NUMBER PAGES, SET PAGE NUMBER	FILTER *PR.ROUTE	SETS PRINTER ROUTING ON or OFF
FILTER *PR.TOF	MOVES PAPER TO TOP OF FORM	FILTER *PR.NEWPG	SET DCB LINE COUNT TO 1
FILTER *KI.ECHO	ECHO KEYS to the PRINTER	FILTER *KI.MACRO	TURN MACRO KEYS ON or OFF
ATTRIB:d.PASSWORD	CHANGE MASTER PASSWORD	DEVICE	DISPLAYS CURRENT CONFIG INFO

All parms above are installed using the new LIBRARY command SYSTEM (parm,parm). Other new LIB options include DBSIDE (enables double sided drive by treating the "other side" as a new independent drive, drives 0-7 supported) and SWAP (swap drive code table #s). Dump (CONFIG) all current high and/or bank memory data/routines and other current config to a disk data file. If your type ahead is active, you can (optional) store text in the type buffer, which is saved. During a boot, the config file is loaded back into high/bank memory and interrupts are recognized. After executing any active auto command, any stored type ahead data will be output. FANTASTIC! Convert your QWERTY keyboard to a DVORAK! Route printer output to the screen or your RS-232. Macro any key,even F1, F2 or F3. Load *01-*15 overlay(s) into high/bank memory for a memory only DOS! Enter data faster with the 256 byte type ahead option. Run 4MGHZ error free as clock, disk I/O routines are properly corrected! Spool printing to high/bank memory. Link spooling to disk (spooling updates DCB upon entering storage). Install up to 4 different debugging monitors. Print MS-DOS text files, ignoring those unwanted line feeds. Copy, Lprint, List or CATalog DOSPLUS, LS-DOS, LDOS or TRSDOS 6.x.x. files and disks. Add top/bottom margins and/or page numbers to your hard copy. Rename/Redate disks. Use special printer codes eg: LPRINT CHR$(3); toggles printer output to the ROUTE device. Special keyboard codes add even more versatility. This upgrade improves date file stamping MM/DD/YY instead of just MM/YY. Adds optional verify on/off formatting, enables users to examine *01-*15, DIR, and BOOT sectors using DEBUG, and corrects all known TRSDOS 1.3. DOS errors. Upgrade includes LIBDVR, a /CMD driver that enables LIBRARY commands, such as DIR, COPY, DEBUG, FREE, PURGE, or even small /CMD programs to be used within a running Basic program, without variable or data loss.

By special arrangement with GRL Software,
SYSTEM 1.5. is now distributed exclusively by TRSTimes magazine.

ORDER YOUR COPY TODAY!

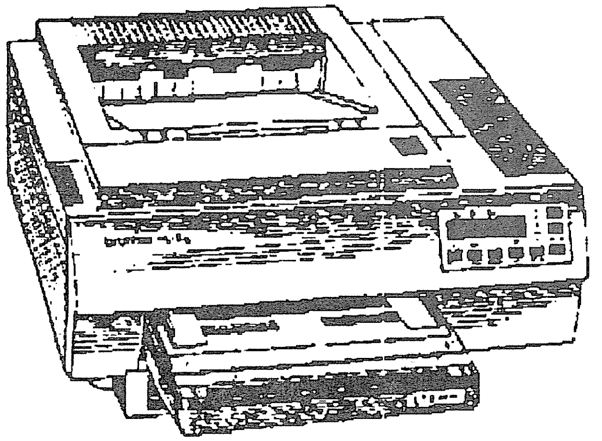
Send $39.95 (U.S. funds) to:

TRSTimes - SYSTEM 1.5.
5721 Topanga Canyon Blvd., Suite 4
Woodland Hills, CA. 91367

AllWrite! and the HP DeskJet Plus

(or Fonts and Widths and Codes, Oh My!)

By George D. Madison



Sometime back, the C.Itoh 8510 dot-matrix printer that I had been using for years finally gave up the ghost; a logic chip failed, and as the only way to repair it was to replace the entire motherboard -- an option that cost far more than a 4 year old printer is worth -- I decided it was time to get a new printer.

It didn't take me long to settle on the Hewlett-Packard DeskJet Plus. A good friend of mine has the original DeskJet, and I had been impressed with the quality of its output. Since I knew that ProSoft had brought out a LaserJet option for AllWrite!, my only concern at the time was getting hold of a copy, since I knew that all HP printers use various versions of HP-PCL (Printer Control Language). I anticipated only a minimum of difficulty.

Boy, was I wrong.

To be fair, most of the time that I wound up wasting, and a lot of the frustration were my own fault. If I'd had the brains to ask for help earlier, I might have had my new printer online with AllWrite! much quicker. However, there are a few GOTCHAS! I can warn you about if you have any plans to follow in my footsteps.

First, however, I would like to thank Lee Rice of Marquette University and Gary Shanafelt of McMurry College. These two gentlemen have done a great deal of work on using AllWrite! with the HP Laser- and DeskJet printers, and without their help, I'd probably still be fumbling around.

To begin with, one must realize that the latest and greatest from HP at the time the LaserJet Option Package was put together was the LJ 500+; the LaserJet Series II hadn't come out yet. This is a long way of saying the package is, in certain important ways, OLD.

The first place that this comes up is the fact that the width tables for the Times-Roman font that come with the package are not usable. Not only does the DeskJet deal with the fonts in a slightly different way that would in itself cause spacing problems, but HP has refined the font over the years to look better, which has introduced further spacing changes.

So, one must obtain the latest width data from HP on MS-DOS disks, and figure out a way to get at it. (I used a borrowed copy of SuperCross.)

The next problem is figuring out how to USE it, and this is where I wasted most of my time and got most of my frustration. The data is, unfortunately NOT well documented. What HP doesn't make clear is that the first disk in the 2-disk set is for the Times-Roman cartridge for the ORIGINAL DeskJet (22706P) -- which is different from the Times-Roman cartridge optimized for the DeskJet Plus (22706R), the data for which is on the SECOND disk. I only found this out when I sent a frustrated letter off to Gary Shanafelt, who told me that HE only recently noticed the problem because the character set that he uses is only on the second disk, so he never realized I might get confused.

(These printers support a variety of "character sets" -- different sets of characters for the codes from 128-255. I chose to use PC-8, which gives me access to line drawing and other useful special characters.)

The next problem is to reduce the width tables to usable form, since they are provided simply as tabular data. Fortunately, one does not have to re-key all this data. Rice and Shanafelt have come up with a number of useful utilities to make life a little easier, and one of them called WID2TAB is what I used here.

I used AllWrite!'s editor to quickly "massage" the width table into raw data "/WID" files, with the format expected by WID2TAB. I then compiled the data into the necessary /TAB files. The /TAB files aren't finished after this step, however; one must then add the proper control code used to call the font described by the width table before the /TAB file can be used.

To do this, one can use HPWIDTAB/BAS (originally written by ProSoft, and much enhanced by Shanafelt and Rice), or do what I did -- use a file zapping program to modify the files directly. The structure of a /TAB file is very straightforward, and with a "map" of the first sector of an example file, I had no trouble correctly adding the necessary control codes. For someone who isn't comfortable with a file zapper, however, I'd suggest using HPWIDTAB.

Unfortunately, ProSoft used the extension "/TAB" for several VERY different file types, so it's important not to get them confused. A "/TAB" file can be either:
(1) A HP Laser/DeskJet font width table
(2) an AllWrite *daisywheel* width table
(3) a complete AllWrite printer driver.

Once one has all the width table files set up, one makes a list which assigns a pitch number to each width table. This number will be used with AllWrite!'s ";pi" control word to select the various fonts. Even the fonts built in to the printer have to have width tables defined for them; otherwise, the driver will have no "knowledge" of the font.

Finally, one is ready to use the HPINSTAL program from ProSoft -- and here lay several more bugaboos! First off, the program will ONLY accept input in UPPER CASE -- so be careful when typing in the name of your list file.

Second, there were two versions of HPINSTAL released by ProSoft, and the package I got included both of them. The LATER version, dated October '86 is the version to use.

Lastly, the driver that HPINSTAL generates uses CR/LF pairs, since the early LaserJets were locked into the MS-DOS standard of using CR/LF's to terminate a line. The TRS-80, however, uses only a CR, and the DeskJet can be set to accept this. However, if one sets the DeskJet to work properly with all the other TRS-80 software, the AllWrite! driver will double-space.

Here, I'm afraid, one MUST use a disk zapper, and page through the first three sectors of the driver, searching for "0D0A" pairs and zapping them to be "0D00" pairs.

Once one works through all this, one gets absolutely beautiful proportionally spaced output -- better, in fact, than I've seen from many MS-DOS word processors. I use MicroSoft WORD 5.0 at work, and its proportional spacing is positively crude compared to that of AllWrite!

There is still work to be done, however; the age of the HPINSTAL program shows in that it does not take advantage of new features added since it was written; I am trying to decipher the format of AllWrite!'s printer drivers so that I can take advantage of these features, and any help would be much appreciated!

In case my tale of woe has discouraged any of you, realize that those of you reading this who decide to get a DeskJet won't have the trauma I did, since I will be more than happy to assist any of you who decide to move up to the fantastic print quality of the HP DeskJet printer. I just wanted all of you to appreciate the value of the advice and assistance you'll be getting. (GRIN!)

USED TRS 80 SALE

USED TRS 80 SALE

RADIO SHACK TANDY OWNERS

Find The Computer Equipment That Tandy No Longer Sells

Computers

Mod 3 2 Drive	$255
Mod 4 2 Drive	$345
Mod 4 2 Drive 128K	$365
Mod 4P 2 Drive	$345
Mod 100 24K	$235

Printers & Hard Drives

Tandy 5 Meg H.D.	$275
Tandy 12 Meg H.D.	$345
Tandy 15 Meg H.D.	$425

All hard disks include cable & software

DMP 105	$105
DMP 120	$145
MD 4 Multi Plexer & Hard Drive	$245
CGP 220 Ink Jet	$215
DMP 130	$175
DMP 200	$155
DMP 420	$325
DMP 430	$365
DMP 2100 24 Pin	$395
DWP II	$325
DWP 410	$245
DWP 210	$215
DWP 510	$395
Line Printer 5	$195
Line Printer 6	$135
410 Tractor (New)	$75
Tractor for 2100	$115
DW II Tractor	$115
DW II Sheet Feeder (New)	$245
CGP 115 Color Graphic	$95

Software and Miscellaneous

MD 3 Scriptsit Dictionary	$22
MD 3 Superscripsit	$55
MD 3 Time Manager	$14
MD 3 Videotex Plus	$25
MD 3 Checkwriter 80	$18
MD 3 Cobol	$45
MD 3 Disk Stock Market Trend	$24
MD 3 Fortran	$45
MD 3 Pascal	$45
MD 3 Profile	$15
MD 3 Profile Plus	$29
MD 3 Scripsit Disk	$29
MD 3 Home Accounant	$39
MD 3 DOS Plus	$24
MD 3 Advanced Statistical Analysis	$19
MD 3 Zaxon	$12
MD 3 T80 Flight Simulator	$25
MD 3 Maxi Manager Database	$20
MD 4 Assembly Language Development	$45
MD 4 Superscripsit Dictionary	$25
MD 4 TRS DOS Training Course	$26
MD 4 P.F.S. File	$45
MD 4 P.F.S. Target Planner Calc	$35
MD 4 Electric Webster Proofing System	$28
MD 4 Double Duty	$34
MD 4 M S Script	$34
MD 4 Allwrite	$45
MD 4 Accounts Payable	$55
MD 4 Accounts Receivable	$55
MD 4 W-2 Writer	$29
MD 4 Deskmate	$69
MD 4 Business Statistics Analysis	$34
MD 4 Real Estate Loans Analysis	$34
MD 4 T.K. Solver	$45
MD 4 TRS 80 Computer Graphics	$35
MD 4 Video Tex Plus	$39
MD 4 Superscripsit	$55
Mod 4 TRS DOS & Manual	$24
Mod 4/4P Technical Reference	$29
Printer Cables Mod 3-4	$15
Mod 3/4 Disk Drives Internal	$55
Modem 4P	$35
Printer Selector Switch	$45
RS 232 Selector Switch	$39

• All equipment is guaranteed to be in good working order. • Equipment is cleaned and tested.
• Drives are cleaned and timed as needed.

We accept VISA & MasterCard or C.O.D. The above prices do not reflect shipping cost. Inventory changes daily; please call for availability. If you don't see what you need, please call and we will do our best to locate it for you.

***Pacific Computer Exchange** The One Source For Used Tandy Equipment!*

(503) 236-2949

PACIFIC COMPUTER EXCHANGE

1031 S.E. Mill, Suite B • Portland, Oregon 97214

Is Your RND Really Random?

Testing the BASIC RND Function

By Delmer D. Hinrichs

If you want to be absolutely correct, the answer to the title question must be 'No.' The BASIC RND function generates each new 'random' number from the previous number, starting from an initial 'seed' number, by a simple calculation. Thus these apparently random numbers are more correctly called 'pseudo-random' numbers. For simplicity, from now on we will just call them 'random' numbers. Which still leaves us with the question: Are these apparently random numbers close enough to being truly random to be useful?

But useful for what? If you use the RND function merely to get a different starting point for a Lunar Lander game, it doesn't make much difference how random it is. If you use RND to simulate dealing cards for a poker game, you may lose more games than you should if the RND is not random, but still it isn't really serious. However, if you use RND in a Monte Carlo simulation to determine the pricing policy for your business, or for some other serious use, you must make sure that your RND is adequately random before you can trust the results. For these critical uses, or even to check on the honesty of your poker dealer, there are tests that can be run on your RND function.

How can we check how random our RND function is? Strangely enough, as the value of each individual random number becomes more uncertain (more truly random), the statistics of a large number of these random numbers become more certain. Therefore we can let our RND function produce a large number of random numbers and check their statistics.

Which statistics, and how do we check them? There are many different statistics that could be used, as random numbers have many different random characteristics. The program listing, RNG/BAS, has fourteen different routines that test for sixteen different random characteristics. A truly random number generator (RNG) would pass all of these tests. Any one RND function may pass some tests easily, but fail others. The uses to which you put your RND determine which tests are most important to you.

How can we tell if an RNG fails to pass a test? We cannot really trust our unaided judgement on this, as a series of numbers that appear random may not be random, and vice versa. For example, if we throw a single die, each of the six faces should come up about one-sixth of the time. If we throw the die 600 times, would we expect each face to come up exactly 100 times? I think that we can agree that this is highly unlikely. But just how much should the number of times that each face comes up differ from 100? One of the most useful statistical tests for this type of problem is the chi square test.

To apply the chi square test to our die-throwing experiment, we compute:

$$X = \Sigma \frac{(n_i - 100)^2}{100}$$

where: X = the chi square statistic (Greek letter chi)
Σ = summation symbol, indicating that all six computed values are to be added together (Greek letter sigma)
n_i = number of times that face 'i' comes up (i = integers 1-6)
100 = 'expected' number of times that each face should come up

Now that we have the chi square, how do we use it? To interpret the chi square, we need a chi square distribution table such as Table 1. There we find another new term, 'degrees of freedom.' This is simply the number of independent choices possible, or one less than the number of possible categories. For throwing a die, there are five degrees of freedom (if the face is not numbers 1 through 5, then it **must** be six, so there are only 5 independent choices).

In Table 1, the 50% probability column is the 'Ideal' chi square level, with the results neither more uniform (probability > 50%, too small a chi square) nor more scattered (probability < 50%, too large a chi square) than would be

Table 1. Chi-Square Distribution Table.

	P=99%	P=95%	P=75%	P=50%	P=25%	P=5%	P=1%
DF=1	0.00016	0.00393	0.1015	0.4549	1.323	3.841	6.635
DF=2	0.02010	0.1026	0.5753	1.386	2.773	5.991	9.210
DF=3	0.1148	0.3518	1.213	2.366	4.108	7.815	11.34
DF=4	0.2971	0.7107	1.923	3.357	5.385	9.488	13.28
DF=5	0.5543	1.1455	2.675	4.351	6.626	11.07	15.09
DF=6	0.8720	1.635	3.455	5.348	7.841	12.59	16.81
DF=7	1.239	2.167	4.255	6.346	9.037	14.07	18.48
DF=8	1.646	2.733	5.071	7.344	10.22	15.51	20.09
DF=9	2.088	3.325	5.899	8.343	11.39	16.92	21.67
DF=10	2.558	3.940	6.737	9.342	12.55	18.31	23.21
DF=11	3.053	4.575	7.584	10.34	13.70	19.68	24.73
DF=12	3.571	5.226	8.438	11.34	14.84	21.03	26.22

"DF" is the Degrees of Freedom.
"P" is the Probability. 50% is the "ideal" value, showing that the scatter from the most likely result is just what would be expected. Greater than 50% (low chi square values) shows scatter from the most likely result to be less than expected, while less than 50% (high chi square values) shows scatter to be greater than expected. Values outside the 1% to 99% range should be rejected, while values outside the 5% to 95% range are suspect.

obtained from truly random events. If each of the six faces of the die came up exactly 100 times, the chi square would be zero, or much more uniform than would be expected from random events. If we had three faces coming up 110 times each, and three faces coming up 90 times each, the chi square would be 6.000, or just slightly more scattered than the ideal chi square value of 4.351. Actually, any chi square value from 1.15 to 11.1 (with probabilities of 95% to 5%) would be quite reasonable for the die-throwing experiment. If the probability is greater than 99% or less than 1%, it is normally grounds for the rejection of the result. Of course, if you run hundreds of trials, you can expect about one of each 100 chi squares to have a probability greater than 99%, and the same for a probability less than 1%.

Random events that occur as a continuous distribution instead of as a small number of discrete categories are not suitable for the use of the chi square statistic. In some of these cases, the 'Z score' statistic may be used to check how well the observed events fit the theoretical probabilities. The 'Ideal' Z score is zero, with a 5% probability of its exceeding ±1.96, and a 1% probability of its exceeding ±2.58.

Some other types of random events are more difficult to treat probabilistically, but we may still gain some added information on randomness by just looking at the results.

Now how do we apply these methods to testing a random number generator? Let us go through the 14 routines of the program listing one by one. I have included the degrees of freedom (DF) for the chi square tests.

1. Distribution Test (DF = 9)

The distribution test (sometimes called the frequency test) checks if the RNG gives a uniform distribution of numbers. That is, if all numbers within its range are equally likely to occur. This is one of the simplest tests, and most RNGs pass it. Random integers (1 through 10) should each occur about 10% of the time.

The batch chi square measures the short-term randomness while the cumulative chi square measures the long-term randomness. The RNG may pass one chi square test but not the other. For example, if the RNG produced a slight systematic excess of high integers, it could still pass the chi square test for each batch, but the cumulative chi square would gradually become larger and larger, showing a greater and greater variation from true randomness.

2. Mean and Standard Deviation

Random numbers (0 to 1) should average about 0.500000, with a standard deviation of about 0.2887. For each batch of 1000 random numbers the program displays the batch number, the batch mean, the batch standard deviation, the cumulative mean of all batches, and the cumulative standard deviation of all batches.

3. Correlation Test

Random numbers (0 to 1) should have a serial correlation coefficient of approximately zero. That is, each new number should not depend upon the previous number in any regular way. A positive correlation shows a tendency to run, while a negative correlation shows a tendency to alternate.

For each batch of 1000 sequential pairs of random numbers, the program shows the batch number, the correlation coefficient, the batch Z score, and the cumulative Z score for all batches taken together.

4. Serial Test (DF = 8)

Random integers (0, 1, or 2) should occur in pairs in equal numbers. That is, the pair 0,0 should occur about as often as 0,1 or 0,2 or 1,0 etc.

For each batch of 1000 pairs, the program shows the batch number, the count of each of the nine possible pairs, the batch chi square, and the cumulative chi square of all batches taken together.

5. Run Test (DF = 4)

Random numbers (0 to 1) should have 'runs up,' where each succeeding number is larger than the last, in a predictable manner. For this test, the count of the number of random numbers needed to give 1000 'runs up' will also vary a little.

For each batch of 1000 'runs up,' the program shows the batch number, the count for the batch, the number of 'runs up' of length zero (the next random number was smaller), one, two, three, and four or more, then the batch chi square and the cumulative chi square for all batches.

6. Gap Test (DF = 5)

Random numbers (0 to 1) should have 'gaps' where no random number is less than 0.5 in a predictable manner. For this test, the number of random numbers needed to give 1000 'gaps' will also vary a little.

For each batch of 1000 'gaps,' the program shows the batch number, the count for the batch, the number of 'gaps' of length zero (succeeding random numbers were less than 0.5), one through four, and five or more, then the batch chi square and the cumulative chi square for all batches.

7. Permutation Test (DF = 5)

If three different random numbers (0 to 1) are created, there are six different possible permutations. That is, LMH, LHM, MLH, MHL, HLM, and HML are equally likely, where L = low, M = medium, and H = high.

For each batch of 1000 sets of permutations, the program shows the batch number, the bin count for each permutation, the batch chi square, and the cumulative chi square for all batches.

8. Mean Square Successive Difference Test

Successive random numbers (0 to 1) should have a predictable average squared difference. This routine calculates both eta and the Z score. Eta is the sum of the squared differences of the successive iterations divided by the sum of the square of the differences between each iteration and the mean. The Z score is then calculated from eta. A positive Z score indicates long trends, while a negative Z score indicates short oscillations.

For each batch of 1000 differences, the program shows the batch number, eta, the Z score, and the cumulative Z score.

9. Poker Test (DF = 3)

A 'hand' of four random integers (1 through 10) has a predictable number of duplications of integers. The matches may be none (garbage), one (1 pair), two (2 pairs), three (3 of a kind), or four (4 of a kind). Since four matches occur so seldom, they are combined with three matches for the chi square calculation.

For each batch of 1000 'hands,' the program shows the batch number, the number of matches for each category, the batch chi square, and the cumulative chi square.

10. Coupon Collector's Test

For random integers (1 through 10), to get at least one of each integer takes an average of 29.29 samples.

For each batch of 100 complete sets, the program shows the batch number, the average number of samples for the batch, and the cumulative average.

11. Pascal's Triangle Test (DF = 8)

For a ten-level Pascal's triangle, if a 'dropping ball' has a 50% chance of being deflected right or left at each level (like a pin-ball machine), the distribution of balls in the eleven bins at the bottom is predictable (see Table 2). Since there will be so few balls in the end bins, these are combined with the next-to-end bins for the chi square calculation.

For each batch of 1024 'balls,' the program shows the batch number, the number of balls in each bin, the batch chi square, and the cumulative chi square.

12. Maximum of 'T' Test (DF = 3)

For three successive random integers (1 through 10), each should be the largest 28.5% of the time, with no maximum (duplication of the high integer) 14.5% of the time.

For each batch of 1000 sets of three random integers, the program shows the batch number, the number of 'no maximum,' the number of times that each of the three sampled integers was maximum, the batch chi square, and the cumulative chi square.

Table 2. Pascal's Triangle

```
                              1
                           1     1
                        1     2     1
                     1     3     3     1
                  1     4     6     4     1
               1     5    10    10     5     1
            1     6    15    20    15     6     1
         1     7    21    35    35    21     7     1
      1     8    28    56    70    56    28     8     1
   1     9    36    84   126   126    84    36     9     1
1    10    45   120   210   252   210   120    45    10     1
```

Pascal's Triangle - Ten-Level. Each number is the sum of the two numbers directly above it. Also, the sum of the numbers at each level is twice the sum of the numbers of the preceding level. If the triangle is considered as a pinball machine, the numbers represent the relative probabilities of a ball ending up in that bin (at any level).

13. Rectilinear Random Walk (Drunkard's Walk)

If walls are set up at X equals 20 and at Y equals 20, and a random walk started at X = 0 and Y = 0, on the average it should take 472 random steps (right or left, up or down) to hit a wall, with a standard deviation of 260 steps. After 25 walks, the average number of steps should be between 420 and 524. Also, X and Y, plus and minus walls should be hit equally often.

For each random walk, the program shows the walk number, the X and Y values after a wall is hit, the number of steps required for this walk, the average number of steps for all walks, and (after the first walk) the standard deviation of the number of steps. Note that this test is quite variable.

14. Looping Test

A random number generator, producing floating point random numbers between zero and one, should not reproduce its original random number, nor any random number produced later. If it does, it will loop (produce the same sequence of numbers again). This test checks the random numbers for duplication at user-specified intervals. Up to 100 random numbers may be stored and checked against all new random numbers for duplication.

If a duplication occurs, the sequence numbers of the duplicate random numbers are displayed. Otherwise, only the number of random numbers that have been checked is displayed.

Note that if one duplication is found, many others will follow, as the same loop of numbers is generated again and again. To test one million random numbers, you must set the interval at least as large as 10000.

Results

So, after running all of these RNG tests, how good are the TRS-80 random number generators? As you can see in Table 3, that depends: If you are using Model I or Model III BASIC, it passes all tests, at least up to one million numbers. Model 4 BASIC is almost as good, giving a Permutation test result that is only slightly questionable (the results showed too little scatter for truly random numbers). On most other tests, Model 4 BASIC gave better results than Model I/III BASIC.

However, if you are using the Model I Microsoft BASIC compilers 5.2 (to make stand-alone .CMD files), or 5.23 (to make .CHN files that require a BRUN/CMD run-time module), you may be in trouble.

The earlier 5.2 compiler failed most tests very badly, giving extremely poor results with only ten batches, as indicated by the underlined results. However, the 5.2 compiler RNG did pass the Distribution, Mean, Standard Deviation, and Looping tests. This shows how necessary it is to run a battery of tests on an RNG before deciding that it is good enough to use.

The later 5.23 compiler gave somewhat better results than the 5.2 compiler, but it still was not acceptable. In getting most test results to run at least for one million numbers before showing up as 'Bad,' this compiler's RNG now fails the Distribution and Looping tests.

The failure of these two Microsoft BASIC compilers for the Model I is most discouraging. To run serious simulations one would like to use compiled programs because of their added speed, but it is not safe to do so with these two compilers.

Table 3.

	Model I/III	**Model I Compiled BASIC**		**Model 4**	**MS-DOS**
Test	**BASIC**	**MS 5.2**	**MS 5.23**	**BASIC**	**QB4.00**
1. Distribution	Ideal Chi = 8.34 (1000)				
	7.93 (1000)	7.03 (1175)	**34.68** (1000)	11.7 (1000)	11.90 (1000)
2. Mean	Ideal mean = .5000 (1000)				
	.5005 (1010)	.5000 (1475)	.5001 (721)	.5000 (1060)	.5001 (1000)
Standard Deviation	Ideal Std.Dev. = .2886 (1000)				
	.2886 (1010)	.2886 (1475)	.2887 (721)	.2885 (1060)	.2888 (1000)
3. Correlation	Ideal Z-Score = .000000 (999)				
	1.357 (1000)	**20.9995** (10)	-2.44 (1000)	.0623 (1060)	.05202 (1000)
4. Serial	Ideal Chi = 7.34 (500)				
	8.0 (1035)	**786.7** (10)	**85.85** (1000)	10.13 (1100)	8.61 (1000)
5. Run	Ideal Chi = 3.36 (500)				
	2.25 (400)	**376.46** (10)	**43.66** (450)	2.67 (770)	5.21 (500)
6. Gap	Ideal Chi = 4.35 (500)				
	5.72 (550)	**795.51** (10)	**30.22** (800)	6.31 (940)	1.85 (500)
7. Permutation	Ideal Chi = 4.35 (333)				
	1.60 (1000)	**262.32** (10)	**111.2** (1010)	1.06 (1000)	4.27 (200)
8. Mean Square Successive Difference		Ideal Z-Score = 0.000000 (999)			
	.745 (1450)	**20.97** (10)	-2.47 (1000)	.0294 (1000)	.0566 (1000)
9. Poker	Ideal Chi = 2.366 (250)				
	.478 (505)	**226.2** (10)	**27.45** (200)	1.025 (757)	9.97 (1000)
10. Coupon Collector	Ideal Average = 29.29 (341)				
	29.32 (1010)	30.11 (1020)	29.17 (600)	29.36 (938)	29.29 (1000)
11. Pascal's Triangle	Ideal Chi = 7.34 (98)				
	9.68 (110)	**1070.9** (10)	**38.18** (100)	7.91 (148)	3.94 (100)
12. Maximum of 'T'	Ideal Chi = 2.366 (333)				
	3.735 (415)	**62.233** (10)	**40.63** (1000)	2.812 (1000)	**130.9** (333)
13. Random Walk, Average	Ideal Average = 472 (2119)				
	474.8 (2270)	**320** (7500)	**524** (2360)	468.7 (4140)	473.8 (2119)
Standard Deviation	Ideal Std.Dev. = 260 (2119)				
	338.3 (2270)	221 (7500)	362 (2360)	326.9 (4140)	327.5 (2119)
14. Looping	Ideal Looping = Never (1,000,000)				
	1,100,000 OK	2,900,000 OK	**91304=68000**	2,400,000 OK	4,000,000 OK

Results of using the the RNG Test program with various BASICs and compiled BASICs.

The values in parenthesis are the number of batches needed to get a test of one million random numbers, and the number of batches actually run for each BASIC or compiled BASIC.

The bold-faced values are unsatisfactory; the RNG failed that test.
The underlined values are questionable; the RNG gave unlikely results.

For comparison, I also ran these RNG tests on the program compiled with Microsoft QuickBASIC 4.00 under MS-DOS. In general, its tests were good, though it did fail the Maximum of 'T' test and give questionable results on the Poker test.

Other BASIC versions:

The same sequence of random numbers was given on the Model I using TRSDOS 2.3, NEWDOS/80 V1 & V2, DOSPLUS 3.4 & 3.5, LDOS 5.0.0, MultiDOS 2.10 and UltraDOS 4.2 and also on the Model III using TRSDOS 1.3 and NEWDOS/80 V2. Apparently all of these DOS use the ROM-based built-in random number generator. Its 'seed' is in memory locations 16554, 16555, and 16556, and each byte is initially set to 255 on a cold boot.

As a 'Reset' does not reset these bytes to 255, in line 90 of the program '255' is POKEd into these locations to start all tests at the cold boot condition. To start any program that uses the RND function at the same place each time, just POKE the same set of numbers from 0 to 255 into each of these locations. The RANDOM function merely puts an arbitrary number from 0 to 127 into memory location 16555 only, to reset the series. It therefore can give only 128 different random number sequences. Since these BASICs use a three-byte seed, in theory their RND function could give over sixteen million random numbers before repeating (looping).

The Model 4 gave the same sequence of random numbers using either TRSDOS 6.2.0 or LS-DOS 6.3.0; other DOS would probably do the same.

Unlike the Model I/III, the Model 4 restarts the same sequence again after a 'Reset,' as well as after a cold boot.

For comparison, under MS-DOS the same sequence of random numbers was given by the Microsoft QuickBASIC 4.00 compiler used to get the results in Table 3, by QuickBASIC 4.50 and by the Microsoft BASIC Compiler PDS, version 7.00. GWBASIC and all of the other MS-DOS BASIC compilers tried each gave a different random number sequence, except for the Microsoft BASIC Compiler version 5.35 and QuickBASIC 1.01, which duplicated each other's sequence.

Since all of the TRS-80 BASICs studied, and most of the MS-DOS BASICs, were from Microsoft, the variety of results obtained is quite surprising. I would have expected them to find a good random number generator and to then stick with it. But no, after getting a pretty good RNG for the Model I/III BASIC, they had a couple of very poor ones for their Model I BASIC compilers, then a good RNG for the Model 4 BASIC. Then for MS-DOS they have a variety of RNGs, with even their latest compilers not having as good an RNG as their TRS-80 Model 4 BASIC version. Odd, to say the least.

I have heard that more computer time has been spent in testing RNGs than in actually using them. After evaluating the TRS-80's RND function, I can believe it! Some individual tests required an overnight run, and even the fastest took several hours in interpreted BASIC to test one million random numbers. At least we can now have greater confidence in the randomness of the TRS-80's BASIC RND function.

We can also avoid the Microsoft Model I compilers for any serious simulations using the RND function.

Other Systems:

The program listing is correct for running on a Model I or Model III, or for compiling on the Microsoft BASIC compilers. A few minor changes are needed for it to run on a Model 4: Delete the POKEs and OUTs in program lines 70-90. The POKEs set the Model I/III to their 'cold-start' sequence, and the OUTs set for higher clock speed if a speed-up board is installed on the Model I/III, or for running at Model 4 speed in Model III mode on the Model 4. Also for the Model 4, or for MS-DOS machines, delete the semicolons at the end of program lines: 320, 770, and 2140. These are needed only for the 64-character video screens of the Model I/III.

In addition, the RND syntax is different on the Model I/III/4 than on the MS-DOS systems:

	Model I/III	**Model 4/MS-DOS**
Random Number 0-1	RND(0)	RND
Random Integer 1-10	RND(10)	INT(RND*10) + 1

Note that 'Random Numbers' are decimal numbers. The above changes must be made throughout the entire program to convert the program listing for Model 4 or MS-DOS use.

If Your RND Fails the Test:

If the RND function in your BASIC (or compiled BASIC) is not good enough for your requirements, must you give up? No, you can still simulate the RND function with a few BASIC statements. The best form of RNG is called a 'mixed congruential' RNG. To implement this form of RNG, you simply select a seed number 'R' (which may be zero), multiply it by a selected number, add a different selected number, then save only the decimal portion of the resulting number for both the random number and for the seed number for the next random number. That sounds rather complicated, but it's really quite simple: Just use something like P = 21 and Q = 0.3271, and the formula:

R(n + 1) = FRAC(R(n)*P + Q)

A subroutine such as:

```
9000 R=R*P+Q :R=R-INT(R) :RETURN
```

might be used after setting the values of P, Q & R in the main program. Note that the values selected for P and Q are critical, and the greater the precision of calculation, the better. You might try P = 9821, Q = 0.211327 and R = 0; I tried these in the subroutine above in a compiled BASIC version of the RNG test program, and after one million numbers, it passed all but the Maximum of 'T' test. Of course, I used double precision for P, Q, and R to give enough significant digits in the subroutine calculations. Experiment, or check the references. I can't guarantee that these values will work for you, as each BASIC does its calculations a little differently. Good luck!

Conclusions:

The subject of random numbers may seem simple, but it actually gets heavily into statistics. By the use of these RNG tests, you may check whether your BASIC's RND function is good enough for you to base serious programs on it, or whether it can be used only for selecting a starting point for a game. Even if your RND fails, you may be able to replace it with a subroutine.

References:

Heyman, Victor K. — *65 Notes Vol. 4 No. 8, pp. 1 - 6*

Knuth, Donald E. — *The Art of Computer Programming Vol. 2, Seminumerical Algorithms, pp. 1 - 160*

Note: 65 Notes is now called The PPC Journal
It is the publication of The PPC Club, 2541 W. Camden Place, Santa Ana, Calif. 92704

RNG/BAS
Model I/III
Model 4 & GW-Basic (with changes)

```
10 CLS
20 PRINT"- - - Random Number Generator (RNG) Tests - - -"
30 '          (c)  by Delmer D. Hinrichs  1990
40 '                 2116 S. E. 377th Ave.
50 '                 Washougal, Wash. 98671
60  ' Written in TRS-80 Level II BASIC for Model I & III
70 OUT 254,1  '  Speed up for Model I with clock control board
80 POKE 16912,104 :OUT 236,104  ' Speed up Mod 4 in III mode
90 POKE 16554,255 :POKE 16555,255 :POKE 16556,255 'Set RND
Seed
100 DEFINT I-N :S=.25 :H=.5 :T=.75
110 DIM C(10), J(10), P(10)
120 PRINT"Select the test that you want from the following list"
130 PRINT"    1.  Distribution Test"
140 PRINT"    2.  Mean and Standard Deviation"
150 PRINT"    3.  Correlation Test"
160 PRINT"    4.  Serial Test"
170 PRINT"    5.  Run Test"
180 PRINT"    6.  Gap Test"
190 PRINT"    7.  Permutation Test"
200 PRINT"    8.  Mean Square Successive Difference Test"
210 PRINT"    9.  Poker Test"
220 PRINT"   10.  Coupon Collector's Test"
230 PRINT"   11.  Pascal's Triangle Test"
240 PRINT"   12.  Maximum of 'T' Test"
250 PRINT"   13.  Rectilinear Random Walk Test"
260 PRINT"   14.  Loop Test";
270 PRINT TAB(45); :INPUT"Which Test No."; A$ :A=VAL(A$) :B=0
280 ON A GOTO 310,470,600,760,930,1130,1320,1570,1730,1990,
     2130,2360,2570,2720
290 PRINT"Your Entry '";A$;"' is Illegal.  Try Again";:GOTO 270
300 '
310 CLS:PRINT"(1) *  *  *  *  Distribution Test for RNG's  *  *  *
*"
320 PRINT"Bin--> 1   2    3   4    5   6    7   8    9   10   Chi
     Cum.Chi";
330 PRINT"Ideal 100 100  100 100  100 100  100 100  100 100  8.34
     8.34"
340 FOR I=1 TO 1000
350  IR=RND(10) :J(IR)=J(IR)+1
360  NEXT I
370 X=0 :C=0 :B=B+1
380 FOR I=1 TO 10
390  X=X + (J(I)-100)*(J(I)-100)/100
400  C(I)=C(I)+J(I)
410  C=C + (C(I)-100*B)*(C(I)-100*B)/(100*B)
420  NEXT I
430 PRINT USING"#### ### ###  ### ###  ### ###  ###
###  ### ### ###.## ###.##"; B; J(1); J(2); J(3); J(4); J(5);
J(6); J(7); J(8); J(9); J(10); X; C
440 FOR I=1 TO 10 :J(I)=0 :NEXT I
450 GOTO 340
460 '
470 CLS:PRINT"(2)   *  *  Mean and Standard Deviation of RNG's
*  *  *"
480 PRINT, " Mean    Std.D    Cum.Mean  Cum.SD"
490 PRINT"Ideal         0.5000  0.2887     0.5000  0.2887"
500 R1=0 :R2=0
510 FOR I=1 TO 1000
520  R=RND(0) :R1=R1+R :R2=R2+R*R
530  NEXT I
540 B=B+1 :T=1000*B :C=C+R1 :D=D+R2
550 SD=SQR((R2-R1*R1/1000)/999)
560 CD=SQR((D-C*C/T)/(T-1))
570 PRINT USING"####          #.####   #.####     #.####
#.####"; B, R1/1000, SD, C/T, CD
580 GOTO 500
590 '
600 CLS:PRINT"(3)  *  *  *  *  Correlation Test for RNG's  *  *  *
*"
610 PRINT, " Cor.Coef", " Z Score", "Cum. Z Score"
620 PRINT"Ideal", " 0.000000", " 0.000000", " 0.000000"
630 X1=0 :X2=0 :Y1=0 :Y2=0 :XY=0 :X=RND(0)
640 FOR I=1 TO 1000
650  Y=X :X=RND(0)
660  X1=X1+X :X2=X2+X*X :Y1=Y1+Y :Y2=Y2+Y*Y
:XY=XY+X*Y
670  NEXT I
680 B=B+1 :C1=C1+X1 :C2=C2+X2 :D1=D1+Y1 :D2=D2+Y2
:CD=CD+XY
690 C=(1000*XY-X1*Y1)/SQR((1000*X2-X1*X1)*(1000*Y2-Y1*Y1))
700 Z=(C+1/999)/((1/999)*SQR(997000/1001))
710 W=1000*B :CC=(W*CD-C1*D1)/SQR((W*C2-C1*C1)*(W*D2-
D1*D1))
720 CZ=(CC+1/(W-1))/((1/(W-1))*SQR(W*(W-3)/(W+1)))
730 PRINT USING"####          ##.######      ##.######
##.######"; B, C, Z, CZ
740 GOTO 630
750 '
760 CLS:PRINT"(4)  *  *  *  *  Serial Test for RNG's  *  *  *  *  *"
770 PRINT"Pair-> 1   2   3   4   5   6   7   8   9   Chi  Cum.Chi";
780 PRINT"Ideal 111  111  111  111  111  111  111  111  111   7.34
     7.34"
790 FOR I=1 TO 1000
800  K=3*INT(3*RND(0)) + INT(3*RND(0))
810  J(K)=J(K)+1
820  NEXT I
830 B=B+1 :X=0 :C=0
840 FOR I=0 TO 8
850  X=X + (J(I)-111)*(J(I)-111)/111
860  C(I)=C(I)+J(I)
870  C=C + (C(I)-111*B)*(C(I)-111*B)/(111*B)
880  NEXT I
890 PRINT USING"#### ###  ###  ###  ###  ###  ###
###  ###  ### ###.## ###.##"; B, J(0), J(1), J(2), J(3), J(4),
J(5), J(6), J(7), J(8), X, C
900 FOR I=0 TO 8 :J(I)=0 :NEXT I
```

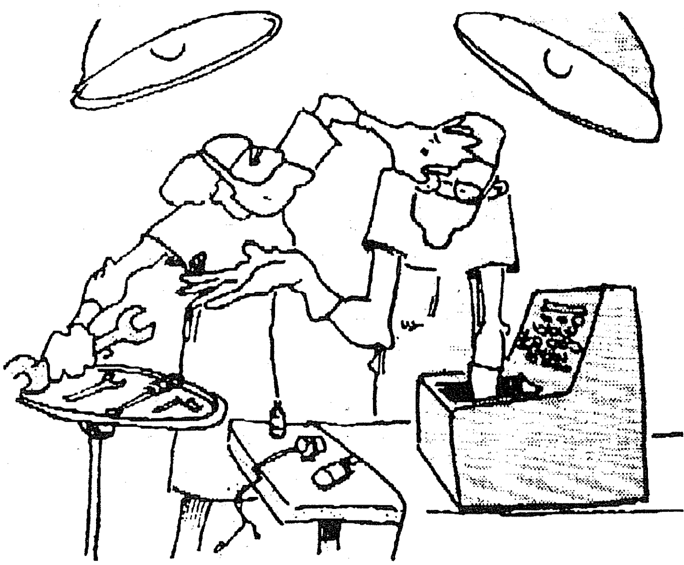
```
910 GOTO 790
920 '
930 CLS:PRINT"(5)  *  *  *  *  *  Run Test for RNG's  *  *  *  *
*"
940 PRINT"      Count      0   1   2  3   4+     Chi   Cum.Chi"
950 PRINT"Ideal   2718      500  333  125  33  8     3.36   3.36"
960 P(0)=500 :P(1)=333.3 :P(2)=125 :P(3)=33.33 :P(4)=8.333
970 FOR I=1 TO 1000
980   D=RND(0) :K=0
990   R=RND(0) :IF R>D THEN D=R :K=K+1 :GOTO 990
1000  IF K>4 THEN K=4
1010  J(K)=J(K)+1 :M=M+K+1
1020  NEXT I
1030 B=B+1 :X=0 :C=0
1040 FOR I=0 TO 4
1050  X=X + (J(I)-P(I))*(J(I)-P(I))/P(I)
1060  C(I)=C(I)+J(I)
1070  C=C + (C(I)-P(I)*B)*(C(I)-P(I)*B)/(P(I)*B)
1080  NEXT I
1090 PRINT USING"####   ####      ### ### ### ## ##
##.##   ##.##"; B, M+1000, J(0), J(1), J(2), J(3), J(4), X, C
1100 FOR I=0 TO 4 :J(I)=0 :NEXT I
1110 M=0 :GOTO 970
1120 '
1130 CLS:PRINT"(6)  *  *  *  *  *  Gap Test for RNG's  *  *  *  *
*"
1140 PRINT"      Count     0   1   2  3  4  5+     Chi   Cum.Chi"
1150 PRINT"Ideal   2000      500  250  125  63  31  31     4.35   4.35"
1160 P(0)=500 :FOR I=1 TO 4 :P(I)=P(I-1)/2 :NEXT I :P(5)=P(4)
1170 FOR I=1 TO 1000
1180  K=K+1 :IF RND(0)>H THEN L=L+1 :GOTO 1180
1190  IF L>5 THEN L=5
1200  J(L)=J(L)+1 :L=0
1210  NEXT I
1220 B=B+1 :X=0 :C=0
1230 FOR I=0 TO 5
1240  X=X + (J(I)-P(I))*(J(I)-P(I))/P(I)
1250  C(I)=C(I)+J(I)
1260  C=C + (C(I)-P(I)*B)*(C(I)-P(I)*B)/(P(I)*B)
1270  NEXT I
1280 PRINT USING"####   ####      ### ### ### ### ###
###     ##.##   ##.##"; B, K, J(0), J(1), J(2), J(3), J(4), J(5), X, C
1290 FOR I=0 TO 5 :J(I)=0 :NEXT I
1300 K=0 :GOTO 1170
1310 '
1320 CLS:PRINT"(7)  *  *  *  *  Permutation Test for RNG's  *  *
*  *"
1330 PRINT"      Bin-->    1   2   3   4   5   6    Chi    Cum.Chi"
1340 PRINT"Ideal          167  167  167  167  167  167    4.35   4.35"
1350 P=166.67
1360 FOR I=1 TO 1000
1370  D=RND(0) :E=RND(0) :F=RND(0)
1380  IF D>E GOTO 1410
1390  IF F>D GOTO 1440
1400  K=2 :GOTO 1450
1410  IF F>D THEN K=3 :GOTO 1450
1420  IF E>F THEN K=0 ELSE K=1
1430  GOTO 1450
1440  IF E>F THEN K=4 :ELSE K=5
1450  J(K)=J(K)+1
1460  NEXT I
1470 B=B+1 :X=0 :C=0
1480 FOR I=0 TO 5
1490  X=X + (J(I)-P)*(J(I)-P)/P
1500  C(I)=C(I)+J(I)
1510  C=C + (C(I)-B*P)*(C(I)-B*P)/(B*P)
1520  NEXT I
1530 PRINT USING"####          ### ### ### ### ###
###  ##.##   ##.##"; B, J(0), J(1), J(2), J(3), J(4), J(5), X, C
1540 FOR I=0 TO 5 :J(I)=0 :NEXT I
1550 GOTO 1360
1570 CLS:PRINT"(8)  *  *  Mean Square Successive Difference
RNG Test  *  *"
1580 PRINT, "  Eta", " Z Score", "Cum. Z Score"
1590 PRINT"Ideal", " 2.00000", " 0.000000", " 0.000000"
1600 X=RND(0) :D2=0 :X1=0 :X2=0
1610 FOR I=1 TO 1000
1620  Y=X :X=RND(0)
1630  D2=D2+(Y-X)*(Y-X) :X1=X1+X :X2=X2+X*X
1640  NEXT I
1650 B=B+1:W=1000*B:CD=CD+D2:C1=C1+X1:C2=C2+X2
1660 ETA=D2/(X2-X1*X1/1000)
1670 Z=(1-ETA/2)/SQR(998/999999)
1680 CE=CD/(C2-C1*C1/W)
1690 CZ=(1-CE/2)/SQR((W-2)/(W*W-1))
1700 PRINT USING"####          ##.######      ##.######
##.######"; B, ETA, Z, CZ
1710 GOTO 1600
1730 CLS:PRINT"(9)  *  *  *  *  Poker Test for RNG's  *  *  *  *
*"
1740 PRINT"   Matches-->   0   1   2   3   4         Chi   Cum.Chi"
1750 PRINT"Ideal", " 504  432  27  36  1", " 2.366  2.366"
1760 P(0)=504 :P(1)=432 :P(2)=27 :P(3)=37
1770 FOR I=1 TO 1000 :K=0
1780  J1=RND(10) :J2=RND(10) :J3=RND(10) :J4=RND(10)
1790  IF J1=J2 THEN K=1
1800  IF J1=J3 THEN K=K+1
1810  IF J1=J4 THEN K=K+1
1820  IF J2=J3 THEN K=K+1
1830  IF J2=J4 THEN K=K+1
1840  IF J3=J4 THEN K=K+1
1850  IF K>4 THEN K=4
1860  J(K)=J(K)+1
1870  NEXT I
1880 B=B+1
1890 J3=J(3) :J4=J(4) :J(3)=J(3)+J(4) :J(4)=0 :X=0 :C=0
1900 FOR I=0 TO 3
1910  X=X + (J(I)-P(I))*(J(I)-P(I))/P(I)
1920  C(I)=C(I)+J(I)
1930  C=C + (C(I)-P(I)*B)*(C(I)-P(I)*B)/(P(I)*B)
1940  NEXT I
1950 PRINT USING"####             ### ### ### ### ##
##.### ##.###"; B, J(0), J(1), J(2), J3, J4, X, C
1960 FOR I=0 TO 4 :J(I)=0 :NEXT I
1970 GOTO 1770
1990 CLS:PRINT"(10)  *  *  *  Coupon Collector's Test for RNG's
*  *  *"
2000 PRINT, "Average", "Cum.Avg"
2010 PRINT"Ideal", 29.29, 29.29
2020 FOR I=1 TO 100
2030  FOR L=1 TO 10
2040   K=RND(10) :M=M+1 :IF J(K)=1 GOTO 2040
2050   J(K)=1
2060   NEXT L
2070  FOR K=1 TO 10 :J(K)=0 :NEXT K
2080  NEXT I
2090 B=B+1 :C=C+M
2100 PRINT USING"####          ###.##        ###.##"; B,
M/100, C/(100*B)
2110 M=0 :GOTO 2020
2120 '
2130 CLS:PRINT"(11)  *  *  Pascal's Triangle Test for RNG's  *  *
*  *"
2140 PRINT"Bin-> 1  2  3  4  5  6  7  8  9  10  11  Chi
Cum.Chi";
2150 PRINT"Ideal 1 10  45  120  210  252  210  120  45  10  1   7.34
    7.34"
2160 P(1)=11 :P(2)=45 :P(3)=120 :P(4)=210 :P(5)=252
2170 P(6)=210 :P(7)=120 :P(8)=45 :P(9)=11
2180 FOR I=1 TO 1024 :K=10
2190  FOR L=1 TO 10
2200   IF RND(0)>H THEN K=K+1 ELSE K=K-1
2210   NEXT L :J(K/2)=J(K/2)+1
2220  NEXT I
2230 B=B+1
2240 J0=J(0) :J1=J(1) :J9=J(9) :JT=J(10)
```

```
2250 J(1)=J(1)+J(0) :J(9)=J(9)+J(10)
2260 X=0 :C=0
2270 FOR I=1 TO 9
2280  X=X + (J(I)-P(I))*(J(I)-P(I))/P(I)
2290  C(I)=C(I)+J(I)
2300  C=C + (C(I)-P(I)*B)*(C(I)-P(I)*B)/(P(I)*B)
2310  NEXT I
2320 PRINT USING"#### ## ##  ## ###  ###  ###  ###
### ##  ## ##  ##.## ##.##"; B, J0, J1, J(2), J(3), J(4), J(5),
J(6), J(7), J(8), J9, JT, X, C
2330 FOR I=0 TO 10 :J(I)=0 :NEXT I
2340 GOTO 2180
2360 CLS:PRINT"(12)  *  *  *  Maximum of 'T' Test for RNG's  *  *
*  *"
2370 PRINT"   Max.->    None  1   2   3          Chi  Cum.Chi"
2380 PRINT"Ideal", " 145  285  285  285", " 2.366   2.366"
2390 P(0)=145 :P(1)=285 :P(2)=285 :P(3)=285
2400 FOR I=1 TO 1000
2410  J=RND(10) :K=RND(10) :L=RND(10)
2420  IF J>K THEN IF J>L THEN J(1)=J(1)+1
2430  IF K>J THEN IF K>L THEN J(2)=J(2)+1
2440  IF L>J THEN IF L>K THEN J(3)=J(3)+1
2450  NEXT I
2460 J(0)=1000-J(1)-J(2)-J(3)
2470 B=B+1 :X=0 :C=0
2480 FOR I=0 TO 3
2490  X=X + (J(I)-P(I))*(J(I)-P(I))/P(I)
2500  C(I)=C(I)+J(I)
2510  C=C + (C(I)-P(I)*B)*(C(I)-P(I)*B)/(P(I)*B)
2520  NEXT I
2530 PRINT USING"####         ### ### ### ###
##.### ##.###"; B, J(0), J(1), J(2), J(3), X, C
2540 FOR I=0 TO 3 :J(I)=0 :NEXT I
2550 GOTO 2400
2560 '
```

```
2570 CLS:PRINT"(13)  *  *  Rectilinear Random Walk RNG Test  *
*  *  *"
2580 PRINT, "X and Y", "Steps  Avg.", " Std.Dev"
2590 PRINT"Ideal", "Values", 472; " 472.00", " 260.00"
2600 D=0 :J=0 :K=0
2610 R=RND(0) :D=D+1 :IF R>H GOTO 2650
2620 IF R>S THEN J=J+1 ELSE J=J-1
2630 IF ABS(J)<20 GOTO 2610
2640 GOTO 2670
2650 IF R>T THEN K=K+1 ELSE K=K-1
2660 IF ABS(K)<20 GOTO 2610
2670 B=B+1 :C=C+D :C2=C2+D*D
2680 IF B>1 THEN SD=SQR((C2-C*C/B)/(B-1))
2690 PRINT USING"####        ### ###      #### ####.##
###.##"; B, J, K, D, C/B, SD
2700 GOTO 2600
2720 CLS:PRINT"(14)  *  *  *  *  *  RNG Looping Test  *  *  *  *
*"
2730 DIM A(100) :R=RND(0) :A(0)=R
2740 PRINT :INPUT"Test 'RND' at Intervals of"; W$ :W=VAL(W$)
2750 IF W>0 AND W=INT(W) GOTO 2770
2760 PRINT"Your Entry '";W$;"' is Illegal.  Try Again":GOTO 2740
2770 PRINT :PRINT"Number of RND's Tested = "
2780 FOR I=1 TO 100 :J=I-1 :P=W*J :PRINT P;
2790  FOR U=1 TO W
2800  R=RND(0)
2810  FOR K=0 TO J
2820   IF R=A(K)THEN PRINT:
PRINT"RND #";P+U;"Duplicates";K*W
2830   NEXT K
2840  NEXT U
2850  A(I)=R
2860  NEXT I
2870 PRINT W*100 :PRINT :PRINT"Looping Test Complete"
2880 END
```

TIPS FROM JACK

Hardware by Jack Eich



Those who know me also know that I'm an electron chaser. I am not so sure I'd know what to do with a computer if I had one running. Instead I choose to play with the guts of the machine, touching the keyboard only long enough to run diagnostics, or to try out some new hardware thing. From time to time I enjoy contributing information that may help you extend the life of your computer, and this time I'll present a few tips that, though common, are usually completely overlooked.

Heat is the absolute number one killer of computers - this can be prevented by using a fan.

Lay a small "muffin" fan (or other small fan) over the vents on top, towards the back, and make sure it is positioned so it sucks the hot air out. All too often I've seen people placing the fan backwards, in essence making it blow the hot air back inside. If you are having problems, this is certainly the easy way to find out if heat is the culprit.

A better solution, of course, is to mount the fan inside the computer so it is activated with the computer's ON/OFF switch.

When mounting a fan inside a Model 3 or 4, I always put it about 3 inches from where the line (110 V input cord) enters the case in the left rear, an inch or two from the left side. Point the fan towards the back of the disk drives so that it also carries away the heat from the power supply(s). Tie it in place with three plastic ties looped through the case vents and the mounting holes in the fan structure. Wire it in with twisted pair wire, running the wire over to and under the keyboard, over to the output (front) side of the power switch. Keep the wiring as far from the other wires in the computer as possible. Make sure you use twisted wire, because even Radio Shack knows that a twisted pair helps cancel out the AC electrostatic field around an AC power carrying wire - you'll notice that they use twisted pair for the AC wiring in the computer.

I've seen a small ten dollar fan at Sears that will do the job nicely. You don't need a gale, just enough fan to make the air move. If the fan is too noisy, it can be made to run slower and quieter with an AC "dimmer" switch. Some dimmers will do the job, however most of the ones available won't. You need one that will allow the half cycles (positive and negative) of the AC to reach maximum, and then cut off all or part of the second half of both the positive and negative halves of the power cycle. You can recognize such "dimmers" by the presence of a tiny clear glass diode - about the size of a 1N19 diode. Adjust the dimmer switch so the fan is bit noisy with the computer's case off, since the case, when mounted, will reduce the noise.

If you can find a Sprite muffin fan, number SU2C1, it's quiet, and about the right size (3 1/8 inches square by 1 5/8 inches thick). The Sears fan I mentioned is called "My Personal Fan", and is model EP1548. Sears lists it as part number 42B8106 and sells it for $9.88. In some areas, it is listed as number 42M8106. I've used the SU2C1 often, but the Sears fan is also quiet and a bargain at the price. The "dimmer" I've used can be ordered from R&D Electronic Supply, 100 E. Orangethorpe Avenue, Anaheim, Ca 92801 (phone 714 773-0856) and sells for $4.95. It has no provisions for mounting, and will have to be attached with double-sided foam tape.

The next step to take to keep your computer running is to pay a little attention to the power cord - specifically where and what you plug it into.

You'll notice that Radio Shack provides a three wire cable with a three prong plug. Please don't insult your computer by using a three prong to two prong adapter.

I've refused to fix, or help fix, computers when people bring them to me with that kind of an adapter, and they refuse to change it. Their excuse usually is, "It ran over a year like that..." My standard reply is: "OK, but why do you think you're bringing it to me now?"

Wall outlets in kitchens and bathrooms usually have three pronged sockets, as do outlets for electric heaters, stoves, and furnace fans. This is usually the minimum per the local electrical codes by law. If you look closely at a 110 volt wall socket for a three prong plug, this is what you'll see: to the right is an opening for one blade of the plug; to the left is a similar opening, but wider in the up/down direction. The wider prong opening is "ground". By "ground" I'm referring to the fact that the wire to that pin is grounded at the switch box, or as one of the three wires that bring power to your building from a "pole pig" (a pole pig is the transformer that supplies power to you). Usually there is a pole pig for every 1 to 5 homes in your area. The highest wires on the power pole make up the distribution line to your area. They are the most dangerous, and so are mounted the highest. They carry perhaps 5000 volts (the higher the voltage, the smaller the wires can be for same amount of power). The 5000 volt line goes to the pole pig, and, yes, it too is center tapped to the primary and secondary windings. The secondary is 220 to 240 volts (center tapped) and goes to your switch box. Thus the 5000 and 220 volt center taps go to your outlets, to the side that takes the wide prong only.

Three prong outlets have a third hole for a round pin. You need that round pin because it goes directly to ground, usually to a water pipe, and is distinctly separated from the wire that goes to the center tap of the 5000 volt distribution line. Further, that center tap is connected through step-down transformers via 50,000 - 200,000 volt lines all the way back to the power source that supplies your area (along with being connected to any lightning strikes to the power lines!). Am I getting through? Reminds me of the fellow who had an expensive line filter - he told me that he paid $100 for it. He got really mad at me when I pointed out that the output socket had only openings for two prongs, and that I thought it wasn't heavy enough to filter the voltage from a lightning strike. Maybe it would make a good boat anchor.

Now, what can you do if you don't have a three prong outlet? Don't just switch the two prong outlet for a three prong one and leave the ground pin unconnected (floating). Your computer's switching power supply bypasses noise generated by the switching to that third prong and so to the ground, BUT ONLY IF IS GROUNDED!

To install a ground wire, use an outlet on the outside wall. Turn off the power supply to that outlet. Get some heavy, insulated 14 or 16 gauge wire. Remove the two prong outlet and drill a hole (using a long drill bit at a downward angle so that water can't get in through the hole) through the outside wall. Feed the wire through the wall to out of doors. Get yourself a three pronged socket (usually a double socket). Hook up the old power lines to the new socket, being sure that the wire that went to the wide blade of the old socket goes to the wide blade of the new socket. Once those two original wires are installed attach the new ground wire to where it will connect with the ground pin (possibly the metal frame of the outlet), take up the slack, and reinstall the new socket on the wall. Outside, dig a small ditch along the wall to the nearest water pipe or faucet. Clean the metal pipe, strip enough wire to wrap it around the pipe, and clamp it on, preferably using a stainless steel hose clamp from an auto supply store. Smear epoxy or paint around the clamp and wire to keep it from corroding. Presto! Turn on the power, plug in the computer, and the job is finished.

Well, it is now 2 AM, so I'll say goodnight. See you next time.

THE SWAP MEET

FOR SALE: PD GOOD GAMES FOR MODEL I/III.
GAMEDISK#1: amazin/bas (maze), blazer/cmd (arcade), breakout/cmd (arcade), centipede/cmd (arcade), elect/bas (a simulation of the 1980 election), madhouse/bas (adv), othello/cmd (board), poker/bas (almost better than going to las vegas - well, cheaper!!), solitr/bas (great solitaire card game). towers/cmd (puzzle game).
GAMEDISK#2: crams2/cmd (chase), falien/cmd (arcade), frankadv/bas (adv), iceworld/bas (adv), minigolf/bas (putt-putt on the trs-80), pingpong/cmd (1 or 2 player arcade game), reactor/bas (simulation), solitr2/bas (another good solitaire card game), stars/cmd (2 player race game), trak/cmd (maze).
GAMEDISK#3: ashka/cmd (d&d), asteroid/cmd (arcade), crazy8/bas (card game), french/cmd (space invaders in french), hexapawn (board), hobbit/bas (adv), memalpha (adv), pyramid/bas (good solitaire card game), rescue/bas (arcade), swarm/cmd (arcade).
Price per disk: $5.00 (U.S.)
or get all 3 disks for $12.00 (U.S.)
TRSTimes - PD GAMES.
5721 Topanga Canyon Blvd,. Suite 4.
Woodland Hills, CA. 91367

WANTED: I'm looking for a disk version of the INSTANT SORT/SEARCH DATA BASE program, written by Gordon Hatton. I've been using the tape version for years. Please contact me if you have a copy.
Harrie Stellberg, P.O Box 844, Bellville, TX 77418
(409) 865-5618

FOR SALE: TRS-80 SOFTWARE, Models 1/3/4/4P/4D. Many useful programs. Economical prices.
Send $3 for listing. Practical Programs, 1104 Aspen Drive. Toms River, NJ. 08753 (201) 349-6070

WANTED: I am looking for back issues of the Radio Shack Computer Catalog to complete my collection. I specifically need the following issues: RSC-1, 3, 13, 18E, 19E, 20E, 21E, and Software Buyers Guide 1st Edition. Any assistance would be greatly appreciated.
Roy Beck. 2153 Cedarhurst Dr.
Los Angeles, CA. 90027

SWAP: Willing to swap Mod III/4 software (and buy if may be) MMMS Forth79, Sota Fig Forth, Howe Diagnostic, Formation, etc. for APL, Pascal, Home Accountant, Videotex Plus, Profile Plus for Model III, and T.K.Solver, Adv. Stat Statistics, Cobol and Cobol Query for Model 4.
R. Yves Breton. C.P. 95, Stn. Place D'Armes
Montreal, Quebec, Canada H2Y 3E9

FOR SALE

New, unused TRS-80 Model 4D
with Deskmate. 2DD. 64K.
$225.00 + $15.00 UPS

Also many new and used Radio Shack items.
Send $1.00 for list.
Werner L. Jordan
10 N. Morley St.
Baltimore, MD 21229

FOR SALE: Printer Buffer, Centronics Port compatible (IBM PC + others), 64 KBytes (25 pages), Reset/Bypass/ Copy buttons, 8 LED indicators (status + memory fullness), 5x7x2 inch metal case, 2 pounds, AC/DC with Power supply, builtin Selfcheck, 1 year guarantee, includes shipping - $119.
Call/write: Practical Programs,
1104 Aspen Drive.
Toms River, NJ. 08753 (201) 349-6070

WANTED: The MARK 3 utility collection for LDOS. Also looking for DOS PLUS hard disk drivers.
Roy Beck. 2153 Cedarhurst Dr. Los Angeles, CA 90027

HELP-HELP: Laura Coan, 20 Cotswold Way, Yorkville, IL 60560, has glitched her only copy of Radio Shack's ASTROLOGY program (cassette version - level 2). If anyone has an unused working copy - please send it to her. *Ed.*

NEW: Starting a support group for Model II, 12.....6000; also a BBS (9600 bps: Xenix/Unix oriented). Bi-monthly 1st issue due out in September. FREE. Send your comments, articles, etc. to: R. Yves Breton, P.O. Box 95, STN Place D'Armes. Montreal, Quebec. Canada H2Y 3E9. Fax: 514 768-5905

TRSTimes on DISK #5
Issue #5 of TRSTimes on DISK is now available, featuring the programs from the January, March and May 1990 issues.

U.S. & Canada: $5.00 (U.S.)
Other countries:$7.00 (U.S.)

Send check or money order to:
TRSTimes on DISK
5721 Topanga Canyon Blvd, Suite 4
Woodland Hills, CA. 91367. U.S.A.

TRSTimes on DISK #1, 2, 3 & 4
are still available at the above prices