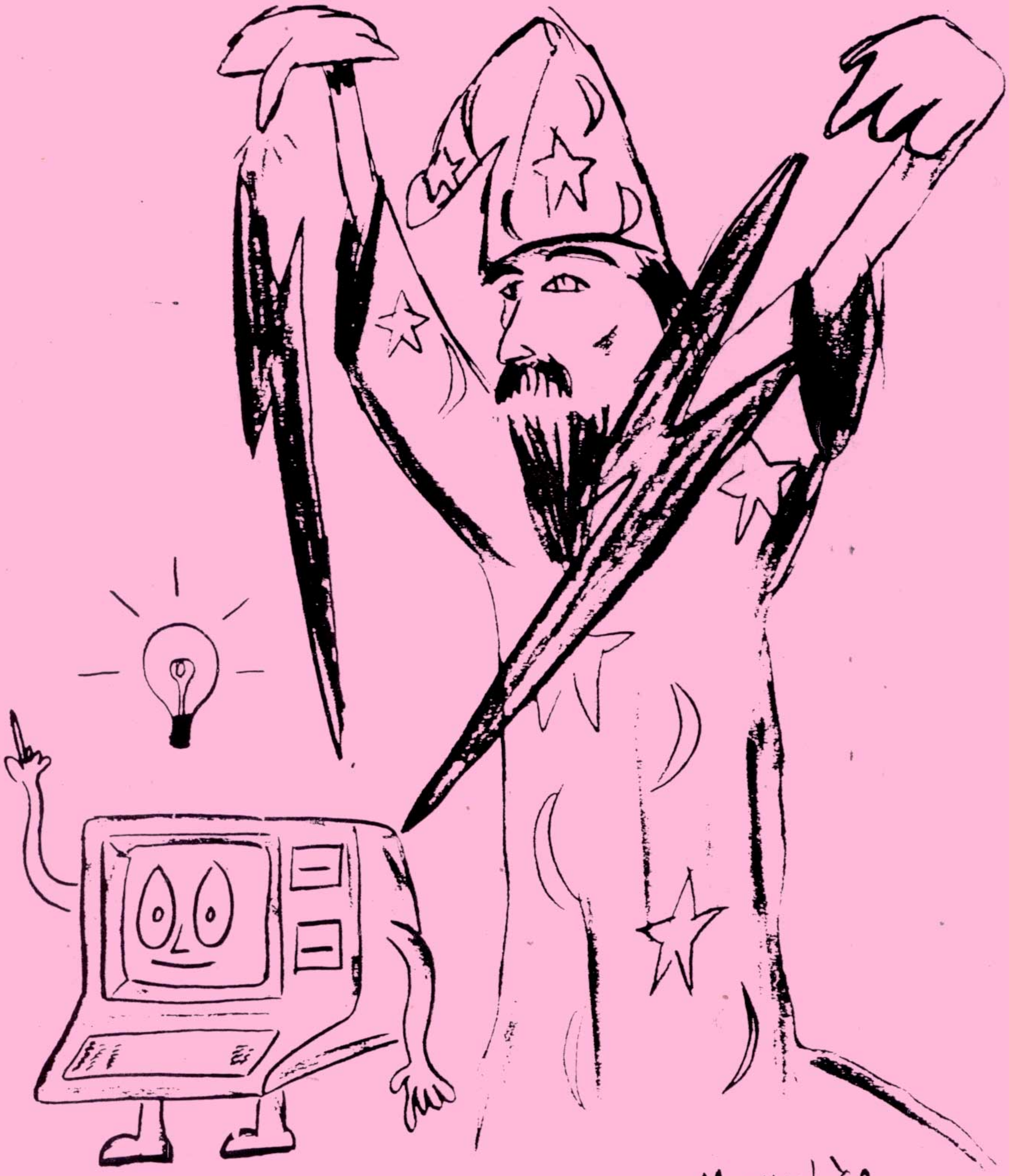
TRSTIMES

Volume 2. No. 4. - Jul/Aug 1989 - $4.00



LITTLE ORPHAN EIGHTY

In the CLOSE#3 editorial in our last issue, I brought up the subject of article submissions. Let me quote a portion of it verbatim:

"Though I haven't discussed this with Stan (Slater, publisher of CN80), I imagine that he shares my views on the following subject:

Both CN80 and TRSTimes continue to exist because you, the readers are kind enought to send your articles to us for publication. We do appreciate it, believe me. The problem occurs when an author sends the same article to both magazines. We may both like the material and, not knowing the other also has it, publish it simultaneously. This just is not fair. It is not fair to CN80; it is not fair to TRSTimes and, most importantly, it is not fair to the readers. Many subscribe to both magazines, and they deserve to get new and fresh information from both magazines each and every issue. So please, if you submit an article to CN80, do not send it to TRSTimes; if you send it to us, please don't send it to CN80."

In their June 1989 issue, Computer News 80 replies to my comments. Again, I quote verbatim:

"In the last issue of TRSTimes, Vol 2 No. 3, Lance Wolstrup makes a statement about article submissions to both TRSTimes and CN80, and although he does mention that he did not discuss it with us, he assumes that we would agree with his statement of 'If you submit an article to CN80, do not send it to TRSTimes; if you send it to us, (TRSTimes) do not send it to CN80.'

Well if he had discussed it with us he would have found out that we DO NOT agree. We may at times publish duplicate articles, and the reason we feel that we should is because having a larger circulation we would deprive those who do not subscribe to both magazines from some possibly very good information.

We are sure that those who do subscribe to both do not like to see the duplication, but for those who do not subscribe to both magazines, they too are entitled to the same information. So we will leave it up to you, if you want to send your articles to CN80, feel free to send it to others also. We will make the editorial judgement to publish it or not, based upon the content of the article, hint, tip, program or whatever, and not upon the fact that you decided to send it to another publication.

There was a time not too long ago when a new TRS-80 product would come out, you could read about it in every publication you picked up. That is true today with the MS-DOS magazines, and I also relied upon as many reviews of a product as I could find before I would purchase. The more I could learn about something, from different viewpoints the better. At times I was convinced to buy, and at times a difference of opinions expressed by several reviewers made me decide I better hold off on the purchase.

As you know we depend heavily upon our readers who contribute material to CN80, and our feeling is if YOU feel you would like to see your article, hint, tip, program, review or etc. published in CN80, then you will send it to us, if not then you won't sent it."

First, I certainly wish to apologize to Stan Slater for putting words in his mouth. It was never my intention to dictate editorial policy to CN80, or to anyone else for that matter. HOWEVER, having said that, I feel a overwhelming urge to climb up on the Wolstrup soap box to voice my opinion on some of the points made in CN80's reply:

Having two magazines dedicated to our machines is good. This gives TRS-80 users a luxury not available for some time, a choice. Should CN80 and TRSTimes proceed to publish the same 'word for word' articles, we are taking that away. This does not serve the TRS-80 community well; all it does is fill pages in a magazine.

The 4th paragraph mentions that reviews in the different publications, at times, helped the writer decide whether or not to buy a particular product. That is exactly the point. These reviews were DIFFERENT. They were written by different authors, each using different approaches, thus relating different experiences with the product. Only a complete dullard would read the same verbatim review, written by the same reviewer, in 4 different magazines before he made up his mind whether or not to buy the product. After all, would you go to the same doctor for a second opinion?

To be sure, I have no problem with the same SUBJECTS being covered simultaneously in the magazines. The popular programs, such as Scripsit, LeScript, PFS:file, etc., as well as DOS, hardware, and the pitifully few new things coming out, are all any of us have to write about, so it is only natural that article material will overlap. But, darn it, let's do our best to present it with a different slant.

Before stepping down from the box, let me just say that TRSTimes does need your articles, programs, hints, tips, reviews, etc., and each and every submission is much appreciated. I do, however, reserve the right to tack on that final criteria for getting published in this magazine: If we know it has appeared previously in CN80, no matter how worthy the material might be, we won't publish it. We will do our best to be as fresh and original as possible. That's a promise.

The above comments were made with absolutely no malice, or other negative feelings, towards CN80 or any of our common contributors. I simply felt that it was time for TRSTimes to publicly take a stand that satisfies my personal ethics.

So, off the box - and *welcome to TRSTimes 2.4.*

Lance W.

TRSTimes - Volume 2. No. 4. Jul/Aug 1989

TRSTimes magazine is published bi-monthly by TRSTimes publications.
20311 Sherman Way suite 221. Canoga Park, CA. 91306. U.S.A.

1989 subscription rates (6 issues):
United States & Canada: $18.00 (U.S.)
All other countries: $23.00 (U.S.)

THE MAIL ROOM

VIRUS

With reference to TRS-80 viruses, yes there is/was a virus of sorts, a real NASTY one. This virus was a creature of natural occurrence. It wasn't planned. It existed in early versions of Model III Superscripsit and was known as the Block Freeze Error. Once you noticed the virus and tried to get rid of it, it would spread all over the disk. With time and various patches, some users learned to control and minimize it. If you put this virus on a hard disk it would be a disaster.

If you have an old version of Mod. III SS, try leaving the INSERT function on and then close and open the document a number of times. This may make the virus appear in the document. You might also try freezing a paragraph while doing this. It usually appears in hex or scrambled data. If the virus doesn't appear, try increasing the file size to over 14 pages and doing the above. If you see a problem, try to alter the text. If you get the message "A frozen paragraph cannot be altered", the virus is in the file. You may be able to transfer this file (virus and all) via ASCII to other systems. Every time you try to get rid of the virus, it may be spreading like cancer to places you cannot see it.

Dr. Manley Fox
Manhattan Beach, CA.

Very interesting. Having never used Scripsit in any of its many forms, I am not familiar with this one. Does anyone out there have similar experiences?

Ed.

ITEMS WANTED

There must be lots of hardware, software and reference books out there somewhere, gathering dust in closets and on shelves. How to get buyers and sellers together? Any chance of devoting, say, half a column per issue (or whatever it takes to meet the demand) to a simple "Wanted" and "Available" listing of items, with names and addresses or phone numbers? Or do BBS's adequately fill the demand - how many readers have routine contact with them? I don't - no room for a permanent set-up. I have two III's, a 4 and a Kaypro 2 - all picked up as discards on the cheap. Looking through magazines of the early 80's, I see many things I'd like (or just think I need), mostly from

companies no longer in business.

For instance, I'd like to find: (1) A floppy drive test disk, to exercise the drive for use with the Dysan test disk; (2) The books "BASEX" by Paul K. Warms (Byte Publications, 1979) which has the source listings of an interesting hybrid of assembly-Basic, and also "C Basic User Guide" by Osborne/McGraw Hill; (3) schematics, installation instructions, or contact with someone who knows about a combined 80 col/RS232 board made by Hurricane Labs of San Jose, called Compactor IV; and finally (4) information on light pen software, or an operating pen.

Also, any leads on Spanish language manuals, or other learning aids. Radio Shack, Education Group disclaim any knowledge of any such under RS sponsorship. I'd like to give my III's to a worthy Mexican charity, but without Spanish helps, I hesitate. Do you suppose there are still RS products being used in Texas, New Mexico or Southern California Schools, in Spanish?

Keep up the good work - every issue seems to manage to contain one or more real gems.

Harold May
428 Phillippa
Hinsdale, ILL. 60521
(312) 325-1910

I have no knowledge about Spanish language software, so let's kick it out to the readers. How about it, can anyone help? TRSTimes will certainly set aside space to print letters from readers looking for particular items. We will do this at no charge. However, as TRSTimes does have real down-to-earth expenses, anyone wishing to sell something should take out an ad. We do charge a reasonable price for this. Fair enough? (see ad rates elsewhere in this issue)

Ed.

HELP

I recently bought an old Model III motherboard as a backup and/or parts. It had another board piggybacked on it that I needed information for. It is numbered SP1-3 Rev A and was made by Process Control Technology in Stockton, CA. in 1981-1983. I have been unable to make contact with them. Old ads in 80 Micro indicate it might be a versatile disk controller and/or speedup kit. My biggest problem is that the wire from Pad "A" was disconnected and I don't know where it should go. Does anyone have any documentation, installation info or description of such a board or know where the Pad "A" wire should go?

Any info would be appreciated. Send copies to:
Frank Gottschalk
785 Maya Court
Fremont, CA. 94539
or call collect to: (415) 651-2313

WHAT'S WITH RAPIDOS?

On my primary computer, I am running two internal double-sided 40 track drives, 2 external double-sided 80 track drives, and a 12 meg hard disk. In addition, I have the Grafyx Solution Hi- Resolution graphics board along with the Alpha Tech 6 mhz speed up board and a clock calendar. I had planned to add the Alpha Tech memory board, but because of a problem that has appeared, I am waiting until I find a solution.

The operating system, RAPIDOS, that Micro Labs furnishes much of their High Resolution software on, will NOT boot reliably with either the Alpha Tech speed up or with the XLR8er board from MISOSYS installed. I am told that RAPIDOS will not boot AT ALL with either the Alpha Tech or the XLR8er extended memory boards installed. So far, I haven't been able to disassemble the BOOT/SYS of RAPIDOS. I feel fairly certain that the problem can be overcome with a simple patch to allow extended timing on boot up. Can you help here or can you find someone that can provide the technical data/source code for RAPIDOS? I have tried to order this operating system from the author, but so far haven't been able to locate them. I've talked with Ted Carter at Micro Labs, but he has no solution. Alpha Tech has no answer, either. Tim Sewell at The File Cabinet is trying to find a solution to this problem, also. He told me that Rapid Dynamic Software (authors of RAPIDOS) have apparantly gone out of business.

One positive thing I can report is the use of the Micro-Labs Mouse Interface. I like it! It connects to the 50 pin I/O port, so is about as easy an installation as you can make. It does have one draw back. If you leave it powered up, and you have a hard disk connected, you will not be able to access the hard disk! Simple solution: there is room within the black box of the Mouse Interface to drill a small hole, install a subminature SPST switch, and wire it in series with the positive lead coming from the power supply. Leave the Interface turned off while you boot up and get your Hi-Res drawing program installed. Load whatever file you want to work on, and THEN turn ON the Interface. You can use almost any device that is designed to plug into the COCO Joy Stick port with this interface. I use a Deluxe Joy stick, a Color Mouse, and a Kola Touch Pad with mine. You must turn off the power to the interface before you can save your file, or exit the program. All in all, I am pleased with this new capability of my computer!

BY THE WAY, I also have the Orchestra 90 Interface connected on my I/O port simultaneously with the Mouse Interface and the Hard Disk without any other problems.

In response to Roy T. Beck's article, "Copy Limited Files Under TRSDOS - LSDOS 6.x", TRSTimes 2.1, pp 30f, I have been frustrated with both pfs:file as well as pfs:report for the Model 4. Roy states that his copy of pfs:file came on TRSDOS 6.2.0! I have now personally inspected 5 different copies of pfs:file, breaking the seal on each, and have yet to find one furnished on anything but TRSDOS 6.1.2! The same holds true for every copy of pfs:report that I have checked. I have successfully moved the pfs:file programs to LSDOS 6.3, BUT I have been unsucessful in making pfs:report function on LSDOS 6.3. With the necessary system files SYSRES into memory, the pfs error "Insufficient Memory to load program" greets me! Any solutions to this one? Regarding his method for copying the limited files, his research is interesting, but I just turn off the PASSWORD prompt in my COPY routine with the patch:

```
. Disables password checking by COPY command
in LSDOS 6.3
PATCH  SYS2/SYS.LSIDOS  (D02,33=18:F02,33=28)
```

Some time ago in TRSTimes 1.5, on pages 13 and following, Fred Blechman had an excellent article on buying and selling used computers entitled "Old Computers Never Die!" In that article he gave the caution, "If a computer, printer or monitor has a serial number removed, it might have been stolen." This is good advice; however, I personally purchased 45 used computers from the Tandy Retail Outlet at their wharehouse in Ft. Worth that had obviously NEVER had serial numbers on them! They had been built up in-house in cases that had minor factory defects for use on their own assembly lines. I went through most of these computers with a fine-tooth comb and put them in like-new condition, and then resold them. They don't even have factory labels on them since they were never really intended for retail sale by Tandy. So, just because they don't have serial numbers does NOT mean they were stolen! I have a legal bill of sale for these machines! The responsible buyer should check the source if the subject machine has no serial number. If there is evidence that it had a serial number that has been removed, THEN I would suspect it had been stolen.

Arthur N. McAninch, Jr.
Borger, TX

My version of PFS:file came on TRSDOS 6.1, I believe, and has long since been converted to 6.2. Obviously, Tandy must have shipped it on whatever Dos was current at the time of shipment.

Being intrigued that REPORT would not run under LS-DOS 6.3., I made a backup of the master LS-DOS 6.3. disk. Then I purged ALL visible files, as well as the unnecessary files from the invisible section (Basic, patch, format, backup, etc.) This gave me more than enough room to copy PFS/CMD, FILE/CMD and REPORT/CMD from my working 6.2. disk to the new LS-DOS 6.3. disk.

I booted up with this LS-DOS 6.3. disk and ran REPORT with no problems.

The solution to your dilemma is to avoid SYSRESing system files. They are NOT needed in memory as long as they are on your disk.

Ed.

TRSDOS 1.3. WILDCARD COPYING

I am the newsletter editor for the Mid-Cities TRS-80 Users Group which is nestled between Fort Worth and Dallas, Texas in the city of Arlington. One of my duties is to collect the club's mail. I like that job because I get to TRSTimes first. We just got your May/June issue and I was reading Maurice Superville's letter. On the part where he was asking about wildcard copying for his BASIC files, you answered that TRSDOS 1.3. was not capable of this. I must call your hand. This DOS does have a wildcard-like COPY function.

On pages 34 and 35 of the TRS-80 Model III Disk System Owner's manual, it shows how you can copy all files with a specific 3 character extension from one drive to another. The HELP command also shows how to use this function. Mr. Superville can copy all of his BASIC files from drive :0 to drive :1 by issuing the command: COPY /BAS:0 :1

This is assuming that all of his BASIC files have /BAS as an extension. Only the extension may be used as the wildcard and it must be 3 characters, no less. Any valid 3 character extension can be used.

As for users groups in Houston, I know of only one. It is a CoCo club, but they may know of another group there. The address is: CoCo Loco, 1237 Cedar Post Lane #B4, Houston, TX. 77055.

Rob Yoder
c/o MCTRUG
P.O. Box 170566
Arlington, TX. 76003

I'd like to respond to Maurice Superville's letter in the May 1989 issue of TRSTimes. Actually, TRSDOS 1.3. does already have a limited form of wildcard handling with the COPY and KILL commands. The difference is, it's limited to different files having the same extension only. The format for COPY is **COPY /ext:d1 :d2** where both drive numbers are mandatory (even if you're copying from the master drive). KILL is **KILL /ext:d**, again with a mandatory drive number. Thus, if as Mr. Superville suggested, you wanted to copy BASIC files from drive :0 to drive :2, you would enter the command: **COPY /BAS:0 :2**.

John C. Fowler
Los Alamos, NM.

Thanks to both Rob and John for keeping 'ye faithful editor' on his toes. Though I used TRSDOS 1.3. on a regular basis at one time, I never knew about it's wildcard ability. I just learned something. I love it.

Ed.

ERRORS

I was able to correct an error in the listing for EDITOR/BAS which was interferring with it running properly. Line 50 reads as follows:

50 IFTCKTHEN30ELSERESTORE

By changing the 'C' to a '<' the program ran smoothly, created EDITOR/CMD, and the rest follows per Gary Campbells article.

Also, I am having a problem with ROTATE/BAS. After going over my typing five or six times and being very sure that there are no remaining typos, the alphabetic array refuses to handle attempts to rotate any position but, 1,5, and 9 properly. Any attempt to rotate any of the other positions causes a rotation at one of these three positions, and usually causes a letter to appear at more than one place on the board. Maybe someone else has had this problem and has figured out what to do.

Now, I have a problem which I would like some help on. In the BASIC which came with LS-DOS 6.3. there are some handy additional editing commands, such as the command **C num1,num2**, which allows copying a line at one position in a program to another position; very handy if the line is long and complicated. There is also a Find, a Move, and a Search command available. These are in the file BASIC/OV2 on this disk. I also have BASICG, Radio Shack's high-resolution Basic and on some disks will have both Basic files available. The problem is that there are times when these additional editing commands would come in handy when I am working in high-resolution with BASICG, but they are not available in that file. Is it possible to PATCH BASICG so that these four commands can be available? I would appreciate it if this could be done.

Robert O'Neal
Quincy, IL.

EDITOR/BAS does indeed contain the error you pointed out. The author of the program, Gary Campbell, is completely innocent. His code is correct. Somehow, during the conversion from TRSDOS to PC, the C replaced the < character. My fault, I should have caught it. Sorry about that. ROTATE/BAS works perfectly on disk, as written. Checking the magazine listing against the disk listing, I could not find an error. Re-check the section of code from line 500 to line 620. Make sure that line 550 is J=I and not J=1. Not being into high-rez and, thus, not familiar with BASICG, I can only tell you that saving your working file in ASCII would allow you to load it into LS-DOS BASIC where you can modify it to your hearts content, using the additional editing features. Can anyone else help?

Ed.

USER GROUP

It occurs to me that it would be very much to the advantage of the readers if they could be organized into

a TRSTimes users group. Along with that, the users group could develop a directory of their membership (something such as that used by the TANDY CAPITAL COMPUTER USER'S GROUP) which would portray the member's name, address, telephone number, hardware/software used, and the willingness to help others in their expertise. In such fashion, computing becomes far easier and more pleasant for everyone, and especially the novice.

Hubert L. Johnston
Sanford, NC.

That is not a bad idea. While I don't think it should be named the 'TRSTimes readers users group', I am certainly willing to collect, and publish, a list of readers who are interested in helping others. How about it out there - what do you think - any takers?

Ed.

MORE USER GROUPS

I would like to get in touch with other persons interested in the TRS-80 here in HOUSTON. Maybe we could form a users group.

Maurice Superville
5572 Aspen Street
Bellaire, TX. 77401
(713) 667-1453

See above letter, as well as the one from Rob Yoder.

Ed.

MAGIC MATH PLUS

Just received TRSTimes Vol. 2, No. 3 yesterday, another marvel to behold. Thank you and Dr. Allen Jacobs for reviewing my Magic Math Plus in it (last issue). If I may, I would just like to clarify a few points and respond to some interesting and legitimate issues that the review raised, as well as to make a special offer to fellow TRSTimes readers.

First, a minor point: Our company name is Recreational Mathemagical Software (not "Mathematical"). I'm sure the bank will accept the checks, but I think the distinction reinforces the imagery that Dr. Jacobs raised, namely of mathemagic and fun that use some math, but not straight math per se.

Second, for the record, the authors of the package are two in number, not four: myself (Dr. M. Ecker) and David B. Lewis. Jim Kyle and Edward Roberts are not additional authors per se, but rather contributed material to the package that enabled me to create the menus (Mr. Kyle) and do touch-ups (Mr. Roberts).

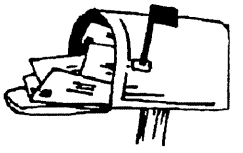
Third, Allen, your reviewer, is correct in that we do indeed offer a much more advanced standalone loan amortization utility than Fastloan. Fastloan 3.0 is the full-featured, current version that even goes beyond the Fastloan II cited. Yes, it does print out amortizations and a whole lot more. I'll send a free blurb to anybody who sends us a self-addressed, stamped envelope and specifies his/her request(s) for specific information (Fastloan 3.0, Magic Math Plus, etc.). If readers and you would like, I can also forward a copy to you for review by Dr. Jacobs (who indicated he would like to see it).

Fourth, Allen mentioned that he would like to see more such recreations and wondered aloud about more offerings. I think readers should know that I am also the person behind Recreational & Educational Computing, or **REC**, now in its fourth year. (Most issues of TRSTimes have been carrying the ads, but haven't mentioned my name, so readers may not have noticed the connection.) As an important BONUS for TRSTimes readers, we are going to include three issues of **REC** (a $10.50 value) FREE with every Magic Math Plus order for the TRS-80 version ($40 with the shipping) ordered by TRSTimes readers. **REC**'s subscription price is $24 per year of 8 issues for 1989, with same price for back issues for 1988; it's $20 for 1986 or 1987; $72 for all four years. Anybody who wants may buy a three-issue trial sub without purchasing Magic Math Plus for $7 (instead of $10.50). And, any heretic with an Apple II (boo!) or PC (even Sanyo 555!) may have the version of Magic Math Plus for his machine for $30. (The PC version, while smaller, has lots of color graphics added by another collaborator, Mr. Loren Krienke.)

Finally, anybody on a tight budget who wants the fun but can't justify the expense is reminded that I will be continuing my semi-regular "Recreational Computing" column right here in TRSTimes, with my second one this issue. In fact, this column also addresses two of the questions Dr. Jacobs raises, namely the Collatz/Ulam conjecture with the question of proof, and his query about multiplying any number from 1 to 100 by 99 and always getting an answer whose digits sum to 18. Allen is right on the money in associating this with Magic Math Plus, and in fact, some versions of Magic Math Plus employ similar tricks based on the same underlying principle (casting out nines, if anybody is dying to know). It is also precisely the kind of thing we do in **REC**.

In a world where even the slightest mention of math ordinarily sends people into cardiac arrest, thanks again for using TRSTimes, itself a wonderful publication, as a vehicle for spreading the joys of "mathemagic" with the readers.

Michael W. Ecker, Ph.D.
President, Recreational Mathemagical Software
Editor/Publisher, Recreational & Educational Computing (**REC**)
129 Carol Drive
Clarks Summit, PA 18411
(717) 586-2784



MAXIT

a game for Model 4

by Lance Wolstrup

Saturday nights, usually way past midnight, I will turn on my PC-clone and call some of the local IBM bulletin boards in search of material for TRSTimes that is not otherwise found in the TRS-80 world. I will enter the down-loading section, read the file descriptions, and down-load anything that may be of interest.

During such a search I found a game called MAXIT. This program has been around for a while. I believe it was originally written for the PET (you know, the Commodore *before* VIC 20 and C-64), then translated to the Apple and, eventually, to IBM. I ran it, liked it and, since to my knowledge it has never existed on a TRS-80, I decided to translate the code to run on the Model 4. As in real life, certain things do not translate well, so a fair portion of the code has been completely rewritten.

MAXIT is played on an 8x8 board filled with positive and negative numbers. The object of the game is to collect the numbers so that, at the end of the game, you have a higher point total than your opponent. The opponent can be another human, or you can choose to play head on against the computer. Should you select the latter, be warned, the Model 4 is no pushover. It plays a fairly mean game (I beat it the first time we played, but have not managed to win three in a row yet).

The game begins with player number 1 choosing a number from the horizontal row where the two asterisk are displayed. Pressing the SPACEBAR moves the cursor to the next number; pressing ENTER captures the number the cursor is on. Player 2 (human or TRS-80) must now select a number from the vertical column indicated by the two asterisks. Note that player 1 always moves horizontally, while player 2 always moves vertically.

The game ends when all the numbers have been selected, or when a player does not have a move. The winner, in either case, is the player with the highest point total.

```
10 GOTO 1000

40 ON A+1 GOTO 100,110,120

100 ON B+1 GOTO 101,102,103,104,105

101 AV=0:AH=0

102 A$=CHR$(31):GOTO 114

103 A$=CHR$(30):GOTO 114

104 AH=AH+LEN(A$):A$=CHR$(14):GOTO 114

105 A$=CHR$(15):GOTO 114

110 ON B+1 GOTO 114,111,112,113

111 AH=0:GOTO 114

112 AH=INT((A9-LEN(A$))/2):GOTO 114

113 AH=A9-LEN(A$)

114 PRINT@(AV,AH),A$;:RETURN

120 A$=INKEY$:IF A$="" THEN 120 ELSE A0=ASC(A$):
A1=VAL(A$):RETURN

1000 DEFINT A-Z:DIM BD(7,7),VA(64):A9=80

1010 TP$="":FOR X=1 TO 8:TP$=TP$+CHR$(156)+
STRING$(5,140)+CHR$(172):NEXT:M1$="":FOR X=1 TO 8:
M1$=M1$+CHR$(149)+STRING$(5,32)+CHR$(170):NEXT:
M2$="":FOR X=1 TO 8:M2$=M2$+CHR$(181)+
STRING$(5,176)+CHR$(186):NEXT

1020 A=0:B=0:GOSUB 40:B=4:GOSUB 40:A=1:B=1:AV=0:
AH=0:A$="TRSTimes Presents:":GOSUB 40:B=2:
A$="M A X I T":GOSUB 40:B=3:A$="Translated by Lance
Wolstrup":GOSUB 40:B=1:AV=1:A$=STRING$(A9,131):
GOSUB 40

1030 AV=3:A$="The object of MAXIT is to get as many points
as possible. Two players can play":GOSUB 40:AV=4:
A$="against each other, or one against the computer.":
GOSUB 40

1040 AV=6:A$="You get points by moving the cursor to a
space with a number in it. The first":GOSUB 40:AV=7:
A$="player always moves horizontally, and the second player
moves vertically.":GOSUB 40

1050 AV=8:A$="You indicate the place you want to move to
by pressing the <SPACEBAR>, and then":GOSUB 40:AV=9:
A$="pressing <ENTER> to capture that piece.":GOSUB 40
```

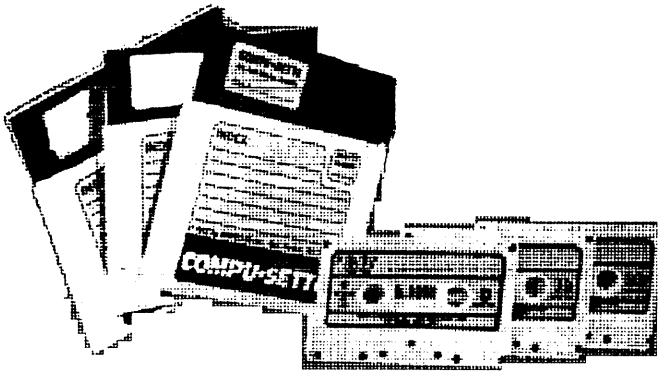
```
1060 AV=11:B=2:A$="Good luck":GOSUB 40

1070 A=1:B=1:AV=15:A$="1 or 2 players: ":GOSUB 40:A=0:
B=3:GOSUB 40

1080 A=2:GOSUB 40:A=1:B=0:GOSUB 40

1090 A=0:B=4:GOSUB 40:NP=A1:IF NP=1 THEN
P2$="TRS-80":GOSUB 1100:GOTO 1120 ELSE IF NP<>2
THEN A=0:B=2:AV=15:AH=15:GOSUB 40:GOTO 1070 ELSE
GOSUB 1100:GOSUB 1110:GOTO 1120

1100 A=1:B=1:AV=17:AH=0:A$="What is your name #1: ":
GOSUB 40:A=0:B=3:GOSUB 40:LINE INPUT P1$:
P1$=LEFT$(P1$,8):B=4:GOSUB 40:RETURN

1110 A=1:B=1:AV=18:AH=0:A$="What is your name #2: ":
GOSUB 40:A=0:B=3:GOSUB 40:LINE INPUT P2$:
P2$=LEFT$(P2$,8):B=4:GOSUB 40:RETURN

1120 A=0:B=1:AV=2:AH=0:GOSUB 40:RANDOM

1130 A=1:B=0:AV=3:AH=12:A$=TP$:GOSUB 40

1140 AH=12:FOR X=4 TO 18 STEP 2:AV=X:A$=M1$:
GOSUB 40:AV=X+1:A$=M2$:GOSUB 40:NEXT

1150 FOR X=1 TO 64:VA(X)=X:NEXT

1160 FOR X=64 TO 1 STEP -1:READ PC:P1=RND(X):
J=VA(P1)-1

1170 IF P1<X THEN FOR I=P1 TO X-1:VA(I)=VA(I+1):NEXT

1180 I=INT(J/8):J=J-8*I:BD(I,J)=PC:GOSUB 1280:NEXT:
RESTORE:A=0:B=3:GOSUB 40

1190 S1=0:S2=0:GOSUB 1680

1200 PL=1:GOSUB 1330:IF FL=0 THEN 1220

1210 PL=2:GOSUB 1330:IF FL THEN 1200

1220 A=0:B=2:AV=22:AH=0:GOSUB 40:A=1:B=2:
ON SGN(S2-S1)+2 GOSUB 1250,1260,1270

1230 A=0:B=2:AV=23:AH=0:GOSUB 40:A=1:B=2:
A$="Do you want to play again (Y/N) ":GOSUB 40:
A=0:B=3:GOSUB 40

1240 A=2:GOSUB 40:IF A0=89 OR A0=121 THEN 1120 ELSE
A=0:B=0:AV=0:AH=0:GOSUB 40:END

1250 A$=P1$+" won by "+STR$(S1-S2)+" points":
GOSUB 40:RETURN

1260 A$="It's a tie !!":GOSUB 40:RETURN

1270 A$=P2$+" won by "+STR$(S2-S1)+" points":
GOSUB 40:RETURN

1280 PC=BD(I,J):A=1:B=0:AV=I*2+4:AH=J*7+14

1290 IF PC=100 THEN A$="**":GOSUB 40:C1=I:
C2=J:GOTO 1320

1300 IF PC=-100 THEN A$=" ":GOSUB 40:GOTO 1320

1310 A$=RIGHT$(" "+STR$(PC),2):GOSUB 40

1320 RETURN

1330 IF PL=2 THEN 1370

1340 FL=600:FOR J=0 TO 7:FL=FL+BD(C1,J):NEXT

1350 IF FL=0 THEN RETURN

1360 NM$=P1$:DX=1:DY=0:GOSUB 1430:RETURN

1370 FL=600:FOR I=0 TO 7:FL=FL+BD(I,C2):NEXT

1380 IF FL=0 THEN RETURN

1390 NM$=P2$:DX=0:DY=1:GOSUB 1430:RETURN

1400 FL=600:FOR I=0 TO 7:FL=FL+BD(I,C2):NEXT

1410 IF FL=0 THEN RETURN

1420 NM$=P2$:DX=0:DY=1:GOSUB 1430:RETURN

1430 Y=C1:X=C2:FX=1

1440 IF NP=2 OR PL=1 THEN 1470

1450 A=0:B=2:AV=23:AH=0:GOSUB 40:A=1:B=2:
A$=NM$+", your turn. ":GOSUB 40:IF P2$="TRS-80"
THEN 1460 ELSE A=0:B=S3:GOSUB 40

1460 GOSUB 1690:GOTO 1610

1470 ON FX GOTO 1480,1490

1480 A=0:B=2:AV=23:AH=0:GOSUB 40:A=1:B=2:
A$=NM$+", your turn. ":GOSUB 40:A=0:B=3:GOSUB 40

1490 A=2:GOSUB 40

1500 IF A0<>32 THEN 1590

1510 '

1520 OX=X:OY=Y

1530 Y=Y+DY:IF Y>7 THEN Y=0

1540 X=X+DX:IF X>7 THEN X=0

1550 PT=BD(Y,X):IF ABS(PT)=100 THEN 1530

1560 MD=1:I=OY:J=OX:GOSUB 1280

1570 MD=2:I=Y:J=X:GOSUB 1280

1580 GOTO 1490

1590 IF A0<>13 THEN 1490

1600 IF ABS(BD(Y,X))=100 THEN 1490
```

```
1610 IF NP=1 AND PL=2 THEN I=Y:J=X:GOSUB 1280

1620  I=C1:J=C2:BD(I,J)=-100:GOSUB  1280

1630  I=Y:J=X:PT=BD(I,J):BD(I,J)=100:GOSUB  1280

1640 IF PL=1 THEN S1=S1+PT

1650 IF PL=2 THEN S2=S2+PT

1660 A=0:B=4:GOSUB  40:A=0:B=2:AV=22:GOSUB  40:
A=1:B=2:A$="Last  taken:"+STR$(PT):GOSUB  40

1670  GOSUB 1680:RETURN

1680  A=0:B=4:GOSUB  40:A=0:B=2:AV=21:AH=0:
GOSUB 40:A=1:B=2:A$=P1$+"'s score ="+STR$(S1)+
"  "+P2$+"'s score ="+STR$(S2):GOSUB 40:RETURN

1690 MT=-100:GG=-1:FOR LP1=0 TO 7:PC=BD(LP1,C2):
IF ABS(PC)=100 THEN 1830

1700 MX=-100:FOR LP2=0 TO 7

1710 IF LP2<>C2 THEN PK=BD(LP1,LP2):IF PK<>-100 AND
PK>MX THEN MX=PK

1720 NEXT LP2

1730 IF MX<>-100 THEN 1750

1740 IF PC>MT THEN MT=PC:GG=LP1:GOTO 1830

1750 IF GG<0 THEN GG=LP1

1760 FOR LP2=0 TO 7:PQ=BD(LP2,SV):IF PQ=-100 OR
LP2=LP1 THEN 1820

1770 MY=-100:FOR LP3=0 TO 7:PW=BD(LP2,LP3):
IF LP3=SV THEN 1790

1780 IF ABS(PW)<>100 AND PW>MY THEN MY=PW

1790 NEXT LP3

1800 IF MY=-100 THEN MY=0

1810 DT=PC-MX+PQ-MY:IF DT>MT THEN MT=DT:GG=LP1

1820 NEXT LP2

1830  NEXT LP1:Y=GG:RETURN

1840   DATA   15,10,9,9,8,8,7,7,7,6,6,6,5,5,5,5,4,4,4,4,3,3,3,3,3

1850 DATA  2,2,2,2,2,2,1,1,1,1,1,0,0,0,0,0,0,-1,-1,-1,-1,-1

1860   DATA  -2,-2,-2,-2,-3,-3,-3,-4,-4,-4,-5,-5,-6,-6,-7,-9,100
```

Model III

TRANSFER YOUR TAPES TO DISK

by Robert R. Keegan



Though MACHRELO/BAS was written some years back, I have noted that people still occasionally have a need to convert an old cassette tape program (in machine language) to reside on disk for easy access. In fact I found that I wanted to do this a few months back, and I vaguely remembered the program. It is still working and fairly user friendly, not cordial, just friendly.

I will skip documenting the internal workings of the program because, 1) I've forgotten how I managed to make it work and 2) Writing "documentation" is something I do for a living. However, it essentially automates a machine language procedure that was published in TRS-80 Microcomputer News, Vol. 4, Issue 3, March, 1982, Pages 11-13.

Also in that article is a table of byte addresses for Radio Shack cassette programs which are needed for the conversion. My program only needs the START, END, and TRAnsfer addresses exactly, and a temporary address which is not critical. (8000H works for temporary address ususally or 8100H should handle the rest assuming you have 64K of memory, doesn't everybody now). I am talking about the Mod III now, Mod I owners are on their own.

There is a possible error in the table for the start of SERIES I EDITOR ASSEMBLER which I have as 4646H instead of 4AEE. Also for GAMMON GAMBLER--START = 4300H, END = 7FFFH, TRA = 4600H, and temporary address should be 8100H. You can get the necessary addresses from a monitor utility program with tape capabilities; I use TASMON. If a reader can't handle this part maybe a friend with a monitor can get the addresses he or she needs.

Cassette machine language programs overlap DOS working memory; hence this conversion lets the program load into higher memory by adding a routine that siezes control and relocates the program before transferring control to the program in its proper environment.

Now if someone has a conversion problem, this program and the proper address data for the subject program should make the job fairly easy.

MACHRELO/BAS

```
0 CLS:
PRINT STRING$(3,32);"MACHINE LANGUAGE PRGRM
RELOCATER FOR CONVERTING":
PRINT STRING$(17,32);"TAPE PRGRMS TO DISK":
MS! = 55554:
POKE16562,MS!/256:
POKE16561,MS!-INT(MS!/256)*256:
CLEAR1000:
DEFINTA-Z

1 DEFFNSI%(A1!) = -((A1! > 32767)*(A1!-65536))-((A1! < 32768)
*A1!)

2 DEFFNIS!(A1%) = -((A1%)*(65536 + A1%) + ((A1% > = 0)*A1%))

3 DEFFNIA%(A1%,A2%) = (65536-(A1% + A2%))*((A1% + A2%)
> 32767) + ((0-A1% + A2%)*-((A1% + A2%) < -32768)) + (A1% +
A2%)*-(((A1% + A2%)) < 32768)AND((A1% + A2%) > -32769))

4 DEFFNRE#(A1#,A2#) = A1#-INT(A1#/A2#)*A2#

24 DEFFNH2$(A1%) = MID$("0123456789ABCDEF",
INT(A1%/16)      + 1,1) + MID$("0123456789ABCDEF",A1%-
INT(A1%/16)*16 + 1,1)

25 DEFFNH4$(A1%) = FNH2$(ASC(MID$(MKI$(A1%),2)))
+ FNH2$(ASC(MKI$(A1%)))

26 DEFFNDH!(A$) = INSTR("123456789ABCDEF",
MID$(A$,1,1))*4096 + INSTR("123456789ABCDEF",
MID$(A$,2,1))*256 + INSTR("123456789ABCDEF",MID$(A$,3,1))
*16 + INSTR("123456789ABCDEF",MID$(A$,4,1))

32 DEFFNRR$(A1%,A2%,A3$) = MID$(A3$,(A1%-1)*A2%
+ 1,A2% )

33 DEFFNRC%(A1$,A2$,A3%) = (INSTR(A1$,LEFT$(A2$ +
STRING$(A3%,"  "),A3%))-1)/A3% + 1

50 PRINT:
PRINT" The addresses below may be 4-digit hex or 5-digit
dec."

100 INPUT "  ENTER PROGRAM START ";S$:
GOSUB 1000:
```

```
IF N1%<66 THEN GOSUB 1100:
GOTO100

110 S1%=N1%:
S2%=N2%:
S!=N!

120 INPUT "  ENTER PROGRAM END ";S$:
GOSUB 1000:
IF N1%<S1% THEN GOSUB 1100:
GOTO120

130 E1%=N1%:
E2%=N2%:
E!=N!

140 INPUT " ENTER PROGRAM ENTRY ADDRESS (TRA) ";S$:
GOSUB 1000:
IF N1% <1 THEN GOSUB 1100:
GOTO140

150 T1%=N1%:
T2%=N2%:
T!=N!:
S$="8000"

160 INPUT "  ENTER TEMPORARY START ADDR
(DEFAULT=8000)  ";S$:
GOSUB 1000:
 IF N!<E+31 THEN GOSUB 1100:
GOTO160

170 M1%=N1%:
M2%=N2%:
M!=N!

180 L!=E!-S!+1:
S$=STR$(L!):
GOSUB 1060

190 L1%=N1%:
L2%=N2%:
L!=N!

200   X1%=221:X2%=213:X!=56789
'     X=MOVEUP START ADDRESS

210 MX!=M!+L!+15
'     MX=MOVE BACK START ADDRESS

220 X%=FNSI%(X!):
MX%=FNSI%(MX!)
'      CHANGE TO INTEGER FOR POKE ADDRESS

300 C=X%:
POKEC,33:
POKEC+1,S2:
POKEC+2,S1:
POKEC+3,17:
POKEC+4,M2:
POKEC+5,M1:
POKEC+6,1:
POKEC+7,L2:
POKEC+8,L1

310 POKEC+9,237:
POKEC+10,176:
POKEC+11,195:
POKEC+12,0:
POKEC+13,0

350 C=MX%:
 POKEC,33:
POKEC+1,M2:
POKEC+2,M1:
POKEC+3,17:
POKEC+4,S2:
POKEC+5,S1:
POKEC+6,1:
POKEC+7,L2:
POKEC+8,L1

360 POKEC+9,237:
POKEC+10,176:
POKEC+11,195:
POKEC+12,T2 :
POKEC+13,T1

370 PRINT"  <ENTER> TO PRINT DUMP ADDRESS
INSTRUCTIONS  NEEDED  LATER,":
INPUT"          PRINTER ON?";E$

380 LPRINT"  After entering Level II you will load source tape,
but do":
LPRINT"not execute with / . Instead enter  /"X!"

385 LPRINT"  TRSDOS will reboot. Create disk file as follows:":
LPRINT"DUMP   filename/CMD   (START=0";FNH4$(FNSI%
(M!));  ",  END=0"FNH4$(FNSI%(M!+L!+32))",  TRA=0";
FNH4$(FNSI%(MX!))")"

400 PRINT:
PRINT"    EXIT TO LEVEL II WITH <RESET BREAK>"

500 END

1000 IF LEN(S$)<4 OR LEN(S$)>5 THEN N1%=0:
RETURN

1010 IF LEN(S$)=5 THEN1060

1020   N2$=RIGHT$(S$,2):
N1$=LEFT$(RIGHT$(S$,4),2)

1030 N1%=FN DH!("00"+N1$):
N2%=FN DH!("00"+N2$)

1040   N!=256*N1%+N2%

1050 RETURN

1060 N!=VAL(S$):
N1%=INT(N!/256):
N2%=N!-256*N1%

1070 RETURN

1100 PRINT"Sorry, illegal entry, try again.":RETURN
```

HINTS & TIPS!

BRANDNEW PATCH FOR TRSDOS 1.3.

by Gary Campbell

Here's a *great* TRSDOS 1.3 patch never before published! Be sure to use a backup disk when installing these patches. *(Compliments of GRL Software, the distributor of SYSTEM 1.5, the 1.3 Dos Upgrade.)*

The ORIGINAL COPY command copies files sector by sector, unless same drive copying is selected. (Same drive copying reads the source file from disk into memory, then dumps the file. It is MUCH faster than the usual sector by sector read/write method) When same drive copying is used, COPY incorrectly uses $4415h, (the top of memory pointer) instead of $4411h, (the bottom of protected memory pointer), to calculate the size of the read buffer. This causes crashes when large files are copied to the same disk. Another bad habit of copy is that is does not check free space before copying new files. Each time 3 sectors are written, it extends the file. If all space on the disk gets used an error is reported. This can waste alot of time when you are copying large files.

Multiple drive copying is also a pain when you copy a large file with a small LRL. Copy will read/write one record at a time. If the LRL=1, a file with 65000 records takes 65000 loops, instead of 65000/256 loops. Add to this the processing time it takes to display "Copying Record 00000" each time a byte is copied, the entire process wastes too much time.

After these patches are in place, before copying any records, the destination file is extended to the source file size. If an out of disk space error occurs, the error will be reported, and you will be returned to DOS. Otherwise, the source file will be read into memory, 256 bytes at a time, to the $4411h high memory pointer. The LRL is ignored. The file will then be dumped from memory to the destination disk, in continuous 256 byte sector writes. The file will not require being extended. Once the file has been copied, the source files LRL and EOF bytes are copied to the destination files DCB. Note that the "Copying Record 00000" routine is now skipped, also reducing the amount of processing required to copy a file.

Try some timing tests BEFORE and after the patches, with some small files, large files, and files with small LRL's. You will be quite suprised! The patches also work on System 1.5 disks. Future releases of System 1.5 will not require these patches.

```
PATCH  *09  (ADD=59A8,FIND=32E054,CHG=000000)

PATCH  *09  (ADD=5358,FIND=15,CHG=11)

PATCH  *09  (ADD=54EA,FIND=C22352,CHG=C3955F)

PATCH  *09  (ADD=540E,FIND=3AA6549047C5,
CHG=2A6561C3DB54)

PATCH  *09  (ADD=54D0,FIND=CDF35221FFFF,
CHG=3EC9322A53CD)

PATCH  *09  (ADD=54D6,FIND=22BA52CD9B,
CHG=F352C35753)

PATCH  *09  (ADD=54DB,FIND=59C2235206,
CHG=110063B700)

PATCH  *09  (ADD=54E1,FIND=119461210064,
CHG=ED527C47B7CA)

PATCH  *09  (ADD=54E7,FIND=CD2044C22352,
CHG=3854C5C31454)

PATCH  *09  (ADD=5F95,FIND=43616E277420,
CHG=C22352ED4BD7)

PATCH  *09  (ADD=5F9B,FIND=416374697661,
CHG=56CD4244CD39)

PATCH  *09  (ADD=5FA1,FIND=746520447561,
CHG=44CAAD5F0000)

PATCH  *09  (ADD=5FA7,FIND=6C207768696C,
CHG=000000C30944)

PATCH  *09  (ADD=5FAD,FIND=6520524F5554,
CHG=CD3F443AA453)

PATCH  *09  (ADD=5FB3,FIND=4520697320,
CHG=47AFC3B153)
```

A HELPFUL HINT:

by Allen Jacobs

Smart manufacturers should always try to use "off the shelf" components in the products they make, if possible. Often, they don't. In some cases, however, they do, (if you happen to know what shelf these components are from). A smart consumer should look for these opportunities to make his life simpler.

If the band to advance the uptake reel on your daisy wheel ribbon cartridge breaks as mine did, go to a plumbing or hardware shop and ask for the replacement "O"

rings used in kitchen faucets on their swinging spigots. Many of these "O" rings come in a variety of sizes common to both spigots and ribbon cassettes.

The investment in an "O" ring runs in the neighborhood of 20 cents but it will save you from having to buy another multistrike or cloth ribbon casette just for its "O" ring. If you make the mistake of purchasing an otherwise unneeded cassette, from then on you are always an "O" ring behind. Additionally, the ink in the now open new cartridge begins to dry out prematurely. This is especially true if you were to purchase a cloth ribbon cartridge. This problem occasionally happens if you either buy a lot of ribbon casettes at once or you do not use one of your cassettes often (such as the multistrike cassette used only for "special" letters or the "final" draft).

The way to determine the right size "O" ring you are looking for is actually simple if you think about it. If you have the ring from another casette, just match it as closely as you can. If you do not have another "O" ring, take your cassette to the store and try out a few different sizes. If you are unable to take your cassette with you, just measure the distance between the outer surfaces of the pulleys. Don't worry about including the widths of the pulleys unless they are large in comparison to the distance between them. This unaccounted-for length will provide a good allowance for the required tension on the ring when it is in place. Just stretch a couple of "O" rings over a ruler until you find one that is not quite as long as your estimated length.

Having successfully replaced this band will give you a tremenduous sense of accomplishment. It may even encourage you to complete that programming project you have had on the back burner for from one month to ten years. It's true! Trust me on this one.

PATCH FOR SYSTEM 1.5.

by Gary Campbell

Some Z80B speed up kits will not boot up a 1.5 Dos, as it must use the original disk drive step rate during boot up. Other users with older drives also may have boot troubles. This patch corrects all problems associated with the faster 1.5 step rate.

```
PATCH UPGRADE/ NOW (ADD =578D,FIND = 18,CHG =1C)
```

Another note to XLR8R users. MISOSYS published a patch to alter some instructions that load the IX and IY registers separately. The *07 MISOSYS patches are not required on System 1.5. Future releases of 1.5 will not require any XLR8R patches.

WRITE LESCRIPT FORMATTED FILES WITH MODEL 102

by Marc Luyens

I own both a Model 4 and a Model 102 and, consequently, transfer a lot of files between them. My wordprocessor in Mod 4 and III mode is LESCRIPT, while I use TEXT in the Model 102. Most of my letters are written on the Model 102 and, when convenient, the files are then ported over to the Model 4.

In the past, I had to enter all the control codes and formatting commands while in LESCRIPT. This was extra work, so I figured that there must be a better way.

Writing a LESCRIPT file, saving it in ASCII, and transferring it to the 102, I found that the imbedded control codes show distinct characters. I checked the characters against the Model 102 ASCII table and found the following:

LESCRIPT code	displayed code	keyboard code
<clear><;>		<graph></>
<clear><enter>		<graph><'>
<clear><B>		<graph><x>

These are just a small example. Most, if not all, of the other commands and control codes are also available. You can now write your LESCRIPT formatted text with Model 102.

Editors note: This also works with Model 100.

The World's Smallest Word-Processor made smaller

by Howard W. Mueller

Edgar Martin's "World's Smallest Word-Processor" (Hints & Tips - Mar/Apr 1989, page 23) is a "monster" to enter from the keyboard. At a minimum his requires 38 exhausting keystrokes. I have one that is several bytes shorter. It works on the Model III (the only computer I have) but, as it is standard Microsoft Basic, it will work on all TRS-80 machines with just minor changes. I'm putting my word-processor in the Public Domain. Sorry, I cannot provide any support other than the following documentation:

DOCUMENTATION:

Name: The World's Smaller Smallest Word-Processor
Variables: None
Keystrokes: 20 (Models I & III)
19 (Model 4)
Features: Same as "The World's Smallest Word-Processor

Model I & III version: 1LPRINTINKEY$;:GOTO1

Model 4 version: 1LPRINT INKEY$;:RUN

If you save the file under the name: "WORDPROC/BAS" on drive:1, password protect it with .SMALLWP and run it with the command: RUN"WORDPROC/BAS.SMALLWP:1" it takes fewer keystrokes to type it in each time than to run it. (Of course some idiot will probably want to destroy that advantage with a filename such as "W" and choosing not to password protect this gem at all.) It doesn't really matter. Eventually someone will come along and write a shorter word-processor.

Some very interesting differences between Model I/III Basic and Model 4 Basic are shown in the above listings.

First, Model I/III does not require spaces (most of the time), so no spaces are included, not even between the line number and LPRINT. We do, however, have a real problem if we attempt to use RUN instead of GOTO1. Model I/III forgets the semicolon after INKEY$ and issues a carriage return and linefeed after each keystroke.

Model 4, on the other hand, does require spaces between keywords. We can get away with not putting a space between the line number and LPRINT, but the other spaces must be there. Amazingly enough, RUN respects the semicolon after INKEY$, thus suppressing unwanted carriage returns and linefeeds.

Another interesting item is that the ? (shorthand for PRINT) cannot be used with LPRINT; that is, L? will not work. Model I/III will list the program line correctly as LPRINT but, when the program is RUN, it will produce a "SYNTAX ERROR". Model 4, in its infinite wisdom, simply inserts a space between L and PRINT, also creating a "SYNTAX ERROR" when the program line is RUN.

Not to be outdone, TRSTimes presents the "World's Smallest Assembly Language Word-Processor", using just 8 bytes of pure machine power:

Model I/III version (EDTASM)

```
00100         ORG     7000H
00110 START   CALL    49H
00120         CALL    3BH
00130         JR      START
00140         END     START
```

The Model 4 version can be written, using only 9 bytes.

Model 4 version (EDTASM - patched)

```
00100         ORG     3000H
00110 START   LD      A,1
00120         RST     40
00130         LD      C,A
00140         LD      A,6
00150         RST     40
00160         JR      START
00170         END     START
```

We can cut this listing to match the 8 bytes of the Model I/III if we 'cheat' and use direct CALLs instead of Roy Soltoff's SuperVisor Calls method.

```
00100         ORG     3000H
00110 START   CALL    628H
00120         CALL    63DH
00130         JR      START
00140         END     START
```

Keep in mind that returning to DOS from any of the above Assembly language listings is accomplished by pressing <RESET>. Anyone wanting a clean return to DOS should purchase the 16 byte luxury version, available at $109.95 plus $5.00 S&H.

Ed.

ANOTHER LESCRIPT GOODIE

by Dennis Burkholz

I've found what I believe to be an undocumented feature in LeScript 1.81. While I like the spelling checker, there are times I know I won't be using it. With that in mind, I checked the manual to find a way to avoid loading it. I could not find any mention of this, so I started to fiddle with various combinations of keystrokes. After many unsuccessful tries, I eventually struck gold. The ! is used as a parameter to the program filename to indicate that the spelling checker should not be loaded.

LESCRIPT ! <ENTER>
Runs LESCRIPT without the spelling checker. The program comes up, as usual, on the graphic, telling you to press a key to continue.

LESCRIPT ! filename
Runs LESCRIPT without the spelling checker, loads the indicated file, and comes up on the edit screen.

Hope this is of use to the readers.

Recreational & Educational Computing

Semi-Regular Column by Dr. Michael W. Ecker

Hello, fellow TRSTimes readers and TRS-80 users! This is the second of an irregular sequence of articles, programs, tutorials, and challenges to appear in this publication. For the first installment, see the March/April 1989 TRSTimes.

This issue's feature:

The Conversational Computer

Remember the movie "War Games"? In it, the protaganist does lots of keyboarding. This ultimately gets him in trouble with the military establishment when he telecommunicates with a company fronting for the Pentagon. Of course, he innocently believes it is an ordinary software company, a company with a BBS offering free games, including the infamous Global Thermonuclear War.

However, the actor - Matthew Broderick, if memory serves - was widely reported as having no typing skills. So, how did he do all that proficient typing on his computer?

The key idea - an implicit assumption - is that we believe that what appears on the screen is precisely what is typed. Naturally, by my telling you this, your little light bulb should be going on right now. You should be able to guess that this tacit assumption is pernicious insofar as it is false. In fact, our poor hero was banging away almost randomly to produce the screens shown.

But, the question remains: *How did he do it?*

Along that line of fun for here, and to answer the question, how about something comparable in which what you type is not necessarily what appears on screen - and vice-versa?

I call this piece The Conversational Computer, an item previously used in my publication, **REC**, or Recreational & Educational Computing (after which this column is named, incidentally). Besides appearing in **REC**, the program has also been offered commercially, but I offer it free for you to type in. I believe you'll enjoy playing the program after entering it - and after I explain it. Though I also have versions for MSDOS (including Tandy and others), I'll show the TRS-80 / TRSDOS version (of course).

The Conversational Computer

This is a fun program for playing a trick on friend and foe alike. With it, you will persuade the victim that your computer is answering his or her questions. You will need to be a reasonably careful typist, however, so count on being methodical.

The idea is this: You tell your friend that your computer will answer almost any question. It doesn't even matter whether the question is really that hard. The important thing is that the form of the reply is what must make sense.

Load Basic for your machine, and then type in the Basic program. Tell the friend to ask a question (verbally). Tell him next that you need to type in an innocuous hint, such as "What big eyes you have!" Surely, nobody will object to that!

In truth, when the time comes to type in this hint, what you really type is whatever you wish the screen to display as the answer later. To clarify, say the question your friend asks is: "Who was the sixteenth president of the United States?" If you know the answer, type in something appopriate (without the quotes): "Easy... it was Abraham Lincoln."

If you don't know the answer, type in something close, but hedging: "I forget, but it wasn't George Washington." The mere fact that the answer is even relevant and makes some sense should be enough of a shock to your friend. If the question is to identify the captain of the Titanic, at least make the answer appear appropriate: "It sure wasn't Captain James T. Kirk!"

Now, understand that your friend and/or observer should be positioned so he or she doesn't see your hands typing, because he will not yet see what you really typed appearing on the screen. Instead, he'll see some goofy statements appearing (the fake hints) as you are typing. Naturally, he'll believe that those words are what you are typing.

So, while you are typing in the actual, correct - or close - answer, the screen will show a fake "hint" (the aforementioned "What big eyes you have!"). He won't know that you are typing in the actual answer, as he doesn't see the answer on the screen. But do be careful to type about 50 characters worth of actual answer, or you could blow your cover.

The program provided has three dummy "hints", but it should not be hard to modify the program in a couple of places, notably the data statements and the check for a counter of 4 (make it one more than the number of data statements, the number of which is currently three). Experiment with the program, and have lots of fun with it:

```
10 CLS: CLEAR 1000: REM TRS-80 -- FOR STRING SPACE
15 PRINT "THE CONVERSATIONAL COMPUTER, by Michael
W. Ecker, Ph.D."
20 PRINT "Copyright 1987-89. All Rights Reserved.": PRINT
21 PRINT "Recreational & Educational Computing Newsletter,"
22 PRINT "and Recreational Mathemagical Software."
30 PRINT "129 Carol Drive": PRINT "Clarks Summit, Pa 18411"
40 PRINT: PRINT "Ask me questions, type in hints, and I will
reply."
50 PRINT: FOR DL=1 TO 2000: NEXT DL: REM Time Delay
60 PRINT "Okay, I, the conversational computer, am ready!"
70 PRINT "Ask me a question verbally, and my friend will give a
hint."
80 PRINT: PRINT "(To stop, use your BREAK key...)"
90 PRINT: PRINT: S$="": CT=CT+1: L=0: REM S$ is sen-
tence, CT is counter
100 IF CT=4 THEN RESTORE: CT=1
110 READ X$: X=LEN(X$)
120 I$=INKEY$: IF I$="" THEN 120
130 S$=S$+I$: L=L+1: PRINT MID$(X$,L,1);
140 IF I$<>CHR$(13) THEN 120: REM Check to see whether
key is <ENTER>
150 PRINT: PRINT "Let me concentrate on that ..."
160 FOR DL=1 TO 1500: NEXT DL
170 PRINT: PRINT "Aha... I have it...": PRINT: PRINT S$
180 FOR DL=1 TO 2000: NEXT: PRINT: GOTO 60
190 DATA "WHY GRANDMA - WHAT BIG EYES YOU HAVE!
" :REM 10 spaces at end
200 DATA "THE BETTER TO SEE YOU WITH, MY DEAR!      "
:REM 10 spaces at end
210 DATA "BABY BEAR'S PORRIDGE WAS JUST RIGHT!
":REM 10 spaces at end
```

(Ed. Note: If you don't want to type it in yourself, send Mike $5 with your name and address and your machine brand. He'll send you a disk with the program on it. If you include an extra dollar, he'll also include a sample issue of the Recreational & Educational Computing Newsletter (REC), ordinarily $3.50 alone. Alternatively, Dr. Ecker advises that he will send the program on disk FREE to any TRSTimes reader who so identifies himself and requests it free when he or she subscribes to the REC Newsletter for at least one year, $24 per 8 issues.)

The Collatz/Ulam Conjecture

Here's a little recreation to explore on your own. The program is so easy to do that I'll ask you to write one in Basic yourself.

Here it is. Start with a natural number (1,2,3,....). If odd, triple it and add 1. If even, divide by 2 (alternatively, take half). Iterate (i.e., repeat). Must you eventually hit 1?

More precisely, must you eventually hit the sequence of length three: 4,2,1, 4,2,1, 4,2,1, etc.?

Consider the example of starting with 6. Since it's even, we divide by 2 to get 3. Since 3 is odd, we triple and add 1 to get 10. In turn, 10 leads to 5, 5 leads to 16, which leads to 8, which leads to 4, then 2, then 1, then 4, then 2, then 1, etc.

Does this always happen no matter what number you begin with? Most who encounter this teaser believe so, but this has proven to be a conjecture difficult to establish. It appears to have first been posed around 1930 or so by a German mathematics student or professor L. Collatz. There was interest in the U.S., perhaps shortly afterwards, around Syracuse (thus the name of the Syracuse Problem in some circles), particularly associated with the late Prof. S. Ulam.

Though this problem is interesting in its own right, I have a secondary motive for mentioning it here. If you read Dr. Allen Jacobs' review of my Magic Math Plus package in the previous issue of TRSTimes (see my letter to the editor this issue as well), then you may note that I am alluding to the same question as Allen brought up. He noted that this is one of three dozen programs on the TRS-80 version of Magic Math Plus (this one being one of the smallest ones), and specifically, he asked whether this assertion could be proven.

As of 1989, the question remains unsolved by professional mathematicians (including yours truly, a Penn State math professor) and amateurs alike, resisting efforts at conclusive proof. On the other hand, computer investigations have failed to produce any counterexamples using numbers so large as to have as many as thousands of digits. That is, every such starting number has resulted in 4,2,1... This, of course, doesn't prove there is not some very large number you start with that doesn't eventually lead to this 4,2,1 cycle. Still, nobody has produced a number that doesn't eventually lead to it.

Casting Out Nines

Dr. Jacobs also mentioned, in the course of the Magic Math Plus review, seeing a newspaper item that made an interesting claim, and he asked for proof. Specifically, the assertion is that any number from 1 to 100, when multiplied by 99, results in a number whose digits sum to 18.

Professional mathematicians will recognize this as just a special case of so-called "casting out nines", the basis of many interesting tricks. The idea is this: A number is divisible by 9 if and only the sum of its digits is.

For instance, is 1,111,111,111,111 divisible by 9? Well, the sum of the digits is 13. Since 13 is not divisible by 9, neither is this large number. In fact, we know more: Since the remainder upon dividing 13 (the sum of the digits) by 9 is 4, the remainder upon dividing this large number by 9 is also 4.

Note that 99 is a multiple of 9. Hence, any multiple of 99 is also a multiple of 9. So, the newspaper item is partially confirmed: We know that multiplying 99 by any of 1,2,3,...,99,100 does result in a number whose digits sum to a multiple of 9. But why do they sum to 18 instead of 9 itself, or to 27, etc.?

As Dr. Jacobs points out, a computer program could confirm this directly:

```
10 CLS: PRINT "Program to confirm assertion that 99 times
any natural number"
20 PRINT "from 1 to 100 results in a number whose digits sum
to 18.": PRINT
30 PRINT "Hit <ENTER> when you are ready to watch
results...";: INPUT X$
```

```
40 FOR N=1 TO 100
50 M=99*N: REM M is the multiple of N
60 M$=STR$(M): REM Convert multiple to string to pick out
digits
70 L=LEN(M$): REM Length of string
80 FOR D=1 TO L
90 S=S+VAL(MID$(M$,D,1)): REM sum the digits
100 NEXT D
110 PRINT "For number = "; N; "- multiple = "; M; " with digit
sum = "; S
120 S=0: REM Reinitialize variable for digit sum
130 NEXT N
```

The program confirms the assertion. Note that 99x101 = 9999, with digit sum = 36, not 18 (nor 9 or 27). One can debate whether any special significance can or should be ascribed to this result using just 1 to 100.

Reader Responses

So far, perhaps because there was only one column, there has been little reader response to the million-dollar word. Depending on dictionary accepted, we have as many as two or three words with value exactly equal to one million. Recall that the value of a word was defined as the product of the letter values, with A = 1, B = 2, ..., Z = 26. For instance, the value of CAT is 3x1x20, a paltry 60.

The key to finding such a word is to notice that the only possible letters are those whose values are divisors of 1,000,000, namely 1, 2, 4, 5, 8, 10, 16, 20, and 25. This reduces the problem to construction of possibilities followed by the search for a legitimate word.

As for the problem with dimes, nickels, pennies, N cents (N > 1) can be given in exactly N different ways (using only dimes, nickels, pennies, at most) if and only if N = 81. I will omit the details.

New Challenge: The Permuted-Change Problem

In December of 1976 in New York City, a quarter-pounder, large order of fries, and Coke cost $1.85 after tax. I remember that because I went to a McDonald's restaurant, gave the clerk a ten-dollar bill, and got back $8.15 in change.

The fact that the change was a permutation of the price - i.e., the digits of the change and the price were the same, but (possibly) scrambled - struck me as curious.

What other prices have this property?

Though I analyzed this without computer benefit, it makes an interesting programming exercise to let a machine answer the question if a pure mathematical analysis isn't your preference. For purposes of this question, we will consider leading zeros as permissible and counted, as in $0.76 (instead of $.76).

I welcome your programs, disks, solutions, comments, improvements, and orders for REC or Magic Math Plus. Write to me directly about this column or REC, but enclose a SASE if you expect or would like a reply:

Dr. Michael W. Ecker
TRSTimes / Recreational Computing
129 Carol Drive
Clarks Summit, PA 18411
(717) 586-2784

I have lots more, but I need to hear from you. Would you like to see more such challenges? Type-in programs in Basic? Tutorials? Recreations? Math? "Mathemagical" tricks? What??

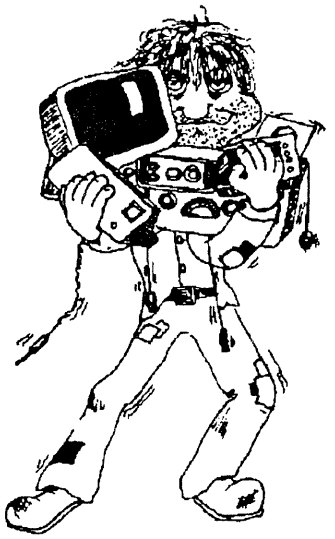
Until next time, happy recreational computing!

Jack's Tip Sheet
MODEL 4 KEYBOARD PROBLEMS
Hardware by Jack Eich

Keyboards, rather I should say keys, do give problems. Since it usually starts with 1 key and then just a few. I have seen whole keyboards replaced for one bad key by Radio Shack repair centers. What a job! First they must desolder the 20 wire cable connector at the keyboard end of the keyboard cable and resolder it, or a new connector, to the new keyboard at a cost of more than $100. By comparison it is much less work to desolder even a dozen keys, clean them and then reinstall them back in place! I'll talk about ALPS keys only, since they are most common on late Model III's and Model 4's. Others are most difficult, particularly the cluster arrow-key jobs with the flexible printed circuit cables, there is no hope for those. Also, the 4P keyboards also look hopeless.

The ALPS are the best from the keybounce standpoint and ease of cleaning. They are easily recognized by having 4 terminals out of the bottom of the keys. The horizontal pairs lengthwise of the keyboard are connected together internally such that there are only contacts shorted together when you press the key.

First, if you have more than 1 key bad, check the keyboard schematic in a tech manual. If the bad keys are connected to the same data or address line, it may be a cable line open, rather than a key. I'll leave repairing the cable to you. It will usually be one or a few keys.



Next, remove the computer top cover and the keyboard and cable so you can work on it. Mark or otherwise indentify each key on the solder side so you can remove the correct keys. Heat up your soldering iron, a 25 watt one is about right. Don't get the key connections too hot! You may spoil the printed circuitry. Using the soldering iron and solder wick, suck up the solder at each of the four connections. The connections are usually sticking straight through their holes in the circuit board. If they have been bent over, use an Exacto knife to lift the bent pin. Make sure each of the 4 pins to each key are free of its hole. When all of the bad keys are desoldered, turn the keyboard over to remove the keys.

I use a regular screw driver to remove the key caps. Stick it under the cap and twist to get it off. You may have to remove several additional key caps in order to make room to remove the keys themselves. I use a #11 Exacto blade in a straight shank holder to slip under the exposed top of the key case to lift up. Insert near one of the two latches that hold the key in place. Using a small jeweler's screwdriver, push the latch nearest the knife blade towards the key center. That side of the key should raise up a bit. Now move the small screwdriver to the other latch and press it in. With a little practice you can pop the keys out in this manner. All the keys are the same so there is no problem mixing them up.

Now that you have the keys out, it's time to open them up and clean them. Assuming you are right handed, hold a key in your left hand with your thumb pressing the two latches that held the key to the board and the key terminals pointing to your right. Put the blade tip of the Exacto knife just to the left of closest pair of the 4 key terminals. You'll feel and see a step in the case which is the edge of a catch that holds the top of the key to the bottom part. Slip the blade under the catch and lift it, unsnapping this side. Turn the key over and unsnap the other side.

DANGER! You are about to release some small parts which are easily lost. You now have in front of you the top and bottom of the key case; the key stem with it's 4 legs; a small spring; and a small rubber cup with a stem to hold the spring. Inside the cup you will see an attached black disk made of carbonized rubber. The latter is the moveable contact of the key assembly. Inside of the key case, you will see two hemi disks of (dirty) bright metal.

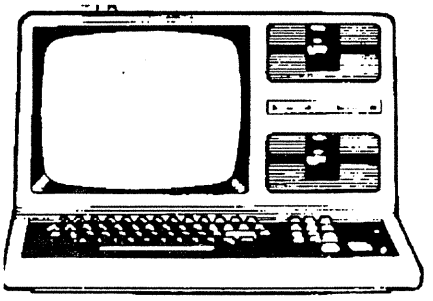
Now to clean it and put it back together! You need a nearly new pencil with a soft eraser. Stick the eraser into the key body and twirl it back and forth 10 or 20 times, then blow it out. Now you can see the two semicircles of bright metal. Each of them goes to two of the external solder connectors. Twirl the eraser until the metal if perfectly clean. You can now see a protruding reversed dimple on each of the two pieces of metal. Contact is made when the carbonized rubber hits these dimples. Now clean up all the parts (a pipe cleaner works well!) and we'll reassemble. First, put the spring on the stem of the rubber

cup and pull back one side of the cup, holding them between your left thumb and middle finger so that the black disk connector is exposed. Take the back of the Exacto blade and make one wipe across the contactor, cleaning the surface. Look closely at that black contactor, you may be able to see where those two pimples on the two semicircles made contact. Keeping track of those two dots that the contacts made, reassemble the spring and cup so that the pimples make contact in an unused area. Drop the key cap stem into place; it will go in any of 4 ways, each is OK. Put the main top on and snap it into place.

How do you like that? Actually, pretty dang simple, once you've done it. Need I say, go ahead and clean the rest of the keys, assemble them into the keyboard, solder in place, put on the key caps, etc, etc??

If you really do need a new keyboard, a friend of mine, Dave Dalager, vice president of the Mid-cities User Group in Arlington, TX, advises to order (specifically) a Model 4 keyboard, computer catalog number 26-1069, Tandy stock number AXX-0243. The price is somewhere in the neighborhood of $31.00. Also, if you are replacing one of the cluster arrow keyboards you need a keyboard bezel, Tandy stock number AZ-0095, for about $7.86. Tandy National Parts toll free number is: 1-800-322-3690. Unfortunately, the new keyboard comes without the keyboard cable. I suggest you cut off the old keyboard cable next to the solder or connector at the keyboard end. Use a 20 pin DIP male (both sides) and solder it to the new keyboard with the remaining pins sticking up. If you can find the pins in a 90 degree bend configuration, point the open pins to the rear. Makes it a bit handier. Now attach a 20 pin IDC connector to the cut-off end of the old cable. You may have to peel back just a bit of the shield on the cable. If this is too much for you, send me the cable and I'll attach the IDC (that's insulation displacement connector) and return it with a male-male connector for you to solder in. For this I ask $5.00. My address is: Jack Eich, 1643 Bolingridge Dr. Orange, CA. 92665

Dave is handy to Ft. Worth and Tandy surplus sales has some real goodies like 4P main boards both gate array and non-gate array for less than $60 and video boards for less than $30. Hi-rez boards for $1200, etc.,etc. You can write him at: 1313 A Timberlake Dr. Arlington, TX. 76010



Assembly 101
Z-80 without tears

by Lance Wolstrup

In the last issue I promised to cover the Assembly language version of INKEY$ and INPUT. So, without further ado, let's do just that.

INKEY$

The Basic INKEY$ function scans the keyboard to see if a key has been pressed. If a keystroke is detected the ASCII value is stored in the variable assigned to the INKEY$ function. For example:

```
10 I$=INKEY$
```

The problem with line 10 is, of course, that the INKEY$ function is so fast that the keyboard scan is executed long before we can possible press a key. Therefore we must add code to keep line 10 scanning the keyboard UNTIL a key is pressed:

```
10 I$=INKEY$:IF I$="" THEN 10
```

Deep down in the dark jungles of ROM, both Model I and III, we have routines that behave identically to the two Basic examples. The first is located at 2BH. A CALL to this routine will scan the keyboard once. If a keypress is detected, register A will contain the key ASCII value (in hexadecimal), otherwise register A will contain 0. This CALL is useful for making things happen while a key is NOT pressed. A good example might be an animated screen sequence until a keypress is detected.

```
INKEY   CALL    2BH         ;ROM INKEY$ routine
        CP      0           ;is register A=0?
        JP      NZ,KEYYES   ;not zero-so jump
        ..      ..          ;A=0, so do animation
        ..      ..          ;animation code here
        ..      ..
        ..      ..
        JP      INKEY       ;go back to INKEY
KEYYES  ..      ..          ;action if key was pressed
        ..      ..          ;action continues
```

As you can see from the above example, whatever animation we wish to perform will take place whenever a keystroke is NOT detected. At the conclusion of the sequence we go back and scan the keyboard again. This is repeated until a keystroke IS detected. Then register A will contain a non-zero value, and we exit the loop and jump to the KEYYES routine.

The other version of INKEY$ is found at 49H. It behaves identically to the latter of the Basic examples: It returns only when a key has been pressed. The ASCII value (in hexadecimal) of the pressed key is stored in register A.

This is the version of ROM INKEY$ used most often. It is certainly the one we will use when we build our multiple keystroke INKEY$ routine later in this series of articles.

```
INKEY   CALL    49H         ;ROM INKEY$ routine
        CP      13          ;did we press ENTER?
        JP      NZ,INKEY    ;if A is not 13 - go back
        ..      ..          ;it was ENTER - so process
        ..      ..
```

Here we check to see if the ENTER key was pressed. If so, we proceed to the code that takes care of whatever it is we do when that key is pressed. If some other key was pressed, we go back and wait for the next keystroke.

RECAP

CALL 2BH - gets character from keyboard.
if A=0 - no character
if A<>0 - A=character
Register DE is changed

CALL 49H - wait for character from keyboard
A=character (always)
Register DE is changed

Do keep in mind that neither the Basic, nor the two assembly language INKEY$ routines, will echo the key press to the screen.

INPUT

The Basic INPUT command allows the user to type a line from the keyboard which is echoed on the screen. The line is terminated when the <ENTER> key is pressed, each character stored in the variable associated with INPUT. For example:

```
10 INPUT I$
```

Before termination, typos can be corrected by using the <LEFT ARROW> key to backspace, erasing the character to the left of the cursor. This is a good enough command for beginning Basic but, as one gathers more experience writing programs, it is easy to see that it gives the user entirely too much freedom to type responses that might be destructive. Hence, I will usually use an INKEY$

routine rather than the INPUT command when writing Basic programs.

The Assembly language version of INPUT is quite a bit better. Here we have the features described above AND we can control the maximum number of characters we will allow the user to type.

The INPUT routine is also found in ROM. To be specific, it is located at 40H. Careful now, there is more to it than just a CALL to 40H. Several things have to be done before we can enter the ROM routine.

First, we should position the cursor to the place on the screen where we want the INPUT to take place. This is usually taken care of by simply terminating the prompt with 03H, thus suppressing a carriage return.

Secondly, we have to have somewhere to store the INPUT. Basic takes care of that by allowing us to store the response in a variable. Assembly language does not use variables, so we must create a place in memory to store the input ourselves. This is done by using the assembler command DEFS (*define space or define storage*). This area should be large enough to hold one character more than the allowable input (to make room for the < ENTER > character).

Register HL must point to the storage area.

Register B should hold the maximum characters allowed.

Here is an example:

```
00100          ORG     7000H

               ;10 CLS
00110  START    CALL    1C9H        ;clear screen

               ;20 PRINT@128,"Type EXIT to return to DOS ";
00120          LD      HL,15488    ;screen location 128
00130          LD      (4020H),HL  ;position cursor
00140          LD      HL,MSG      ;point HL to prompt
00150          CALL    21BH        ;display. M1=CALL 4467H

               ;160 INPUT I$
00160          LD      B,4         ;max number of charaters
00170          LD      HL,BUFFER   ;point HL to storage area
00180          CALL    2BH         ;call INPUT ROM routine

               ;190 IF IMID$(I$,1,1)="E" THEN 230 ELSE 110
00190          LD      HL,BUFFER   ;point HL to storage area
00200          LD      A,(HL)      ;get first character
00210          CP      'E'         ;is it E?
00220          JP      NZ,START    ;not E - start over

               ;230 IF MID$(I$,2,1)="X" THEN 270 ELSE 110
00230          INC     HL          ;point HL to next character
00240          LD      A,(HL)      ;get second character
00250          CP      'X'         ;is it X?
00260          JP      NZ,START    ;not X - start over

               ;270 IF MID$(I$,3,1)="I" THEN 310 ELSE 110
00270          INC     HL          ;point HL to next character
00280          LD      A,(HL)      ;get third character
00290          CP      'I'         ;is it I?
00300          JP      NZ,START    ;not i - start over

               ;310 IF MID$(I$,4,1)="T" THEN 350 ELSE 110
00310          INC     HL          ;point HL to next character
00320          LD      A,(HL)      ;get fourth character
00330          CP      'T'         ;is it T?
00340          JP      NZ,START    ;not T - start over

               ;350 CMD"S"  'REM return to DOS)
00350          RET                 ;it was EXIT - so goto DOS


               ;MSG holds out prompt
00360  MSG     DEFM    'Type EXIT TO RETURN TO DOS '
00370          DEFB    03H         ;use 03H to keep cursor
00380                              ;positioned immediately
00390                              ;to the right of the space
00400                              ;after the word 'DOS'

               ;this is our storage area - room for 5 characters
00410  BUFFER  DEFS    5           ;allow room for 4 key-
00420                              ;strokes + <ENTER>
00430          END     START
```

RECAP

CALL 40H - get a line of input from keyboard

before entry: HL points to storage area
storage area must hold max. number of chrs + 1
register B = max. number of chrs

on exit: HL points to beginning of storage area
register B = number of chrs entered
register DE is changed

Before we get on with writing the mailing label program we need to take care of several items, the first being a command we already know.

ANOTHER GOTO COMMAND

Basic has only one GOTO command; Assembly language has two. We already know that we can branch to anywhere in our program by using the JP (jump) command. This command uses 3 bytes and it is just fine to use it anywhere. However, we have a version of the command that uses only 2 bytes, so it is more elegant to use it, when possible.

The new version is JR (jump relative) and it can be used whenever you are not branching too far back or forward. The limits are 128 bytes back and 127 bytes forward, relative to where the JR instruction is located in the program. Since the above demonstration program is short, all the JP instructions could have been replaced with JR, making the program 4 bytes shorter. Another advantage is that using JR instead of JP will help make the program relocatable, something you'll want to do when you begin writing ML routines to be accessed from Basic.

THE STACK

When we write a Basic program and need to store some value temporarily, we invent a variable and store the value there. As mentioned earlier, Assembly language does not use variables; instead we use registers and memory locations to store our values.

So what happens when we need to store a value AND all the registers already contain values we need? Good question! We could define as many storage areas as needed with the DEFS or DEFB assembler commands. This is done when there is absolutely no other way, as this is, by machine language standards, very slow. The best way is to use the STACK.

The STACK is an area in memory where data values from the CPU can be stored and retrieved extremely fast. Up to this point we have only talked about the 8 bit registers A, B, C, D, E, H, L, the flag register F, and the combination 16 bit registers AF, BC, DE, and HL. We have another register called SP (stack pointer). The SP is a 16 bit register in the CPU that contains the address of the current location that is the "top" of the stack. Understanding the use of the stack is very important when writing any Assembly language program; if the stack or stack pointer ever get destroyed, your program will disappear into never-never land and locking up the the computer, requiring a reset.

The stack is organized into a "last in - first out" system. When new values are "pushed" onto the stack, they are saved backwards in memory, and the stack pointer is decremented by 2. When values are "popped" out of the stack, the SP is incremented by 2.

All uses of the stack are for the double registers only. They can be copied onto the stack with the PUSH instruction, and retrieved with the POP instruction.

For example, let's suppose that we need to have the value 64 stored in register DE. The problem is that we also need the current value of DE. The solution is to save the current value of DE on the stack, load DE with 64, use it, and then retrieve the original value of DE from the stack.

```
PUSH    DE        ;save DE to stack
LD      DE,64     ;the value 64 now in DE
ADD     HL,DE     ;add 64 to HL
POP     DE        ;restore original value in DE
```

This example will be used often in our label program. Register BC is used for something which, for now, is not important to our discussion; register DE points to a series of graphic characters; HL points to the the screen location where a character is to be displayed. Now, whenever we need to move down one line, since the Mod I & III screen width is 64, register HL needs to upped by 64. But, we cannot lose the address pointed to by DE, so we quickly save the value of DE onto the stack, load DE with 64 and perform the ADD; then we POP the address back in DE.

The stack can also be used to copy 16 bit value from one double register to another. For example, if we needed to copy the value of HL to DE, we could do the following:

```
LD      D,H
LD      E,L
```

This is fine, but could be done as easily with:

```
PUSH    HL
POP     DE
```

As you see from this example, you need not POP a value back into the same register pair that PUSHed it.

The stack pointer is also used with the CALL and RET instructions. Another register (PC, for program counter) always points to the next instruction in memory. When the computer encounters a CALL instruction, the PC contains the RETurn address and this value is therefore PUSHed onto the stack. The stack pointer is decremented by 2, and the computer branches to the specified CALL address. When the RETurn instruction is encountered, the address is POPped off the stack, the SP is incremented by 2 and, thus, we return to the instruction immediately after the CALL.

Do be careful with the stack. If during the subroutine you PUSH a value onto the stack and forget to POP if off, the RET instruction will get lost and CRRAASSHH!!

SPECIAL LOOPS

As you know from programming in Basic, loops are worth their weight in gold, saving us from writing an enormous amount of code. The same, of course, is true in Assembly language. The loop we used in the demonstration program kept going back to START if the compared character was not the one we were looking for, thus setting up a series of 4 loops that would only be broken if the characters were correct.

Basic has the famous FOR/NEXT loop; Assembly language has one that is very similar. It is the one invoked with the DJNZ instruction. DJNZ means *decrement register B and jump to the label as long as B is non-zero.*

Here is how it works:

Register B must be loaded with the value that indicates how many times to repeat. The code to be looped should be marked with a label, followed by the code itself. Finally, we issue the instruction: DJNZ *label.*

For example:

```
        LD      HL,15424    ;point HL to screen location 64
        LD      B,64        ;loop counter
LOOP    LD      (HL),131    ;POKE CHR$(131) to location 64
        INC     HL          ;next screen location
        DJNZ    LOOP        ;keep looping until B=0
```

This could be written in Basic as follows:

```
10 HL=64
20 FOR X=1 TO 64
```

```
30   PRINT@HL,CHR$(131)
40 HL=HL+1
50 NEXT X
```

Both programs draw a line of CHR$(131) across the second screen line. Keep in mind that Basic's FOR/NEXT loop is sensitive to premature exits and, if this happens too many times, can cause the program to fail. The DJNZ loop does not have this flaw; all that happens if the program logic forces an exit from the loop before B=0, is that register B contains the remaining value. No big deal!

The most powerful loop in the Z-80 is the LDIR instruction. It can copy a block of memory from one place to another so fast that you can't even think about blinking your eye. LDIR means *load the memory location that register DE points to with the contents of the memory location pointed to by register HL; increment both HL and DE; decrement register BC; continue this procedure until BC reaches zero.*

Let's use the DJNZ example to demonstrate how the LDIR instruction works; that is, let's use it to draw a line across the second screen line.

```
          LD      HL,15424    ;HL points to source
          LD      (HL),131    ;CHR$(131) now in HL
          LD      DE,15425    ;DE points to destination
          LD      BC,63       ;BC is counter
          LDIR                ;copy HL to DE 64 times
```

As mentioned, this instruction is used to move whole blocks of memory from one place to another. For example, suppose we write a program where we can bring up a help screen and, when done with the help, we want to return to the original screen with its data intact. The way to do that is to save the current screen in a buffer before overlaying it with the help screen; then, when done, we copy the contents of the buffer back to the screen.

```
          ORG     7000H
BUFFER    DEFS    1024        ;storage area
START     ..      ..          ;program code here
          ..      ..          ;program continues
;now we come to part of the program where we will call the
;help screen
          LD      HL,15360    ;HL=start of screen
          LD      DE,BUFFER   ;DE=storage area
          LD      BC,1024     ;store entire screen
          LDIR                ;copy screen to buffer
          CALL    HELP        ;bring up the help screen
;we have now returned from the help screen, so we need to
;restore the screen we saved
          LD      HL,BUFFER   ;HL=start of buffer
          LD      DE,15360    ;DE=start of screen
          LD      BC,1024     ;bring back entire screen
          LDIR                ;bring it back
          ..      ..          ;program continues
          RET                 ;program returns to DOS
HELP      ..      ..          ;this is the help subroutine
          ..      ..          ;more help
          RET                 ;return to caller
          END     START
```

There are several variations of LDIR but, since we have covered quite a bit of territory in this installment, we will skip them for now. Instead, we will get started on our project to write a program, called TRSLABEL, that will generate mailing labels on your printer.

The program is very simple. It will allow the user to type a name, street address, city - state & zip, country, and another line of information (if needed). When done, an editing routine allows changing the information in any of the fields. When everything is correct, pressing <0> prints a label on the printer. Labels are made one at a time, so If another is needed with the same information, press <0> again.

When the user wants to type a different label, pressing <ENTER> leaves editing mode and the program starts over, prompting for name, street address, etc.

For the fun of it, I added a feature that will let you store your own name and address for label printing. This can be brought up at any time by pressing <CLEAR>. Sure beats typing it in every time you need a label for a return address.

To exit the program, type <SHIFT><UP-ARROW>.

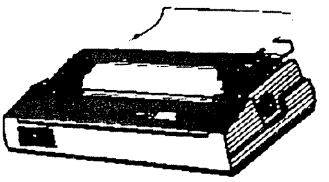
The original plan was to have both the Basic and Assembly language version of TRSLABEL in this issue. However, since we are getting short on space, we will have to settle for just the Basic program this time.

Type it in, I think you will find it useful. I have commented each section of the code, so it should be fairly easy to follow. Study the program until you know exactly what is going on, because in the next issue we will take each section and translate it as verbatim as possible into Assembly language.

TRSLABEL/BAS

```
1  'TRSLABEL/BAS
2  '(c) 1989 TRSTimes & Lance Wolstrup
3  'written for Jul/Aug Assembly 101
4  'the Basic version
5  '
100  CLEAR 500

set up the screen

110 CLS
120 PRINT@0,"TRSTimes  Presents:"
130  PRINT@28,"TRSLABEL"
140  PRINT@46,"(c) Lance Wolstrup"
150  PRINT@80,"a quickie mailing label program"
160    PRINT@128,STRING$(64,131)

set up the prompt fields

170  PRINT@192,CHR$(31)
180  PRINT@269,"Name:  "
190  PRINT@330,"Address:  "
200  PRINT@384,"City, State & Zip:  "
```

```
210 PRINT@458,"Country: "
220 PRINT@524,"Other: "

draw line at first field and enter INKEY$ routine

230 LO=275
240 GOSUB 810
250 IF I$=CHR$(31) THEN 580
260 LB$(1)=A$

draw line at second field and enter INKEY$ routine

270 LO=339
280 GOSUB 810
290 IF I$=CHR$(31) THEN 580
300 LB$(2)=A$

draw line at third field and enter INKEY$ routine

310 LO=403
320 GOSUB 810
330 IF I$=CHR$(31) THEN 580
340 LB$(3)=A$

draw line at fourth field and enter INKEY$ routine

350 LO=467
360 GOSUB 810
370 IF I$=CHR$(31) THEN 580
380 LB$(4)=A$

draw line at fifth field and enter INKEY$ routine

390 LO=531
400 GOSUB 810
410 IF I$=CHR$(31) THEN 580
420 LB$(5)=A$

display label and options

430 PRINT@192,CHR$(31)
440 PRINT@271,"1 - ";LB$(1)
450 PRINT@335,"2 - ";LB$(2)
460 PRINT@399,"3 - ";LB$(3)
470 PRINT@463,"4 - ";LB$(4)
480 PRINT@527,"5 - ";LB$(5)
490 PRINT@640,CHR$(31)
500 PRINT@646,"Press number of field to change - or press 0
to print"
510 PRINT@708,"Press (ENTER) for new label - press (CLEAR)
for YOUR label"
520 PRINT@783,"Press (SHIFT)(UP ARROW) to quit"
530 I$=INKEY$:IF I$="" THEN 530

check to see which key pressed

540 IF I$=CHR$(27) THEN CLS:END '(shift)(up-arrow)
550 IF I$=CHR$(13) THEN 170 '(enter)=new label
560 IF I$="0" THEN 730 '(0)=print label

replace LB$(1)-LB$(5) with your own name and address

570 IF I$<>CHR$(31) THEN 640
580 LB$(1)="TRSTimes magazine" '(clear)=your label
590 LB$(2)="20311 Sherman Way. Suite 221"
600 LB$(3)="Canoga Park, CA. 91306"
610 LB$(4)="U.S.A."
620 LB$(5)=""
630 GOTO 430

edit routine - first check if key-press is valid

640 I=VAL(I$) 'edit routine
650 IF< I OR I>5 THEN 530

key press is valid, so edit the chosen field

660 PRINT@640,CHR$(31)
670 PRINT@651,"Change: ";LB$(I)
680 PRINT@719,"To: ";

draw line and enter the INKEY$ routine for edit

690 LO=723
700 GOSUB 810
710 LB$(I)=A$
720 GOTO 430

send label to printer - make sure it is on and on-line

730 LPRINT LB$(1) 'print routine
740 LPRINT LB$(2)
750 LPRINT LB$(3)
760 LPRINT LB$(4)
770 LPRINT LB$(5)
780 LPRINT
790 GOTO 430

800 END

draw line to indicate the allowable length of the field

810 A$=""
820 PRINT@LO,STRING$(34,95);

INKEY$ routine

830 L=0
840 I$=INKEY$
850 IF I$="" THEN 840
860 IF I$=CHR$(13) THEN PRINT@LO+L,CHR$(30);:
RETURN
870 IF I$=CHR$(8) AND L=0 THEN 840
880 IF I$=CHR$(9) THEN 840
890 IF I$=CHR$(10) THEN 840
900 IF I$=CHR$(8) THEN L=L-1:A$=LEFT$(A$,L):
PRINT@LO+L,CHR$(95);:GOTO 840
910 IF I$HR$(31) THEN 840
920 IF I$=CHR$(31) THEN RETURN
930 IF L=34 THEN 840
940 PRINT@LO+L,I$;:A$=A$+I$:L=L+1:GOTO 840
```



CP/M - The Alternate DOS for Model 4

Some thoughts on CP/M disk formats

by Roy T. Beck

Due to the availability of several formats in Monte's CP/M, a decision must be made as to which format to use for any particular disk. Some factors to consider include:

Single vs double sided drives
Data vs System disks
Number of floppy drives in system
Presence of Hard drive
Size of data files to be saved.

Floppy-only Systems

If no hard disk is available, then the floppies must be formatted for best overall utility. Unless you are rigorous in marking your disks with notations showing their format, I strongly urge you to opt for the fewest possible number of different formats for your system. Even if you are careful always to mark the formats on the labels, it can still be a nuisance to have to reset your configuration when you change from one disk format to another.

I would also recommend all your drives be the same, if possible. Having a mixture of 35, 40, and 80 track and SS and DS drives may be interesting, but you will waste a lot of time trying to resolve the question of which drive to format in what way to solve a particular problem.

When no HD is available, you must have Drive A formatted for a system disk. Your choice here is between SS and DS. If your drive is DS, then go for DS system disks for the extra space. But go the whole way and make all your system disks DS. Naturally, when you reboot, your Drive A config will match your system disk, whatever it is. The only problem with settling on DS system disks as your standard is that your system disk will not boot in a friend's machine if his Drive A is only SS. This is really only a minor consideration.

If large data files are needed, then data only disks may be favored for physical drives B, C, and/or D. If data disks are chosen, and double sided drives are installed, then go for DS data disks of 400K as your standard.

But there is another option which should be considered. That is to make all your disks system disks, especially if your drives are DS. This has the great virtue that only one configuration is normally used on all your drives, and more particularly, any disk can be booted. This latter item is a great plus when you pick up a disk which you forgot to label at 2AM a couple of nights before. *What is it? What's on it?* Simple, just boot it and look at the directory. Try that with a data disk! And even after you determine a disk is a data disk, which format is it? Having all disks formatted as bootable system disks is much simpler.

Another bad habit (of which I am also guilty) is purchasing a library from someone who was using an earlier BIOS, and not converting the useful programs to your current BIOS and format. If the old disks are to be saved, put them aside as a group, with each disk carefully marked as to system/data, SS/DS, and BIOS version. If the disks are from a Kaypro, etc, then the machine and model must also be indicated. A little time invested in labeling while your memory is fresh can pay great dividends when you want to make use of the disks at a later time.

HD Systems

If you are fortunate enough to have a hard disk on your system, then you will surely have your system files on it, and it will be configured as your Drive A. In this case, you must construct two configuration files, one on your boot disk and one on your HD. If your floppies are SS, there is no choice for your boot disk. It must be a SS system disk. If you have DS floppies, then I would still go with a SS system disk, as you only need it to start your HD, and there is no need for a lot of space on the boot disk. The HD config table will have the HD as drive A, and may have other partitions assigned to other logical drives. Your floppies can be assigned to B and C, or higher letters as you please. For convenience in handling your boot disk, two of your floppies should have the same configuration as the boot disk. One small consideration in assigning logical drives in a HD system is the fact we have only 3 function keys on a Mod 4, and therefore only 3 logical drives can conveniently have their directories called by a single function key stroke. If B and C are also on the HD, then you must type the command **D:DIR <CR>** to get a floppy directory, as opposed to the single function key stroke available for drives A, B or C.

80 tracks - A special case

By their nature, HD's are susceptible to failure, and backups of the stored data are mandatory. Regardless of my earlier comments about having all drives alike, (which also implies all 40 track), a rational case can be made for having two 80 track DS drives available for backing up HD files. Of course, HD's can be backed up to 40 track floppies, but it takes twice as many of them. While one

drive meets the minimum requirements, I never feel comfortable with only one drive of a special type. *If it fails, what do I do next?* Having two drives allows direct copies and provides insurance if one 80 track drive packs up. Since the Mod 4 also runs the HD under TRSDOS, the same logic applies about using 80 tracks to backup the HD files there. The 80 track is a nice convenience for backing up purposes. While you might argue that all drives on a system should be 80 track, I would resist this on the basis that the standard of interchange is 40 track, whether for commercial software or private software, and an 80 track-only system would be effectively isolated from the rest of the world. I only suggest the use of 80 track for bulk storage of backups.

Summary

The table below attempts to summarize my feelings on operation of CP/M on a Mod 4 or 4P. One entry is for "4 drives", with no distinction between a 4 and a 4P. This is deliberate, as some people have added outboard drives to their 4P's. Physical drives are identified by 0, 1, 2, or 3, as usual. You may also wonder why the sequence of physical drives for drives C and D is the reverse of the sequence for drives A and B in the "second choice" entries. This is done to facilitate the making of backups. The disk to be backed up remains in its normal drive, whether system or data. To me, this seems to simplify the backup process, with less confusion about which disk goes where.

RECOMMENDED CONFIGURATION

Logical Drive	A	B	C	D
No Hard Drive				
2 SS Drives				
1st choice	0 SS syst	1 SS syst	*	*
2nd choice	0 SS syst	1 SS data	1 SS Syst	0 SS data
2 DS Drives				
1st choice	0 DS syst	1 DS syst	*	*
2nd choice	0 DS syst	1 DS data	1 DS Syst	0 DS data
4 DS Drives	0 DS syst	1 DS syst	2 DS syst	3 DS syst
W/ Hard Drive				
2 SS Drives				
Boot Disk	0 SS syst	1 SS syst	*	*
HD Config	HD syst	0 SS syst	1 SS syst	*
2 DS Drives				
Boot Disk	0 SS syst	1 SS syst	*	*
HD Config	HD syst	0 SS syst	1 SS syst	*

*** Configure as needed**

I have not mentioned MEMDISKs in the above. I don't make much use of them, mainly for two reasons. The MEMDISK is too small (63K) to hold an entire disk, and I haven't developed the knack of making the MEMDISK the system drive, although I am sure it can be done. If you do it, then you have to reconfigure after each reboot, I believe. Maybe it's just laziness or ignorance or a combination of these factors that holds me back, but I just don't do it. Any comments by those who do? Feel free.

FOG anyone?

Let me put in a plug here for FOG. Some of you may be thinking, "*What the heck does FOG have to do with my Mod 4? Or did he mean SMOG?*". No, it's not a typo either. FOG is the short name for FOG International, located in Daly City. FOG began life as The First Osborne Group, a user group of Osborne owners. I believe all Osbornes were CP/M machines, so you can begin to see where this genealogy is leading. When Osborne went belly-up, the user group survived, and encouraged membership by other CP/M machine owners. As time went along, the IBM began to dominate, and so FOG opened its membership to IBM and its clones. They now even have some MAC owners. FOG issues two newsletters each month. One is named FOGHORN and is devoted to our CP/M needs. The other is the FOGLIGHT and it is written for the IBM members. (Some members pay extra to receive both publications).

Why join another group? Enlightened self interest is probably the best answer. The advantages of FOG membership include access to a large library of CP/M programs, an international forum for questions, essays, complaints, political speeches, etc and a place for advertisers to be noticed. What does it cost? $30 per year. I don't know the added cost to receive both newsletters, but it is shown on the membership application. For an application, drop a postcard to FOG International, Box 3474, Daly City, CA. 94015. In case you are wondering, Daly City is a small city close to the southern edge of San Francisco. The word "International" in their name is appropriate, as they have international members. They need us, we need them. I urge you to sign up.

Roy

TRSDOS 1.3. CORNER

The DATE prompt

by Lance Wolstrup

Though SYSTEM 1.5. provides a terrific alternative to TRSDOS 1.3., many of our readers are obviously just too comfortable using our original DOS. Their letters indicate that, while mostly satified with 1.3., there are some changes they would like. In other words, would we do some more patches? Sure, why not? So, once again, the disassembly of 1.3. is on the table in front of me; let's see what we can do.

The most requested change was to eliminate the DATE and TIME prompts at boot-up. This can be done in many ways, the simplest being:

```
PATCH   *0   (ADD=4EA9,FIND=CA,CHG=C3)
```

TRSDOS 1.3. uses 42B5H and 42B6H to store a 2-byte flag indicating whether or not a valid date is present. The initialization routine loads register HL with this 2-byte flag, loads register DE with the 'valid date' flag. It then proceeds to subtract DE from HL. If the result is 0 (HL and DE being equal), the Z flag is set, otherwise the NZ condition occurs. At location 4EA9H a decision is made: If the Z flag is detected, we skip the the date prompt by jumping directly to 4F2EH. If NZ, we go to the date prompt routine. The actual code looks like this: **JP Z,4F2EH**

This, of course, is a conditional jump. If we make it unconditional we will bypass the prompt regardless. Therefore, by changing the above code to **JP 4F2EH** we will have accomplished our mission.

A slightly more elegant way is to not only bypass the DATE and TIME prompts, but to set the date to 00/00/00 and the time to 00:00:00 in the process.

```
PATCH   *0   (ADD=4E9E,FIND=2AB542,CHG=C3C94E)
PATCH   *0   (ADD=4ED4,FIND=3A814F,CHG=C3234F)
```

Here we skip over the very beginning of the date routine by immediately jumping to 4EC9H, where the three bytes starting at 4F81H (all zeroes) are moved to the boot-up date storage beginning at 42B7H. Upon completing this task, DOS usually checks to see if the input is correct (zeroes are not allowed). By modifying the code at 4ED4H from **LD A,(4F81H)** to **JP 4F23H**, we go directly to the portion of the routine where the same three 0 bytes starting at 4F81H are copied to the time storage area. The routine now goes on normally and we have stored a date of 00/00/00 and a time of 00:00:00 without prompts. Incidentally, the date can still be set by typing, for example: **DATE 06/17/89 <ENTER>**. The date is good until you reset the machine. Not bad!

Now, some of you may say: "I want the prompt, but if I don't feel like typing the date, I want to be able to bypass it by pressing <ENTER>."

OK, good enough! Let's do just that.

```
PATCH   *0   (ADD=4EB5,FIND=212051,CHG=C3FE4E)
```

This patch allows you to respond to the date prompt any way you wish. You may type a date, you may just press <ENTER>, or you may type anything else that strikes your fancy. Whatever you do, you will proceed to the time prompt where you've always had the capability of simply pressing <ENTER>. if you didn't desire to set the time. As in the example above, you can enter the date by typing **DATE mm/dd/yy** at any time from DOS. This one stays until you turn the machine off, or change the date with another 'DATE mm/dd/yy' command.

The patch overwrites the beginning of the error routine at 4EB2H. Instead of displaying the "TRY AGAIN" message and repeating the prompt, we simply write a JumP to 4EFEH, which is the beginning of the time prompt.

Keep in mind that these patches can NOT be used together. Select the one you prefer and leave the others alone.

Other readers have asked if we could replace the frustratingly cryptic DOS error codes with the actual error messages. This is probably the easiest patch of them all, as DOS already provides for this. Overlay 4 (the DOS error handler) loads in at 4E00H. After checking to see if "DO" is active, the actual error code is POPped off the stack and eventually copied into register B. Bit 6 of the error code indicates whether or not long error messages should be used. If bit 6 is set (1) we get the long error messages; if bit 6 is off (0) we get the cryptic error codes. The default for bit 6 is 0. We could certainly make sure that bit 6 is always set, but the easier way is to wait until we get to 4E28H. Here we have the instruction to make a relative jump on NZ up to 4E3AH. This is in response to the previous instruction: BIT 6,B. In other words, bit 6 of register B is checked. If it is on (1) we jump, otherwise we fall through to the code to display the error codes. Thus, if we make the conditional jump unconditional, we get the long error messages regardless of the status of bit 6

```
PATCH   *4   (ADD=4E28,FIND=20,CHG=18)
```

This will be all for this issue, but I got my folder with the disassembly out, and I've caught the 'patch-bug' again, so we'll probably have some other goodies in September.

Happy patching - but do be careful.

ATTENTION TRSDOS 1.3 USERS!

GRL SOFTWARE PROUDLY ANNOUNCES "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 to 1989! SYSTEM 1.5 supports 16k-32k bank data storage and 4 MGHZ clock speed (4/4P/4D). DOUBLE SIDED DRIVES ARE NOW 100% UTILIZED! (all models).

CONFIG=Y/N	CREATES CONFIG BOOT UP FILE
TIME=Y,N	TIME BOOT UP PROMPT ON or OFF
BLINK=Y,N	SET CURSOR BOOT UP DEFAULT
LINES='XX'	SET *PR LINES BOOT UP DEFAULT
ALIVE=Y/N	GRAPHIC MONITOR ON or OFF
TRON=Y/N	ADDS an IMPROVED BASIC "TRON"
TYPE=B/H/Y/N	HIGH/BANK TYPE AHEAD ON or OFF
SLOW	2 MGHZ SPEED (MODEL III's)
CPY (parm,parm)	COPY/LIST/CAT LDOS TYPE DISKS
SYSRES=Y/N	DISABLE/ENABLE SYSRES OPTION
SPOOL=H/B,SIZE='XX'	SPOOL to HIGH or BANK MEMORY
SPOOL=N	TEMPORARILY DISABLE SPOOLER
SPOOL=RESET	RESET (NIL) SPOOL BUFFER
SPOOL=CLOSE	CLOSES SPOOL DISK FILE
FILTER *PR,IGLF=Y/N	IGNORES "EXTRA" LINE FEEDS
FILTER *PR,FILTER	ADDS 256 BYTE PRINTER FILTER
FILTER *PR,FIND='XX',CHNG='XX'	TRANSLATE PRINTER BYTE to CHNG
FILTER *PR,LINES='XX'	DEFINE NUMBER LINES PER PAGE
FILTER *PR,TMARG='XX'	ADDS TOP MARGIN to PRINTOUTS
FILTER *PR,PAGE=Y/N or 'XX'	NUMBER PAGES, SET PAGE NUMBER
FILTER *PR,ROUTE=*DO	ROUTE PRINTER to VIDEO if on
FILTER *PR,TOF	MOVES PAPER to TOP OF FORM
FILTER *PR,SPEC=Y/N	ADDS LPRINT CHR$(1-7) CONTROLS
FILTER *KI,CLICK=Y/N	"CLICK" KEYBOARD SOUND on/off
FILTER *KI,LENGTH='XX'	SETS KEYBOARD "CLICK" LENGTH
FILTER *KI,ECHO=Y/N	ECHO KEYS to the PRINTER
FILTER *KI,FILTER	ADDS 256 BYTE KEYBOARD FILTER
FILTER *KI,FIND='XX',CHNG='XX'	TRANSLATE KEYBOARD BYTE to CHNG
FILTER *KI,DVORAK	GOODBYE QWERTY, HELLO DVORAK!
ATTRIB:d,NAME="DISKNAME"	RENAME DRIVE:d DISKETTE
ATTRIB:d,PASSWORD	CHANGE DRIVE:d MASTER PASSWORD
DATE=Y/N	DATE BOOT UP PROMPT ON or OFF
CURSOR='XX'	DEFINE BOOT UP CURSOR CHAR
CAPS=Y/N	SET KEY CAPS BOOT UP DEFAULT
WP=d,Y/N (WP)	WRITE PROTECT ANY or ALL DRIVES
TRACE=Y/N	TURN (SP) MONITOR ON or OFF
MEMORY=Y/N	BASIC FREE MEMORY DISPLAY MONITOR
FAST	4 MGHZ SPEED (MODEL 4's)
BASIC2	ENTER ROM BASIC (NON-DISK)
SYSRES=H/B, 'XX'	MOVE /SYS OVERLAY(s) to HI/BANK MEMORY
MACRO='XX',TEXT STRING + (??)	DEFINE ANY KEY TO MACRO
SPOOL=D,SIZE='XX'	LINK MEM SPOOLING to DISK FILE
SPOOL=Y	REACTIVATE DISABLED SPOOLER
SPOOL=OPEN	OPENS, REACTIVATES DISK SPOOLING
FILTER *PR,ADLF=Y/N	ADD LINE FEEDS BEFORE PRINTING 0Dh
FILTER *PR,HARD=Y/N	SEND 0Ch to PRINTER (FASTEST TOF)
FILTER *PR,ORIG='XX',CHNG='XX'	TRANSLATE PRINTER BYTE to CHNG
FILTER *PR,RESET	RESET PRINTER FILTER TABLE
FILTER *PR,WIDTH='XX'	DEFINE PRINTER LINE WIDTH
FILTER *PR,BMARG='XX'	ADDS BOTTOM MARGIN to PRINTOUTS
FILTER *PR,ROUTE=Y/N	SETS PRINTER ROUTING ON or OFF
FILTER *PR,ROUTE=*RO	ROUTE PRINTER to RS-232 if on
FILTER *PR,NEWPG	SET DCB LINE COUNT to 1
FILTER *KI,EXTKBD=Y/N	ENTER GRAPHICS FROM KEYBOARD
FILTER *KI,TONE='XX'	SETS KEYBOARD "CLICK" TONE
FILTER *KI,PORT='XX'	SEND "CLICK" SOUND TO PORT XX
FILTER *KI,MACRO=Y/N	TURN MACRO KEYS ON or OFF
FILTER *KI,ORIG='XX',CHNG='XX'	TRANSLATE KEYBOARD BYTE to CHNG
FILTER *KI,RESET	RESET *KI FILTER TABLE
FILTER *KI,SPEC=Y/N	ADDS SPECIAL CODES Cntl 1-7
ATTRIB:d,DATE="DO/DO/DO"	REDATE DRIVE:d DISKETTE
DEVICE	DISPLAYS CURRENT CONFIG INFO

All parms above are installed using a new LIBRARY command **SYSTEM (parm,parm)**. Other new LIB options include **DBSIDE** (enables double sided drive use by treating the "other side" as new independent drive, drives 0 - 7 supported) and **SWAP** (swap drive code table #'s). Previous PATCHER/CMD (DBSIDE) customers may upgrade to SYSTEM 1.5 for only $9.95 US funds, original PATCHER disk must be returned. Dump (CONFIG) all current high and/or bank memory data/routines and other current config data, to a disk data file. If your type ahead is active, you can (optional) store text in the type buffer, which is saved. During a boot, the config file is loaded back into high/bank memory, and interrupts are recognized. After executing any active auto command, any stored type ahead data will be output FANTASTIC! Convert your QWERTY keyboard to a DVORAK! Route printer output to the screen or your RS-232. Macro any key, even F1, F2 or F3. Load *01 - *15 overlay(s) into high/bank memory for a memory only DOS! Enter data faster with the 256 byte type ahead option. Run 4 MGHZ error free as clock disk I/O routines are properly corrected! Spool printing to high/bank memory. Link spooling to disk. (Spooling updates DCB upon entering storage.) Install up to 4 different debugging monitors. Print MS-DOS text files ignoring those unwanted line feeds. Copy, Lprint, List, or CATalog DOSPLUS, LS-DOS, LDOS or TRSDOS 6.xx files & disks. Add top/bottom margins and/or page numbers to your hard copy. Rename/Redate disks. Use special printers codes eg: LPRINT CHR$(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 LIB/DVR, 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!

ORDER TODAY!

FREE WITH YOUR ORDER!

The first 50 customers will receive 2 Disks Full of Choice Model III Public Domain Programs.

32k 48k Model III's, 48k 64k 128k Model 4 4P 4D's. Send $39.95 US funds, plus $4.00 postage/handling to:

GRL Software, Suite 209, 1051 KLO Road, Kelowna, British Columbia, Canada V1Y 4X6

Attention SYSTEM 1.5.

FREE WITH YOUR ORDER!

STORAGE POWER

Models I, III, IV, IVD, IVP

HARD DISK DRIVES

5 Meg Hard Disk.....$295.00

10 Meg.....$425.00 15 Meg.....$495.00 20 Meg.....$545.00

30 Meg & up..$Call Bare hard drive bubbles available..$Call

Hard floppy combinations available.

Hard disk drivers..... $49.95

supports most Hard drives & DOS's

III/IV INTERNAL DISK DRIVE KITS

Complete with controller, drive stands, power supply, cables, Add drives & Dos.

2 FH Drives $149.95 4 HH Drives $159.95

FDCController only.....$ 89.95

Internal 20 pin flat ribbon cable .....$ 4.95

Internal 2 Drive Cable.....$ 9.95

Metal drive stands.....$ 29.95

EXTERNAL DISK DRIVES

Complete w/case, power supply and cable.

2 - 40 track HH DS DD..$249.95 2 - 80 track HH DS DD..$269.95

1 - 40 & 1 - 80 track.....$259.95 2 - 3.5" 80 track.....$299.95

1 - 80 track FH DS DD.....$119.95

BARE DRIVES

40 track DS DD FH refurb 360 k.....$ 69.95

Replacement for SS Mod III & IV

40 track DS DD HH 360k.....$ 89.95

80 track DS DD HH 720k.....$ 99.95

80 track DS DD FH 720K.....$ 49.95

3.5" 80 track 720K.....$114.95

DRIVE CASES w/Power supply

Hard drive 1 FH or 2 HH w/fan.....$109.95

Floppy 1 FH or 2 HH .....$ 59.95

MOD IV MEMORY SETS

8 4164 - 200ns new.....$ 19.95 Pulls.....$ 15.95

8 4164 - 150ns new.....$ 24.95 Pulls.....$ 19.95

Pal chip for non Gate/ array.....$ 9.95

MOD IV SPEED UP KITS

Non Gate Array 5Mhz....$ 34.95 Gate Array 6Mhz.$ 34.95

DISKETTES w/ sleeves & labels

Pkg of 10 (5.25").....$ 4.25 (3.5").....$ 11.95

Pkg of 25 (5.25").....$ 9.95 (3.5").....$ 25.95

100 5.25" Disk storage w/ lock.....$ 11.95

70 5.25" Disk storage w/ lock.....$ 9.95

40 3.5" Disk storage w/ lock.....$ 8.95

80 3.5" Disk storage w/ lock.....$ 12.95

65w Power supply...$29.95 CRT Tube green/ amber...$ 74.95

Model I Double Density Board.....$ 89.95

Cables: Printer 6ft.....$14.95 12ft.....$ 19.95

Disk Drive 2 drives 3ft..$ 9.95 Custom cables.$ Call

We can supply most of the parts (new & used) that you will need in repairing & upgrading your Mod I, III. IV's.

Call or write for availability & price.

Look for our BBS Coming soon.

STORAGE POWER

10391 Oakhaven Dr. Stanton, Calif. 90680

(714) 952-2700 (9:30 am - 8:00 pm PST)

All C.O.D. orders are cash only.

Prices are FOB and subject to change and availability.

Calif. residents require 6% sales tax.

*** NEW ***

Recreational & Educational Computing

Have you been missing out on the only publication devoted to the playful connection of computers and math?

The REC Newsletter features programming challenges and recreational math, such as:

the Magic of Schram's 123 String - the probability of an N game at Bingo - time to complete a collection - 6174 - Next Number in Sequence - Locate the Bomb - perfect numbers - Fibonacci numbers - prime number generation and contest - self-reference and paradoxes - self-listing program challenge and solution - pi - mystery programs explained - probability - Monte Carlo simulations

Also:

Fractal art - the world's best card trick (based on algebra) reviews of best software and books - editorial - humor - cartoons - art - reader solutions and more!

Programs supported for: TRS-80, Tandy, MS-DOS and others. REC is available for $24.00 per calendar year of 8 issues

REC Newsletter
129 Carol Drive
Clarks Summit, PA. 18411
(717) 586-2784

NEW PROGRAMS

from the Valley TRS-80 Hackers' Group public domain library for Model I, III & 4

Send SASE for annotated list
Sample disk $5.00 (US)

VTHG

BOX 9747
N. HOLLYWOOD, CA. 91609

STORAGE POWER

Storage Power introduces a new addition to their line of Hard and Floppy Disk drives for the Radio Shack Models I, III & IV's, a 10 Meg removable cartridge hard disk.

This is an excellent unit for backing up those large hard disk files and for security of sensitive data.

Run different DOS's on the same system.

The unit comes complete with case, power supply, cables and removable cartridge, for $1145.00.

For more information please call or write to

STORAGE POWER
10931 Oakhaven Dr. Stanton, CA. 90680
(714) 952-2700

COLOR SLIDES FROM A TRS-80

by Robert M. Doerr

Over a quarter century ago, Eastman Kodak issued a publication in which the qualities of good text and line (as opposed to conventional continuous-tone) slides were described. The four main points are:

1) Limit each slide to no more than 20 words.
2) Such slides should have a black background.
3) Within a slide, bright, warm colors, especially rich yellow attract attention.
4) Because some auditoria cannot handle vertical slides, keep all slides in the horizontal format.

Soon, some presentations included slides of gripping superiority. Unfortunately, all too often, one still sees at meetings crowded slides introduced with "I know you can't read any of this, but...." and black-on-white slides that all but blind the audience in a darkened room.

Sophisticated, but costly, equipment now exists to create great color slides quickly and easily, directly from computer systems.

A number of people, working together, developed a method to produce excellent color slides without elaborate equipment. The method consists of using the computer and desk-top publisher to print the text, black-on-white, on paper, photographing it on 'line' film, applying color tapes to the film, and photographing the film by transmitted light onto slide film. Graphics slides can be prepared similarly, by use of a graphics package. The slides are of high quality; they just take a little longer to ready than when using sophisticated equipment. Unlike sophisticated systems, this method lends itself well to hand-made artwork, either as additions to computer-generated slides or as stand-alone art.

PROCEDURE

The computer is a 128-K Model 4.

The software includes three main packages, DDUTY, DOTWRITER, and an ASCII editor. Under DDUTY, the ASCII editor is in one bank and DOTWRITER in the other. This provides instant switching between file creation and trial printing. (DDUTY, 128K, and Model 4 are not essential; I have done slides on a Model III, but slowly.)

Each slide is carefully created on paper, and preferably occupies the full width of the paper, by use of large fonts. The 2:3 height:width ratio of 35 mm slides is maintained; a goal is to fill the 35 mm slide. Fonts are chosen to avoid those that have small openings for lower-case e's and a's; small openings are difficult to photograph. The font, BIGBOLD/ PR, proved generally good, but lacks a good matching subscript/superscript font, and allows some confusion between the 5 and the capital S.

Any hand work is then added. Paste-ups are entirely feasible.

When the paper versions of the slides are laid out satisfactorily, they are evenly lighted, photographed on Kodalith III black-and-white negative film, and processed in a suitable high-contrast developer. We had an old 5 X 7 camera, and used 5 X 7 sheet film.

It is almost always necessary to apply some opaque to pinholes that form in the emulsion of high-contrast films. Paste-up edges often require opaqueing, which is easy to do on 5 X 7 film.

Different transparent or translucent color tapes, such as Zip-a-Tone or Chart Pak from graphic arts stores, are then applied to selected segments of the text (or graphics) on the film. All of the text (graphics) is so colored. It takes a little trial and error to learn what colors do well, but that is a 1-time learning process, which depends on which tapes one happens to get. A rich yellow is best for highlighting the key point(s) of a slide, oranges or reds next, and blues and greens are good for headings, etc. At times, double-thick taping is useful. We found 5 X 7 film easy to tape; we thought that smaller film, such as 120 roll film, would pose a real problem.

Paints for coloring the text were found to be unsatisfactory.

All this work is done on the emulsion side of the black-and-white film negative. No positive is made.

Lastly, the film is placed on a light box or otherwise evenly back-lit and photographed onto color slide film, which is processed and mounted normally, ready to project.

EXAMPLE

A procedure that proved valuable is this. By way of an example, imagine a presentation on, say, word processing. An outline slide might look like this:

WORD PROCESSING

SETUP
CREATE DOCUMENT
EDIT DOCUMENT
PRINT DOCUMENT
SAVE TO DISK

One might make seven copies of this outline slide, all with the heading in blue. During the introduction, the first copy, with the rest of the slide in orange or red, would be projected.

Then there might be other introductory slides.

When beginning the discussion of SETUP, one might show a copy of this outline as before, but with the word SETUP in rich yellow instead of orange or red.

The slides on the subject of setup would then be shown and discussed.

Then, when beginning the discussion of document creation, a copy of this outline with the words CREATE DOCUMENT in rich yellow (heading still in blue and other topics in orange or red) would be of maximum clarity to the audience.

And so on through the five topics.

Finally, the first copy of this outline (with the five topics in orange or red) could be repeated as a summary of major topics.

The old adage, "Communicate not to be understood but so as not to be misunderstood" still applies.

Note: If you must photograph a bit of text or a black-and-white line work for a slide, please use black-and-white print film, then mount and project the negative. At least, you won't blind your audience.

Robert M. Doerr can be reached at:
39 McFarland Drive,
Rolla, MO 65401-3828
(314) 364-1275

CLOSE#4

In the last issue I stated that it was once again time to ponder on the future of TRSTimes. Boy, I had no idea of the mail it would generate. My mailman, who for obvious reasons would like to see all of this foolishness end, has been giving me the evil eye for the past week, grumbling something about a hernia.

Seriously, thank you for the encouragement and support. I just can't imagine another machine having such wonderful and loyal owners. From the letters received, it is clear that you are sticking with the TRS-80, and that you want TRSTimes around to support it.

The kids are encouraging me to keep the magazine going. They have their own devious motive, figuring that Dad might not check the homework quite as carefully as he would if not absorbed in TRSTimes. *Kids are great!*

My wife, being a writer herself (her 2nd book is coming out in November), understands the creative process and is a constant source of kind words, inspiration and help. (*Yes, I know, I'm very lucky*.)

Thus, the bottom line was: Do I have something left to contribute to the TRS-80 world in 1990?

I deliberately avoided touching any and all computers for over a week. I even tried not to think about them; what a fiasco that was. I know it would be easier to quit smoking than to kick the computer habit. I had actual withdrawal pains, getting restless and cranky. At night I would dream about good looking TRS-80's with provocative double-sided drives (well, almost). I wanted to program, write, tinker, explore DOS, or otherwise just play around with the machine.

This pretty well tells the story. The urge, the excitement, the fun, it is all still there. I am not doing TRSTimes because I *have* to; rather, I do it because I *want* to. Therefore, it certainly should not surprise anyone that the decision is:

TRSTimes again in '90.....

Before closing, I would like to thank Roy Beck, Robert M. Doerr, Jack Eich, Gary Campbell, Dr. Allen Jacobs, Marc Luyens, Howard Mueller, Dennis Burkholz, Robert R. Keegan and Dr. Michael W. Ecker for making this issue possible. A special thanks goes out to Michael Soth for recreating every TRS-80 owners favorite wizard for the cover.

Next issue will feature a clever article about the TRS-80 graphic characters from Fred Blechman, Assembly 101 will bring the machine language version of TRSLABEL, Roy Beck will be back with more on CP/M, there will be more patches for TRSDOS 1.3., and much more.

Don't forget that TRSTimes is actively seeking articles, programs, hints, tips, reviews, and anything else pertaining to our favorite machines. If you have something of interest to the TRS-80 world, send it to us for possible publication.

Until September.....THINK TRS-80!!

Lance W.