# TRSTimes

Volume 3. No. 6. - Nov/Dec 1990 - $4.00

HOLIDAY GREETINGS FROM SOUTHERN CALIFORNIA

# LITTLE ORPHAN EIGHTY

Might as well start with the official announcement: **TRSTimes WILL continue in 1991** with 6 bimonthly issues.

I have kept the subscription rate for U.S and Canada the same as last year ($18.00).

However, because of the impending postal hike, I am forced to raise the overseas subscription rates. They are as follows:

Europe, Central & South America: $23.00 for surface mail or $29.00 for air mail.

Asia, Australia & New Zealand: $25.00 for surface mail or $32.00 for air mail.

I do apologize to our foreign subscribers. Unfortunately, to balance the books, there simply was no alternative.

On another note, our supply of unused material - articles, programs, etc. - is getting VERY short. In order for TRSTimes to do a good job in 1991 (and, hopefully, again in 1992), I urge you, the readers, to participate by sending in your programs, articles, reviews, hints & tips, or anything else that might be of interest to other TRS-80 users. If motivated by nothing else - do so in self-defense - you certainly don't want to read 6 issues of TRSTimes written entirely by me!

As you will see elsewhere in this issue, another 12 disks have been added to our Model I/III public domain library. There will be more made available, but the cataloging process is being interrupted while this issue of TRSTimes is being put together. There will also be additional Model 4 public domain disks coming, but they will have to wait until the Model I/III work is finished. Hopefully, the Jan/Feb issue will list the additional Model 4 disks.

My Model I blew up earlier this year. Various people looked at it, and all declared the patient completely dead. I started looking around for another, but didn't have a lot of success. Either the system was not in as good a shape as I would have liked, or the price was unreasonable. Well, a few weeks ago, "Sandy" Sandusky came to the rescue. He donated a complete Model I system, expansion interface, 3 drives and the works, to TRSTimes. Boy, this system was in good shape. It certainly had been taken care of. It looks (and works) almost as new. I started to work on it as soon as I brought it home and, as you can see in this issue, it produced an article.

Thanks, Sandy, I really appreciate it.

My office has two Model 4's (one desktop with a 5 meg hard drive, the other a 4P with a 15 meg hard drive), a Model III, a Model I, a PC clone, a LaserJet series II printer, a Citizen MSP50, and a DMP200.

That ought be enough equipment for any one person, right? WRONG!! The other day Roy Beck called and informed me that he had just made a deal for a Model 16B. He said that there was another one available - did I want it? I looked around the office - I could still see the floor - so, of course, I said yes.

The computer has not as yet arrived - it is coming from overseas - but it is supposed to come with one drive and a 15 meg hard drive, along with a fair share of software. I really do look forward to getting it so I can play with it. This will be a nice learning experience for me as I have never touched a 16B, and know next to nothing about it.

All I know is that it has Z-80 chip and a 68000 chip, allowing the machine to run both TRSDOS and the Radio Shack version of UNIX, known as XENIX. The TRSDOS part should be interesting. The "SOURCE" (the official commented disassembly of the Model 4 DOS) indicates that a version of TRSDOS 6.2 is capable of running in the 16B (Model II mode). If that version is still available from Roy Soltoff (Misosys), I will get it just to play around and see how many programs I can get to run on both machines. Programs written in Basic shouldn't be much trouble. The fun part will be the assembly language programs. It seems to me that, if a program is well behaved (using only official SVC's), it ought to run on both machines. Well, we'll see!

I'm not too sure about the XENIX aspect of the machine. The closest I've come to this operating system is playing around with OS-9 on the Color Computer - and I hated it. I think OS-9 is the sole reason that I've once again dismantled the CoCo and stored it in the garage. Hopefully XENIX will prove to be a little more civil - and maybe even faster.

Before ending this column, I would like to thank the fine people who contributed to this issue. First, "USING THE I/O BUS OF THE MODEL 4" was written by Heine A R de Miranda from Holland. It is an excellent article, written in excellent English. His command of the language puts most of us to shame, doesn't it?

Roy Beck, once again, comes up with a winner. Crashing a hard disk is what we all have nightmares about. Now we have something to go by if the unmentionable should happen.

Dr. Allen Jacobs handed me a disk and said: Here is a little something about ALLWRITE". 'Little something', indeed. It was 40K of great information for users of that word processor, so we had to split the article into two parts.

Along with the above, to Jim King, Art McAninch, Frank Slinkman, Roy Garcia and Sam McFarland, a big thank you from TRSTimes.

# TRSTimes magazine

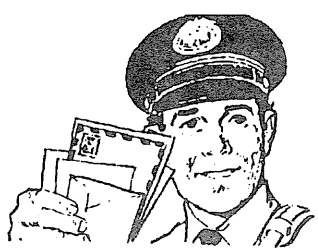
## Volume 3. No. 6. - Nov/Dec 1990

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

TRSTimes magazine is published bi-monthly by TRSTimes Publications. 5721 Topanga Canyon Blvd, Suite 4. Woodland Hills, CA. 91367. (818) 716-7154.
Publication months are January, March, May, July, September and November.

1990 subscription rates (6 issues):
UNITED STATES & CANADA:
$18.00 (U.S. currency)
ALL OTHER COUNTRIES:
$23.00 for surface mail, or
$29.00 for air mail.
(U.S. currency only)
Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible, a disk with the material saved in ASCII format. Any disk format is acceptable, but please note on label which format is used. Also, please make sure that your name and address is written legibly on both hardcopy and disk label.

# THE MAIL ROOM



## SUPERSCRIPSIT

I have a question that perhaps you or your readers can, or already have, answer. I am using a Model 4 with a 15 meg hard drive and a DWP-220 printer, running LS-DOS 6.3.1. On of the programs I use is SuperScripsit, Version 01.01.00. My question is, does anyone know of a way to change from one pitch to another within the same document?

In order to change the pitch of the printer, two control codes must be sent at the same time (27 & 14 for 12 pitch; 27 & 15 for 10 pitch), and from everything I've tried, SuperScripsit will not allow you to send two control codes at the same time.

To approach the problem from a different angle, I've tried moving a block of text with a different pitch than the original document, but that doesn't work either. As soon as the block is moved, it defaults to the original document's pitch.

If anyone has an answer to this problem, I would appreciate it if they would contact me.

Walt Danylak
115 Upland Rd.
Syracuse, NY 13207-1119
(315) 479-5879

*I am not a (Super) Scripsit user, but according to local sources it is not possible. Whoa, that can't be right - everything is possible on the TRS-80.*

*OK, SuperScripsit experts, let's hear from you.*

*Ed.*

## CLAN INFO

Good news for the many users of CLAN. This program now has a users group. Contact:

CLAN USERS GROUP
ARTHUR C. HURLBURT
1919 N. CLARK ST.
DAVENPORT, IA 52804
(319) 391-1606
This information is current as of 9/13/90.
Mike Lingo
Orange, CA

## MORE ON THE "RUMMY BUZZARD"

I have more information about the "RUMMY BUZZARD" mentioned on page 14 of the September/October issue. First of all, you don't need any special utilities to see the message; it can be done using the DEBUG utility supplied with the DOS. From the "TRSDOS Ready" prompt, just enter "DEBUG" to begin, "F" to enter file mode, "HERZ50/BLD for the filename, and ";" to view the second sector of the file. Second, there is more to the message than what is that sector. Back up to the first sector of HERZ50/BLD, and you'll see the "JOE:" which precedes the message.

I once read somewhere (sorry, but I can't find the exact source right now) that "JOE: HELLO YOU RUMMY BUZZARD" used to be the format pattern used by the FORMAT command when initializing a new diskette. It was used in the pre-release versions and perhaps in some early released versions. The format pattern was later changed to a bunch of hexadecimal 'E5' bytes, followed by a copyright message in the last sixteen of each sector. My guess is that either HERZ50/BLD was originally created on a disk using the old formatting pattern (and accidentally picked up some of it), or someone at Tandy got sentimental and hid the message there for old times' sake.

As for who Joe is, I don't know.

One last thing: I know the decision whether or not to continue publishing TRSTimes in 1991 is coming up. I would like to encourage you to continue. I think the TRS-80 community is still going strong and will continue to have interest in TRSTimes in the near future. TRSTimes has become my personal favorite TRS-80 magazine since 80 Micro dropped support. Thanks for all you've done.

John C. Fowler
La Jolla, CA

*I completely missed the "JOE:" in sector 1 of HERZ50/BLD. Now, as for the theory of the message being a previous format pattern, I must take issue with that. To my knowledge it just isn't possible, as the disk controller is only capable of accepting certain bytes, such as E5 B5 and a couple of others, for this special task. Eric Maloney of 80 Micro speculates that "Joe, you rummy buzzard" was used in a prototype format program's verification utility and got inserted at the end of each sector in the early TRSDOS 1.3 disks. He further speculates that the verification utility was later changed to insert "(c) 1980 Radio Shack".*

*Eric Maloney may have something there, but it does not explain why the message appears several times in sector 2 of HERZ50/BLD. I still feel that it is much more likely that one of Tandy's programmers got cute (or defiant) and put the message there 'just for the pure heck of it' during a late night session with the editor/assembler and a bottle of his favorite adult beverage. What strikes me interesting is that "JOE:" is not the last 4 bytes of sector 1; there are actually 32 more bytes following this message. I wonder what was there originally.*

*As you will see elsewhere in this issue, the decision on the future of TRSTimes has been made. TRSTimes will continue in 1991. I fully agree with you that the interest in the TRS-80 machines is still high. Who would have believed that in September 1987?*

*Ed.*

## TERMINAL EMULATION

The library association in Portland, Oregon has set up a communication link for access to its card catalog. Tandy's COMM software fails to connect.

The library indicates a choice of terminal emulations: ADDS Viewpoint, ADDS VP Enhanced Mode, Wyse 50, and VT-100.

Does anyone know of existing software for the Model 4 that will meet these requirements?

The library would be happy to have any information to pass on to Model 4 owners who call with enquiries. I'll get whatever information I recieve to them.

Charles Stones
Portland, OR

*As I am not knowledgable in the telecommunications field, I can only recommend that you try FASTERM. This is, as far as I know, the most advanced communications program for the TRS-80. You can get this program from the author: Mel Patrick, 13699 70A Ave, Surrey, British Columbia, Canada V3W 2J8.*

*Does anyone have the answer?*

*Ed.*

## NOTES ON RND

I thorougly enjoyed Delmer D. Hinrichs article "Is Your RND Really Random?" in the September issue of TRSTimes. The tests certainly show the TRS-80's as the capable machines they are. Many thanks to Mr. Hinrichs for a truly fine and interesting article.

I did find one small error. On page 25, about the middle of column 2, Model 4 Basic is grouped with MS-DOS Basic (GW-Basic) as being able to produce a random number from 0 to 1 by using RND, as well as being able to produce a random integer from 1 to 10 with INT(RND*10) + 1.

While this works fine in GW-Basic, it produces nothing but syntax errors in Model 4 Basic. The Model 4 should have been grouped with Model I & III as the RND syntax is identical for all three machines.

Robert G. Pinto
Brooklyn, NY

*Right you are. Somehow we inserted a tab before "Model 4". The tab should have been placed before "MS-DOS"*

*Ed.*

## WILDCARDS & MORE

Got the new TRSTimes the other day, and have had fun reading it. Thought I'd reply to your query on p. 16 and also follow up on a few other things.

First, regarding your query about backing up invisible and system files using the wildcard method. Since I have done this for quite some time on LDOS 5.3, I was surprised that an old-time hacker like yourself didn't know. But I guess we all have more to learn.

"Now, class, take out your 'TRSDOS Version 6' manual and turn to page 1-27." The first example given is a wildcard backup of all files from :0 to :1, including system and invisible files. "BACKUP $:0 :1 (SYS,INV)" is the example. I have used a variant of this in LDOS 5.3 to create what I call a "microsystem" disk. What is a "microsystem" disk? Well, I am now a "dual mode" TRS-80 and MS-DOS user. I have a Model III with two DS drives, and a 386SX clone. I also use a Zenith Turbo XT clone at work. When using SuperCross/XT to transfer files back and forth, I want to move as many files on one pass as possible. So I format an LDOS data disk at 43 tracks (the limit my machine can handle) in drive :1. I then run the following file, which I call MICROSYS/JCL:

```
ATTRIB :1 (NAME="SX3DISK")
BACKUP SYS8/SYS:0 :1 (S)
BACKUP SYS1$/SYS:0 :1 (S)
KILL SYS11/SYS.P3UF:1
BACKUP SYS2/SYS:0 :1 (S)
BACKUP SYS3/SYS:0 :1 (S)
BACKUP SYS4/SYS:0 :1 (S)
```

This puts exactly the minimum number of LDOS system files needed to handle the functions called for under SuperCross/XT. For LS-DOS, I assume that changing (S) to (SYS) and "KILL SYS11/SYS.P3UF:1" to "REMOVE SYS11/SYS.SYSTEM6:1" would create the same disk. I don't know if there is a way to backup ONLY invisible, or ONLY system, files without the filespec or partspec, but this definitely does the job. And, of course, the QUERY function is supported within BACKUP, so by adding Q=Y as a parameter, you could backup only what you wanted.

Speaking of "dual mode", I would find it helpful if TRSTimes would include coverage of the problems faced by users operating in dual mode; such as file transfers, how to interface a TRS-80 printer to a PC, etc.

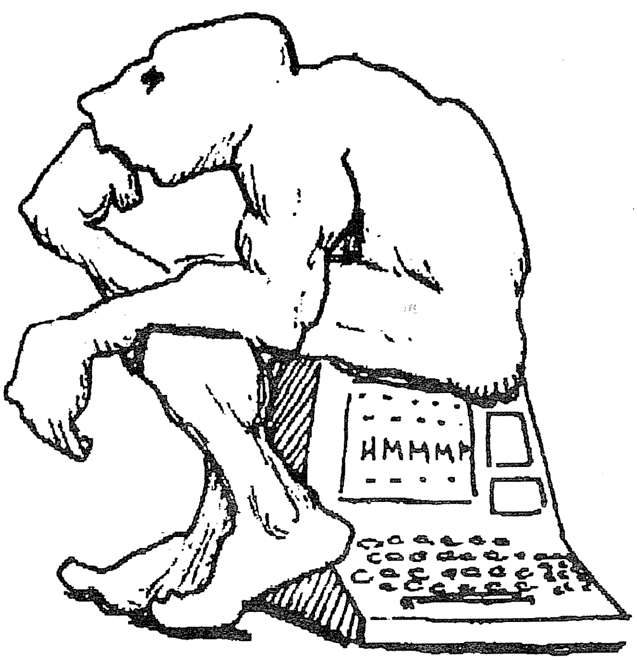
Michael E. Webb
Gainesville, FL

*Thank you for the information. Yes indeed, I have a lot more to learn. That is why I am still fascinated by computers after all these years - there's still more.*

*If we can come up with articles from the TRS-80 point of view of the "dual mode", I will consider your request. I just don't want to cover MS-DOS in this magazine. Maybe, when no one else want anything to do with Big Blue, I'll publish PCTimes. (don't hold your breath! ) Ed.*

# HOW TO SALVAGE A HARD DISK

By Roy T. Beck



## THE PROBLEM

I have a 15 Meg Hard Drive.
Recently I turned it on to AUTOBOOT.
It wouldn't AUTOBOOT.
The floppy boot disk for the drive wouldn't work.
I couldn't remember the assignment of cylinders and heads in the partitions.

About all I knew for sure was that the drive had TRSDOS 6.3.1 on its system partition, and that the drive has 6 heads and 306 cylinders. What to do?

As is usual in a situation like this, I immediately panicked! After a little while I settled down and began to take stock of the situation. I knew the drive was formatted under TRSDOS, and had many working files on it. I have TRSDOS and RSHARD6 by MISOSYS to work with. I also have SuperUtility and ToolBelt.

## ANALYSIS

What do we need to know to reestablish the missing partitions? Assuming we are using RSHARD6, the following questions must be answered:

- 1. The drive address. This is #1, unless you have more than one bubble connected to your hard disk controller.
- 2. The stepping rate. Consult drive data. (Most drives will accept 10 microseconds). Enter 10 for 10 microseconds.
- 3. Total cylinder count on the drive. Consult drive data.
- 4. Total head count on the drive. Consult drive data.
- 5. The starting head and total number of heads assigned to each partition.
- 6. The starting cylinder and total number of cylinders assigned to each partition.

With the above information known, the partition can be reestablished under TRSDOS 6.X and RSHARD6.

## DOS INFORMATION

What knowledge of the partitions can we recover from the hard drive, assuming we can get into it in some way? I proceeded to review the following documents in pursuit of information:

Soltoff's RSHARD Hard Disk Driver Manual
Soltoff's VRHARD Hard Disk Driver Manual
Soltoff's Programmer's Guide to LDOS/TRSDOS 6
Radio Shack's Technical Manual for the Model 4

In most respects these documents agreed. But there were some differences and I opted to follow the RSHARD Driver Manual for this problem. Be aware that differences exist, attributable to the different hard disk controllers in existence.

- 1. The first sector of BOOT/SYS contains a pointer in byte 02H which identifies the logical cylinder containing the directory. But there are two caveats to be observed. The first is that this ignores the DBLBIT, which is set in the event the cylinder account exceeds 203. If the DBLBIT is set, then the value in byte 02H must be doubled in order to make the physical cylinder offset agree with the logical value. More on DBLBIT below. The second is that the cylinder count is only within the unknown partition; it does not necessarily match the physical cylinders of the drive.

- 2. The first directory sector of any TRSDOS 6.X directory contains the GAT, the name of the disk, and a few crucial bytes of information which will allow us to reconstruct the partition. Bytes F9H through FFH correspond to the last 7 bytes of the DCT used to write the partition. Almost everything we need is in these 7 bytes.

- 3. Byte CCH gives the total number of cylinders in the disk (partition) stated in excess 35 notation. In other words, a zero means 35 tracks, 5 means 40 tracks, etc.

- 4. Byte F9H is the first of two flag bytes. Bits 1 and 0 are prefixed to byte FBH to form a 10 bit value. See byte FBH below.

- 5. Byte FAH is the second flag byte whose bit 5 is the DBLBIT. If this bit is unset (0) then each logical cylinder is one physical cylinder; if this bit is set (1), then there are more than 203 physical cylinders in the partition, and all references to logical cylinders must be understood to mean twice as many physical cylinders. Further, if an odd number of physical cylinders was assigned, then the number used will be reduced by one during the partitioning. That is, if you specify 303 physical cylinders, RSHARD6 will reduce the count to 302, and use 151 as the number of logical cylinders in its internal logic.
  Bit 3 is the hard drive address.
  Bits 2-0 represent the starting head number for the partition, with the heads numbered 0 to 7.

- 6. Byte FBH whose decimal value ranges from 0 to 1023. This is the starting cylinder of the partition. This value is affected by DBLBIT (see byte FAH). If DBLBIT is set, then the starting cylinder value must be doubled. As an example, 31H represents 49D, which is the 50th cylinder, with the cylinders numbered from 0-49. If DBLBIT is set, then the highest value will be 511, as the hard disk controller is limited to 1024 physical tracks, maximum.

- 7. Byte FCH contains the highest numbered logical cylinder in the partition. This value must be doubled if DBLBIT is set.

- 8. Byte FDH contains two numbers. Bits 7-5 contains a value which is one less than the highest head number assigned to the partition. The maximum number of heads is 8, so 3 bits holds the values 0-7.
  Bits 4-0 contain 0-1FH, corresponding to 0-31D. This is one less than the number of sectors/logical cylinder in the unknown partition.
  Example: Byte FDH contains 5FH, or 0101 1111. Rewrite this as 010 11111. We now see the values 2H and 1FH. The head count is 2+1=3; the sectors per logical cylinder is 1FH+1=20H=32D. The unknown partition has 3 heads assigned to it, and has 32 sectors per logical cylinder.

- 9. Byte FEH also contains two numbers. Bits 7-5 contain one less than the granules per logical cylinder, and bits 4-0 contain one less than the number of sectors per granule.

  Example: Byte FEH contains AFH, or 1010 1111. Rewrite this as 101 01111. We now see the values 5 and FH. The granules per cylinder is 5+1=6; the sectors per granule is FH+1=10H=16D. The unknown partition has 6 granules per logical cylinder and 16 sectors per granule.

- 10. Byte FFH contains the number of the directory logical cylinder. In an example, FFH contains 19H, or 25D; the directory is on logical cylinder 25D. This is also subject to DBLBIT.

## VERIFICATION

As a cross-check on our interpretations, the following equation will be satisfied by the numbers found above:

$$\frac{\textbf{Sectors per track=32}}{\textbf{Sectors per granule}} = \frac{\textbf{granules per logical cylinder}}{\textbf{number of heads assigned}}$$

Using the numbers from our examples, the equation comes out as follows:

$$\frac{32}{16} = \frac{6}{3} = 2$$

The above information includes most of the data we need to reestablish the unknown partition.

## MISSING INFORMATION

The missing pieces are the starting cylinder and starting head of each partition. Since the first partition must begin with head 0, cylinder 0 and sector 0, we have everything we need for the first partition, assuming we can access the values stored in the BOOT/SYS and the GAT sector.

Since partitions are located contiguously by RSHARD6, once the first partition is defined, then the next cylinder above it must be the first cylinder of the second partition, and so on up through the cylinders. You should be preparing a map on paper as you locate the partitions, and this will allow you to keep track of both the cylinder and head assignments in graphic form. When you are entirely done with the first head, then begin working upwards from the first unassigned cylinder of the second head, if any, and so on with the remaining heads.

## HARD DISK ACCESS

Now, how to gain access to the unknown drive?

SuperUtility was considered and immediately discarded, as it cannot address a hard disk. Kim Watt did not write it that way. But ToolBelt, also by Kim Watt can normally work with a hard disk. But not mine; not yet at any rate. Why? Well, ToolBelt has to work through your hard disk driver, and in my present fix, I didn't have any hard disk driver installed and working. But could I install one under these circumstances?

I decided I could create a dummy partition just for the sake of running ToolBelt, which in turn would allow me to explore the drive and locate the directories. How can I get away with this? Let's consider how RSHARD6 operates. RSHARD6 is implemented by setting up a drive control table and a driver in RAM without writing anything to the hard drive. So far, so good. But what partition can I assign to the drive control table? TRSDOS has some limitations which must be observed. One is that the cylinder count must not exceed 406. Another is that the partition size must not exceed approximately 13.3 Meg.

My general line of attack was as follows:

Create a fresh DOS disk and boot up from it. Create dummy partitions on the drive in increments of 200 cylinders, with all heads included. For example, drive :4 can be the first 200 cylinders, drive :5 the next 200, and so on. Yes, I know the DOS will allow up up to 406 cylinders, but this would mean adding and subtracting increments of 406 which is not as easy as adding and subtracting 200. I chose 200, which combined with 6 heads gave me 9.6 Megs per partition. I could create these dummy partitions without altering the contents of the hard drive, because the partitioning information is only stored in RAM, not on the disk. After creating dummy partitions, (which need not match the lost partitioning at all), I could begin to explore the drive via ToolBelt.

## CAUTION

One very important precaution to be observed:

**DO NOT FORMAT THE DUMMY PARTITION,
OR ALL IS LOST!**

By stepping through the cylinders, looking only at the 0 sectors, it is easy to find the BOOT sector of any real partition which happens to fall in this dummy partition. And where is the directory in a partition? Use the pointer at 02H of the Boot sector. If this does not bring you to the directory, double the track number and look there, as the DBLBIT may be set because the physical cylinder count exceeded 203. When you find the GAT, record and analyze the bytes as above. If there is more than one real partition in your dummy partition, proceed as above with each partition to find its GAT data, and repeat as required.

## MAPPING

At this point, you should begin to map out the partitions as you locate them. I use a sheet of quadrille paper with the cylinder numbers running vertically, and the head numbers running horizontally. By marking the partition boundaries on this chart as I discovered them, I was quickly able to plan my attacks on the unidentified areas of the drive. When all of the first head of the hard drive is accounted for, begin on the next head, continuing as indicated until all the drive is accounted for.

Note, you must reboot to get rid of a partition, dummy or otherwise. SYSTEM (drive = n, DISABLE) is not good enough, as it does not release the disabled cylinders for reassignment.

There is a limit of 256 entries in a directory (limited by the space available for the HIT), and with 8 entries per sector, no more than 32 sectors can be part of any one directory. When the GAT and HIT are added, then the largest directory will consist of 34 sectors. With TRSDOS always having 32 sectors per track on hard drives, then the directory can exceed one track, extending onto a second head if more than one head is in the partition, or extending onto the next physical track.

## diskDISK

Caution! Not every directory you will discover is part of a regular partition directory. diskDISK creates a small directory structure for each of its subdirectories. These can appear anywhere, without obvious rhyme or reason, so such "peculiar" directories should be considered as possible diskDISK subdirectories. They will be smaller than the typical HD directory, and need not begin on the first head of a partition.

## RESETTING PARTITIONS

When you have deduced the total partition setup on a drive, it is time to find out if it is valid. Reboot the machine again, and partition it in accordance with what you have learned in your exploration.

To find out if your deductions are valid, call the directory of each new partition you have discovered and reestablished. If the directory display makes sense, you have probably correctly set the number of cylinders and heads when you partitioned. A rubbish display means you have something set incorrectly.

The next test is to attempt to load or execute a program from the directory just found. If the program won't load due to "load file format error", you still have something wrong. Try again. Remember to reboot the machine before each new partition reconstruction.

## SYSGEN

When all the partitions (there can be more than 4) are identified, SYSGEN the active partitions onto your new floppy boot disk.

Don't forget to backup a fresh copy of the DOS onto the hard disk when you have everything working again. Failure to do this could bomb the machine when you try to swap drive 0 onto the hard disk.

Now make the system partition on the hard disk the active system drive :0 and then create a new boot disk by SYSGENing onto the DOS in the floppy drive. Also, record the partitioning information (drive, head and cylinder counts) on paper and file it where you can find it next time!

## CONCLUSION

I assure you, it is a nice feeling of accomplishment when the directory command again works normally and the machine will again boot and execute the programs saved on the hard disk!

In my actual case, I only had to reestablish the original hard disk system partition correctly, backup a fresh copy of TRSDOS onto it, and then reboot with my old boot floppy, which immediately restored all the missing partitions I was describing above. My original problem was that I had previously unknowingly garbled the DOS on partition :0 of the hard drive. You can be sure I immediately documented the heads and cylinders on to a neatly drawn map, which is now in my machine's "working manual", which stays with the machine and contains the peculiarities of its hardware and software. I was fortunate in that I had write protected my boot floppy, and so I could trust it to be valid after I restored a fresh copy of TRSDOS onto the system partition of the hard drive.

In any event, the episode forced me to learn something new, and I hope this may save some of you the headache of starting a large hard drive from scratch!

## PROGRAM SOURCE

Note: All the above applies strictly to TRSDOS 6.X and LDOS 5.X. Kim Watt's ToolBox will allow access to LDOS 5.X in the same way that ToolBelt accesses TRSDOS 6.X. Both of these programs are now available from MISOSYS, who bought out PowerSoft.

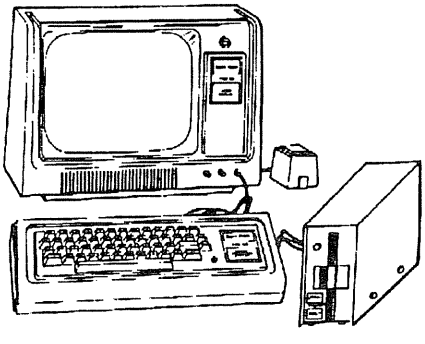
While DOSPLUS is said to be quite similar in its directory and file structure, I am not acquainted with it, and cannot advise on recovery of DOSPLUS partitions, or partitions established by other DOSes. I would love to see articles like this by other knowledgeable people on the other DOSes. Do I hear any volunteers?

*-Roy-*

# UNPROTECT TRSDOS 2.3

## MODEL I

By Lance Wolstrup



In previous issues of TRSTimes I have made it fairly well known that I am not a fan of password protection. As my Model I is once again working, I thought it might be fun to strip TRSDOS 2.3 of this irritating feature.

The password is checked whenever a file is requested to open. If the password is correct, DOS will proceed to open the file; on the other hand, an incorrect password will make DOS refuse to open the file, giving the 'FILE ACCESS DENIED' message instead.

All of this takes place in SYS2/SYS. The decision whether to open the file or give the error message is made according to the status of the Z flag. If Z status, then the password matched and the file is opened. If NZ status, the password did not match and the error message is issued.

When SYS2/SYS is in memory, locations 4EACH and 4EADH contain 28 and 21, which translates to the assembly language mnemonics JR Z,4ECFH. In other words, if the passwords matched we jump directly to 4ECFH where the file will be opened with full access. By jumping directly to 4ECFH, we jump over the code at 4EAEH which denies the file access.

Thus, if we change the conditional JR Z,4ECFH to the unconditional JR 4ECFH, we will jump over the error portion of the code and open the file with full access - regardless whether the password matched or not.

The byte 28 means JR Z; the byte 18 means JR. It now becomes plain that we need to change the byte at 4EACH from 28 to 18. However, since SYS2/SYS is a DOS overlay, it is very transitory; that is, it is loaded from disk whenever it is needed and the memory it occupies is reused by the other SYS files when they are needed. The bottom line is that changing the byte at 4EACH in memory will not do us any good. We need to change the byte in the SYS2/SYS file itself.

Unfortunately, TRSDOS 2.3 protects itself by not having a 'patch' utility included with the DOS. So, we must use a 'zap' utility, such as Super Utility, or other similar program to do our dirty work for us. If you have such a utility available, select the track and sector mode and choose track 16, sector 6. The byte we want is located at position 63H. Choose the modify mode, place the cursor over the 28 at 63H and change it to 18. Now write this sector back to disk and the job is done - no more password protection in TRSDOS 2.3.

Should you want to re-enable the password protection, follow the instructions above, but change the 18, which is now at location 63H, back to 28.

If you do not own a 'ZAP' program, but instead have an editor/assembler, the following assembly language program will do the trick. Key it in, assemble it, and then run it to automatically remove the password protection.

When you run the program, you will be prompted to insert a backup of the original 2.3 DOS disk in drive :1. Pressing Q will return you to TRSDOS; pressing <ENTER> will read track 16, sector 6 of the disk in drive :1 into a buffer. The byte at relative position 63H in the buffer is then compared to 28H. If that byte is not 28H, the program aborts with an error message, leaving the disk in drive :1 unaltered. If byte 28H is found at relative position 63H in the buffer, it is changed to 18H and the buffer is then written back to the disk in drive :1. A message tells you that the operation was successful, and you are returned to DOS.

Do keep in mind that you must boot the system with the altered disk before the passwords are disabled.

```
00080 ;NOPW/SRC
00090 ;for Model I TRSDOS 2.2 & 2.3 only
00100          ORG    7000H
00110 START    CALL   1C9H         ;cls
00120          LD     HL,TOPMSG    ;display topmsg
00130          CALL   4467H
00140 ASK      CALL   49H          ;wait for key press
00150          CP     81           ;is it 'Q'
00160          JR     Z,QUIT       ;yes - so quit
00170          CP     113          ;is it 'q'
00180          JR     Z,QUIT       ;yes - so quit
00190          CP     13           ;is it (ENTER)
00200          JR     Z,READ       ;yes - read sector
00210          JR     ASK          ;not Q, q or (ENTER)
00220 READ     LD     C,1          ;drive :1
00230          LD     DE,1006H     ;track 16, sector 6
00240          LD     HL,BUFFER    ;point hl to buffer
00250          CALL   46DDH        ;read disk sector
00255          OR     A            ;is A 0
00260          JR     NZ,RERROR    ;no - abort on error
00270          LD     A,(BUFFER+63H) ;prot byte to a
00280          CP     28H          ;is it 28h
00290          JR     Z,FOUND      ;yes - we found it
```

```
00300          LD    HL,NOGOOD ;no - so display
00310          CALL  4467H     ;error message
00320          JR    QUIT      ;and abort
00330 FOUND    LD    HL,GOTIT  ;display the gotit
00340          CALL  4467H     ;message
00350          LD    A,18H     ;stuff unprot byte
00360          LD    (BUFFER+63H),A ;into buffer
00370          LD    C,1       ;drive :1
00380          LD    DE,1006H  ;track 16, sector 6
00390          LD    HL,BUFFER ;point hl to buffer
00400          CALL  46E6H     ;write buf to sector
00405          OR    A         ;is A 0
00410          JR    NZ,WERROR ;no - jump on error
00420          LD    HL,DUNMSG ;display success
00430          CALL  4467H     ;message and
00440 QUIT     RET             ;return to dos
00450 ;
00460 RERROR   LD    HL,RDERR  ;display read error
00470          CALL  4467H     ;message
00480          JR    QUIT      ;and abort to dos
00490 ;
00500 WERROR   LD    HL,WRTERR ;display write error
00510          CALL  4467H     ;message
00520          JR    QUIT      ;and abort to dos
00530 ;
00540 TOPMSG   DEFM  'NOPW - (c) copyright 1990 by '
00550          DEFM  'Lance Wolstrup'
00560          DEFB  0AH
00570          DEFM  'disable password protection on '
00575          DEFM  'TRSDOS 2.3'
00580          DEFB  0AH
00590          DEFB  0AH
00600          DEFM  'Insert copy of the TRSDOS 2.3 '
00610          DEFM  'master disk in drive :1'
00620          DEFB  0AH
00630          DEFM  'press (ENTER) to disable the '
00640          DEFM  'password    protection - or'
00650          DEFB  0AH
00660          DEFM  'press (Q) to quit'
00670          DEFB  0AH
00680          DEFB  13
00690 RDERR    DEFB  0AH
00700          DEFM  'UNABLE TO READ SECTOR'
00710          DEFB  0AH
00720          DEFM  'FUNCTION ABORTED'
00730          DEFB  13
00740 WRTERR   DEFB  0AH
00750          DEFM  'UNABLE TO WRITE SECTOR'
00760          DEFB  0AH
00770          DEFM  'FUNCTION ABORTED'
00780          DEFB  13
00790 NOGOOD   DEFB  0AH
00800          DEFM  'INCORRECT BYTE FOUND'
00810          DEFB  0AH
00820          DEFM  'FUNCTION ABORTED'
00830          DEFB  13
00840 GOTIT    DEFB  0AH
00850          DEFM  'CORRECT BYTE FOUND'
00860          DEFB  0AH
00870          DEFB  13
00880 DUNMSG   DEFB  0AH
00890          DEFM  'TRSDOS 2.3 DOS DISK IN '
00895          DEFM  'DRIVE :1 IS NO LONGER '
00900          DEFM  'PASSWORD PROTECTED'
00910          DEFB  0AH
00920          DEFB  13
00930 ;
00940 BUFFER   DEFS  256
00950 ;
00960          END   START
```

I realize that not everyone has a 'zap' program or an editor/assembler readily available. So, for those people, here is a Basic program that will do the same thing.

The program, NOPW/BAS, handles the problem a little differently than the assembly language version. Rather than reading and writing to specific tracks and sectors, the program treats SYS2/SYS as a normal file (which it is). The password (NV36) has been documented in most of the TRS-80 publications of the seventies and eighties, so that is not a problem. We then open SYS2/SYS as a random file and, since the password protection byte is the 100th byte of record 2, we GET 1,2. To make sure we are doing the right thing, byte 100 is tested to see if it contains 40 (28H - JR Z). If not, we KNOW something is wrong. We therefore display an error message, close the file, and terminate the program. If the byte does equal 40, it is changed to 24 (18H - JR) and the record is written back to disk with PUT 1,2. The file is then closed, and we display the success message.

```
5 'NOPW/BAS FOR MODEL I TRSDOS 2.3 ONLY
10 CLEAR 500:CLS
20 PRINT"NOPW/BAS - (C) COPYRIGHT 1990 BY
LANCE WOLSTRUP"
30 PRINT"DISABLE PASSWORD PROTECTION ON
TRSDOS 2.3"
40 PRINT
50 PRINT"INSERT COPY OF THE TRSDOS 2.3
MASTER DISK IN DRIVE :1"
60 PRINT"PRESS <ENTER> TO DISABLE
PASSWORD PROTECTION - OR"
70 PRINT"PRESS <Q> TO QUIT"
80 I$=INKEY$:
IF I$="Q" OR I$="q" THEN CLS:END
ELSE IF I$=CHR$(13) THEN 90
ELSE 80
90 OPEN"R",1,"SYS2/SYS.NV36:1"
100 FIELD 1,255 AS A$
110 B$=A$
120 GET 1,2
130 IF ASC(MID$(B$,100,1))< >40 THEN PRINT:
PRINT"INCORRECT BYTE FOUND":
PRINT:CLOSE 1:END
140 MID$(B$,100,1)=CHR$(24)
150 LSET A$=B$
160 PUT 1,2
170 PRINT:
PRINT"TRSDOS 2.3 DOS DISK IN DRIVE :1 IS NO
LONGER PASSWORD PROTECTED":
PRINT
180 CLOSE 1
190 END
```

# ALL ABOUT ALLWRITE pt. 1

## (well - almost!!)

By Dr. Allen W. Jacobs

Whenever we use a program frequently, we become more familiar with it. This usually means that we find better ways of accomplishing the tasks for which we use the program. We usually do this by experimenting and by hearing or reading about helpful techniques and methods. From these sources, we develop still further refinements in our use of the program. This process of improvement is often called experiencing a "learning curve".

Since word processors are acknowledged to be the most widely used programs in the realm of microcomputers, it is reasonable that word processors benefit a great deal from the learning curve effect. I use Allwrite. I am writing this article for selfish reasons. I hope that I will get a like amount of feedback from other users of Allwrite in order to push my "learning curve", "around the corner".

For beginners, this is not a tutorial on Allwrite. The manual is quite complete for that purpose, although it is a bit wordy and not as well indexed as I would like it to be. My first wish for an improvement in Allwrite would be to have the manual in some form of hypertext format on line. There is a HELP feature but it is only marginally useful. Anybody learning to use Allwrite still has to READ THE BOOK.

Don't feel slighted by this requirement relative to the MS-DOS or MAC worlds. There is a large aftermarket industry in "advanced" manuals out there for MS-DOS word processing programs such as Word Perfect, Word Star, and Microsoft Word. There is even a monthly magazine devoted only to Word perfect. Check out your nearest book store and you will see. Any full featured word processor is capable of executing many options in many ways. No matter what they tell you, you have to know what editing and formatting commands your word processor has or, at least, that those commands exist in order to use them. To fully use any word processor, there is no shortcut to knowing what it can do. Allwrite is as full featured as the best of them.

After looking over a Word Perfect 4.2 manual, I found nothing that Allwrite could not do except in text math and parallel columns. Admittedly, Word Perfect 5.0 with its graphics import capability, multiple font handling on a laser printer, mathematical symbol, and automated table creation capabilities surpasses Allwrite. However, I have found that if you create a document with any impact type of printer (dot matrix, daisy wheel), you will be able to create it with Allwrite. Note that laser capability is also currently available for Allwrite but I do not know how extensive it is.

The following is a disorganized collection of uses, techniques, and hints that help make Allwrite more useful for me and hopefully for you. I even have some criticisms that maybe someday I, or one of you brave programmers, may correct.

Don't look for any explanation of Allwrite variable logic in this article. I don't use Allwrite for form letters. Regrettably, Allwrite's logic only seems to work on Allwrite variables which are only used in form letters, however, I have never tried this out. If someone is knowledgeable about this, let's hear from you.

My hints for document testing (ie. review to screen) are mostly self-evident, but may have been overlooked for some time. The more obvious hints are hardware hints, but not all. My advice in regards to hardware is to use the fastest disk access you can. Allwrite works best with a hard disk but even better with a ram disk. This gives you the best performance, saves you a lot of time, and reduces disk drive wear. Regrettably, the Allwrite editor and formatter will not fit into the standard 64K memdisk of the Model 4. It might have, but the memdisk is only 57K when formatted.

If you have a 128K machine and a Radio Shack Hi-rez Board, there is a public domain program called Grafdisk that adds the memory in the Hi-rez board to the standard memdisk. This configuration provides you with a 96K ram drive (less with a Microlabs board). That is enough to put all of Allwrite into memory, after it is installed, but not all the system files. Therefore system calls are still made to the disk. Since I have a hard disk, I have not bothered with this option myself and so I have not attempted to generate a "minimum system" for diskless operation. If somebody has done this, please let us in on the result.

Admittedly, the formatter is slow to load and execute. The reason it is so slow is that it is capable of so many options; thus, it requires a lot of time to execute them. Remember, it is doing what a much faster MS-DOS machine does in less time. Allwrite only seems slower because most MS-DOS machines today are very fast compared with a Model 4. The Allwrite formatter is much larger than most TRS-80 programs (about 40K) and therefore takes a long time to load. For comparison, the Model 4 DOS is only 12K.

Software hints for document testing (ie. review to screen) consist of adding ";vi" or ";vs" and also ";pi 12" to the beginning of your document, while it is in development. It will look neat to your view and will fill the screen. You will have to delete the ";pi 12" command and review the document without a neat right edge screen if you are going to use proportional pitch in the printed draft. I used to retype ";vs ;pi 12" on the options line every time I wanted to review the document by printing it to the screen. Now, I simply type them once at the beginning of the document. When I want to print the document, I simply insert comment labels (";cm" ) in front of these instructions and they are subsequently ignored. The ;cm command is just like a REM statement in basic. It allows you to prevent code execution while leaving the code in place, as documentation.

While I don't write form letters, I do often write individual letters. To do this I use a form file and a skeleton file. Which I call a letter skeleton. It serves four purposes. It saves typing, prevents copying errors of the addressee and my return address, permits storing a number of different "callable" formats in one file, and it allows letters to be shorter. I have it set up to address two sizes of standard envelopes, if your printer can address an envelope directly. If you use labels, you can alter these files any way you see fit. They are presented to relay the concept that much of the form of a letter çan be "automated" and indeed need not even be present in the letter being written. While your needs and preferences may be different, my skeleton letter hopefully provides a method you can adopt for yourself. I guess that this method is the forerunner of MS-DOS Ventura Publisher's "style sheets".

The best way to explain this is to illustrate it. Therefore, I have have two files which I use to write a letter. The first is a list of labeled "imbed" files. The second is the skeleton of a letter that I use to fill in the body of the letter I want to write. I call these files LETTER/SKL, SKL, and SK. SK is simply a copy of SKL from which I deleted a lot of the extra comments. I first made SKL and later found that I was familiar enough with it not to need comments which I found myself deleting after writing each letter, to save space. SK is the shortened file I currently use. If you do not need some of the options I use, you can shorten it even more. It's all personal preference.

The option that makes it all possible is the "imbed" command. Imbed commands tell the formatter to stop printing the text in the current file and to proceed to a label or other specified text in another file! When that text ends, the formatter then proceeds with the original text. Imbeds can even be nested to four levels but I have never had the need or nerve to try that.

To follow the action, start with either the SK or SKL file and when it says ";im", go to the indicated file and to the label within that file. Print what's there until the ";en", which means "end of labeled text" of that label occurs. Then, go back to SK or SKL to the line below where you left the file.

If all this is too much for you at this time, just substitute your return address for the example and use the files "as is". You may have to alter them just to use them in your situation and you are sort of "stuck" with my format unless you develop the ability to change it. At least you know that there is an alternate way of creating a new business style letter other than copying a previous letter, renaming the file, and editing it. In some cases, I do that also. Whatever works.

## LETTER/SKL

```
;lb letstart
;ll 64
;in 40
;fo off
;cm Your return address goes here.
5721 Topanga Canyon Blvd.
Suite 4
Woodland Hills,
CA  91364
;in
;sk2
;tb + 41
;cm Addressee and date with " + " separator goes after this
comment line (in SKL).
;en letstart
;lb letend
;sk
Thank You Very Much.
;fo off
;sk
;in 41
Sincerely,
;sk3
Lance Wolstrup, Editor
TRSTimes magazine
;in
;fo
;pa
;en letend
;lb letencl
;sk
```

```
Thank You Very Much.
;fo off
;sk
;in 41
Sincerely,
;sk 3
Lance Wolstrup, Editor
TRSTimes magazine
;sk 2
;in
encl. (ck.)
;pa
;en letencl
;lb addenv
;ls 1
;pn off
;in 0
;fo off
;ll 30
TRSTimes magazine
5721 Topanga Canyon Blvd.
Suite 4
Woodland Hills,
CA  91364
;en addenv
;lb addenv2
;ll 80
;in 40
;fo off
;sk 5
;cm address of addressee is inserted here.  In SK or SKL.
;en addenv2
;lb addenv2s
;ll 60
;in 30
;fo off
;sk 3
;cm address of addressee is inserted here.  In SK or SKL.
;en addenv2s
;lb addenv3
;pa
;en addenv3
```

## SKL

```
;cm THIS SKELETON PUTS THE START AND END ON A BUSINESS LETTER IN A #10 OR SHORT ENVELOPE.
;cm ALL YOU HAVE TO DO IS INSERT THE BODY AND ADDRESSEE, AND DELETE THE COMMENTED LINES.
;im letter/skl,letstart
;cm insert the return address directly on the line below with a "+" and the date, with no spaces.
;cm (IE.: "XYZ Co.+Month 99, 1999".) No other punctuation.
__________________________
__________________________
__________________________
;sk
Dear ____________:
;fo on;cm DELETE THIS LINE FOR RAGGED RIGHT EDGE.
;ls 1.5;cm DELETE THIS LINE TO SHORTEN LETTERS.
   ;pp
;cm Start or insert the body of the letter directly on this line & RENAME FILE
________________________________________.
;cm Remember to choose one letter ending and delete the other,
;cm with only an enter after the end of the body of the letter text.
;im letter/skl,letend
;im letter/skl,letencl
;cm For envelopes, delete either addenv2 or addenv2s and leave the other.
;cm "2" is for size #10 envelopes while "2s" is for the shorter size.
;im letter/skl,addenv
;im letter/skl,addenv2
;im letter/skl,addenv2s
;cm Insert the address of the addressee here, minus the date.
;im letter/skl,addenv3
```

## SK

```
;im letter/skl,letstart
____________________ +___
__________________________
__________________________
;sk
Dear ____________:
;fo on;cm DELETE FOR RAGGED RIGHT EDGE.
;ls 1.5;cm DELETE TO SHORTEN LETTERS.
;pp
RENAME FILE AS YOU START TYPING OVER THIS LINE.
______________________________
;im letter/skl,letend
;im letter/skl,letencl
;im letter/skl,addenv
;im letter/skl,addenv2
```

```
;im letter/skl,addenv2s
;cm ADDRESSEE HERE
;im letter/skl,addenv3
```

One of the uses I put my "business" letter form to is that I often place orders for hardware, software, etc. The only thing "special" about ordering is that I use "print time tabbing", as suggested in the Allwrite manual. I notice that less frequent users of Allwrite use "on screen" tabbing. It appears to be more "WYSIWYG" than print time tabbing, but is really less flexible and nearly impossible to use with proportional spacing. Again, the best way to see how it works is to experiment with it.

My typical order form uses the best form of tabbing, which is relative tab stop. They are the most flexible. This letter to MISOSYS is a pretty good example. In order to use it, you must have LETTER/SKL on any available drive. It was derived from SK but it can be edited to be to anybody for anything. If you wanted to turn it into a billing of some sort, the method is obvious. Since Allwrite doesn't have on screen math, all of that must be done with a calculator. If you were doing a lot of these letters, it might be better to use a data base and a form letter with Allwrite. My example is just for the casual home user.

## MISOSYS/LET

```
;cm ;vs
;cm ;pi 12
;im letter/skl,letstart
Misosys, Inc.+August 31, 1990
P.O. Box 239
Sterling, VA  22170-0239
;cm Telephone number: 1 (800) MISOSYS
;sk
Dear Sirs:
;sk
;fo on
I would like to purchase the following:
;sk
;fo off
;tb+  +25,+15d,+15d
TMQ Disk Notes+Vol 4 No 4+ +$10.00
TMQ Disk Notes+Vol 5 No 1+ +$10.00
Misosys Quarterly+Vol 5+ +$30.00
DoubleDuty Version 2.6+ + +$49.95
   (or highest current version)
;cm LS DOS 5.3.1+Update+ +$15.00
Shipping & Handling+ + +$5.00
+ + +TOTAL $104.95
;fo on
;pp
Enclosed is a check for $104.95. My customer number is
#XXXXX. Please ship the above listed items to me
through the above address.
;im letter/skl,letencl
;im letter/skl,addenv
;im letter/skl,addenv2
Misosys, Inc.
P.O. Box 239
Sterling, VA  22170-0239
;im letter/skl,addenv3
```

The thing to remember is that tabs are measured in inches and tenths of an inch and that those measurements are INDEPENDENT of pitch. It is also helpful to remember that if your text runs past a tab, the next tab will be at the tab stop AFTER the one you intended. If you can't change the text or the tab, try changing the pitch. It is a poor third choice but sometimes the only one you have. I almost never use it.

You will notice that I am prepared to order the LS DOS 5.3.1 update when it comes out but I did not, as yet. I placed it in a comment line so that it's there and so that I will not forget to order it, at a later time. I often place the telephone number associated with the addressee in a comment line.

Until relatively recently, I have avoided using the soft key features of Allwrite. These are analogous to the highly touted macro keys in Word Perfect. We of long time TRS-80 use know of this general capability as keystroke multiply (KSM), among other names. It has been a feature of many DOS'es and add-on programs. Basically, it just means assigning a number of user programmable keystrokes to a specific soft key;

....but that will have to wait for next issue's installment.

# WHO KNOWS Z-80?

## 'ASSEMBLY LANGUAGE volume X: REFERENCE by Chris Fara'

reviewed by Lance Wolstrup

- Bo knows baseball!
- Bo knows football!
- Bo knows Z-80 assembly language???

While I have no argument about football and baseball, I have serious doubts that the famous Bo knows assembly language for the TRS-80.

However, I will tell you who does know:

- CHRIS KNOWS.

Chris Fara of Microdex, that is, and he has just completed his latest work, a book called:

ASSEMBLY LANGUAGE
FOR THE TRS-80 AND OTHER Z-80 COMPUTERS
VOLUME X: REFERENCE
Published & distributed by: Computer News 80
P.O. Box 680, Casper, Wyoming 82602-0680

The book is priced at $12.95 and, true to its title, it is a complete reference for the Model I/III & 4 assembly language programmer, or 'would be' programmer. It is divided into three sections.

The first section (A) is a detailed reference of the Z-80 instruction set with a description of each instruction.

This is where most books about TRS-80 Assembly language fail miserably. All too often an author, though a brilliant programmer, will ramble on for pages at a time, trying to impress the reader with his vast and in-depth knowledge. The result is usually that the reader is utterly confused, having more questions raised than answered.

Not so in this book. Chris doesn't try to impress anyone by talking over their heads. Quite the contrary, he demonstrates his knowledge of the subject by explaining each instruction in clear, concise and, most importantly, in plain and easy to understand English. Now, that's impressive.

Along with the explanation, each instruction also shows the flag settings, while many examples are given to further clarify the usage. Each example shows the byte length and the T-state (execution speed) of the particular version of the instruction.

Section B is called "Technical notes". Don't let that heading scare you! Instead, this section should be titled: "Technical notes - in English".

Here Chris covers Registers, Flags, Memory, Source code, Labels, Instructions, Pseudo-ops, Operands, Operators, Addressing modes, Assembled code, Execution speed, and Instruction patterns.

This material can be found in most any book about Z-80 programming - certainly Bill Barden, Rodnay Zaks, Hubert Howe and Lance Leventhal covered this in their various writings - BUT, somehow Chris Fara manages to bring the subject down to earth. He doesn't overwhelm the reader with long-winded, techo-speak drivel - he explains, simply and effectively.

For example, how many of you out there know what the N-flag does? Hmmm... I don't see too many hands raised! Well, don't feel bad - I didn't know either - until I read Chris' explanation.

My only complaint about section B, is that the subject of Pseudo-ops is treated rather skimpily. After giving just one example, the author advises the readers to study their particular assembler manual for further information.

Now, I understand that covering pseudo-ops could turn out to be a book in itself - and a rather large one at that. Pseudo-ops, as Chris explains, are not Z-80 instructions; rather, they are specific instructions to the assembler, and since there are many assembler packages around for the TRS-80's, these instructions vary from package to package, making it impractical to give much detail. I do wish a couple more examples would have been included. However, while a complaint, it is a minor one.

Section C is the part I enjoy the most. Having two parts, the first is a listing of all Z-80 instructions with all possible combinations of operands in alphabetical order; the second part is a listing of the same instructions and operands, but in the numerical order of the machine code generated by the assembler.

This information, once again, can be found elsewhere. I have used Hubert Howe's listings in the back of his book "TRS-80 Assembly Language" for years to translate those weird hexadecimal numbers you see in Debug into actual Z-80 mnemonics. However, that book has now been quietly put back in the bookcase, replaced by Chris' clear, complete, and much easier to read listings.

Will "ASSEMBLY LANGUAGE FOR TRS-80 AND OTHER Z-80 COMPUTERS, VOLUME X: REFERENCE teach you assembly language? No, probably not. That is not its purpose. It is a reference and, as such, will serve as an extremely useful tool to anyone with an interest in Z-80 assembly language. It will help a programmer to quickly and efficiently look up needed information, especially the alphanumeric listings in section C will be of value. The budding programmer will find much comfort in the easy to understand description of the instruction set, as well as the explanation of the flags.

This is a fine and useful book, and I'm pleased to see, even at this late stage of the life of the TRS-80, that quality products are still being made available for our machines.

Keep 'em coming Chris.

# HINTS & TIPS

## HOW TO BEAT 'QUEST FOR THE KEY OF NIGHTSHADE'

### Model I/III

By Lance Wolstrup

The February 1983 issue of 80 Micro featured 'Young Programmers Contest'. David Schmidt (age 16, from Loveland, CO) won the grand prize with an unusually fine graphic-adventure game called 'Quest for the Key of Nightshade'. This is one of the toughest games I have seen for the TRS-80. My son, Alan, has played 'Nightshade' numerous times, and each time he has come away winless and frustrated. The other night he began to play it again and, as usual, died over and over. Well, I felt sorry for him, so I did my fatherly duty - I helped him cheat! I modified the code so he would no longer be bothered by the fatigue factor, never run out of food nor water, and best of all, he wouldn't die. Even with this amount of help, the game will still take many hours to play, while remaining fun. Below are the modifications:

Change line 320 to:

```
320 FA%=0:FO%=25:WA%=25
```

(to keep the fatigue level at 0, and the food and water at 25)

Change line 6020 to:

```
6020 FA%=0:GOSUB 200:RETURN
```

(to keep fatigue level at 0)

Change line 7520 to:

```
7520 F=0:T=T-F:IF T<0 THEN WR%=WR%+T:
T=0
```

(simply insert F=0 at the very beginning of the present code - this keeps you from getting killed)

You normally need 5 pieces of a special key to gain entrance to Castle Night Shade. If you feel this is too many, you may change the code at the very beginning of line 2910 to:

```
2910 IF K<NUMBER THEN etc...
```

(NUMBER should be replaced with the actual number of the key-pieces you wish to collect in order to gain entrance to the castle. I used 3 for Alan's game - and it kept him busy for hours)

Just goes to show you - if you can't win by conventional methods - CHEAT! However, please confine this policy to computer games.

## (SEMI) AUTOMATIC /BAK

by James E. King, MSEE, C10

One of the nicest features on a mainframe computer was the fact that when you saved a program that you were working on, the system automatically copied the previous disk version to a new file and gave it the extension /BAK. If you messed up the 'improvent/change' to the current file, at least you had your last /BAK version.

I have long wished for this feature in my operating systems, but the writers of the miscellaneous TRS-80 DOSes just never obliged. I was on my own...

Well, after a couple of thinking sessions, I came up with a way to do it almost automatically in MULTIDOS.

This DOS has the capability of doing any CMD from BASIC, so the code is something like this:

```
CMD"COPY PROGNAME PROGNAME/BAK":
SAVE"PROGNAME
```

This did it. The 'almost automatic' part is that with this command DOS thinks you are saving to a different disk in drive 0, and asks you to press <ENTER> when that disk is in the drive. Just press <ENTER> at the prompts.

In my own style of programming I always begin by initializing the program in line 0 and then jump over the common subroutines to a menu located at line 99. This menu is followed by a series of IF-THEN tests, lasting until line 199.

```
0 CLEAR 1000:
CLS:
DEFSTR Z:
DEFINT I-N:
ZN="PROGNAME/BAS:
GOTO 99

7 IF LEN(Z) THEN FOR L=1 TO LEN(Z):
N=ASC(MID$(Z,L,1)):
MID$(Z,L,1)=CHR$(N+(32*(N>96))):
NEXT:
GOTO 98
ELSE 98

8 Z=INKEY$:
IF Z="" THEN 8
ELSE IF Z=CHR$(31) THEN END
ELSE 7
```

**from 9 to 96 are additional common subroutines**

```
97 I=INSTR(ZN,"/"):
IF I THEN ZU=LEFT$(ZN,I)+"BAK"
ELSE ZU$=ZN+"/BAK"

98 RETURN
```

**beginning of main program**

```
99 PRINT ZN$+": <I>nput, <L> oad, <D>isplay,
Sa<v>e, <P>rint"

101 GOSUB 8 '(Inkey$ routine)

110 IF Z="$"THEN GOSUB 97:
ZC=ZN+" "+ZU:
PRINT"Copying ";ZN;" to ";ZU:
CMD"COPY "+ZC:
PRINT"Saving: ";ZN:
SAVE ZN:
GOTO 99
```

**these are the remaining menu choices for whatever program I am currently writing:**

```
120 IF Z="I"THEN 200
130 IF Z="L"THEN 750
140 IF Z="D"THEN 300
```

**etc......**

```
190 IF Z="V"THEN 700
195 IF Z="P"THEN 800
199 GOTO 99
```

The menu in line 99 displays the choices available for whatever program I am currently writing. For programming purposes I will always use some choices that will not be listed in the menu. Such is the case of the /BAK & SAVE option. This choice is activated by pressing **$** (dollar sign) from the menu.

To make the /BAK & SAVE routine work, you must be sure to always have **ZN="PROGNAME/BAS"** in line 0. (*substitute the actual name of your program for PROGNAME/BAS*).

After the menu is displayed, the program goes to the INKEY$ routine in line 8. Here the <CLEAR> key is checked, and if it is pressed, the program ends. If any other key is pressed we jump to the routine in line 7 where the keypress is converted to upper case, if needed.

Coming back to the main program, the keypress is interpreted and passed along to the proper routines. If the dollar sign was pressed, line 110 takes control, and we immediately activate the subroutine in line 97. There the name of the program is stripped of its extension (if any), and the extension /BAK is added to the filename. This new filename is stored in ZU. We now have the real filename in ZN and the backup filename in ZU.

Returning from the subroutine, back to line 110, we store the real filename (ZN), a space, and the backup filename (ZU) in variable ZC. Then we issue the commands CMD"COPY "+ZC and SAVE ZN. This copies the contents of PROGNAME/BAS to PROGNAME/BAK, and the file in memory is saved as PROGNAME/BAS.

Note that the routine will fail if PROGNAME/BAS does not exist, so be sure to save the file at least once before activating the /BAK routine.

Though written especially for Model III MULTIDOS, this method works even better with LDOS, DOSPLUS and NEWDOS/80, as the <ENTER> key need not be pressed, making the /BAK fully automated. With a couple of slight changes, the routines will also work with Model 4 TRSDOS/LS-DOS 6, DOSPLUS and MULTIDOS.

For Model 4, make the following changes:

Remove **CLEAR 1000** in line 0

Substitute **CMD** with **SYSTEM** in line 110 to read:

```
110 IF Z="$" THEN GOSUB 97:
ZC=ZN+" "+ZU:
SYSTEM"COPY "+ZC:
PRINT"Saving: ";ZN:
SAVE ZN:
GOTO 99
```

It seems to me that Model III method might work on the Model I DOSes that support CMD"DOSCMD". Since my Model I bit the dust some years ago I cannot verify this, but I'd certainly be interested to know.

Hope you enjoy the routines.

---

# TRSDOS 6.3.1 CUSTOM PATCHES

By Lance Wolstrup

Before we get to the custom patches, let me bring a correction to Roy Soltoff's MANDATORY patches listed in the Misosys Quarterly (Summer 1990, vol. IV.iv. page 32, column 2 at bottom). The first four patches are just fine, but the last one is incorrect. It is a series of five patch lines to SYS8/SYS, bringing LS-DOS 6.3.1 up to level 1F, that will correct the release of banks larger than 7. Unfortunately, each patch line address is listed exactly 4 bytes too high, thus resulting in the 'FIND line mismatch' error.

We would normally patch directly from the DOS command line, but line 4 & 5 are in excess of 80 characters, so rather than splitting these lines, we will write a 'DO' file to make the patches for us.

From LS-DOS Ready, type the following:

```
BUILD SPOOL1/FIX <enter>
D1F,A1=21;F1F,A1=CA <enter>
D20,1A=1F;F20,1A=07 <enter>
D20,20=CD EF 29;F20,20=32 8C 2A <enter>
D21,7E=69 26 00 11 8C 2A C5 06 02 3E 5F EF C1 C9;
F21,7E=53 70 6F 6F 6C 65 72 20 61 6C 72 65 61 64
<enter>
D22,20=78 20 66 72 65 65 64 20 20;F22,20=20 72
65 6C 65 61 73 65 64 <enter>
<control><shift><@>
```

Now that the 'DO' file is saved, type:

```
PATCH SYS8/SYS.SYSTEM6 USING SPOOL1/FIX
<enter>
```

This should do the job. Finally, bring the level up to 1F with this patch:

```
PATCH BOOT/SYS.SYSTEM6
(D02,1F=46;F02,1F=45) <enter>
```

Ok, now that we got this over with, we can concentrate on customizing LS-DOS 6.3.1. For reference, the patch to restore the change back to normal is listed directly under the initial patch.

Also, because of our 2-column format, most of the patches are listed on two lines. When patching, be sure to enter the entire patch on one line only.

**Replace the REMOVE command with KILL.**

```
PATCH SYS1/SYS.SYSTEM6
(D02,15=4B 49 4C 4C 20 20;F02,15=52 45 4D 4F 56 45)
```

**Replace KILL with REMOVE.**

```
PATCH SYS1/SYS.SYSTEM6
(D02,15=52 45 4D 4F 56 45;F02,15=4B 49 4C 4C 20 20)
```

**Replace DIR with D**

```
PATCH SYS1/SYS.SYSTEM6
(D01,AE=20 20;F01,AE=49 52)
```

**Replace D with DIR**

```
PATCH SYS1/SYS.SYSTEM6
(D01,AE=49 52;F01,AE=20 20)
```

**Replace CAT with C**

```
PATCH SYS1/SYS.SYSTEM6
(D01,76=20 20;F01,76=41 54)
```

**Replace C with CAT**

```
PATCH SYS1/SYS.SYSTEM6
D01,76=41 54;F01,76=20 20)
```

**Disable password protection.**

```
PATCH SYS2/SYS.SYSTEM6 (D02,50=18;F02,50=28)
```

**Re-enable password protection.**

```
PATCH SYS2/SYS.SYSTEM6 (D02,50=28;F02,50=18)
```

**Boot up in uppercase.**

```
PATCH SYS0/SYS.SYSTEM6 (D00,80=20;F00,80=00)
```

**Boot up in lowercase.**

```
PATCH SYS0/SYS.SYSTEM6 (D00,80=00;F00,80=20)
```

**Boot up with CLOCK on**

```
PATCH BOOT/SYS.SYSTEM6
 (D05,53=00 00;F05,53=28 04)
```

**Boot up with CLOCK off**

```
PATCH BOOT/SYS.SYSTEM6
(D05,53=28 04;F05,53=00 00)
```

**Extended error messages**

```
PATCH SYS0/SYS.SYSTEM6 (D00,88=48;F00,48=08)
```

**Normal error messages**

```
PATCH SYS0/SYS.SYSTEM6 (D00,88=08;F00,88=48)
```

# MORE TRSDOS 6.3.1 TIPS

By Art McAninch

Thank you for the information about using the period (.) with the DIR, CAT, and BACKUP commands in conjunction with the "wild card" feature. I was already aware of the capability of limiting multidrive DIR commands with the hyphen (-) character. This is particularly useful on a hard disk.

In response to your request, you may BACKUP invisible and system files using the "wild card" method in exactly ths same way you do visible files. The only added requirement is to follow your command line with an I or a S in parenthesis. For example:

```
BACKUP /CMD:0 :1 (I)
```

You may also do this:

```
BACKUP /SYS:0 :2 (S,Q)
```

and the system will Query you about each /SYS file before it copies it!

Another possibility that I use frequently is:

```
BACKUP :0 :1 (DATE="01/01/90-",I,Q)
```

in order to selectively copy all files of certain dates.

You may desire to copy newly created files to a target disk from a disk you have recently made a lot of changes on. There may be many more files on this source disk that you do not want to copy to the target. Do this:

**BACKUP :0 :1 (MOD,I,S)**

thus you will copy only the files that have been modified to the target disk. If you would like to use this source disk to copy the modified files to several target disks, then write protect the source disk first. Then as the copying occurs, you will see the message SOURCE DISK WRITE PROTECTED, UNABLE TO UPDATE. This way you will retain the + mod flag on the modified files, and you can repeat the operation with another target disk.

You may update the operating system on a target disk without changing the application programs by:

**BACKUP :0 :1 (O,I,S)**

this "O" parameter will cause the BACKUP routine to check the Target directory against the Source directory and copy ONLY the files over that already exist on the Target -- they will get the newer files of the Source.

Then, again, you may want to copy only the files from a Source disk that do not presently reside on the Target to the Target:

**BACKUP :0 :1 (N,I,S)**

and the "N" parameter will allow only the files that would be NEW to the Target to be copied.

Now, another application is to apply the same procedure to the PURGE command in order to PURGE selectively. We really do have a very flexible operating system! It is vastly superior to MS-DOS!

# MANDATORY PATCHES FOR GIF4MOD4 V.2

By J.F.R Slinkman
(author of GIF4MOD4)

Users of GIF4MOD4 Version 2 for the TRS-80 Model 4 should apply the following patch. This patch implements a change in the GIF87a spec to allow what are called "delayed clear codes" in the GIF code stream, which permit better compression ratios.

Compuserve has requested all GIF Developers to have the users of their programs upgrade to a version permitting delayed clear codes by 01-Jan-91, the date the inclusion of these codes will be authorized.

Without this patch, GIF4MOD4 will crash if a delayed clear code is encountered in a GIF file, and could concievably destroy programs or data on disk, especially if an interlaced file is being processed and/or if the +d switch is set.

This patch also changes the mailing address in the program "billboard" to reflect the fact I've moved from my cruddy little apartment to a nice, big house. I admit this is not of major concern to most users, but you never know when they may need to contact me.

```
.GIFL2W/FIX -- apply to GIF4MOD4/CMD Version 2
. permit delayed clear code
D0D,E0=C3 38 8E
F0D,E0=2A 75 36
D11,0D=2A 75 36 CB 64 C2 34 8A C3 1D 8B
F11,0D=00 00 00 00 00 00 00 00 00 00 00
. Change mailing address in billboard
D12,18=31 35 31 31 20 4F 6C 64 20 43 6F 6D 70 74 6F
6E
F12,18=34 31 30 38 2D 43 20 46 61 69 72 6C 61 6B 65 20
D12,28=20 52 6F 61 64 2C 20 52 69 63 68 6D 6F 6E 64
2C
F12,28=4C 61 6E 65 2C 20 47 6C 65 6E 20 41 6C 6C 65
6E
D12,38=20 56 61 2E 20 32 33 32 33 33
F12,38=2C 20 56 41 20 32 33 30 36 30
.eop
```

The original version of GIF4MOD4 is not so easily modified, and is no longer supported anyway. Those using the old version should exercise caution when decoding GIF files created after 31-Dec-90, specifically making sure they have backups of all programs and data on all disk drives accessable to the system before attempting to decode such files. Those using the original version are STRONGLY advised to upgrade to Version 2 before 01-Jan-91.

On August 1, Compuserve released the spec for GIF Version 89a, and authorized the release or Version 89a files starting September 1.

This new version includes several enhancements, one of which is the inclusion of Pixel Aspect Ratio information. The "standard" pixel is 1:1 (i.e. it is as wide as it is tall). TRS-80 pixels, however, are not 1:1, but are 1:2 (i.e. they are half as wide as tall).

At the present time, some GIF decoder programs will choke when seeing version 89a GIF files. Just seeing the "89a" stops some of them. For this reason, especially if you are using HR2GIF to create files intended to be transported to other hardware, you may wish to maintain two copies of HR2GIF -- one in its original form, to create Version 87a files, and one patched version to create Version 89a GIF files.

The advantage of Version 89a is that it tells the GIF89a decoder on the other computer that the height of the image must be doubled (or the width halved) to render it in its original proportions. The disadvantage, as mentioned above, is that many existing decoders on other computers will not even attempt to render the image at all.

As time goes on, and a greater proportion of decoders in the field are upgraded to handle Version 89a GIF files, this disadvantage will gradually disappear.

The patch to HR2GIF follows:

```
. HR2GIF/FIX -- apply to HR2GIF/CMD
. Change version number from 87a to 89a
D00,25=39
F00,25=37
. change Pixel Aspect Ratio from "no information to 1:2
D00,2D=11
F00,2D=00
. change mailing address in program billboard
D09,7A=31 35 31 31 20 4F 6C 64 20 43 6F 6D 70 74 6F
6E
F09,7A=34 31 30 38 2D 43 20 46 61 69 72 6C 61 6B 65 20
D09,8A=20 52 6F 61 64 2C 20 52 69 63 68 6D 6F 6E 64
2C
F09,8A=4C 61 6E 65 2C 20 47 6C 65 6E 20 41 6C 6C 65
6E
D09,9A=20 56 61 2E 20 20 32 33 32 33 33 20
F09,9A=2C 20 56 61 2E 20 20 32 33 30 36 30
.eop
```

If you make the above change to HR2GIF, you will also need to change one line in APENDGIF/BAS. Line 30 should be changed to:

```
30 CHECK$="GIF89a"+MKI$(640)+MKI$(240)+
MKI$(128)+CHR$(17)+STRING$(3,0)+
STRING$(3,255)
```

Be aware that this change to APENDGIF/BAS will render it unable to work with Version 87a GIF files; so if you plan to maintain two versions of HR2GIF/CMD, you should also maintain two versions of APENDGIF/BAS.

Also be aware that APENDGIF/BAS, whether line 30 is changed or not, will not be able to append a Version 87a file to a Version 89a file, or vice versa.

The new Version 89a spec will not have too much effect on GIF4MOD4 Version 2 users. The program will successfully handle virtually all Version 89a files. However, you will start seeing some "Function Block Type ### -- not Processed" messages as the image base slowly begins to shift to Version 89a files. You WILL get an image out of GIF4MOD4 when decoding a Version 89a file.

The only Version 89a enhancement which may cause GIF4MOD4 some trouble is the new "transparent" color. The area of the overlay image which is supposed to be transparent (i.e. retain the information already on the screen) will instead contain some color (probably, but not necessarily, black), and will therefore obliterate the portion of the image intended to be preserved.

# ADD A JOY-STICK TO YOUR MODEL III/4/4D/4P

By Roy Garcia

You'll be adding the Joy-stick in the same place your RAM size label is, on the right side of the keyboard, below the orange reset button. Model 4P will add the Joy-stick under the Radio Shack name tag on the keyboard. Do keep in mind that making this modification is at your own risk and that Radio Shack's warranty, if any, will become void.

Before beginning this project, please make sure you read (and understand) the directions completely.

- 1. Unplug A.C. powercord from wall.
- 2. Turn computer over & remove (11) screws. On Model 4P turn keyboard over and remove (8) screws.
- 3. Remove computer cover, carefully watching video cable & ground wire (4P - skip).
- 4. Note where RAM size label is and remove keyboard cover (6) screws (4P - skip).
- 5. CAUTION: cut on PC board where RAM size label would be (4P - skip).
- 6. Replace keyboad to computer & replace cover (4P - skip).
- 7. Remove RAM size (4P name tag) label & drill through both keyboard & cover.
- 8. File in for DB-9 male connector.
- 9. Drill in keyboard for two screws to hold DB-9.
- 10. Cut wire's 6" long, strip each end 1/8", and solder both ends.
- 11. Solder to DB-9 pins (backview - see below).
- 12. Solder to 20 pin (16 pin on 4P) connector on the keyboard (see below).
- 13. Replace keyboard cover & screws.
- 14. Replace computer cover & screws (4P - skip).
- 15. Plug in and test with most Commodore or Atari joy sticks.

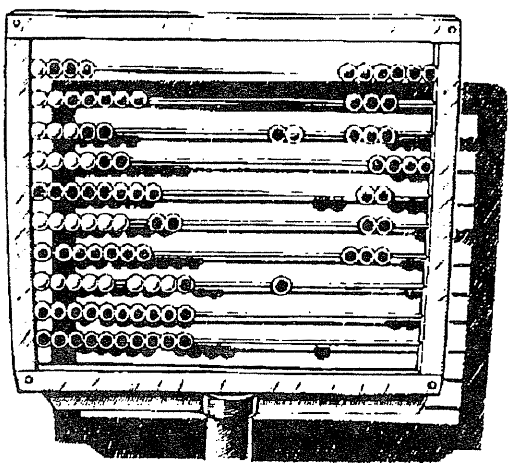
```
DB-9 male      !5  4  3  2  1 !
back view      ! 9   8  7  6  !
```

| DB-9<br>male<br>9 pin | Color<br>Option | Keyboard<br>20 pin<br>connector |
|---|---|---|
| 1 | brown | 12 |
| 2 | red | 13 |
| 3 | orange | 14 |
| 4 | yellow | 15 |
| 5 | N/C | |
| 6 | blue | 16 |
| 7 | N/C | |
| 8 | green | 7 |
| 9 | N/C | |

# REVERSE4

## Another Basic puzzle-game for Model 4

By Lance Wolstrup



As I have mentioned before, I enjoy writing games. Especially, I enjoy taking programs written for other computers and rewriting them for my TRS-80. The game of REVERSE was originally written by Peter Sessions sometime before 1978, and was published by David Ahl in his book 'BASIC COMPUTER GAMES'.

The game requires you to arrange a list of letters in alphabetical order from left to right. To move, you tell the computer how many letters (counting from the left) to reverse. For example, if the current list is:

B C D E A F G H I

and you reverse 4, the result will be:

E D C B A F G H I

Now, if you reverse 5, you win!

Peter Sessions describes the game strategy as follows:

*"There are many ways to beat the game, but approaches tend to be either algorithmic or heuristic. The game thus offers the player a chance to play with these concepts in a practical (rather than theoretical) context.*

*An algorithmic approach guarantees a solution in a predictable number of moves, given the number of items in the list. For example, one method guarantees a solution in 2N-3 moves when the list contains N numbers. The essence of an algorithmic approach is that you know in advance what your next move will be. One could easily program a computer to do this.*

*A heuristic approach take advantage of 'partial orderings' in the list at any moment. Using this type of approach, your next move is dependent on the way the list currently appears. This way of solving the problem does not guarantee a solution in a predictable number of moves, but if you are lucky and clever, you may come out ahead of the algorithmic solutions. One could not so easily program this method.*

*In practice, many players adopt a 'mixed strategy', with both algorithmic and heuristic features. Is this better than either 'pure' strategy?"*

I include the above explanation of the game strategy to illustrate just how seriously any computer program was viewed in the old days. Even a simple game, such as REVERSE, was considered to be a mathematical learning tool for the user, hence the very learned discussion.

In the years since, we have become sophisticated (jaded?), and we now consider a game for what it really is: something fun to waste our time on.

Anyway, back to our program. Though REVERSE4 plays identically to REVERSE, not one line of code remains from the original program. It has been completely rewritten, from scratch, and especially for Model 4.

In the process, I have used two routines that are not documented in the Basic manual. Though both have been discussed in previous issues of TRSTimes, let's review the tricks that can be accomplished with a few strategic PEEKs and POKEs.

Line 4 disables the <BREAK> key. The DOS manual tells us that we can do this with the Basic command: SYSTEM"SYSTEM (BREAK=N)"

Fine, it works as documented, but it has to load an overlay to execute the command and is very slow. A much better way is to set bit 4 of the value in 7CH (124 decimal). This is the memory location holding the SFLAG$ in TRSDOS/LS-DOS 6, and bit 4 controls whether or not the break key is enabled. Bit 4=0 means that break is enabled; bit 4=1 means that break is disabled. By simply ORing the current value at 7CH with 16, bit 4 is set and the break key no longer works. Clean and fast.

The code to re-enable the break key is found in line 280. There, bit 4 of 7CH is reset when the current value is ANDed with 239.

Line 5 holds the other undocumented routine, and it is somewhat more complicated.

One of the major differences between Model III and Model 4 is that VIDEO RAM on the Model I/III is memory mapped, while VIDEO RAM on the Model 4 is not. The authors of DOS chose to switch video in and out to gain additional user memory.

However, as explained in TRSTimes' Nov/Dec 1988 issue (HUNTING FOR BURIED TREASURE), the Model 4 VIDEO RAM can be forced to be memory mapped and, thus, allow faster access to the screen. If we force the memory mapped screen, it begins at 0F800H and goes on for the next 1919 memory locations (a total of 1920 locations).

Since we don't want BASIC to use these locations for its own use, we CLEAR,&HF7FF. BASIC now knows not to use any memory above 0F7FFH.

Now, in order to force the memory mapped screen we have to change the value in memory location 78H. This location is the OPREG$ in the TRSDOS/LS-DOS flag-table and is also known as the 'PORT 84H IMAGE'.

Normally, this value is 135 (87H - or 1000 0111 in binary). Notice the binary value - bit 0 is set (1). This is the key. If bit 0 is set, VIDEO RAM is transitory and is switched in and out as DOS sees fit. If, on the other hand, bit 0 is reset (0), VIDEO RAM becomes stable and memory maps itself at 0F800H. Thus, if we POKE 78H with 134 (86H - or 1000 0110 in binary), bit 0 is reset and our goal is accomplished.

Well, not quite! As mentioned above, 78H is known as the 'PORT 84H IMAGE'. This is because PORT 84H, which controls the video state, always gets its information from memory location 78H. Therefore, in order to complete our goal, we send the same value as is stored in 78H to PORT 84H with the command: OUT &H84,134 (we cheated a little here. The correct command is OUT &H84,PEEK(&H78), but it doesn't make any difference; either way the value 134 is being sent to PORT 84H).

To restore the screen to its normal switching in and out state as the game ends, we first POKE &H78 with 135. Then we send the same value to PORT 84H (OUT &H84,135), and finally, to regain the lost RAM, we reset HIGH$ to the top of memory with the command: CLEAR,&HFFFF.

My reason for including the memory mapped screen feature in REVERSE4/BAS is two-fold. First, it certainly does make the screen access faster and, secondly, it allows us to PEEK and POKE the screen just like on the Model I and III. This means that many programs taking advantage of the memory mapped VIDEO RAM of these machines can now be translated to the Model 4 with little difficulty. As long as the prospective translator remembers that the Mod I/III screen starts at 3C00H (15360 decimal) and that the Model 4 screen starts at 0F800H (63488 decimal), while also taking into consideration that the Model 4 screen is 80x24 as opposed to the Mod I/III screen size of 64x16, the going should be fairly smooth.

One program I would like to see a Model 4 translation for is the 'ATLANTEAN ODYSSEY' written by Teri Li (aka Terry Kepner of 80 Micro fame). It is an enormous adventure game with decent graphics which are POKEd directly to the Mod I/III screen. The entire program listing can be found in 'The Captain 80 Book of Basic Adventures', published in 1981 by 80-Northwest Publishing, Inc.

How about it! Anyone ready for this challenge?

# REVERSE4/BAS

```
1 'REVERSE4/BAS for Model 4
2 '<c> 1989 Lance Wolstrup
3 '

         disable break-key
4 POKE &H7C,PEEK(&H7C) OR 16

         force memory mapped screen
5 CLEAR,&HF7FF:POKE &H78,134:
OUT &H84,134:DEFINT C-Z

         store tops of boxes in B1$
6 B0$=CHR$(151)+STRING$(5,131)+CHR$(171):
FOR X=1 TO 9:B1$=B1$+B0$:NEXT

         store sides of boxes in B2$
7 B0$=CHR$(149)+STRING$(5,32)+CHR$(170):
FOR X=1 TO 9:B2$=B2$+B0$:NEXT

         store bottoms of boxes in B3$
8 B0$=CHR$(181)+STRING$(5,176)+CHR$(186):
FOR X=1 TO 9:B3$=B3$+B0$:NEXT

         set screen width and jump over subroutines
10 SW=80
11 GOTO 100

                    subroutines:

         print routines:
         gosub 20 - prints flush left
         gosub 21 - prints center
         gosub 22 - prints flush right
         gosub 23 - prints wherever you wish
20 H=0:GOTO 23
21 H=INT((SW-LEN(A$))/2):GOTO 23
22 H=SW-LEN(A$)
23 PRINT@V*SW+H,A$;:RETURN

                    beginning of program:
         turn off cursor and erase screen
         then display program name and copyright
100 PRINT CHR$(15);:CLS
110 V=0:A$="TRSTimes Presents:":GOSUB 20
120 A$="REVERSE4":GOSUB 21
130 A$="(c) Lance Wolstrup":GOSUB 22
140 V=1:A$="another puzzle game for Model 4":
GOSUB 21
```

```
150 V=2:A$=STRING$(SW,131):GOSUB 22
```

**draw boxes**

```
160 V=5:A$=B1$:GOSUB 21:
V=6:A$=B2$:GOSUB 21:
V=7:A$=B3$:GOSUB 21
```

**display game information at bottom of screen**

```
165 V=16:A$=STRING$(SW,131):GOSUB 20:
V=17:A$="N = New game":GOSUB 20:
A$="2 - 9 = reverse":GOSUB 21:
A$="Q = Quit":GOSUB 22:
V=18:A$=STRING$(SW,140):GOSUB 20
166 V=20:
A$="Arrange the letters in alphabetical order from left
to right.":GOSUB 21:
V=21:
A$="To move, tell the computer how many letters
(counting from the left) to reverse.":GOSUB 20
167 V=22:
A$="You may reverse from 2 to 9 letters.":GOSUB 21
```

**generate box numbers**
**generate box letters**
**make sure letters are not repeated**

```
170 FOR X=1 TO 9:A$(X)="":C(X)=X:NEXT
180 RANDOM:SC=0:FOR X=1 TO 9
190 C=RND(9)
200 IF C(C) THEN A$(X)=CHR$(C(C)+64):C(C)=0
ELSE 190
210 NEXT
220 FOR X=1 TO 9:C(X)=ASC(A$(X))-64:NEXT
```

**display box letters and box numbers**

```
230 H=11:FOR X=1 TO 9:
V=6:A$=A$(X):GOSUB 23:
V=4:A$=RIGHT$(STR$(X),1):GOSUB 23:
H=H+7:NEXT
```

**check if game is won**

```
240 GOTO 330
```

**display number of moves**
**and prompt for key stroke**

```
250 V=10:H=35:A$="Moves:":GOSUB 23:
H=42:A$="":GOSUB 23:PRINT USING"###";SC
260 V=12:
A$="Reverse how many: "+CHR$(14):GOSUB 21
270 I$=INKEY$:IF I$="" THEN 270 ELSE I=VAL(I$)
```

**get key stroke and evaluate it**
**N will start a new game**
**Q will disable the memory mapped**
**screen, enable the break key and & the game**

```
280 IF I$="N" OR I$="n" THEN PRINT CHR$(15);:
V=10:H=42:A$=STRING$(3,32):GOSUB 23:
GOTO 170
ELSE IF I$="Q" OR I$="q" THEN CLS:
POKE &H78,135:OUT &H84,135:CLEAR,&HFFFF:
POKE &H7C,PEEK(&H7C) AND 239:END
```

**check if number of boxes to reverse is legal**

```
290 IF I<2 OR I>9 THEN 270 ELSE PRINT CHR$(15);
```

**reverse boxes as requested**

```
300 FOR X=1 TO INT(I/2):
Z=C(X):C(X)=C(I-X+1):C(I-X+1)=Z:NEXT
310 FOR X=1 TO I:A$(X)=CHR$(C(X)+64):NEXT
```

**update score**

```
320 SC=SC+1:GOTO 230
```

**check if game is won**

```
330 FL=0:
FOR X=1 TO 9:IF C(X)<>X THEN FL=1:X=9
340 NEXT:IF FL THEN 250
```

**game is won - but check if this has**
**happened on original set-up - don't allow**

```
350 IF SC=0 THEN 170
```

**display win message and**
**ask if user wants another game**

```
360 V=10:H=0:A$=CHR$(31):GOSUB 23
370 A$="You win in"+STR$(SC)+" moves":GOSUB 21
380 V=12:A$="Would you like to try again (Y/Q) ":
GOSUB 21:PRINT CHR$(14);
```

**wait for key stroke**

```
390 I$=INKEY$:IF I$="" THEN 390
```

**Y begins another game**
**Q goes to the routine that resets**
**the liberties we have taken with DOS**
**and then ends the program**

```
400 IF I$="Y" OR I$="y" THEN V=10:H=0:
A$=CHR$(31):GOSUB 23:PRINT CHR$(15);:
GOTO 165
410 IF I$="Q" OR I$="q" THEN 280 else 390
```

# USING THE I/O BUS OF THE MODEL 4

By Heine A R de Miranda

## INTRODUCTION

Someday you may want to use the I/O bus of the Model 4 to control the temperature of your greenhouse or the trains of the model railway. With some knowledge or help on electronics this is not difficult (at least not the computer part ). The Model 4 I/O bus is easier and safer to use than the Model 1 I/O bus, because there are fewer lines, and all the lines are buffered. For those familiar with the Model 1 I/O bus, some comparison with that model is given. This small article closes with a description of a demonstration circuit and programme which you may want to try out. Since the I/O bus of the Model 3 is equal to the 4, most of the information given also holds for the Model 3.

## THE I/O BUS STRUCTURE

At first sight, the Model 4 bus seems to be larger than its Model 1 counterpart. It has a 50 pole connector instead of a 40 pole connector. However, if we look at the number of active pins, we count 24 for the Model 4 and 38 for the Model 1. All the even pins in the Model 4 are grounded.

The following table shows the differences:

| | MODEL 1 | MODEL 4 |
|---|---|---|
| Address lines | 16 | 8 |
| Data lines | 8 | 8 |
| Control lines | 12 | 8 |
| Voltage lines | 2(5V & ground) | 1(ground) |

The important differences are:

- The Model 4 has only 8 address lines, namely the lines A0 through A7
- The Model 4 has only 8 control lines

On the Model 1 we could choose between memory mapping the external device or port mapping. The Model 4 uses only ports. This reduces the number of address- and control lines to 8, which is still more than we need for most applications. We will continue to need the IN and OUT controls, signifying that there will be some action at a port. An important added feature is the line EXTIOSEL, which switches the data buffer to send or receive, and therefore acts as the key to information exchange.

The pin numbering for the I/O connector of the Model 4 is:

| | |
|---|---|
| 02-50 even | GROUND |
| 01-15 odd | DATA LINES D0-D7 |
| 17-31 odd | ADDRESS LINES A0-A7 |
| 33 | IN* |
| 35 | OUT* |
| 37 | RESET* |
| 39 | INT* |
| 41 | WAIT* |
| 43 | EXTIOSEL* |
| 45 | NC |
| 47 | M1 |
| 49 | IORQ |

For most applications we only need 3 control lines, IN*, OUT* and EXTIOSEL*. Therefore these will be discussed. The significance of the other functions can be found in Litt 2 and 3.

## ENABLE EXT I/O

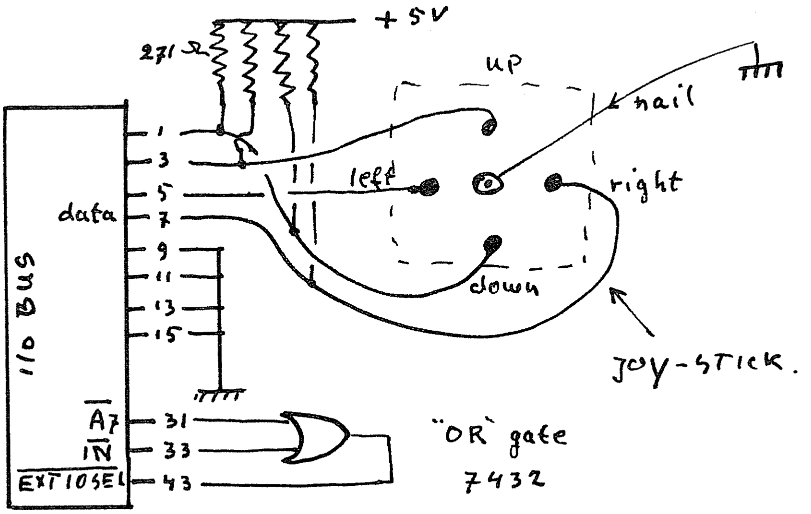
A function which we do find in the Model 4, but not in the Model 1, is the ENEXTIO function. This is an internal control line which must be activated in order to connect the address lines and data lines to the I/O bus. This gating function must be activated by writing to port 0ECH with bit 4 on in the user software. This seems to be an extra protection of the system against erroneous signals at the I/O bus. Fortunately the makers of LSDOS/LSBASIC switched this bit 4 on by default. However when you are using the Model 4 in the 3-mode and use NEWDOS you must turn this bit on yourself and write the resultant byte to memory address 4120H. Later in this article you will see how this is done in an application program.

## EXT I/O SEL

This is a handy new control function of the Model 4. It is the signal you put on pin #43 to open up the data lines for information flow. If this pin is "high"(+5V), outgoing data is selected and if the pin is "low"(0V), incoming data is selected.

By nature this pin is "high" (internal pullup resistor of 150 ohms to +5V) and therefore the computer is protected against outside signals.

However if you want a byte to go into the computer, the necessary condition for EXTIOSEL has to be "low" and your circuitry must provide this condition.



WHAT MAKES A GOOD PORT ?

It looks as if we may freely choose a number from 0 to 255 since we have 8 address lines available. However, that is not completely true. The Model 4 uses port addressing for many essential internal functions like display and keyboard. That is the bad news.

The good news is that all internal port addresses are within the range of 80H-FFH. Avoiding these addresses, we still may choose an address in the range of 0-7FH and be safe of interference. All allowable addresses therefore have bit 7 "low". Thus, if we have only one external device connected the address decoding can be very simple. You may use any port# between 0-127 and require that A7 is "low" for selection. If you use more external channels, you must include more address lines in the selection.

A JOY-STICK DEMONSTRATION CIRCUIT

The BASIC command INP(0), will activate IN* (active "low") and will make A7 "low". This combination must activate EXTIOSEL* (active "low"). An OR-gate like the 7432 will perform this function.

The joy-stick I use, is a small cardboard box with a nail pinched through the bottom and 4 wire contacts around a hole in the top. The contacts are connected to the data lines D0-D3 and pull up resistors to +5V and the nail is connected to ground. The experimental circuit (see fig) can be build on a small piece of cardboard with holes pinched for the wires and some elastic bands to keep the 50 pole flat cable in place. Looking at the rear side of your computer, line# 1 should be at your left.

THE BASIC DRIVER PROGRAMME

The programme below can be used with LS-BASIC. As mentioned before, it is not necessary to set bit 4 of port ECH, since by default this is done by LS-BASIC.

```
        10 CLS
        20 FOR Z=1 TO 10:
        A$=A$+CHR$(152): NEXT:
        A$=A$+CHR$(144)
        30 B$=CHR$(191)+"TRS-TIMES"
        +CHR$(191) +CHR$(140)
        40 FOR Z=1 TO 10:
        C$=C$+CHR$(137): NEXT:
        C$=C$+CHR$(129)
        50 PRINT CHR$(15)
        60 X=0: Y=20: GOTO 110
        70 GOSUB 140
        80 CLS
        85 PRINT P-240
        90 IF Y> =20 THEN Y=20
        ELSE IF Y< =0 THEN Y=0
        100 IF X> =67 THEN X=0
        ELSE IF X< =0 THEN X=0
        110 PRINT @(20-Y,X),A$:
        PRINT @(21-Y,X),B$:
        PRINT @(22-Y,X),C$
        120 GOTO 70
        130 END
140 P=255-INP(0)
150 IF P AND 8 THEN  X=X+1: RETURN
160 IF P AND 4 THEN  X=X-1: RETURN
170 IF P AND 2 THEN  Y=Y+1: RETURN
180 IF P AND 1 THEN  Y=Y-1: RETURN
190 GOTO 140
```

If you want to use NEWDOS-BASIC in the 3-mode, please add the following lines:

```
3 POKE &H4210, PEEK(&H4210) OR 16:
 'setting bit 4 at memory address &H4210
5 OUT &HEC,PEEK(&H4210):
 'setting bit 4 at port &HEC
```

Since the PRINT @ works different in NEWDOS-BASIC, you must change the following lines:

```
60  X=0: Y=0: GOTO 110
90  IF X-64*Y831 THEN Y=-12: X=0
100 IF X-64*Y<0 THEN Y=0:X=0
110 PRINT @(X-64*Y),A$: PRINT @(X-64*(Y-1)),B$:
PRINT @(X-64*(Y-2)),C$
```

BIBLIOGRAPHY

- TRS-80 Microcomputer Technical Reference Handbook.
  Cat. 26-2103 (Model 1)
- TRS-80 Model 4 Technical Reference Manual
  Cat. 26-2110
- Radio Shack Service Manual TRS-80 Model 4 and Model 4 Gate Array
  Cat. 26-1067/8/9
- Microprocessors and Digital Systems
  Douglas V.Hall
  McGraw-Hill Book Company. ISBN 0-07-025552-0

# TRS-80 PUBLIC DOMAIN SOFTWARE BONANZA

**We have bought collections of software from people leaving the TRS-80 world. As fast as we can, we are weeding out the good Public Domain and Shareware from the Commercial programs and the junk. So far, we have come up with 18 disks for the Model I & III, and 3 disks for the Model 4.**

## Model I & III

**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

## Model 4

**M4GOODIES#1:** day/cmd, day/txt, gomuku/cmd, llife/cmd, llife/doc, writer/cmd, writer/doc, writer/hlp, yahtzee/bas

**M4GOODIES#2**: arc4/cmd, arc4/doc, cia/bas, etimer/cmd, index/cmd, index/dat, mail/bas, mail/txt, trscat/cmd, trscat/txt, util4/cmd, xt4/cmd, xt4/dat, xt4hlp/dat

**M4GOODIES#3:** convbase/bas, dates/bas, dctdsp/cmd, dmu/cmd, dmu/doc, dskcat5/cmd, dskcat5/doc, editor/cmd, editor/doc, fedit/cmd, fkey/asm, fkey/cmd, fkey/doc, hangman/cmd, m/cmd, m/src, membrane/bas, miniop2/cmd, miniop2/src, move/cmd, move/doc, othello4/bas, scroll4/cmd, scroll4/src, setdate6/cmd, setdate6/doc, setdate6/fix, spaceadv/bas, taxman/bas, utilbill/bas, utilbill/doc

**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**

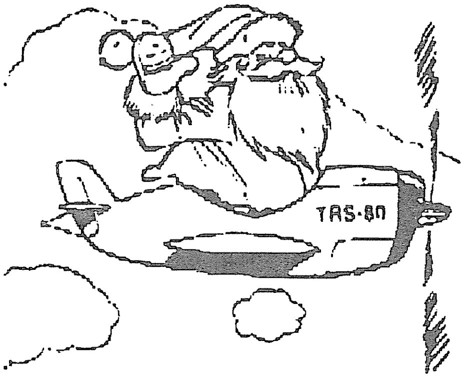
# JUST FOR THE HOLIDAYS

## Model I & III

By Sam McFarland

To put a little Christmas spirit into our TRS-80 Mod I's & III's, I have modified two programs which, I believe, were originally published in either Kilobaud or an early edition of the Microcomputer News.

Happy Holidays to all my fellow TRSTimes readers.



## TRSMAS1/BAS

```
10 DEFINT A-Z:CLS
20 FOR X=0 TO 30:A=32-X:B=32+X:Y=X+11:
SET(A,Y):SET(B,Y):NEXT:FOR X=2 TO 62:
SET(X,41):NEXT
30 FOR Y=42 TO 45:FOR X=31 TO 33:SET(X,Y):
NEXT:NEXT:FOR X=27 TO 37:SET(X,46):NEXT:
FOR X=28 TO 36:SET(X,47):NEXT
40 X=24:FOR I=20 TO 22:FOR C=1 TO 5:
SET(X+C,I):NEXT:X=X+C:NEXT:X=14:
FOR I=30 TO 36:FOR C=1 TO 5:
SET(X+C,I):NEXT:X=X+C:NEXT
50 FOR X=15854 TO 15862 STEP 2:READ F:
POKE X,F:NEXT:FOR X=15914 TO 15930 STEP 2:
READ F:POKE X,F:NEXT:FOR X=16044 TO 16056:
READ F:POKE X,F:NEXT
60 R1=PEEK(16639):R2=PEEK(16640)
70 FOR I=1 TO 35:READ F,G:SET(F,G):NEXT:
POKE 16639,R1:POKE 16640,R2
80 E=E+1:IF E=36 THEN POKE 16639,R1:
POKE 16640,R2:E=1
90 READ F,G:RESET(F,G):FOR I=1 TO 60:NEXT:
SET(F,G):FOR I=1 TO 20:NEXT
100 GOTO 80
110 DATA 77,69,82,82,89,67,72,82,73,83,84,77,65,83
120 DATA 70,82,79,77,32,84,82,83,84,73,77,69,83
130 DATA 32,13,26,26,42,31,15,37,38,37,24,22,
    38,32,13,34
140 DATA 34,40,29,15,32,29,47,30,22,35,32,10,
    29,24,39,21
150 DATA 36,25,56,38,32,36,34,16,19,27,45,35,
    8,39,27,19
160 DATA 39,28,17,32,45,39,31,18,28,30,50,33,
    19,40,36,19
170 DATA 44,26,52,36,26,38
```

## TRSMAS2/BAS

```
10 DEFINT W-Z:GOTO 80
20 Y=12:FOR X=58 TO 82 STEP 2:SET(X,Y):
Y=Y+1:NEXT:Y=32:FOR X=35 TO 72 STEP 2:
SET(X,Y):Y=Y-1:NEXT:SET(65,9):SET(63,11):
SET(57,14):SET(73,16):SET(61,16):SET(61,21):
SET(51,21):SET(40,31):SET(89,32):SET(55,17)
30 SET(61,30):SET(58,32):SET(66,25):SET(69,24):
SET(77,26):SET(75,28):SET(82,28):SET(78,30):
SET(48,30):SET(59,23):SET(66,20):SET(55,27):
SET(68,31):RETURN
40 FOR W=X TO X+Z:SET(W,Y):SET(W,Y+Z):NEXT:
FOR W=Y TO Y+Z:SET(X,W):SET(X+Z,W):NEXT:
RETURN
50 SET(80,28):RESET(80,28):SET(45,30):RESET(45,30):
SET(65,12):RESET(65,12):SET(64,7):RESET(64,7):
SET(77,24):RESET(77,24):SET(48,32):RESET(48,32):
GOTO 20
60 PRINT@110,"MERRY CHRISTMAS";:
PRINT@175,"FROM TRSTIMES";:RETURN
70 PRINT@110,"HAPPY NEW YEAR ";::RETURN
80 CLS:FOR X=0 TO 30:SET(64+X,4+X):
SET(64-X,4+X):NEXT:FOR X=34 TO 94:SET(X,34):
NEXT
90 FOR X=60 TO 68:SET(X,34):SET(X,42):NEXT:
FOR X=34 TO 42:SET(60,X):SET(68,X):NEXT
100 SET(64,2):SET(63,3):SET(64,3):SET(65,3)
110 GOSUB 20
120 X=35:Y=36:Z=8:GOSUB 40:
X=48:Y=38:Z=4:GOSUB 40:
X=78:Y=38:Z=4:GOSUB 40:
X=86:Y=36:Z=6:GOSUB 40
130 Z=1:GOSUB 60
140 FOR V=0 TO 1 STEP 0
150 X=RND(24)+52:Y=RND(16)+17:SET(X,Y):
FOR W=1 TO 100:NEXT:RESET(X,Y):GOSUB 50
160 IF Z/8=INT(Z/8) THEN GOSUB 70
ELSE IF Z/15=INT(Z/15) THEN GOSUB 60
170 Z=Z+1:IF Z=16 THEN Z=1
180 NEXT
```

# THE SWAP MEET

**WANTED:** Any literature, such as owner's or tech manuals for the Model III in Spanish. Basic manuals or anything of likely value to beginners. Will buy or pay to duplicate.
Call Harold May. (312) 325-1910 collect.

**WANTED:** Any software/hardware to allow the Model III or IV to communicate with H.P. xy recorders, analog or digital type.
John Greenland. Box 171,
Kelligrews, New Foundland. A0A 2T0 Canada

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

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

**WANTED:** Looking for Word processing program for TRS-80 Model III.
Diane A. DiGregorio, 1679 E. Williamsburg Ct.
Wheaton, IL 60187 (708) 688-8035

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

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

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

## 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. 91367**

## TRSTimes on DISK #6

**featuring the programs from the July, September and November 1990 issues.**

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

**TRSTimes on Disk #6 and the special 'Little Brother data disk': add $4.00**
**Please specify if Model 4 or TRSDOS 1.3**

**TRSTimes on DISK**
**5721 Topanga Canyon Blvd. #4**
**Woodland Hills, CA. 91367. U.S.A.**

**TRSTimes on DISK #1, 2, 3, 4 & 5 are still available at $5.00 per disk.**

**FOR SALE:** Radio Shack Model 100 Laptop Computer, 288K Ram Memory, 5 ports, Printer with Paper, Barcode Wand, Data Recorder, Disk Drive, Battery UPS System, All items rechargeable and portable, All manuals, 20 disks with software.

Computer, Radio Shack Model 4, Dual 10 megabyte (20Mbyte) Hard disk disk drive, Two 360/720K disk drives builtin, One 360K external drive, parallel and serial ports, builtin monitor, All manuals, DOS and other software.

Computer, Radio Shack Model 4P, Transportable, Two 360K/720K disk drives, parallel and serial ports, builtin monitor and modem, All manuals, DOS and other software.
Call (201) 349-6070 for Info.

---

**WANTED:** I've glitched my copy of "THE LAST ONE" for Model II. If any kind soul could supply a copy, I'd be most grateful.
Also, our group library is desperately short of Model 16 (Model II mode) software, and even more desperately short of Model 16 68000 mode software, so any offers of help in this direction would be much appreciated.
Ted Barnes
4 Depleach Road
Cheadle
Cheshire SK8 1DZ
England

---

**WANTED:** To hear from other WANG computer users that have a Wang Professional computer, as well as those having an LVP system.
R. Yves Breton. P.O. Box 95, Stn Place D'Armes
Montreal, Quebec, Canada H2Y 3E9

---

**FOR SALE:** New, unused TRS-80 Model 4D with Desk-Mate. 2DD, 64K. $225.00 + $15.00 UPS.
Also many new and used Radio Shack items. Send $1.00 for list.
Werner L. Jordan, 10 N. Morley St.
Baltimore, MD 21229

---

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

---

**WANTED:** The LDOS Quarterly Volume 1, issue # 1,2 & 3. Volume 2, issue # 1 & 4. All issues from Volume 3 up to the last issue. Also looking for The MISOSYS Quarterly Volume 1, issue # 1,2,3 & 4. Volume 2, issue # 1,2, & 3. Volume 3, issue # 1 & 2.
Lance Wolstrup, 5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

---