TRSTimes

Volume 6. No. 6. - Nov/Dec 1993 - $4.00

HAPPY HOLIDAYS 1993

LITTLE ORPHAN EIGHTY

As you will no doubt notice from the front cover, this is TRSTimes 6.6 - our 36th issue.

When 80-Micro left the TRS-80 community stranded after the December 1987 issue, we attempted to keep the coverage going by publishing six bi-monthly issues in 1988. Our original plan was to publish TRSTimes for just one year, but one year led to the next, and now we've just finished year number six.

Finishing a year means that all subscriptions are now completed - it feels good to have helped to keep our favorite machines going for a while longer, and we wish to thank you, the subscribers, for being with us in 1993.

At the end of each year, as publisher, I always ask myself two questions -- '*is there enough TRS-80 interest out there?*' and '*do I have enough enthusiasm for another year?*' I do spend serious time pondering the answers to these questions, because I just cannot imagine publishing something that is of no interest to anyone or, even worse, getting stuck having to do something that I am no longer interested in doing.

This, of course, is the reason that TRSTimes only accepts calendar-year subscriptions. It gives me a way out at the end of a year, and it insures subscribers that they will get everything that they paid for.

I have been burned more than once, subscribing to computer magazines that didn't last. My Computronics subscription was transferred to 'Basic Computing', extending my existing 'BC' subscription by many months. Wouldn't you know it, shortly thereafter Basic Computing went out of business. I may have pointed this out previously, but in what turned out to be their last issue, 3-year subscriptions were offered at reduced rates. Such a bargain!

So, I asked myself the above questions again this year and, while the subscription base certainly is shrinking, the bottom line is that I think there is still enough interest from TRS-80 owners, and my enthusiasm is still high. In other words, TRSTimes will continue with 6 bi-monthly issues in 1994. The price will remain at the 1993 level for both the domestic and overseas subscriptions, hoping and praying that the current bunch in Washington are so busy raising taxes on everything else that they will forget to raise the postal rates.

Anyway, don't you forget to subscribe. And please do it as early as possible so we can get the bookkeeping chores out of the way. Thanks.

I would like to take this opportunity to extend my heartfelt thanks to the folks who made this issue possible.

Roy Beck, a good friend, always has something interesting to say. This time it is about the motors associated with your computing.

Chris Fara from Microdex is one heck of a programmer. His latest offering, the Screen Robot for Model III, is very innovative. It's the kind of thing where you say, 'Gee, I wish I'd thought of that!'

Danny Myers tackles the last chapter of Zork. I am impressed - not only with the solution, but with the game itself. I look forward to the time when I can sit down and play through all three chapters in their entirety.

Gary Shanafelt asks the musical question, 'Is the Gate Array Model 4 really better?', and he certainly makes a good case for his point of view. Anyone with an opposing view is invited to submit an article.

Charles Harris, Kelly Bates, Chris Fara and yours truly put together some short goodies, which we hope you will enjoy.

The Model I seems to be reborn. It certainly has been talked about at every user group meeting that I attend. The Model I is now being emulated on MS-DOS machines and it is incredible to see a PC running 'Dancing Demon' or some of the other classics. I have dusted off my Model I collection, transferred a fair share of it to the emulator format, and am having the time of my life writing Z-80 code on the Intel-based machine. My '1 to 4 to PC' article presents utilities for Model I/III and 4 to copy Model I disks over to the virtual disk format that the emulator uses. I hope you find them useful.

Before closing Little Orphan Eighty for this time, I would like to bring you up to date on my weight status. As I mentioned in the last issue, my no-eat diet plan had taken me from 230 lbs down to 201 lbs. Well, I looked through some old scrapbooks from my soccer-playing days and I found a game program which listed the player profiles. Under my picture, the profile read: age 20, 185 lbs. Believe me, that brought back some memories and, then and there, I decided to go all the way down to 185. Just a little more time, coupled with a little more willpower! I had hoped to reach my goal by birthday, but I didn't make it until two days later. Still, not too shabby! I am mighty proud of it - I now weigh the same as when I was 20 - and I haven't started smoking again.

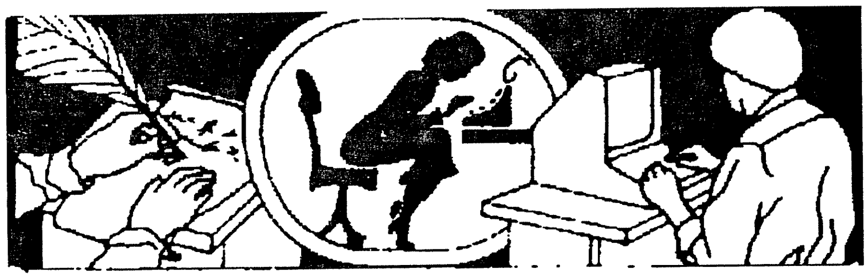
And now... **Welcome to TRSTimes 6.6**

TRSTimes magazine

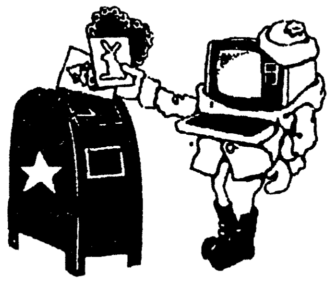
Volume 6. No. 6 - Nov/Dec 1993

PUBLISHER-EDITOR
Lance Wolstrup
CONTRIBUTING EDITORS
Roy Beck
Dr. Allen Jacobs
Dr. Michael W. Ecker
TECHNICAL ASSISTANCE
San Gabriel Tandy Users Group
Valley TRS-80 Users Group
Valley Hackers' TRS-80 Users Group

TRSTimes magazine is published bi-monthly by TRSTimes Publications. 5721 Topanga Canyon Blvd, Suite 4 Woodland Hills, CA. 91364. (818) 716-7154.
Publication months are January, March, May, July, September and November.
Entire contents [c] copyright 1993 by TRSTimes publications.
No part of this publication may be reprinted or reproduced by any means without the prior written permission from the publishers.
All programs are published for personal use only. All rights reserved.
1993 subscription rates (6 issues):
UNITED STATES & CANADA: $20.00 (U.S. currency)
EUROPE, CENTRAL & SOUTH AMERICA: $24.00 for surface mail or $31.00 for air mail. (U.S. currency only)
ASIA, AUSTRALIA & NEW ZEALAND: $26.00 for surface mail or $34.00 for air mail. (U.S. currency only)
Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible, a disk with the material saved in ASCII format. Any disk format is acceptable, but please note on label which format is used.



THE MAIL ROOM



TRS-80 MODEL I EMULATOR

Thank you very much for your registration fee and copy of TRSTimes, received august 31st. It was quite nice to see such an enthusiastic review of my program.

To answer a few questions, first off, the difficulties with FORMAT may have a variety of causes. With TRSDOS 2.3, I believe the error you get is "DISK UNUSABLE, HARD SECTORED!" The problem in that case is that the FORMAT program times the pulse interval from the index hole sensor on the disk. The timing of this pulse is difficult to match, as emulator timing in no way resembles that of the real machine. In any case, the pulse is too fast, resulting in TRSDOS believing the disk is hard sectored. (Disk timer programs have a fit trying to time the virtual drives.) Unfortunately, the best solution I can currently provide *within* the the emulatoris to change the character at file offset 2D5H in TRSDOS's FORMAT/CMD from 19H to 37H. This will disable the disk timing check.

As a better alternative however, you will find on the enclosed disk a series of programs called the "Virtual Disk Utilities" (subdirectory "VIRUTIL"). They are an "unadvertised" part of the package with registration. Among these is a program called VFORMAT. This command will, from the MS-DOS prompt, create virtual disks of any track count from 18 to 80 (default 35). I'd recommend this as a preferred format method for virtual disks even with DOSes that do not encounter difficulties, as it is much faster than format from within the emulator.

As for the trouble with LDOS and MULTIDOS, this sounds to be attributable to the treatment of the directory track. As the format structute (data address marks, etc.) is not retained in the .DSK files, the emulator assumes that sectors on track 17 have "read protect" status, while the others are normal. If I remember correctly, these DOSes may write the directory track with a different data address mark that "read protect" (perhaps the same code as the Model III), though I don't have copies of them myself to verify this.

I'm waiting to see how the response is to this package, but time permitting, I may in the future be writing a more complete disk emulation which could handle non-standard formats (including most copy protection schemes). This may be combined into a new emulator capable of behaving as both a Model I and a Model III. (Currently, I'm putting a TRS-80 Colour Computer II emulator through its paces.)

The "track <9" message of MULTIDOS is likely a product of a bug that has been fixed in the registered version. (The RLD and RRD Z-80 opcodes were interpreted incorrectly in version 2.05 of the emulator.)

Reading your article, I'd like to suggest another method of disk transfer that you may prefer. By patching the BOOT/SYS directory entry, offset 14H on sector 2 of the directory track from 05 00 00 00 FF FF FF FF to 5E 01 00 1F 101F 20 05, BOOT/SYS becomes a 350-sector file spanning the entire disk. The file can then be transferred to an MS-DOS disk using the same method as the file created by your GETDISK/CMD program. This was in fact the method I initially used. The parallel-port cable was developed afterwards as a more universal approach, not requiring double density or the ability to patch disks.

Incidentally, the registered version of the emulator undergoes fairly constant upgrading, and currently includes several features not listed in the documentation of 2.05. You'll find these in the file README.REG on the disk enclosed. Registered users are notified of updates, and so you don't have to worry about registering "too soon".

Two things that I should point out about registration though: The address in the documentation you received is missing the line "Vancouver, B.C." before "Canada V6T 1Z9". It will still reach me, but it may take an extra week or so. Secondly, as specified in MODEL1.DOC, you should either send me a disk to put the program on, or an Internet-accessible e-mail address to whick it can be sent. This was not clear in your article.

I'd just like to say again that I was very grateful for your review in TRSTimes. It's encouraging to know that the program was well-received. I'd really be interested to hear any comments or suggestions that you or your readers may have.

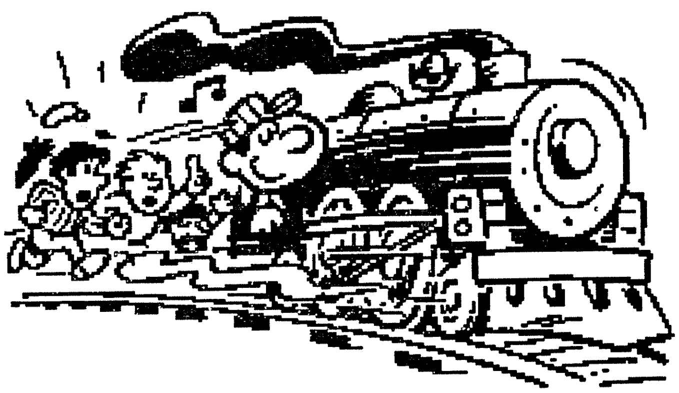
Jeff Vavasour
c/o Department of Physics
University of British Columbia
6224 Agricultural Road
Vancouver, B.C.
Canada V6T 1Z1

The Model I emulator program is responsible for the most fun I have ever had on my MS-DOS machine. While there are other programs emulating Model III and 4, they do no hold a candle to this one - and the registered version is even better. I recommend any PC-owning TRS-80 fan with a Model I background to get a copy of the shareware version, play with it a while, and then get the registered version from Mr. Vavasour.

Ed.

IS THERE A MOTOR IN YOUR COMPUTER?

by Roy T. Beck



Yes, there probably is a motor in your computer; in fact, it likely has a half dozen or more, if you look carefully. And what kind of motors might you expect to find? DC motors, AC induction motors, synchronous motors, and stepping motors are all possibilities.

DC Motors

DC motors are now a rarity, but they were more popular in times past. You may find them driving cooling fans, but that's about all. The main reason is lack of speed control. Since DC motors have no inherent operating speed, their actual speed depends upon the applied voltage and the shaft load connected to the motor.

AC Induction Motors

AC induction motors have a top speed determined by their design. This is their "synchronous" speed. Their actual operating speed is always a little bit less than the synchronous speed, due to a phenomenon known as "slip". At no load, an induction motor operates very near to its synchronous speed, but as load is increased, the shaft speed slows a little. The usual full load speed is about 4% less than the synchronous speed.

Synchronous Motors

Synchronous motors come in a variety of forms, but the only one likely to be found in home equipment is the synchronous reluctance type, which is only built in small sizes. Its two most popular applications are in AC powered electric clocks and in small electronic devices such as disk record players and audio and video tape machines. Some 8" disk drives also used them.

Stepper Motors

Stepping motors are a cross between a boy scout's magnetic compass needle and yet another form of synchronous motor. A stepper motor usually has a three phase winding on its stator (the stationary part), and no windings on its rotor (the rotating part). The rotor sometimes incorporates a permanent magnet to improve its power capability, but usually consists of specially shaped metal stampings, which appear to have "teeth" around the perimeter. Because magnetic flux prefers to flow in iron rather than in air, the teeth create preferred flux paths. When external power is applied to a stator winding, it causes a magnetic field to appear in the stator, and also forces magnetic flux to jump across the air space between the stator and rotor and flow in the rotor.

Because of the preference to flow in iron, the magnetic flux creates a torque in the motor which tends to align the rotor teeth with the flux coming from the stator. This torque will bring the rotor to a predetermined position relative to the stator. As the magnetic flux in the stator is moved around the stator by energizing successive windings, the rotor will also move to maintain its preferred flux path relative to the stator. This is "stepping", and gives the motor its name.

If the stator flux remains stationary (by applying DC power to one of the windings), the rotor will remain stationary. If the DC power is successively applied (slowly or rapidly) to the other windings of the stator, the rotor will step or turn (slowly or rapidly) to follow the applied voltages. It "steps" to follow the flux in the stator winding, hence the name.

Some of the uses of motors in computers includes cooling fans, floppy drive spindle and head stepping motors, hard drive spindle and head positioning motors, and printer line feed and head stepping motors. There may be a few other specialized applications, such as a motor in a scanning device to move the optical system. Laser printers have both cooling fans and paper drive motors. Big commercial tape drives have a synchronous motor to turn the capstan which moves the tape past the heads, and some DC torque motors which keep forward tension on the takeup reel and back tension on the supply reel.

Printer Motors

All printers before lasers usually had two motors, one to advance the paper and one to move the head back and forth. Some very large, high production office type printers also had a cooling fan.

Looking at my trusty old EPSON MX-80F/T, which has many reams of paper and years on it, I see the typical two motors. One serves only to advance the paper, the other to drive the head back and forth. Since the stepper motor can step in either direction depending upon the sequence in which the windings are energized, the paper can be stepped up or down. My old printer was only designed to advance the paper, so that's all its paper feed motor is permitted to do. Many printers can feed the paper either way.

The print head motor, however, can and must drive in both directions, due to the fact that it is a bidirectional printer; that is, it can print in either direction, which speeds its operation.

An important point in a printer is that the printhead position should be known to the computer logic at all times. Since all printers have at least 80 print position, and sometimes much more, how does the printer logic know where the head is? If the stepper motor is randomly energized, as at startup, the print head might be at any position across the page, and the motor can only lock in where it is at the moment. To solve this, the startup logic continually steps the head towards the left until a zero position flag informs the logic that the head has reached the extreme left printing position. This zero detector is usually a small metal flag on the print head carriage which intercepts the light path between and LED and a photo diode. Once the head has reached the left column zero, the logic knows where it is and can step the head around as required, accurately placing the head at the desired column.

All the above implies the stepper motor only has 80 stable positions, corresponding to the 80 print columns. Can life really be this simple? No way!

If only 80 columns were used, with 80 stable positions, several bad effects would arise. First is hysteresis. Anytime you send a mechanical part to a position to be held by magnetic forces alone, there is some possibility of friction causing the part to come to rest at slightly different positions. Especially is this true if the part is to approach from one direction one time and the other the next time. And this is precisely what we are asking a bidirectional printer to do! The result could be that the characters in a vertical row on a piece of paper would "snake", or wiggle back and forth down a column. A second bad effect is that the printer would be limited to only one pitch, say 10 characters per inch, with no possibility of compression or smaller pitches. To overcome these effects, the actual head movement due to one step, which is the least possible movement from stable position to stable position in a stepper motor, is designed to be a very small fraction of an inch. In my EPSON, I believe this is 1/72", and is determined by the gear ratios in the drive train between the stepper motor and the platen and also by the number of magnetic poles wound into the stepper motor. Since 1/72" is about 0.014", and if hysteresis is held to about 5%, then the error from column to column would be held to 0.0007", which is probably less than the eye would notice on a sheet of paper.

Standard vertical pitches are 6 and 8 lines per inch. 6 into 72 is 12. Thus, 12 steps will advance the paper 12/72", the correct amount for 6 lines per inch. If the motor is only stepped 9 times per line, then the printer will operate at 8 lines per inch. For graphics purposes, the paper can be advanced as little as 1/72" at a time. This gives vertical position resolution of 72 dots per inch (dpi). Because the head has 9 printing pins, the actual vertical resolution is much better.

Similarly, the head motion of my EPSON is in steps of 1/216" per step, which is only 1/3 of the least motion of the paper feed system. This is 216 dpi.

I believe the vertical feeds of 24 pin printers is about the same as my MX-80, but in order to achieve greater dpi horizontally, they probably use smaller steps than the 216/inch mine is capable of. Someone who really knows 24 pin printers ought to write an article about their capabilities. (Hint, hint).

Floppy Drives

Turning to floppy drives, motors are used to turn the spindles and to position the heads. The early floppies used a belt drive system.

Some of the 8" drives used a hysteresis synchronous motor to turn the disk at a precise speed; This was controlled by the electric utility, and in most places is a very constant number, usually either 50 or 60 hertz. For whatever reason, other 8" drives were fed 24 volt DC power, which immediately implies either a DC motor of uncertain speed or a stepping motor system.

All 5.25" drives use 12 volts DC to operate the motor system. It is my belief that all current drives use a stepper motor scheme with a servo feedback to insure accurate rotational speeds. Most 5.25" drives rotate at 300 RPM, but the 1.2 Meg unit turns at 360 RPM when operating in the high density mode, and 300 RPM when operating with medium density disks. The stepper motor scheme is used here to drive the spindle. Since a stepper motor needs no brushes, it is a durable, long-lived motor. Stepping and speed control logic have become so sophisticated and with LSI require so few chips, that spindle motors now are a form of a stepping motor. Since the motors have been turned "inside out", with a large diameter, symmetrical rotor mounted outside of the wound stator, it has become essential to operate the motors as induction motors. As speed varies with load, due to changing friction in the floppy envelopes, etc, the stepping speed is continually adjusted by logic circuitry to maintain the desired spindle speed.

All present floppy drives use stepper motors to move the heads in and out on the disks. Again, a scheme to find track zero is employed. Every drive has a logical detector to determine when the head is at track 0, the outermost track. This detector must be accurately calibrated, because the proper operation of the drive depends upon the head being accurately positioned to track 0 when FORMATting, and also when intending to read and write files. The floppy driver program knows to locate the head by stepping it out until the track 0 flag comes up, analogously to the way the printer determines where the leftmost column is located.

Stepper motors are universally used in this application. Since the floppy drivers are all written to expect one step pulse to move the head by one track, then 40 track and 80 track drives must have some means to accommodate the fact that the tracks are spaced 36 per inch on 40 track drives and 72 tracks per inch on 80 track drives. This is handled in the "gear ratio" of the mechanism, established when the drives are designed. Of course some drivers can read a 40 track disk in an 80 track drive, but this simply requires two step pulses per track, which is handled in software.

Hard Drives

Hard drives have the same two needs for movement, spindle rotation and head positioning. Spindle motors usually rotate at 3600 RPM, but a stepper motor is used to allow accurate speed control. Except for the difference in speed, the hard drive and the floppy spindle motors are quite similar.

Head motion in hard drives was originally controlled the same as in a floppy; that is, a stepper motor and a track 0 detector were used. This worked adequately well until the total number of tracks went beyond the practical limits of the stepper system. As more tracks were required, and the track spacing became desperately small, thermal problems became unwieldy.

Consider that the platters in a hard drive expand radially as they warm up. As the track spacing became smaller, the required accuracy of positioning of the heads increases. The problem is that the center of the 0 track actually moves outward from where it is when the drive is cold to where it is when the drive has reached full operating temperature. With a 153 track drive having track spacings of about 0.008", center to center, a radial change of 0.001" might be easily accommodated. But with track counts exceeding 1000, the center to center spacing may be anywhere from 0.001 down to perhaps 0.0005". Now a warmup position change of 0.001" would be a logical disaster. What to do?

The answer was to get away from the old method of a fixed track 0 detector. The head positioning system could no longer be allowed to be positioned in steps, and so a system having a magnetic plunger in a solenoid was devised. There is a spring to return the heads to a known mechanical position, and the current in the solenoid is adjusted by transistors to pull the plunger and its attached heads to the desired track. One head and one whole platter side were dedicated to compensating for thermal and hysteresis errors. This is why large drives now are listed with an odd number of heads. The extra head and platter side are there, alright, just not advertised. The trick is that the factory uses the extra head to record a special magnetic signal on each track under carefully controlled conditions at the factory. Normally, these tracks are never altered by the user, and the internal drive logic continually refers to this head and its associated tracks to determine where it is, and to continually reposition the head system to keep all the heads on the centers of the tracks being read or written.

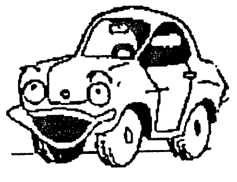
This whole scheme is referred to as Voice Coil positioning, because is is analogous to the magnetic structure which moves the cone in a loudspeaker, and the driving coil there is traditionally known as the voice coil.

Obviously the voice coil system is not much like an electric motor, and yet when you get to fundamentals, both the stepper motors and the voice coils accomplish the same thing. However, voice coils are not normally thought of as motors, so in this usage, hard drives now only have spindle motors.

Incidentally, it is advisable to allow a hard drive to warm up before FORMATting it. The reason is to allow it to expand to its operating condition before applying the format, which will slightly improve its long term data reliability. 30 minutes is recommended. This is just a passing thought, but there is some merit to it.

Conclusion

As you can see, there are quite a number of motors in your computer, all of which are important. Now you know a little more about that expensive toy you (we) love so much.



PROGRAMMING TIDBITS

Copyright 1993 by Chris Fara (Microdex Corp)

SCREEN ROBOT III

Why couldn't a computer automatically write programs for us? It has been attempted, but so far with more hype than meat. Some day, maybe. Meanwhile, instead of wasting money on dubious promises, we might try something simple yet useful on our own. For instance, one tedious programming chore is the layout of screens: menus, help screens, data entry forms, and similar displays. So here is an idea that might help a bit. It will instantly produce a file with a screen display subroutine that later can be simply MERGEd into any BASIC program. Just so you Mod-III people know we care, we'll start with a Mod-III version (Mod-4 later).

SCREEN EDITOR

First we need some sort of "editor" to lay out the screen. We'll sketch here only its simplest skeleton. If you like the idea, you can make it more fancy later.

```
10 clear 5000: cls: defint a-z
11 x=0:y=0:xx=64:yy=16'video size
12 d$=chr$(9)+chr$(8)+chr$(10)+chr$(91)
13 d$=d$+chr$(13): q$=chr$(34)
20 print @ y*xx+x, chr$(14);'cursor
30 k$=inkey$: if k$="" then 30
31 print chr$(15);'cursor off
32 on instr(d$,k$) goto 41,42,43,44,50
33 if k$>=" " then print k$;:else 20
41 x=x - (x+1 < xx): goto 20'right
42 x=x + (x > 0)  : goto 20'left
43 y=y - (y+1 < yy): goto 20'down
44 y=y + (y > 0)  : goto 20'up
50 '-----save
```

After the usual start-up chores (clear string space, etc) define variables: X and Y are cursor coordinates, XX and YY are screen limits, D screen rows

```
55 poke varptr(a$)+1, y*64 and 255
56 poke varptr(a$)+2, &H3C + fix(y/4)
```

We name the file SCREEN/SUB but you can change it to anything you want. Similarly, the lines of our subroutine will be 10000 and up, but could be anything convenient. The first line will simply clear the screen. If for some reason you don't want to clear the screen every time the subroutine is called, then omit the CLS, but make sure that we have a line 10000 (or whatever number you choose), perhaps with some REMaroodes match then the program "falls thru" to line 33: if we have a displayable character (blank space or higher) then display it. Otherwise go back for another key.

After displaying the character (semi-colon prevents cursor from dropping to the next screen row) our program goes to next line which also happens to be handling the right arrow. Even though PRINT already moved the cursor to the right, we need to calculate the new X-coordinate here: it will increase by 1, but only if the cursor is not yet atthe right edge of the screen. In BASIC a comparison such as..... A < B generates -1 if true, 0 if false. So, if adding one will result in a column number X lower than XX (ie. 63 or less) then the cursor will advance by 1 step, because..... X - (X+1 < XX) = X - (-1) = X + 1

Otherwise the comparison is "false" and leaves the cursor position unchanged...
X - (X+1 < XX) = X - (0) = X

In a similar way coordinates are calculated in lines 42-44 for other arrows. Then go back to line 20, turn on the cursor at the calculated position and wait for a key.

Even with this rudimentary "editor" we can now easily scribble all over the screen: use arrows to move around, type something, change, erase with SPACE bar, and so on. More code can be added to copy and move text, etc, but that's not the point here.

PROGRAM WRITER

When the screen looks right, press ENTER to start generating a BASIC program file. Three steps are involved: get the screen rows from memory, trim all blanks from left and right sides of each row, and construct BASIC command lines for output to file. The program lines in that file will have this "syntax".....

```
PRINT @ position,"text";
```

The semicolon at the end comes in handy when something is displayed in the bottom line: it prevents scrolling of text (as long as no character is in the rightmost corner of the screen). Let's try it.

```
50 '------save screen
51 open "O", 1, "SCREEN/SUB"
52 print #1, "10000 CLS"
53 a$ = string$(64,32)
54 for y=0 to yy-1'get screen rows
55 poke varptr(a$)+1, y*64 and 255
56 poke varptr(a$)+2, &H3C + fix(y/4)
```

We name the file SCREEN/SUB but you can change it to anything you want. Similarly, the lines of our subroutine will be 10000 and up, but could be anything convenient. The first line will simply clear the screen. If for some reason you don't want to clear the screen every time the subroutine is called, then omit the CLS, but make sure that we have a line 10000 (or whatever number you choose), perhaps with some REMar useless blanks from the resulting subroutine file.

The next, final segment of our "robot" program writes a line to the disk. For educational purposes the PRINT#1 command has been split here into three pieces, but it could be just as well written in one line. In any case on the disk it will create one BASIC command line.

```
70 print #1,10000+1+y;'write
71 print #1, "PRINT @" y*xx+j-1 ",";
72 print #1, q$ mid$(a$,j,k) q$ ";"
80 next'go get next row
81 print #1,"10029 RETURN"
82 close: end
```

cond row #1 is stored at hex'3C00 plus 64, and so on. Our POKEs calculate the low and high byte of those addresses for each Y=row number. When Y*64 gets to be 256 or more (after row #3), the AND will mask out the excess of 256. Thus for the row #4 we'll poke 0 (same as for the top row), for the row #5 we'll poke 64, and so on. At the same time the high byte poke will be incremented for every fourth row whenever we pass the 256-byte boundary. If you're not comfortable with such "binary" capers then just take my word for it: it is so.

Anyway, after the two POKEs the picture of a screen row sits in the string A$. But we don't want to clutter our subroutine file with useless blank spaces, so the next task is to trim all leading and trailing blanks. And, if in the process we discover that an entire row is blank, then we'll ignore it altogether.

```
60 j=xx+1'trim string
61 for x=1 to xx'leading blanks
62 if mid$(a$,x,1)>" "then i=x:x=j:j=i
63 next: if j>xx then 80
65 for x=xx to 1 step -1'trailing
66 if mid$(a$,x,1)>" "then k=x:x=1
67 next: k=k-j+1
```

First scan the string from the beginning. When a non-blank character is found, then the counter variable X is swapped with J. Therefore now J contains the position X in the string where the first non-blank was found (the column on the screen where the text begins) while X equals XX+1 and thus forces NEXT in line 63 to terminate the scan.

Now, if all characters were blank then after NEXT the value of J is still XX+1 (it was pre-set in line 60 and never got swapped), J>XX is true, and we skip to line 80 to try the next row (see next program segment below). If the string is not blank then lines 65-67 scan backwards for trailing blanks, and calculate the length K of that portion of A$ which will remain after discarding any leading and trailing blanks.

You will notice that, when the screen is mostly blank, the "trimming" takes a couple of seconds (hundreds of MID$ comparisons are made). But this is time well spent, because it eliminates tons of useless blanks from the resulting subroutine file.

The next, final segment of our "robot" program writes a line to the disk. For educational purposes the PRINT#1 command has been split here into three pieces, but it could be just as well written in one line. In any case on the disk it will create one BASIC command line.

```
70 print #1,10000+1+y;'write
71 print #1, "PRINT @" y*xx+j-1 ",";
72 print #1, q$ mid$(a$,j,k) q$ ";"
80 next'go get next row
81 print #1,"10029 RETURN"
82 close: end
```

First, a line number is constructed by adding the current row number (0-15) to a "base" number (in our case 10000+1, because the "header" line in our subroutine was at 10000). Then the PRINT@ keyword and screen position is added. The position is calculated from row Y and column J where the non-blank part of the screen row begins. We must subtract 1 because J=1 if the string starts at the leftmost column, but in PRINT@ this must be column 0, etc. Finally, write the text of A$ sourrounded by quotes O$, plus semicolon to stop the cursor from dropping down.

Note that the subroutine lines generated by our "robot" are numbered sequentially, but not necessarily consecutively: for example if the top screen row (row 0) is blank then the subroutine will not have line 10001, and so on. That's why it is a good idea to start the subroutine with some "neutral" line that will be always present, as we have done with our line number 10000.

After all rows are done, write the RETURN command and close the file. The file looks just like any BASIC program file saved with the "A" (ASCII) option and can be now included in any program.....

```
MERGE "SCREEN/SUB"
```

Then, to display the screen.....

```
GOSUB 10000
```

Pretty sleek for being so simple, isn't it? Next time we'll look at Mod-4. By the way, both Mod-III and Mod-4 versions of the "screen robot" are available, along with other "goodies" on the... "Goodies" disk from Microdex ($12.95+SH; address and phone number see Microdex ad elsewhere in this issue).

BEAT THE GAME

by Daniel Myers

ZORK III

Well, you've come a long way since you first stood by the mailbox outside the house in the forest. You've defeated the thief, outwitted the Wizard of Frobozz, and now, you stand at the foot of the endless stairs, ready to embark on the final part of your journey. So, pick up the lamp, turn it on, and head along due South until you come to the shore of the lake.

Drop the lamp <say goodbye to it; you won't be seeing it again>, and jump into the lake. Brrrr!!! Pretty cold! So, don't stay in there long; swim west and then go South into the Scenic Vista. Kind of a strange place, with changing numbers on the wall and a bare table...not quite all that scenic, eh? Well, get the torch, and wait for the number to change to "II." Then, touch the table.

My oh my! You're in a room from Zork II....Room 8, as a matter of fact.

However, you don't have much time to sight-see, so get the can of Grue repellant, then try moving East, and you will find yourself back in Scenic Vista again. Now wait for the number to change to "III," then touch the table again. This time, you're in a Damp Passage. Drop the torch, and just wait there until you're pulled back to Scenic Vista.

Okay, you're finished here, so move along North to the shore, and again jump in the lake. Splash! It hasn't gotten any warmer; in fact, you just dropped the can of repellant. So, go Down, and you will be on the lake bottom. Ah, there it is! But, could there be something else there, too? "Get all," and you will have not only the repellant but also an amulet. This is one of those "wonderful" variable things; it may take more than one try on your part to get both items. In the meantime, you can't stay in the icy waters too long, and sooner or later a hungry fish will come looking for you.

Therefore, it's best to save the game before you jump in from the Western Shore. So if you die in the water, or get eaten by the fish, or picked up by the Roc <while you're swimming on the surface>, you don't have to start all the way back at the beginning. By the way, this is the only one of the Zorks where you don't lose points if you die. But, all the items you've collected so far get scattered all around, and it's time-consuming to go look for them.

Okay, now you have the can and the amulet, so head Up to the surface, then South to the Southern shore. You can see a cave to the South, and it looks kind of dark. In fact, it *is* dark in there, which is why you have the repellant. So, spray the smelly stuff on yourself, and go South, and you will find yourself in a Dark Place. Go South again, then East, and you will be in the key room. Whew! At least there's some light in here! And by the light you can see a strange key. Get the key, then move the manhole cover and go down.

And here you are on an aqueduct. Since you can't go back <the Grue repellant wouldn't have lasted that long>, you might as well go forward. So, just head along North and you will come to the Water Slide. Go North down the slide, and guess where you are? In the Damp Passage! And there's the torch, so pick it up, because you're certainly going to need a light source...especially when you think of where you're going next.

So, from the Damp Passage hike along West to the Junction <you can't get the sword out of the rock, so don't even try>, then South into Creepy Crawl, and Southwest into the Shadow Land. Here we come to another variable portion of the game. You will have to wander around in the Shadow Land until a cloaked and hooded figure appears. When that happens, the sword will suddenly materialize in your hand, and you will be able to fight.

However, since there's no way of telling when that will happen, you just have to keep moving around until it does. At least you will get a chance to practice some elementary map-making! Also, this is the most dangerous part of the game, as the figure is quite capable of killing you, too! So, best to save before you enter Shadow Land.

When the mysterious figure finally appears, attack him with your sword until he is badly wounded and cannot defend himself. At that point, get his hood. The figure will then disappear, leaving the cloak behind. Get that also.

Now, you have to get out of here, and I can't tell you exactly how, since there's no way of knowing exactly where you were when the fight started.

However, if you go Eastwards, you will exit the Shadow Land at either the Creepy Crawl or the Foggy Room. From either place, go North to the Junction.

From the Junction, it's West through the Barren Area, and West again to the Cliff. Bet you just can't wait to climb down the rope, huh? Well, pick up the bread first, then go down to the ledge. Well, well, a chest! Too bad you don't have a key to open it. In fact, there's no way for you to open it at all. But don't despair, there's a way of doing it. Just wait around and someone will come along the top of the cliff. You may not really trust him, but tie the rope to the chest when he asks, and wait around some more. Eventually, he will return and help you back up the cliff. He will also give you a staff, which is what you're really after here. Take the staff, then go back down to the ledge, and from there, to the Cliff Base.

Now trek South to the Flathead Ocean, and do a little more waiting. Sooner or later a ship will come floating by. As soon as you see it, say: "Hello, Sailor." The man in the ship will throw something onto the beach for you. Take a look, and you will see it's a vial. It'll come in handy later, so pick it up. Now comes the fun part: You have to wait for the earthquake <notice how you've been doing a lot of waiting around? I hope you're a patient person!>. While you're waiting, you might want to wander around a little, although you've been to most of the accessible places by now. In any case, wherever you are, once the earthquake hits, make your way to the Creepy Crawl, and from there East into the Tight Squeeze, then East again into the Crystal Grotto. Then all the way South to the Great Door, and East into the Museum Entrance.

Now, open the East door, then go North into the Museum. Look at the gold machine <it's a time machine, in case you were wondering>, then set the dial to 776. Here comes the fun part: Push the machine South into the Entrance, then East into the Jewel Room. Get into the machine, and push the button. Aha! Now you're back in 776 GUE, but the time machine seems to have vanished! No matter, wait for the guards to leave, then get the ring <and *only* the ring!>, then open the door, go out into the Entrance, open the North door and go North.

By golly, the machine is right there! Put the ring under the seat, turn the dial to 948, get in, and push the button. Whew, you're back in the right time period again. Get out of the machine, look under the seat <you will get the ring automatically when you do this>, then back South, and South again, to the Royal Puzzle.

Okay folks, you are about to enter the absolute nastiest part of the game. You must follow the instructions *EXACTLY* as given, or you will never get out. And, since it would be easy to make a mistake here, I strongly recommend you save the game.

1. Go Down the hole, then push the South wall. Then go East, South, East, East. Push the South wall, get the book, and push the South wall again.

2. Push the West wall twice. Then go East, South, and push the East wall.

3. Now, go straight North until you come to the marble wall, and push the East wall.

4. Now, go West, South, South, South, South, East, East, North, North, North, and push the West wall.

5. From there, go East, South, South, South, West, West, West, West, North, North, North, West, North. Push the East wall three times.

6. Now, West, West, South, South, East, East, South, and push the East wall.

7. Okay, now West, West, West, North, North, North, East, East, and push the South wall two times.

8. From there, West, South, South, East, East, North, and push the West wall two times.

9. Now, South, West, and push the North wall until it won't move any more.

10. Then West and North. Finally! You have maneuvered the ladder under the hole <which was the purpose of all this pushing and running around>, and now you can just go up and out! WHEW!!!!

Okay, you've solved the Royal Puzzle and you have the book, so go North to the Museum Entrance, then open the East door and get your other stuff from the Jewel Room. Then it's back West to the Great Door, and from there back to the Junction. Now, East into the Damp Passage, and NE to the Engravings Room.

Well, we have here yet another <!> of those variable events: Sooner or later, an old man will be sleeping here. If he isn't there the first time you arrive, walk around a little and return. When you finally do see him, wake him up and give him the bread. He will eat it and then make visible to you a secret door. He will then vanish.

Okay, you're getting closer to the end! Open the door, and go into the Button Room, then North to the Beam Room. Put the sword in the beam, then

go back to the Button Room and push the button. Now, back North to the Beam Room and North again into the Mirror Room. There will be an opening in the Mirror, so go North one more time, and you will be inside.

Now, don't let the long and complicated descriptions scare you! It's not really as bad as you think <it's worse! heheheheh..just kidding!>. First, raise the short pole. Then, push the white panel twice. Now, push the pine panel, and go North. Okay, so here you are, standing a little too close for comfort to the Guardians of Zork. If I were you, I wouldn't try going past them quite yet! Open the vial, then drink the liquid. While nothing seems to have happened, you have in fact become invisible. Now you can walk North until you come to the locked door. Knock on the door, and the Dungeon Master will open it and let you in.

All right, hang in there, you have reached the end game! Go North, then West, then North again. The DM will be following you. Go North to the Parapet, set the dial to 4, and push the button. Now, go South, open the cell door, and step inside. The DM will not follow you in.

Once inside, you will notice a bronze door in one of the walls. However, you can't open it yet! Something else has to be done. And it will have to be done by someone else. So, first tell the DM to go to the Parapet. Then tell him to turn the dial to 1, and then tell him to push the button.

All right!! The magic moment has arrived! Unlock the bronze door with the key, open the door, and go South!

***** TA DA!! *****

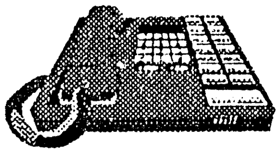
Finally, Zork is finished! You have survived all the perils, pitfalls, and puzzles, and now, *YOU* are the new Dungeon Master. Have fun!

TRSuretrove BBS

8 N 1 - 24 hours

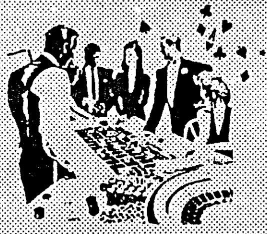
Los Angeles

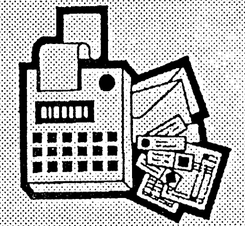
213 664-5056



where the TRS-80 crowd meets

RECREATIONAL & EDUCATIONAL COMPUTING





REC is the only publication devoted to the playful interaction of computers and 'mathemagic' - from digital delights to strange attractors, from special number classes to computer graphics and fractals. Edited and published by computer columnist and math professor Dr. Michael W. Ecker, REC features programs, challenges, puzzles, program teasers, art, editorial, humor, and much more, all laser printed. REC supports many computer brands as it has done since inception Jan. 1986. Back issues are available.
To subscribe for one year of 8 issues, send $27 US or $36 outside North America to REC, Att: Dr. M. Ecker, 909 Violet Terrace, Clarks Summit, PA 18411, USA or send $10 ($13 non-US) for 3 sample issues, creditable.

*** DR. PATCH ***

BRAND-NEW UTILITY FOR TRS-80 MODEL 4 AND LS-DOS 6.3.1

A 'MUST HAVE' FOR ALL LS-DOS 6.3.1 OWNERS.

DR. PATCH MODIFIES LS-DOS 6.3.1 TO DO THINGS THAT WERE NEVER BEFORE POSSIBLE. COMPLETELY SELF-CONTAINED - MENU-DRIVEN FOR MAXIMUM USER CONVENIENCE. FAST & SAFE - EACH MODIFICATION IS EASILY REVERSED TO NORMAL DOS OPERATION.

DISABLE PASSWORD CHECK IN FORMAT/CMD
FORMAT DOUBLE-SIDED AS DEFAULT
FORMAT 80 TRACKS AS DEFAULT
DISABLE VERIFY AFTER FORMAT
CHANGE 'DIR' TO 'D'
CHANGE 'CAT' TO 'C'
VIEW DIR/CAT WITH (I) PARAMETER AS DEFAULT
VIEW DIR/CAT WITH (S,I) PARAMETERS AS DEFAULT
CHANGE 'REMOVE' TO 'DEL'
CHANGE 'RENAME' TO 'REN'
CHANGE 'MEMORY' TO 'MEM'
CHANGE 'DEVICE' TO 'DEV'
DISABLE THE BOOT 'DATE' PROMPT
DISABLE THE BOOT 'TIME' PROMPT
DISABLE FILE PASSWORD PROTECTION
ENABLE EXTENDED ERROR MESSAGES

DISABLE PASSWORD CHECK IN BACKUP/CMD
BACKUP WITH (I) PARAMETER AS DEFAULT
BACKUP WITH VERIFY DISABLED
DISABLE BACKUP 'LIMIT' PROTECTION
DISABLE PASSWORD CHECK IN PURGE
PURGE WITH (I) PARAMETER AS DEFAULT
PURGE WITH (S,I) PARAMETERS AS DEFAULT
PURGE WITH (Q=N) PARAMETER AS DEFAULT
IMPLEMENT THE DOS 'KILL' COMMAND
CHANGE DOS PROMPT TO CUSTOM PROMPT
TURN 'AUTO BREAK DISABLE' OFF
TURN 'SYSGEN' MESSAGE OFF
BOOT WITH NON-BLINKING CURSOR
BOOT WITH CUSTOM CURSOR
BOOT WITH CLOCK ON
BOOT WITH FAST KEY-REPEAT

DR. PATCH IS THE ONLY PROGRAM OF ITS TYPE EVER WRITTEN FOR THE TRS-80 MODEL 4 AND LS-DOS 6.3.1. IT IS DISTRIBUTED EXCLUSIVELY BY TRSTIMES MAGAZINE ON A STANDARD LS-DOS 6.3.1 DATA DISKETTE, ALONG WITH WRITTEN DOCUMENTATION.

*** DR. PATCH *** $14.95

SHIPPING & HANDLING: U.S & CANADA - NONE
ELSEWHERE - ADD $4.00
(U.S CURRENCY ONLY, PLEASE)

TRSTimes magazine - dept. DP
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

HINTS & TIPS

USING MY TRS-80'S
by Charles Harris, MD

Having just renewed my subscription to TRSTimes, I may have missed comments about jet printers. But just in case -- the Epson Stylus 800 does a remarkable job on printer-ready material. It is not as fast as a laser printer, and it has fewer fonts, but it is inexpensive, and downward compatible with the FX, RX and MX series of Epson printers. Thus compatible with all TRS-80 computers. No fixes or gimmicks necessary.

While I have been away, I have used the TRS-80 for word processing (Lazy Writer on Multidos), installed 1 meg of ram (Superdrv ram and Megaram, the latter by Anitek) in several computers, so they move quickly. Can upload several applications side by side and go from there at many more megahertz than believed possible for a TRS-80.

I have used the computer with Profile 4 Plus with Clay Watts wonderful 'look-up' and 'Forms' program in Basic, called PROAID, to complete Medicare forms by the zillion and now have two Tandy's side by side in my office, one for patient addresses, and the other for the illnesses, each downloading to two 80 track disks.

Also write on the marvelous Model 100 laptop, which has text, Basic, and Communications, and can null-modem files up to the Lazy Writer directly, with Lazy Writer then formatting them.

I guess the point of all this is that the little machines are sturdy and utilitarian and, if you have enough of them, are multitasking with none of the exertion needed for MS-DOS, or klutziness of the Mac's (which are very good instruments).

Incidentally, Vern Hester has created a version of Multidos which accesses the Megaram, and possibly has a hard disk driver.

MODEL III KEYBOARDS
by Kelly Bates

Just finished working my Model III keyboard over. Had a few keys that did not function all the time.

Built an adapter cable that now has one connector that comes out of the Mod III just below the right shift key. I just tuck it under the case when not in use. The adapter cable allows me to work on the keyboard outside the Mod III (and also anybody else's that has a problem).

First the adapter - flat ribbon 20 conductor - mount a 3M 3326 or equivalent on one end, then about 1.25 inches up mount a 3M mate to the 3326 (this one does not show a number, but it is a 20 pin socket connector). About 8 inches up from that connector you should build some sort of strain relief (I used a plastic tube with tie wraps to secure it). This portion of the cable built so far remains inside the computer. About 2 inches from the strain relief put another mate to the 3326. This connector allows you to plug any other Mod III keyboard into it, and work on it if necessary. It also lets you extend your keyboard so that, with the computer assembled, you can work on it outside (observe polarity and pin #s).

So, what you have is two 20 pin socket connectors and one 20 ,pin connector with about 12 inches of flat 20 conductor ribbon cable. This adapter will also let you hook up a joy stick if you are so inclined - that was my first use of it back in '86 or so.

The keyboard is a mass of solder connections, and you must remove the key assembly to work on it. Get some solder (brass ground strap) and remove the solder from the key you want to work on. Flip the board over, release the 'catches' (one on each side) and remove the key.

The key assembly also has 2 'catches'. Release them and pull the key apart - CAREFULLY! There is a small spring that can easily get away from you and is very difficult to find on a shag rug.

There are 2 fingers on one of the parts that almost contact the spring when the key is assembled. Bend those 2 fingers slightly more in the same direction, and then reassemble the key.

At this point I check to be sure the key is working with a battery and LED indicator.

Now, put the key assembly back into the keyboard and resolder the connections. The BREAK key is the hardest to verify operation on when the computer is running.

Sometimes it is hard to keep track of which keys you want to work on if more than one, so I always pull the caps (pull them straight up and they pop right off).

To figure all this out, I unsoldered all my keys the first time. Once I knew what it was all about, then it was easy to arrive at this fix. It appears that you cannot use freon to make these work properly. They must be 'resprung' to operate properly, or replaced if you have another assembly of spare parts.

PATCHER4 UPDATE
by Chris Fara

Another tidbit.... It seems that perhaps in PATCHER4/BAS we could simplify line 80...

```
80 F1=VARPTR(#2):RETURN
```

This form of VARPTR returns directly the address of the file buffer (sez "Mod-4 by Chris" on page 4-72). In that case we could also omit line 255 FIELD.... because apparently F1$ is only needed to find the buffer address in line 80. I didn't have the chance yet to actually try it with this modification, but I can't see why it shouldn't work.

MODEL I/III OR 4
by Lance Wolstrup

I was in the process of writing a program for Model I and III when I ran into a problem. Writing in assember, I naturally used all the available DOS routines, however, two of the routines employed a different address on the Model I than on the Model III. I needed to find a way to make the program figure out on which machine it was running. After a few hours research I came up with the answer. It was published in the February 1981 issue of the 'TRS-80 MICROCOMPUTER NEWS' in a short article called 'What Kind of Machine Is It?'.

The solution, it seems, is found in memory location 293. If this location contains the value 73, the machine is a Model III. Any value other than 73 makes the machine a Model I.

From Basic the code would look something like this:

```
IF PEEK(293)=73 THEN A$="Model III" ELSE
A$="Model I"
```

The code might look like this in Assembly Language:

```
        LD      A,(293)         ;get value in 293
        CP      73              ;is it 73?
        JR      Z,MOD3          ;jump if 73
        ;model 1
        ;code
        ;goes
        ;here
        JR      COMMON          ;skip Model III code
MOD3    ;model 3
        ;code
        ;goes
        ;here
COMMON  ;the program continues with
        ;code common to both Mod I & III
```

Over the years Radio Shack has taken a lot of flack for not giving us, the users, enough information about our machines. In a few cases this criticism was justified -- but mostly they gave us the information when they learned it. I suspect that in the beginning (and for some time thereafter) the good folks at Tandy were as much in the dark about the machines as we were. They were at the mercy of people like Randy Cook, who appears to have been very protective of his code.

Well, we learned how to tell a Model I from a Model III, but how does a program know if it is running on a Model 4? Easy! Memory location 125 (7DH) will contain the value 4 if it is a Model 4. Thus, the program can now test for the value 4 in location 125. If 4 is found, we have a Model 4, otherwise we go on to test memory location 293. If we find 73 there, we have a Model III. If any other value is found, the program is running on a Model I.

That's all there's to it.

YES, OF COURSE !

WE VERY MUCH DO TRS-80 !

MICRODEX CORPORATION

SOFTWARE

CLAN-4 Mod-4 Genealogy archive & charting *$69.95*
Quick and easy editing of family data. Print elegant graphic ancestor and descendant charts on dot-matrix and laser printers. *True Mod-4 mode*, fast 100% machine language. Includes 36-page manual. *NEW!*

XCLAN3 converts Mod-3 Clan files for Clan-4 *$29.95*

DIRECT from CHRIS Mod-4 menu system *$29.95*
Replaces DOS-Ready prompt. Design your own menus with an easy full-screen editor. Assign any command to any single keystroke. Up to 36 menus can instantly call each other. Auto-boot, screen blanking, more.

xT.CAD Mod-4 Computer Drafting *$95.00*
The famous general purpose precision scaled drafting program! Surprisingly simple, yet it features CAD functions expected from expensive packages. Supports Radio Shack or MicroLabs hi-res board. Output to pen plotters. *Includes a new driver for laser printers!*

xT.CAD BILL of Materials for xT.CAD *$45.00*
Prints alphabetized listing of parts from xT.CAD drawings. Optional quantity, cost and total calculations.

CASH Bookkeeping system for Mod-4 *$45.00*
Easy to use, ideal for small business, professional or personal use. Journal entries are automatically distributed to user's accounts in a self-balancing ledger.

FREE User Support Included With All Programs !

MICRODEX BOOKSHELF

MOD-4 by CHRIS for TRS/LS-DOS 6.3 *$24.95*
MOD-III by CHRIS for LDOS 5.3 *$24.95*
MOD-III by CHRIS for TRSDOS 1.3 *$24.95*
Beautifully designed owner's manuals completely replace obsolete Tandy and LDOS documentation. Better organized, with more examples, written in plain English, these books are a *must for every TRS-80 user.*

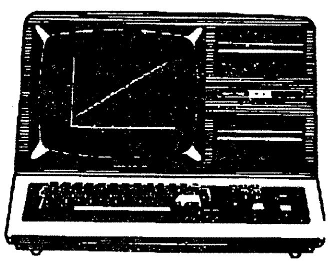
JCL by CHRIS Job Control Language *$7.95*
Surprise, surprise! We've got rid of the jargon and JCL turns out to be simple, easy, useful and fun. Complete tutorial with examples and command reference section.

Z80 Tutor I Fresh look at assembly language *$9.95*
Z80 Tutor II Programming tools, methods *$9.95*
Z80 Tutor III File handling, BCD math, etc. *$9.95*
Z80 Tutor X All Z80 instructions, flags *$12.95*
Common-sense assembly tutorial & reference for novice and expert alike. Over 80 routines. No kidding!

Add S & H. Call or write MICRODEX for details
1212 N. Sawtelle Tucson AZ 85716 602/326-3502

IS THE GATE ARRAY MODEL 4 REALLY BETTER?

by Gary W. Shanafelt



Not long ago, the Fort Worth School District replaced its longtime classroom computers -- over 200 Model 4 TRS-80's. A car repair shop bought the lot, selling off individual machines for $20 if they worked, $10 if they didn't.

I live about three hours from Fort Worth; even though I've already got several Model 4's, I couldn't resist driving in for another, for parts. The sight that greeted me in the storage area of the shop was pretty incredible: white desktop Model 4's piled in rows one on top of the other. There were also boxes of disk drives, printers, and even an old Model I. I was told they were mostly non-gate array models, since the gate array machines had been picked through pretty thoroughly. I found a machine I liked, with no burn-in on the monitor, hauled it off to my car, and got the idea for this article.

I bought a non-gate array Model 4. In fact, all my Model 4's are non-gate array models. To some extent this is accidental, for I once had a gate array as well. I've never been able to figure out why everyone else seems to like the gate-array versions. In fact, the more I've thought about it, the more I've concluded that non-gate array is the way to go.

First of all, if you don't know what I'm talking about, the difference is that the earlier non-gate array Model 4's have a separate mother board, RS232 board, and floppy disk controller board. Newer chips (the gate array technology) allowed the later Model 4's to have everything on one consolidated board, cutting down the chip count and manufacturing expenses.

You can tell the difference on a desktop Model 4 by turning it around and looking for the RS232 port. On the non-gate array model, it's on the bottom with all the other connectors, pointing down. On the gate array model, it's on the back, pointing directly outwards. The reputed advantages of the gate array model seem to be as follows. It is later, so it "must" be better.

It comes with a green screen, not the old black and white one.

The CPU runs at the full 4 mhz clock speed, while the non-gate array version runs somewhat slower (and speedup kits allow the gate array to run faster yet).

It has a keyboard with clustered cursor keys, the standard on the latest computer systems.

It has the upgraded ROM that asks "Diskette?" when you turn on the machine, and the upgraded character generator with a full European character set (not the old Japanese Kana characters) as the alternate special characters.

And I'm told you can leave a disk in the drive when you turn off the machine without the risk of zapping all your data.

Compared to these stunning (?) advantages, what does the non-gate array Model 4 have going for it? The most obvious advantage is that everyone seems to want the other version. Someone once said that if you'd like to collect something, find an item you like but no one else does: you get better pricing and better selection. That certainly holds for the non-gate array models, as I found in Fort Worth. I had my pick of the lot, whereas if I'd wanted a gate array model I probably would have left empty-handed. And that brings up the related question of what to do if something goes wrong.

Since the non-gate array machine consists of three separate boards, you can switch out a bad floppy disk controller board without having to replace the whole motherboard. The gate array is all or nothing, so to speak. If the floppy controller or RS232 go bad, the entire motherboard has to be chucked.

And since non-gate array machines are more common than gate array ones, finding a replacement part is likely to be easier, too.

And guess what: the disk controller and RS232 boards are the same ones used in the Model III, which further improves the availability of parts. I have several extras of each of these parts from old Model III's, and they would be useless in a gate array machine.

Some of the other gate array advantages can, in fact, be found or created on the non-gate array machines. As far as speed goes, if you want to bring your non-gate array computer up to a full 4 mhz, you can pull out one of the pins of the Z80 CPU. I haven't bothered with this on my machines, because I never noticed any significant speed difference between them and the gate array machine I once had.

Later non-gate array machines have both the upgraded ROM chips and the upgraded character generator. If you want these features in an earlier machine, the simple matter of replacing a few chips gets them for you.

Keyboards are a matter of preference, as are monitors. The last non-gate arrays had, in fact, both green monitors and clustered cursor keys. I think the original non-gate array non-clustered keyboard manufactured by ALPS of Japan is unmatched by any Model 4 keyboard since. The keys have a quick, springy response and the contacts seem indestructible. My keyboard is ten years old and still going strong. And I love the cursor keys at opposite ends of the keyboard. Try playing any of the old games like Sea Dragon with clustered cursor keys.

For that matter, about the only thing the clustered cursor keys are good for is inputing data from the numeric keypad with programs like VisiCalc. If you're a touch typist and use your computer for word processing, they're a disaster because you have to constantly move your fingers away from the rest of the keys. My first Model 4 had, in fact, the "new" keyboard, and I refused to take delivery on it from Radio Shack until it was replaced with one of the old ALPS ones.

And what about monitors? I like green more than black and white, though I have machines with both. Most non-gate array machines come with black and white monitors; only the last production models seem to have been fitted with green ones. Unless something has been switched, though, a gate array machine --always-- means a green monitor. But is green really that much better? Monochrome VGA monitors on IBM clones aren't green, nor are monochrome Macintoshes, nor are the monochrome DEC VT420 terminals in our university library. So why are all these supposedly state-of-the art monitors black and white and not green? Maybe black and white screens aren't so bad after all.

Finally, is "later" really "better"? It was what appears to be an error in one of their new chips that caused some gate array Model 4's to create duplicate file names in directories until the DOS was patched to compensate for the chip code (see Roy Beck's article "Two For the Price of One?" on pages 9-10 of the September/October 1990 issue). And it's with the gate array models that Radio Shack began using its own power supply units instead of the earlier Astec version, which seems to be more reliable.

There are also problems with the Radio Shack hi-res board: the Shack had to make a modified version of the board that was compatible with its gate array machine, but either version of the board will work on the non-gate array machine. So if you have a non-gate array Model 4 that you plan to keep for a while, and you worry that you should "upgrade" to the non-gate array version before it's too late -- don't. Get a cheap backup machine for parts, turn on your computer, and enjoy yourself... at least until someone with a gate array machine writes a rebuttal to this article!

Gary Shanafelt can be reached at:
Dept. of History
McMurry College
Abilene, TX 79697

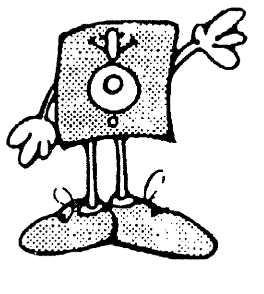
Roy Beck offers the following thoughts:

To add to Gary's comments (with which I fully agree), be aware that the redesign from the non-gate array to the gate array includes a different floppy disk controller chip. Tandy intended the two versions of the machines to be logically identical, and for most purposes they are. The same ROM is used in both the non-gate array and the gate array version of the 4P. But wouldn't you know, a fly crept into the ointment.

Mike Durda, (M.A.D. Software) offered a software only modification of the 4P to allow auto-booting from a hard drive. His original software functioned reliably with the non-gate array 4P, but only worked about 50% of the time in the gate array 4P. Having several of each at the time, I wrote to him to ask assistance. After some analysis on his part, he came back to me that the problem occurred because of slight differences in the internal micro-coding of the two different floppy disk controllers. While he could offer a work-around patch, the real solution was to change the 4P ROM, but ONLY the one in the gate array version. The non-gate array version was fine. The net result of all of this is that non-gate array 4P users need buy only the software from M.A.D. Software to have autobooting. Gate array version owners must buy the software AND a replacement ROM in order to have reliable autobooting of their 4P. (Note, the Model 4 and 4D ROMs were not written to auto-boot from hard drives, and all Model 4's and 4D's require a ROM change to enable auto-booting.)

This whole commentary just provides another argument in favor of the non-gate array machine in the case of the 4P.

Roy



ITEMS OF INTEREST
USERS GROUPS

SADTUG
The San Diego
TRS-80 Users Group

meets the first Sunday of each month at 2pm.
The oldest TRS user group in the US,
established in 1977,
two months after the Mod 1 came out.

Contact Mike Baldwin
(619) 583-1578

VTUG
Valley TRS-80 Users Group

meets the third Friday of each month
at 7:30 pm at the Learning Tree University.
DeSoto & Knapp Streets, Chatsworth, CA

Contact Lance Wolstrup
(818) 716-7154

VTHG
Valley TRS-80 Hackers Users Group

Los Angeles, CA
meets the first Friday of each month at 7:00 pm

for more information

Contact Roy Beck
(213) 664-5059

OCTUG
Orange County TRS-80 Users Group

meets the third Sunday of each month at 11:00 am
in the Community Hall at the City Shopping Center
Orange, CA
Complete support for TRS-80's Model I through 4

Contact Mike Lingo
(714) 545-1059 (before 3 pm)

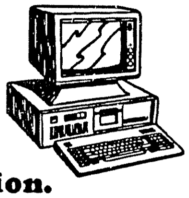
SAGATUG
San Gabriel Tandy Users Group

meets the second Friday of each month at 7:00 pm
at the Arcadia Park Senior Citizen's Center
405 South Santa Anita Avenue
Arcadia, CA
THE CLUB FOR TRSDOS, MS-DOS, CP/M
AND LAPTOP COMPUTERS
Large public domain libraries
remote memberships available

Contact Barbara Beck
(213) 664-5059

Mid-Cities
Tandy Radio Shack
Users Group

MCTRUG supports all of the Tandy computers plus IBM compatibles. We have software available for TRS-80 Models I, III, 4, Color Computers, Model 2000 and MS-DOS.

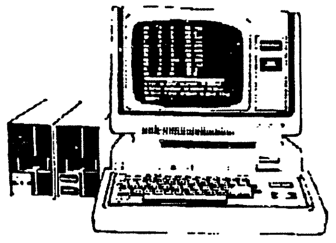
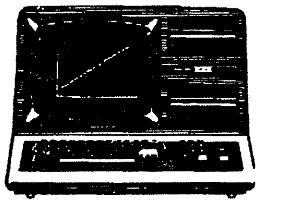
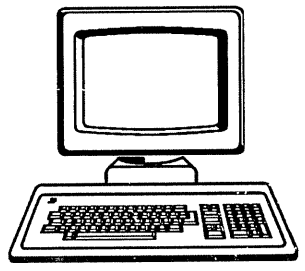


Write us for more information. Please include your name, mailing address, computer model and which Disk Operating System you use.

Write to:

MCTRUG
P.O. Box 171566
Arlington, TX 76003

1 TO 4 TO PC
BORN AGAIN part II
by Lance Wolstrup

In the time since our last issue, I have had an absolute ball playing with and exploring the Model I emulator program running on my MS-DOS clone.

"Oh No", some of you might say, "another PC article! Is TRSTimes slowly converting to MS-DOS?" The answer, of course, is NO. We have covered, and will continue to cover, TRS-80 material exclusively. However, even though the program we will talk about in this article is indeed an MS-DOS program, we consider it TRS-80 material because, for all practical purposes, it creates a TRS-80 Model I right inside your PC-clone - and what could be more TRS-80 than a Model I?

I mailed my check for $25 to Jeff Vavasour, the author, and within a week I received the upgraded version of the emulator. And what an upgrade it was. The new version runs faster than a real Model I and I am really pleased with its performance.

One of the things that did not work in the PD version was the FORMAT utilities in LDOS, DOSPLUS, MULTIDOS, and TRSDOS. For some reason, NEWDOS/80 was capable of formatting a disk from the emulator - while the aforementioned four failed. I was hoping this would be fixed in the upgrade version, and I was a little disappointed to find that it had not. I later realized that the problem was not with the emulator, but with the DOSes themselves, and I then set out to fix FORMAT/CMD in each of those. More on this later.

For those of you who would like to get a copy of this program (the shareware version), you can do it several ways. It is available free for the downloading from the TRSuretrove BBS, (213) 664-5056 (as well as from other boards, I'm sure). The file is called MODEL1/ARC and is located on the A board. You can also obtain the program from TRSTimes. We will mail it to you on an MS-DOS formatted diskette for the sum of $5.00. The smartest way to get the program, though, is to buy the registered version directly from Jeff Vavasour. Send him $25.00 and an MS-DOS formatted disk - I think you'll be glad you did. His address is:

Jeff Vavasour
c/o Department of Physics
University of British Columbia
6224 Agricultural Road
Vancouver, British Columbia
Canada V6T 1Z1

Since our last issue, I have had the opportunity to transfer many programs to the emulator. So far I have seen the following DOSes and programs running without problems:

Ldos 5.1.4, Ldos 5.3.1, Trsdos 2.3, Newdos/80, Multidos 1.6, Dosplus 3.4, Dosplus 3.5, Trsdos 3.0, EdaS (Ldos), Edtasm series 1, Edtasm (Newdos/80), Basic on all Dos'es, Ted (Ldos), FedII (Ldos), Superzap (Newdos/80), Disassem (Newdos/80), Zap (Multidos), Profile, Scripsit, Visicalc, Newscript, Lescript, Enhbas, Time Manager, Aids Plus, Gobbler, Cosmic Fighter 2, Centipede, Scarfman, Hoppy, Super Nova, Galaxy Invasion, Meteor Mission, Sea Dragon, the Scott Adams adventures, Eliminator, Voyage of the Valkyrie (incredible sound), Dancing Demon, Android Nim, Bee Wary, Apple Panik, Time Machine, Pyramid, Haunted Hause, Crazy Painter, Rear Guard, Diplomacy, Lunar Lander, Super Scripsit, and every program I tested from my Softside on Disk collection.

Oh Yes, I did find a cople of programs that objected somewhat to being inside the emulator - SUPER UTILITY and COPYCAT. Because of the nature of these programs, they are from a programming standpoint 'ill-behaved'. Oh well, most of their functions are not needed anyway, so it is no great loss.

As I mentioned earlier, I had severe problems with the FORMAT utility on all DOSes except NEWDOS/80. With it, I was able to format data disks of 35 to 80 tracks. The others refused to complete the format procedure, aborting with a variety of error messages. At first I thought that this was a bug in the emulator program, but when I began playing with the code, I soon realized that it was indeed FORMAT/CMD from the various DOSes that were at fault.

Each DOS gives a different error message, but the bottom line is that the format utility has trouble verifying the directory track. We all know (or should know) that the directory track is formatted with a different set of data address marks (DAM) than the rest of the tracks, so if FOR-

MAT/CMD is capable of formatting a disk correctly on the real Model I, why does it refuse to do it in the emulator. After discussing this with several people, on the BBS and at club meetings, it finally hit me; it works perfectly on the Model I because it is formatting a real disk - and it fails in the emulator because it is trying to format a virtual disk.

Say what!!!!

In other words, FORMAT/CMD reads the data address marks from the disk in the real Model I - because they are there. In the emulator we are dealing with a virtual disk, which is not a real disk - it is a file. And, my friends, disks have data address marks, files do NOT. Thus, when FORMAT/CMD tries to verify the just formatted virtual disk (the file), it does NOT find the DAMs it is looking for and it therefore aborts with an error message.

Well, once I understood what was going on, the obvious thing was to modify the FORMAT utilities to ignore this error. With FEDII and LDOS, I disassembled FORMAT/CMD from LDOS 5.1.4, LDOS 5.3.1, TRSDOS 2.3, MULTIDOS 1.6, DOSPLUS 3.4, and DOSPLUS 3.5. After a weekend all-night session I had found the patches to the two versions of LDOS, MULTIDOS, and TRSDOS 2.3. It certainly felt good to have 'conquered' the problem. Then came the setback. I started to investigate DOSPLUS 3.4, and after many hours, I just could not find where it conked out. It appears to be in the actual DOS-call, which is undocumented and not part of FORMAT/CMD, so I gave up for the time being and began tackling DOSPLUS 3.5. This version of FORMAT/CMD is much, much different than the one in 3.4. I had no trouble finding the patch here. Makes me wonder why version 3.4 is so convoluted!

Here are the patches - make sure that they are made from inside the emulator. In other words, do not patch your real Model I disks, **patch the virtual disks only**.

LDOS 5.1.4

patch format/cmd.rs0lt0ff (d01,9d=00)

LDOS 5.3.1

patch format/cmd.rs0lt0ff (d01,9b=62:f01,9b=e3)

MULTIDOS 1.6

patch format/cmd (rec=1,byte='x5e') 24

TRSDOS 2.3

This DOS does not have a built-in PATCH command, so it will be necessary to use a ZAP program, such as FEDII, ZAP, SUPERZAP, or other.

```
record 2, bytes DA & DB
                 now AD 56
           change to  1D 54
```

DOSPLUS 3.5

This DOS does not have a built-in PATCH command, so it will be necessary to use a ZAP program, such as FEDII, ZAP, SUPERZAP, or other.

```
record 3, byte 21
                 now 28
            change to 18
```

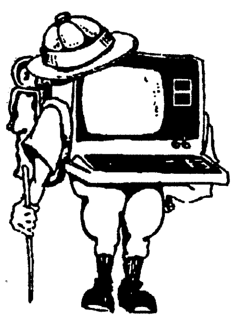
OK, with the formatting problem now solved, once and for all, we can go on to other things, such as speeding up the disk-to-file transfer.

Last month I presented a quick and dirty Model I machine language program that would copy a Model I disk to a data file. It worked just fine - the problem was that it took a few seconds better than 4 minutes to do the job. It was a real hard worker, reading a sector on the disk and then writing it as a record in the file. On a 35-track disk that turned out to be 350 sector reads, each read followed by a record write. Too much swapping back and forth. No question that this could be done better.

It occured to me that most Model I owners also own Model 4's (at least that is the case in the various Los Angeles area TRS-80 clubs), so writing a Model 4 program to copy the Model I disk, taking advantage of the faster clock speed and also tightening up the original code ought to speed up the process significantly.

I got a bare-bones working program together and uploaded it to Gary Shanafelt, who then added the user-friendly touches (such as prompts and default values). He posted the modified version back on the TRSuretrove BBS for me, and I then added the final features. By the time we were done, the program does its work in half the time of the original Model I version if the Model I disk is copied to a file on another floppy disk (2.05 minutes vs 4+ minutes). If, on the other hand, the Model I disk is copied to a file residing on a hard drive, the time is reduced to only 35 seconds - this is my kind of speed!

Before I forget, let me mention that, because of the limitations imposed by the Emulator program, the Model I disk MUST be single-density and the directory MUST be on track 17. In other words, when written to a file, the di-



rectory MUST be in records 170 through 179. Also keep in mind that the file that is created by the program will be slighly larger than the entire Model I source disk, so the destination disk must therefore be formatted to accomodate this larger size. Since it is being done on a Model 4, this should not be a problem. I format all my destination disks to 40 track, double-density.

The program, now called MAKFILE4, prompts the user to type in the source drive number. This should be answered by typing the number of the drive where the Model I disk is inserted. Pressing just ENTER will select the default value of drive :1. Pressing the BREAK key at this point will make the program exit to DOS.

The second prompt will ask the user to type the number of the destination drive. That is the number of the drive where the file will be written. You may answer this with any valid drive number. Pressing just ENTER will select the default value of drive :2. Pressing the BREAK key returns the program to the Source drive number prompt.

The third prompt asks for the number of tracks. If the Model I disk has other than 35 tracks, type that number, otherwise select the default by pressing ENTER. Pressing the BREAK key moves the program back to the previous prompt, the 'Destination' prompt.

The fourth prompt asks for the name of the destination file. Type a standard DOS filename - and use the extension /DSK. Don't bother to attach a drive specification at the end of the filename. It will be ignored in favor of the value chosen at the 'Destination drive number' prompt. Pressing the BREAK key returns the program to the previous prompt, the 'Number of tracks' prompt.

The final prompt asks you to press ENTER when both the source and destination disks are ready. Pressing BREAK instead will move the program to the previous prompt, the 'Name of destination file' prompt.

Assuming that both the source and destination disks are in place, MAKFILE4 will read 10 sectors per track, each time writing 10 records to the end of the newly created file.

MAKFILE4/ASM is written with EDAS from Misosys. The source code is, for the most part, compatible with the other Editor/Assemblers available for the TRS-80's. The notable exceptions will be that the DB and DW pseudo-ops will have to be written as DEFB (or DEFM) and DEFW. Also, if more than one value is defined under DB, it may be necessary to write one value per DB. With that in mind, get out your favorite Editor/Assembler and type in the following code and assemble it as MAKFILE4/CMD

MAKFILE4/ASM

```
;MAKFILE4/ASM v2.0
;by Lance Wolstrup and Gary W. Shanafelt
;Copyright (c) 1993 - all rights reserved
;
;for TRS-80 Model 4
;converts entire single-sided, single-density
;Model I disk to a normal data file
;directory of Model I disk MUST be on track 17
;
        ORG     3000H
START   LD      C,15            ;cursor off
        LD      A,2             ;@dsp
        RST     40
;
        LD      A,105           ;@cls
        RST     40
;
        LD      HL,HELLO$       ;point to header msg
        LD      A,10            ;@dsply
        RST     40
;
ASKSRC  LD      HL,0906H        ;print@(9,6)
        CALL    LOCATE          ;position cursor
;
        LD      HL,SRCMSG       ;point to src prompt
        LD      A,10            ;@dsply
        RST     40
;
        LD      A,1             ;set default
        LD      (SRC),A         ;and store it
;
        LD      HL,BUFFER       ;point to buffer
        LD      BC,0100H        ;max input chr=1
        LD      A,9             ;@keyin
        RST     40
;
        JP      C,EXIT          ;exit if break
;
        LD      HL,BUFFER       ;point to input
        LD      A,(HL)          ;and get it
        CP      13              ;is it just enter?
        JR      Z,ASKDST        ;yes-default chosen-jump
;
        CP      30H             ;is it 0?
        JR      C,ASKSRC        ;jump if less
        CP      38H             ;jump if
        JR      NC,ASKSRC       ;larger than 7
        SUB     30H             ;drv num legal-strip ascii
        LD      (SRC),A         ;and store in buffer
;
ASKDST  LD      HL,0B00H        ;print@(11,0)
        CALL    LOCATE          ;position cursor
;
        LD      HL,DSTMSG       ;point to destination prompt
        LD      A,10            ;@dsply
        RST     40
;
        LD      A,2             ;set default
        LD      (DST),A         ;and store it
;
        LD      HL,BUFFER       ;point to buffer
        LD      BC,0100H        ;max char input is 1
        LD      A,9             ;@keyin
        RST     40
;
        JR      C,ASKSRC        ;prev prompt if break
;
        LD      HL,BUFFER       ;point to input
        LD      A,(HL)          ;and get it
;
        CP      13              ;is it enter
        JR      Z,ASKTRK        ;jump if default
;
        CP      30H             ;is it 0
        JR      C,ASKDST        ;jump if less
        CP      38H             ;is it 7 or more
        JR      NC,ASKDST       ;jump if so
;
        SUB     30H             ;strip ascii
        LD      B,A             ;dst drv num to B
        LD      A,(SRC)         ;get src drv num
        CP      B               ;compare them
        JR      NZ,SETDST       ;jump if different
;
DRVERR  LD      HL,0D00H        ;print@(13,0)
        CALL    LOCATE          ;position cursor
;
        LD      HL,SRCDST       ;point to error msg
        LD      A,10            ;@dsply
        RST     40
;
        LD      A,1             ;@key
        RST     40
;
        CP      13              ;is it enter
        JR      NZ,DRVERR       ;no - prompt again
        JR      ASKDST          ;ask for dest drv num
;
SETDST  LD      A,B             ;retrieve dst drv num
        LD      (DST),A         ;and store it
;
ASKTRK  LD      HL,0D08H        ;print@(13,8)
        CALL    LOCATE          ;position cursor
;
        LD      HL,TRKMSG       ;point to trk prompt
        LD      A,10            ;@dsply
        RST     40
;
        LD      A,35            ;set up default
        LD      (TRK),A         ;and store it
;
        LD      HL,BUFFER       ;point to input buffer
        LD      BC,0200H        ;max input chars=2
        LD      A,9             ;@keyin
        RST     40
;
        JR      C,ASKDST        ;jump if break
        LD      A,B             ;num of key strokes
        OR      A               ;any there
        JR      Z,ASKNAM        ;jump if default
;
        LD      HL,BUFFER       ;point to input
        LD      A,96            ;@dechex
        RST     40
;
        LD      A,C             ;xfer hex number to a
        CP      35              ;is it 35 tracks
        JR      C,ASKTRK        ;jump if less
        CP      81              ;max is 80 tracks
        JR      NC,ASKTRK       ;jump if >80
;
        LD      (TRK),A         ;store num of tracks
;
ASKNAM  LD      HL,0F0FH        ;print@(15,15)
        CALL    LOCATE          ;position cursor
;
        LD      HL,NAMMSG       ;point to name msg
        LD      A,10            ;@dsply
        RST     40
;
        LD      HL,BUFFER       ;point to input buffer
        LD      BC,1800H        ;max input=23+cr
        LD      A,9             ;@keyin
        RST     40
;
        JR      C,ASKTRK        ;prev prompt if break
;
        LD      A,B             ;get num of chrs input
        OR      A               ;any input there?
        JR      Z,ASKNAM        ;yes-assume filename
;
PUTDN   INC     HL              ;find end of filename
        LD      A,(HL)          ;get char
        CP      ':'             ;is drv number attached
        JR      Z,PUTDN1        ;attached-so skip colon
        DJNZ    PUTDN
;
        LD      A,':'           ;append colon
        LD      (HL),A          ;to filename
PUTDN1  INC     HL              ;next filename position
        LD      A,(DST)         ;get dest drv number
        ADD     A,30H           ;make it ascii
        LD      (HL),A          ;append it to filename
        INC     HL              ;next filename position
        LD      A,13            ;append
        LD      (HL),A          ;terminator to filename
;
        LD      HL,BUFFER       ;point to filespec
        LD      DE,FCB          ;point to fcb
        LD      A,78            ;@fspec
        RST     40
;
ASKRDY  LD      HL,1200H        ;print@(18,0)
```

```
        CALL    LOCATE          ;position cursor
;
        LD      HL,RDYMSG       ;point to ready msg
        LD      A,10            ;@dsply
        RST     40
;
        LD      A,1             ;@key
        RST     40
;
        CP      80H             ;is it break
        JR      Z,ASKNAM        ;prev prompt if break
;
        CP      13              ;is it enter
        JR      NZ,ASKRDY       ;no - ask again
;
        LD      HL,1200H        ;print@(18,0)
        CALL    LOCATE          ;position cursor
;
        LD      C,15            ;cursor off
        LD      A,2             ;@dsp
        RST     40
        LD      C,31            ;clear to end of dsply
        LD      A,2             ;@dsp
        RST     40
;
        LD      HL,BUFFER       ;point to i/o buffer
        LD      A,(SRC)         ;get drive num
        LD      C,A             ;put it in c
        LD      DE,0            ;track 0, sector 0
        LD      A,49            ;@rdsec
        RST     40
;
        LD      A,(BUFFER+2)    ;get dir track
        CP      17              ;is it track 17
        JR      Z,CONTIN        ;jump if 17
;
ERR17   LD      HL,NOT17        ;point to message
        LD      A,10            ;@dsply
        RST     40
;
        LD      A,1             ;@key
        RST     40
;
        CP      80H             ;is it break
        JP      Z,ASKSRC        ;1st prompt if break
;
        CP      13              ;is it enter
        JP      Z,ASKRDY        ;yes-to ready prompt
        JR      ERR17
;
CONTIN  LD      HL,BUFFER       ;point to i/o buffer
        LD      DE,FCB          ;point to fcb
        LD      B,0             ;rec len=256
        LD      A,58            ;@init
        RST     40
;
        LD      A,(TRK)         ;get number of tracks
        LD      B,A             ;xfer to b
        LD      A,(SRC)         ;get src drive number
        LD      C,A             ;xfer to c
        LD      A,41            ;@slct
        RST     40
;
        LD      D,0             ;begin at track 0
TLOOP   PUSH    BC              ;save trk loop counter
        PUSH    DE              ;save current trk num
        LD      HL,1200H        ;print@(18,0)
        CALL    LOCATE          ;position cursor
        LD      L,D             ;trk num to hl
        LD      H,0
        LD      DE,DECTRK       ;convert to decimal
        LD      A,97            ;@hexdec
        RST     40
;
        LD      HL,RDMSG        ;display message
        LD      A,10            ;@dsply
        RST     40
;
        POP     DE              ;restore trk number
        POP     BC              ;restore trk loop cnter
        PUSH    BC              ;save track loop
        LD      E,0             ;sector 0
        PUSH    DE              ;save track
        LD      HL,BUFFER       ;point to i/o buffer
        LD      B,10            ;10 sectors
SLOOP   LD      A,49            ;@rdsec
        RST     40
        JR      Z,SLOOP1        ;jump if good read
        CP      6               ;is it dir read error
        JR      NZ,ERROR        ;jump only if other err
SLOOP1  PUSH    DE              ;save sector
        LD      DE,256          ;point to next
        ADD     HL,DE           ;segment of buffer
        POP     DE              ;restore sector
        INC     E               ;next sector
        DJNZ    SLOOP
;
        LD      HL,1200H        ;vertl=18,horiz=0
        CALL    LOCATE          ;position cursor
;
        LD      HL,WRTMSG       ;point to write message
        LD      A,10            ;@dsply
        RST     40
;
        LD      B,10            ;10 sectors
        LD      HL,BUFFER       ;point to i/o buffer
        LD      DE,FCB          ;point to fcb
        PUSH    DE              ;copy fcb pointer
        POP     IX              ;to ix
WLOOP   LD      (IX+3),L        ;stuff address of current
        LD      (IX+4),H        ;buffer segment in fcb
        LD      A,75            ;@write
        RST     40
        JR      NZ,ERROR        ;jump if write error
;
        PUSH    DE              ;save fcb pointer
        LD      DE,256          ;figure new address
        ADD     HL,DE
        POP     DE              ;restore fcb pointer
        DJNZ    WLOOP
;
        POP     DE              ;restore tracks
        INC     D               ;next track
        POP     BC              ;restore counter
        DJNZ    TLOOP
;
        LD      DE,FCB          ;point to fcb
        LD      A,60            ;@close
        RST     40
;
ASKRPT  LD      HL,1200H        ;print@(18,0)
        CALL    LOCATE          ;position cursor
        LD      HL,OKMSG        ;point to message
        LD      A,10            ;@dsply
        RST     40
;
        LD      A,1             ;@kbd
        RST     40
        CP      80H             ;is it break?
        JR      Z,EXIT          ;yes - jump to exit
        CP      13              ;is it enter?
        JP      Z,ASKSRC        ;back to 1st prompt
        JR      ASKRPT          ;no-so prompt again
;
EXIT    LD      C,14            ;cursor on
        LD      A,2             ;@dsp
        RST     40
        RET                     ;back to dos
;
ERROR   POP     DE
        POP     BC
        OR      192             ;set bits 6 & 7
        LD      C,A
        LD      HL,1500H        ;print@(21,0)
        CALL    LOCATE
        LD      A,26            ;@error
        RST     40
        JR      EXIT
;
LOCATE  PUSH    BC
        PUSH    DE
        LD      B,3             ;position cursor
        LD      A,15            ;@vdctl
        RST     40
        POP     DE
        POP     BC
        RET
;
HELLO$  DB      'MAKFILE4',10,10
        DB      'Transfer a single-sided, single-density '
        DB      'Model I disk into a normal data file',10
        DB      'By Lance Wolstrup and '
        DB      'Gary W. Shanafelt',10
        DB      'Copyright ',21,239,21
        DB      ' 1993 v.2.0. All rights reserved',10,10
        DB      'NOTE: The destination disk must '
        DB      'have more '
        DB      'free space than the source disk!!',13
;
SRCMSG  DB      15,31
        DB      'Enter source drive number '
        DB      '(default = 1): ',14,3
;
DSTMSG  DB      15,31
        DB      'Enter destination drive number '
        DB      '(default = 2): ',14,3
;
SRCDST  DB      15
        DB      'The source and destination drives '
        DB      'cannot be the same '
        DB      '- press ENTER to continue ',14,3
;
TRKMSG  DB      15,31
        DB      'Enter number of tracks '
        DB      '(default = 35): ',14,3
;
NAMMSG  DB      15,31
        DB      'Enter name of destination file: ',14,3
;
RDYMSG  DB      15,31
        DB      'Press ENTER when source and '
        DB      'destination disks '
        DB      'are ready ',14,3
;
NOT17   DB      15,31
        DB      'Source disk does not have directory '
        DB      'on track 17 - '
        DB      'press ENTER ',14,3
;
OKMSG   DB      15,31
        DB      'Disk is now transferred to normal '
        DB      'data file - '
        DB      'press ENTER to continue ',14,3
;
WRTMSG  DB      'Writing',3
RDMSG   DB      31
        DB      'Reading track '
DECTRK  DS      5
        DB      13
;
SRC     DB      0
DST     DB      0
TRK     DB      0
FCB     DS      32
BUFFER  DS      2560
;
        END     START
```

With the above program and the program published in our last issue, we are now able to create the virtual disks on Model I's and 4's. In short, we have effectively discriminated against the Model III. In our current 'politically correct' times, lest we should be so accused, we are immediately making amends by presenting the Model I/III source code below. This is written using EDAS3, so make the appropriate translations. explained earlier, if you are using another editor/assembler. Note that MAKFILE/CMD is intended to be used under Model I or III LDOS only, as some of the DOS calls will probably not work on other operating systems.

MAKFILE/ASM

```
;MAKFILE1/ASM & MAKFIL3/ASM v.2.0
;by Lance Wolstrup and Gary W. Shanafelt
;Copyright (c) 1993 - all rights reserved
;
;for TRS-80 Model I/III
;converts entire single-sided, single-density
;Model I disk to a normal data file
;directory of Model I disk MUST be on track 17
;
        ORG     7000H
;
START   CALL    1C9H            ;cls
;
        LD      A,(293)         ;get value from 293
        CP      73              ;is it Mod III
        JR      Z,MOD3          ;yes - jump
        XOR     A               ;set up Model I
        LD      (COPYRG),A      ;nop
        LD      A,31H           ;set to "1"
        JR      MOD13           ;and jump
MOD3    LD      A,33H           ;set to "3"
MOD13   LD      (MODEL),A       ;and stuff into text
;
        LD      HL,HELLO$       ;point to header msg
        CALL    4467H           ;@dsply
;
ASKSRC  LD      HL,0606H        ;print@(6,6)
        CALL    LOCATE          ;position cursor
;
        LD      HL,SRCMSG       ;point to src prompt
        CALL    4467H           ;@dsply
;
        LD      A,1             ;set default
        LD      (SRC),A         ;and store it
;
        LD      HL,BUFFER       ;point to buffer
        LD      BC,0100H        ;max input chr=1
        CALL    40H             ;@keyin
;
        JP      C,EXIT          ;exit if break
;
        LD      HL,BUFFER       ;point to input
        LD      A,(HL)          ;and get it
        CP      13              ;is it just enter?
        JR      Z,ASKDST        ;default chosen-jump
;
        CP      30H             ;is it 0?
        JR      C,ASKSRC        ;jump if less
        CP      38H             ;jump if
        JR      NC,ASKSRC       ;larger than 7
        SUB     30H             ;strip ascii
        LD      (SRC),A         ;and store in buffer
;
ASKDST  LD      HL,0800H        ;print@(8,0)
        CALL    LOCATE          ;position cursor
;
        LD      HL,DSTMSG       ;point to dest prompt
        CALL    4467H           ;@dsply
;
        LD      A,2             ;set default
        LD      (DST),A         ;and store it
;
        LD      HL,BUFFER       ;common buffer
        LD      BC,0100H        ;max char input is 1
        CALL    40H             ;@keyin
;
        JR      C,ASKSRC        ;prev prompt if break
;
        LD      HL,BUFFER       ;point to input
        LD      A,(HL)          ;and get it
;
        CP      13
        JR      Z,ASKTRK        ;jump if default
;
        CP      30H             ;is it 0
        JR      C,ASKDST        ;jump if less
        CP      38H             ;is it 7 or more
        JR      NC,ASKDST       ;jump if so
;
        SUB     30H             ;strip ascii
        LD      B,A             ;dst drv num to B
        LD      A,(SRC)         ;get src drv num
        CP      B               ;compare them
        JR      NZ,SETDST       ;jump if different
;
DRVERR  LD      HL,0D00H        ;print@(13,0)
        CALL    LOCATE          ;position cursor
;
        LD      HL,SRCDST       ;point to error msg
        CALL    4467H           ;@dsply
;
        CALL    49H             ;@key
;
        CP      13              ;is it enter
        JR      NZ,DRVERR       ;not entern
        JR      ASKDST          ;ask for dest drive num
;
SETDST  LD      A,B             ;retrieve dst drive number
        LD      (DST),A         ;and store it
;
ASKTRK  LD      HL,0A08H        ;print@(10,8)
        CALL    LOCATE          ;position cursor
;
        LD      HL,TRKMSG       ;point to track prompt
        CALL    4467H           ;@dsply
;
        LD      A,35            ;set up default
        LD      (TRK),A         ;and store it
```

```
;
        LD      HL,BUFFER       ;point to input buffer
        LD      BC,0200H        ;max input chars=2
        CALL    40H             ;@keyin
;
        JR      C,ASKDST        ;jump if break
        LD      A,B             ;a=num of key strok
        OR      A               ;any there
        JR      Z,ASKNAM        ;jump if default
;
        LD      HL,BUFFER       ;point to input
        CALL    DECHEX          ;convert to hex
;
        LD      A,C             ;xfer hex number to a
        CP      35              ;is it 35 tracks
        JR      C,ASKTRK        ;jump if less
        CP      81              ;max is 80 tracks
        JR      NC,ASKTRK       ;jmp if larger than 80
;
        LD      (TRK),A         ;store num of tracks
;
ASKNAM  LD      HL,0C0FH        ;print@(12,15)
        CALL    LOCATE          ;position cursor
;
        LD      HL,NAMMSG       ;point to name msg
        CALL    4467H           ;@dsply
;
        LD      HL,BUFFER       ;point to input buffer
        LD      BC,1800H        ;max char=23+cr
        CALL    40H             ;@keyin
;
        JR      C,ASKTRK        ;prev prompt if break
;
        LD      A,B             ;num of chrs input
        OR      A               ;any input there?
        JR      Z,ASKNAM        ;filename - jump
;
PUTDN   INC     HL              ;find end of filename
        LD      A,(HL)          ;get char
        CP      ':'             ;is drv num attached
        JR      Z,PUTDN1        ;yes-so skip colon
        DJNZ    PUTDN
;
        LD      A,':'           ;append colon
        LD      (HL),A          ;to filename
PUTDN1  INC     HL              ;next position
        LD      A,(DST)         ;get dest drive num
        ADD     A,30H           ;make it ascii
        LD      (HL),A          ;append it
        INC     HL              ;next position
        LD      A,13            ;append
        LD      (HL),A          ;terminator
;
ASKRDY  LD      HL,0E00H        ;print@(14,0)
        CALL    LOCATE          ;position cursor
;
        LD      HL,RDYMSG       ;point to ready msg
        CALL    4467H           ;@dsply
;
        CALL    49H             ;@key
;
        CP      1               ;is it break
        JR      Z,ASKNAM        ;prev prompt if break
;
        CP      13              ;is it enter
        JR      NZ,ASKRDY       ;no - ask again
;
        LD      HL,0E00H        ;print@(14,0)
        CALL    LOCATE          ;position cursor
;
        LD      A,15            ;cursor off
        CALL    33H             ;@dsp
        LD      A,31            ;clear to end of disply
        CALL    33H             ;@dsp
;
        LD      HL,BUFFER       ;point to filespec
        LD      DE,FCB          ;point to fcb
        CALL    441CH           ;@fspec
;
        LD      HL,BUFFER       ;point to i/o buffer
        LD      A,(SRC)         ;get drive num
        LD      C,A             ;put it in c
        LD      DE,0            ;track 0, sector 0
        CALL    4777H           ;@rdsect
;
        LD      A,(BUFFER+2)    ;get dir track
        CP      17              ;is it track 17
        JR      Z,CONTIN        ;jump if 17
;
ERR17   LD      HL,NOT17        ;point to message
        CALL    4467H           ;@dsply
;
        CALL    49H             ;@key
;
        CP      1               ;is it break
        JP      Z,ASKSRC        ;1st prompt if break
;
        CP      13              ;is it enter
        JP      Z,ASKNAM        ;go ask for diskname
        JR      ERR17
;
CONTIN  LD      HL,BUFFER       ;point to i/o buffer
        LD      DE,FCB          ;point to fcb
        LD      B,0             ;rec len=256
        CALL    4420H           ;@init
;
        LD      A,(TRK)         ;get number of tracks
        LD      B,A             ;xfer to b
        LD      A,(SRC)         ;get source drive num
        LD      C,A             ;xfer to c
        CALL    4754H           ;@select
;
        LD      D,0             ;begin at track 0
TLOOP   PUSH    BC              ;save track loop counter
        PUSH    DE              ;save current track num
        LD      HL,0E00H        ;print@(14,0)
        CALL    LOCATE          ;position cursor
        LD      L,D             ;trk num to hl
        LD      H,0
        LD      DE,DECTRK       ;convert to decimal
        CALL    HEXDEC          ;conv to dec ascii
;
        LD      HL,RDMSG        ;display message
        CALL    4467H           ;@dsply
;
        POP     DE              ;restore track num
        POP     BC              ;restore track counter
        PUSH    BC              ;save track loop
        LD      E,0             ;sector 0
        PUSH    DE              ;save track
        LD      HL,BUFFER       ;point to i/o buffer
        LD      B,10            ;10 sectors
SLOOP   CALL    4777H           ;@rdsect
        JR      Z,SLOOP1        ;jump if good read
        CP      6               ;is it dir read error
        JR      NZ,ERROR        ;jmp only if other err
SLOOP1  PUSH    DE              ;save sector
        LD      DE,256          ;point to next
        ADD     HL,DE           ;segment of buffer
        POP     DE              ;restore sector
        INC     E               ;next sector
        DJNZ    SLOOP
;
        LD      HL,0E00H        ;vert=14, horiz=0
        CALL    LOCATE          ;position cursor
;
        LD      HL,WRTMSG       ;point to write msg
        CALL    4467H           ;@dsply
;
        LD      B,10            ;10 sectors
        LD      HL,BUFFER       ;point to i/o buffer
        LD      DE,FCB          ;point to fcb
        PUSH    DE              ;copy fcb pointer
        POP     IX              ;to ix
WLOOP   LD      (IX+3),L        ;stuff address of
        LD      (IX+4),H        ;buffer segmnt in fcb
        CALL    4439H           ;@write
        JR      NZ,ERROR        ;jump if write error
;
        PUSH    DE              ;save fcb pointer
        LD      DE,256          ;figure new address
        ADD     HL,DE
        POP     DE              ;restore fcb pointer
        DJNZ    WLOOP
;
        POP     DE              ;restore tracks
        INC     D               ;next track
        POP     BC              ;restore counter
        DJNZ    TLOOP
;
        LD      DE,FCB          ;point to fcb
        CALL    4428H           ;@close
;
ASKRPT  LD      HL,0E00H        ;print@(14,0)
        CALL    LOCATE          ;position cursor
        LD      HL,OKMSG        ;point to message
        CALL    4467H           ;@dsply
;
        CALL    49H             ;@key
        CP      1               ;is it break?
        JR      Z,EXIT          ;yes - jump to exit
        CP      13              ;is it enter?
        JP      Z,ASKSRC        ;back to 1st prompt
        JR      ASKRPT          ;prompt again
;
EXIT    RET                     ;back to dos
;
DECHEX  LD      A,(HL)          ;get tens digit
        SUB     30H             ;strip ascii
        ADD     A,A             ;2x
        LD      C,A
        ADD     A,A             ;4x
        ADD     A,A             ;8x
        ADD     A,C             ;10x
        LD      C,A
        INC     HL              ;point to ones digit
        LD      A,(HL)          ;get tens digit
        SUB     30H             ;strip ascii
        ADD     A,C             ;we have hex number
        LD      C,A             ;to c to be compatible
        RET
;
ERROR   POP     DE
        POP     BC
        OR      192             ;set bits 6 & 7
        LD      C,A
        LD      HL,0E00H        ;print@(14,0)
        CALL    LOCATE
        CALL    4409H           ;@error
        JR      EXIT
;
HEXDEC  LD      A,L             ;get number
        LD      HL,TABLE        ;point to table
        LD      B,2             ;only need two digits
HEX0    PUSH    BC              ;save loop counter
        PUSH    AF              ;save number
        LD      B,0             ;subtraction loop counter
        LD      A,(HL)          ;get subtraction number
        LD      C,A             ;store it in c
        POP     AF              ;restore number
        PUSH    AF              ;save it again for later
HEX1    SBC     A,C             ;subtract table value
        JR      C,NEXT          ;until below 0
        INC     B               ;sub good - inc counter
        JR      HEX1            ;and do it again
NEXT    PUSH    BC              ;save subtraction counter
        LD      A,B             ;copy it to a
        OR      A               ;is it zero
        JR      Z,NEXT1         ;yes - so jump
        XOR     A               ;a=0
HEX2    ADD     A,C             ;get the digits value
        DJNZ    HEX2
NEXT1   POP     BC              ;restore subtraction counter
        LD      C,A             ;copy digits value to c
        LD      A,B             ;copy number to a
        ADD     A,30H           ;make it ascii
```

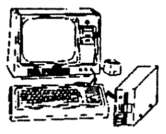
```
         LD      (DE),A          ;store num in buffer
         POP     AF              ;restore original num
         SUB     C               ;subtract digit value
         INC     DE              ;next buffer location
         INC     HL              ;next table value
         POP     BC              ;restore loop counter
         DJNZ    HEX0
         RET
;
LOCATE   PUSH    BC
         PUSH    DE
         LD      DE,15360        ;start of screen
         LD      B,0
         LD      C,L             ;store horiz value
         LD      A,H             ;get vert value
         OR      A               ;is it vert 0
         JR      Z,NOMULT        ;yes - no multiply
         LD      L,H             ;vert value in l
         LD      H,0             ;vert value now in hl
         LD      B,6             ;multiply hl by 64
MULT64   ADD     HL,HL
         DJNZ    MULT64
NOMULT   ADD     HL,BC           ;add in horiz value
         ADD     HL,DE           ;add the mem offset
         LD      (4020H),HL      ;cursor set
         POP     DE
         POP     BC
         RET
;
HELLO$   DB      'MAKFILE'
MODEL    DB      32
         DB      10
         DB      'Transfer Model I disk into a normal '
         DB      'data file',10
         DB      'By Lance Wolstrup and '
         DB      'Gary W. Shanafelt',10
         DB      'Copyright ',21
COPYRG   DB      239
         DB      21
         DB      ' 1993 v.2.0. All rights reserved',10,13
;
SRCMSG   DB      15,31
         DB      'Enter source drive number '
         DB      '(default = 1): ',14,3
;
DSTMSG   DB      15,31
         DB      'Enter destination drive number '
         DB      '(default = 2): ',14,3
;
SRCDST   DB      15
         DB      'The source and destination drives '
         DB      'cannot be the same '
         DB      '- press ENTER to continue ',14,3
;
TRKMSG   DB      15,31
         DB      'Enter number of tracks '
         DB      '(default = 35): ',14,3
;
NAMMSG   DB      15,31
         DB      'Enter name of destination file: ',14,3
;
RDYMSG   DB      15,31
         DB      'Press ENTER when source and '
         DB      'destination disks '
         DB      'are ready ',14,3
;
NOT17    DB      15,31
         DB      'Source disk does not have directory '
         DB      'on track 17 - '
         DB      'press ENTER ',14,3
;
OKMSG    DB      15,31
         DB      'Disk now transferred to normal '
         DB      'data file - '
         DB      'press ENTER ',14,3
;
WRTMSG   DB      'Writing',3
RDMSG    DB      31
         DB      'Reading track '
DECTRK   DS      2
         DB      13
;
SRC      DB      0
DST      DB      0
TRK      DB      0
FCB      DS      32
;
TABLE    DB      10
         DB      1
;
BUFFER   DS      2560
;
         END     START
```

With these programs, you are now able to transfer your Model I disks to standard data files, even if you no longer have your Model I available. The transfer can now be accomplished on a Model III or 4. However, once MAKFILE/CMD or MAKEFILE4/CMD has done its thing, the resulting datafile is still on a TRS-80 formatted disk. It now needs to be transferred to an MS-DOS diskette.

Copying a TRS-80 file to an MS-DOS disk can be done several ways. Using the PC as the transfer machine, TRSCROSS (available from MISOSYS) will do the trick.

Using the Model 4 as the transfer machine, we have programs such as HYPERCROSS, SUPERCROSS and MS-COPY. All three will effectively read the TRS-80 disk and copy the desired file to an MS-DOS formatted disk.

Once you have the datafile (virtual disk) on a PC disk, copy it to the appropriate directory on your hard drive and fire up your emulator. Let me tell you, it is the best Model I you've ever had.



ATTENTION TRSDOS 1.3. USERS!

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

MORE SPEED!! MORE POWER!! NEW LOW PRICE!!

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

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

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

SYSTEM 1.5. is now distributed exclusively by TRSTimes magazine.
ORDER YOUR COPY TODAY!

NEW LOW PRICE $15.95

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

Model I & III Public Domain Disks

PD#1: binclock/cmd, binclock/doc, checker/bas, checker/doc, chomper/bas, cls/cmd, dduty3/cmd, driver/cmd, driver/doc, drivtime/cmd, mazeswp/bas, minibase/bas, minitest/dat, mx/cmd, piazza/bas, spdup/cmd, spdwn/cmd, vici/bas, vid80/cmd, words/dic.
PD#2: creator/bas, editor/cmd, maze3d/cmd, miner/cmd, note/cmd, poker/bas, psycho/cmd, supdraw/cmd, vader/cmd
PD#3: d/cmd, trsvoice/cmd, xmodem/cmd, xt3/cmd, xt3/txt, xthelp/dat
PD#4: cobra/cmd, disklog/cmd, flight/bas, flight/doc, narzabur/bas, narzabur/dat, narzabur/his, narzabur/txt, othello/bas, vid80x24/cmd, vid80x24/txt
PD#5: eliza/cmd, lu31/cmd, sq31/cmd, usq31/cmd
PD#6: clawdos/cmd, clawdos/doc, cocoxf40/cmd, dskrnam/bas, menu/cmd, ripper3/bas, sky2/bas, sky2/his, space/cmd, stocks/bas, trs13pat/bas, vidsheet/bas
PD#7: cards/bas, cities/bas, coder/bas, eye/bas, heataudt/bas, hicalc/bas, life/bas, moustrap/bas, ohare/bas, slots/bas, stars/cmd, tapedit/bas
PD#8: craps/bas, fighter/bas, float/bas, hangman/bas, jewels/cmd, lifespan/bas, varidump/bas, xindex/bas, xor/bas
PD#9: bublsort/bas, chess/bas, finratio/bas, homebudg/bas, inflat/bas, mathdril/bas, midway/bas, nitefly/bas, pokrpete/bas, teaser/bas
PD#10: ltc21/bas, ltc21/ins, lynched/bas, match/bas, math/bas, message/bas, message/ins, portfol/bas, portfol/ins, spellegg/bas, storybld/bas
PD#11: alpha/bas, caterpil/cmd, cointoss/bas, crolon/bas, cube/cmd, dragon/cmd, fastgraf/bas, fastgraf/ins, lunarexp/bas, music/bas, music/ins, planets/bas, volcano/cmd
PD#12: baccarat/bas, backpack/bas, backpack/ins, doodle/bas, dragons/bas, dragons/ins, king/bas, sinewave/bas, snoopy/bas, wallst/bas, wallst/ins
PD#13: atomtabl/bas, boa/bas, chekbook/bas, conquer/cmd, dominos/bas, morse/bas, mountain/bas, quiz/bas, signbord/bas, sketcher/bas
PD#14: autoscan/bas, checkers/bas, craps/bas, ducks/bas, isleadv/bas, nim/bas, rtriangl/bas, sammy/cmd, typing/bas, wordpuzl/bas
PD#15: budget/bas, corp/bas, corp/ins, fourcolr/bas, fullback/bas, grapher/bas, illusion/bas, jukebox/bas, ledger/bas, maze/cmd, reactest/bas, shpspree/bas, states/bas, tapecntr/bas, tiar/bas, tiar/ins
PD#16: amchase/bas, constell/bas, filemastr/bas, foneword/bas, geometry/bas, heartalk/bas, hidnumbr/bas, lgame/bas, marvello/bas, powers/bas, scramble/bas, speed/bas, subs/bas
PD#17: conundrm/bas, eclipse/bas, esp/bas, esp/ins, hustle/bas, jacklant/bas, mindblow/bas, othello/bas, pleng/bas, rubik/bas, trend/bas, ufo/bas, veggies/bas
PD#18: backgam/bas, chess/cmd, cosmip/cmd, distance/bas, hexpawn/bas, music/cmd, stokpage/bas, texted/bas, texted/ins, trex/bas, twodates/bas, wanderer/bas
PD#19: banner/bas, cresta/cmd, lander/bas, medical/bas, moons/bas, par/bas, parchut/bas, pillbox/bas, readtrn/bas, replace/bas, ship/cmd, solomadv/bas, space/cmd, survival/bas
PD#20: bomber/bas, bumbee/cmd, claadv/bas, dice31/bas, dice31/ins, diskcat1/bas, firesafe/bas, flashcrd/bas, hitnmiss/bas, mazegen/bas, mazescap/cmd, roulette/bas, seasonal/bas
PD#21: aprfool/bas, catmouse/bas, d/cmd, escape/bas, header/bas, kalah/bas, mathwrld/bas, nameit/bas, note/cmd, photo/bas, read/cmd, syzygy/bas, timeshar/cmd, timeshar/doc, trace80/cmd, trsdir/cmd, worm/bas, yatz80/bas
PD#22: arcade/bas, cube/cmd, eclipse/bas, lcd/bas, leastsqr/bas, medical/bas, million/bas, pwrplant/bas, round/bas, subway/bas, tapeid/bas
PD#23: artil/bas, artil/ins, baseconv/bas, crushman/bas, dissert/bas, huntpeck/bas, jungle/bas, jungle/ins, messages/bas, monitor/bas, monster/bas, moons/bas, ohmlaw/bas, stockpage/bas, tictacto/bas
PD#24: baslist/asm, baslist/cmd, baslist/doc, cleaner3/cmd, cleaner3/doc, difkit1/bas, difkit1/doc, dirpatch/asm, dirpatch/cmd, e/cmd, ei/doc, i/cmd, newmap/bas, newmap/doc, varlst/asm, varlst/cmd, varlst/doc
PD#25: copy/bas, copy/doc, dirpw/asm, dirpw/cmd, dirpw/doc, dskfmt/bas, dskfmt/doc, himap/asm, himap/cmd, huricane/bas, hv/bas, hv/doc, keydemo/bas, keyin/bas, keyin/doc, lazyptch/asm, lazyptch/doc, salvage/bas, salvage/doc,wpflt/asm, wpflt/flt
PD#26: constell/bas, divisor/bas, frame/bas, heatfus/bas, heatfus/doc, hicalc/bas, mathlprt/bas, mathquiz/bas, molecule/bas, morscode/bas, phyalpha/bas, phyalpha/doc, remaindr/bas, usa/bas, wiring/bas
PD#27: engine/bas, fraction/bas, geosat/bas, grades/bas, julian/bas, lunarcal/bas, mailist/bas, metaboli/bas,musictrn/bas, perindex/bas, potrack/bas
PD#28: chainfil/bas, citoset/bas, convnum/bas, cursors/bas, cursors/doc, datamkr/bas, deprec/bas, gmenuii/bas, ledger12/bas, menui/bas, menuii/bas, minives/bas, ninteres/bas, refinanc/bas, regdepo/bas, rembal/bas, rndbordr/bas

Each disk is $5.00 (U.S.)
or get any 3 disks for $12.00 (U.S.)

please specify the exact disks wanted.

TRSTimes PD-DISKS
5721 Topanga Canyon Blvd., Suite 4
Woodland Hills, CA. 91367

PUBLIC DOMAIN GOOD GAMES FOR MODEL I/III.

GAMEDISK#1: amazin/bas, blazer/cmd, breakout/cmd, centipede/cmd, elect/bas, madhouse/bas), othello/cmd, poker/bas, solitr/bas, towers/cmd
GAMEDISK#2: cram/cmd, falien/cmd, frankadv/bas, iceworld/bas, minigolf/bas, pingpong/cmd, reactor/bas, solitr2/bas, stars/cmd, trak/cmd
GAMEDISK#3: ashka/cmd, asteroid/cmd, crazy8/bas, french/cmd, hexapawn, hobbit/bas, memalpha, pyramid/bas, rescue/bas, swarm/cmd
GAMEDISK#4: andromed/bas, blockade/bas, capture/cmd, defend/bas, empire/bas, empire/ins, jerusadv/bas, nerves/bas, poker/cmd, roadrace/bas, speedway/bas

Price per disk: $5.00 (U.S.)
or get all 4 disks for $16.00 (U.S.)

TRSTimes - PD GAMES
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA. 91364

TRSTimes on DISK #12

Issue #12 of TRSTimes on DISK is now available, featuring the programs from the Jul/Aug, Sep/Oct, Nov/Dec 1993 issues:

TRSTimes on DISK is reasonably priced:

U.S. & Canada: $5.00 (U.S.)
Other countries: $7.00 (U.S.)
Send check or money order to:

TRSTimes on DISK
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

TRSTimes on DISK #1, 2, 3, 4, 5, 6, 7, 8, 9, 10 & 11 are still available at the above prices

FONTS GALORE

FANTASTIC DOT WRITER FONTS

CREATED BY KELLY BATES

$3.00 PER DISK

CONTACT

MICKEY MEPHAM
9602 JOHN TYLER MEM HWY
CHARLES CITY, VA 23030

This is the last issue of your 1993 TRSTimes subscription.

Thank you for your support.

Stay with us in 1994.

Resubscribe now.

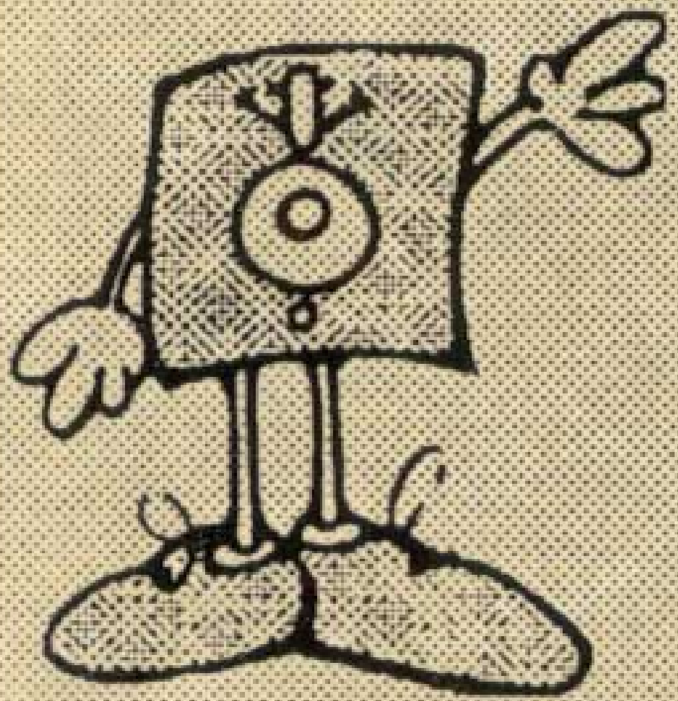
TRSTimes magazine
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

TIRED OF SLOPPY DISK LABELS?
TIRED OF NOT KNOWING WHAT'S ON YOUR DISK?

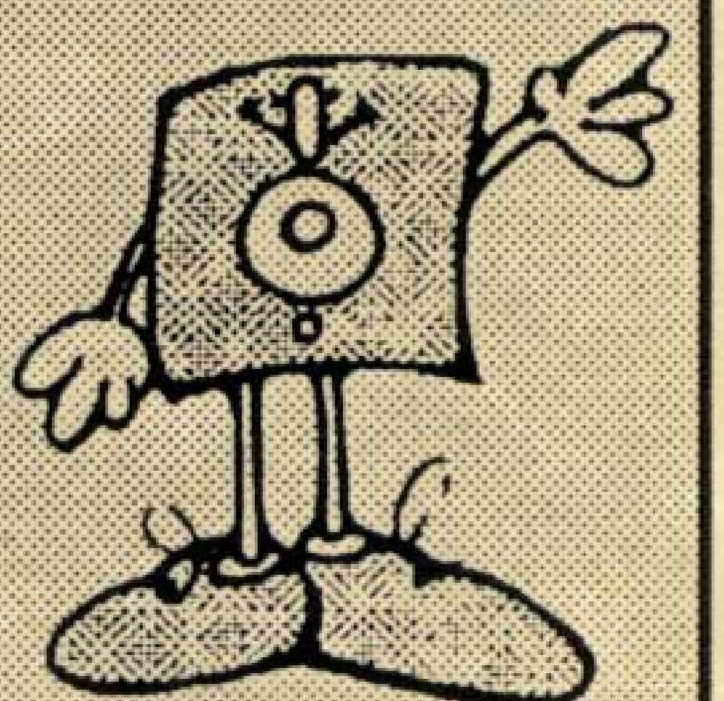
YOU NEED 'DL'

'DL' will automatically read your TRSDOS6/LDOS compatible disk and then print a neat label, listing the visible files (maximum 16).
You may use the 'change' feature to select (or reject) the filenames to print.
You may even change the diskname and diskdate.
'DL' is written in 100% Z-80 machine code for efficiency and speed.

Don't be without it - order your copy today.



'DL' is available for TRS-80 Model 4/4P/4D
using TRSDOS 6.2/LS-DOS 6.3.0 & 6.3.1.
with an Epson compatible or DMP series printer.
'DL' for Model 4 only $9.95



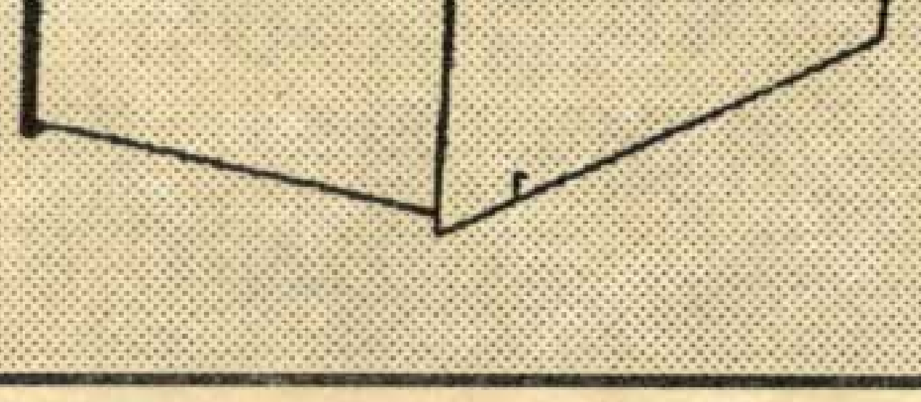
TRSTimes magazine - Dept. 'DL'
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

HARD DRIVES FOR SALE

Genuine Radio Shack Drive Drive Boxes with Controller, Power Supply, and Cables. Formatted for TRS 6.3, installation JCL included.
Hardware write protect operational.
Documentation and new copy of MISOSYS RSHARD5/6 included.
90 day warranty.

5 Meg $175	**10 Meg $225**	**15 Meg $275**	**35 Meg $445**

Shipping cost add to all prices



Roy T. Beck
2153 Cedarhurst Dr.
Los Angeles, CA 90027
(213) 664-5059