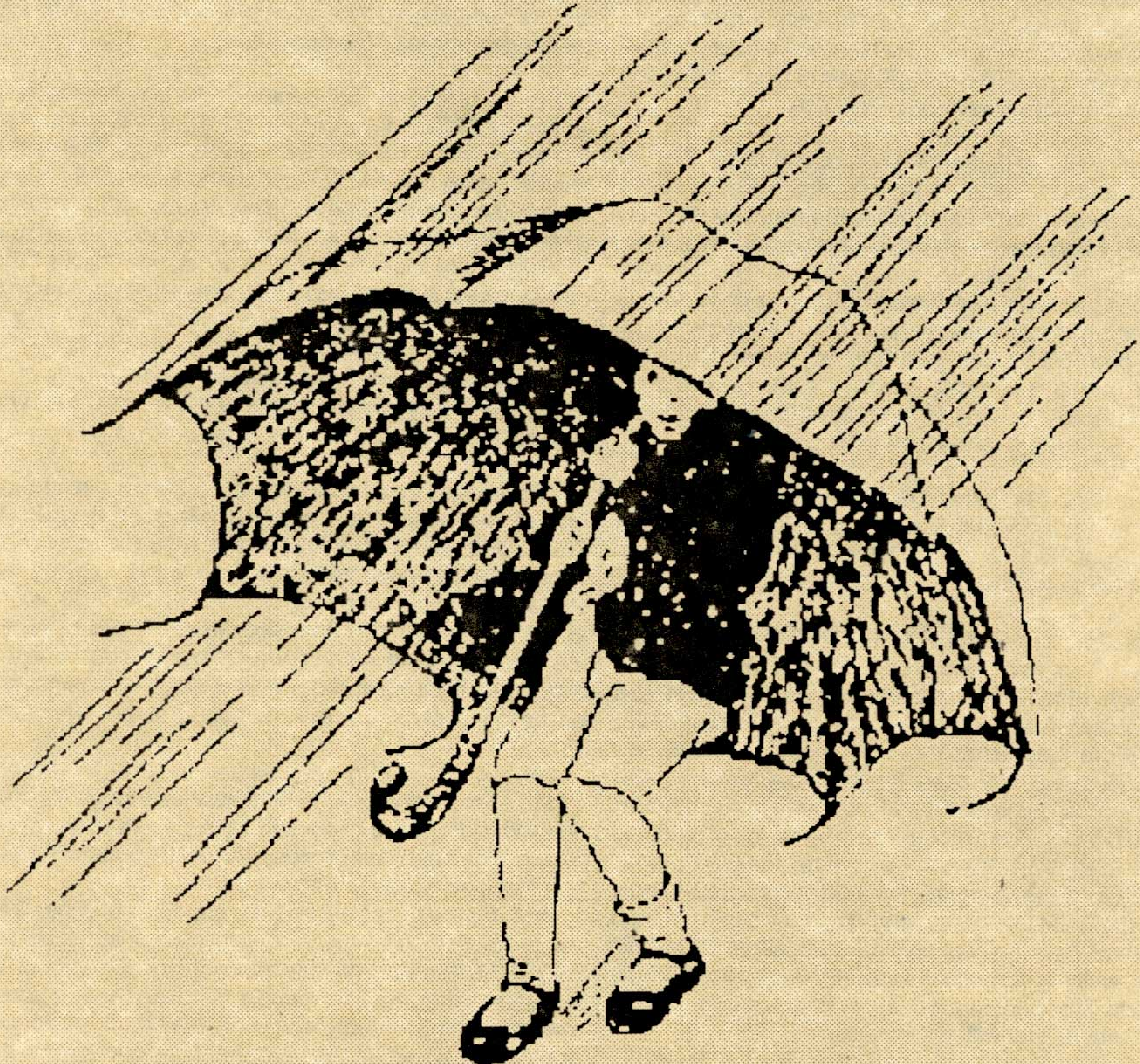
TRSTimes

Volume 5. No. 2. - Mar/Apr 1992 - $4.00



'IT NEVER RAINS IN SOUTHERN CALIFORNIA'

LITTLE ORPHAN EIGHTY

After a five year drought, we finally had rain here in Southern California. Boy, did we ever! It rained and it rained, and if you watched the news, you probably know that parts of the San Fernando Valley were flooded and much damage was done.

TRSTimes is located in the western part of the 'Valley', but we were indeed lucky; the building is on high ground, and we got through the storms with only a roof leak to report. It was very minor, one of the 'stick a bucket under and let it drip' types. Two straight days of heavy rain didn't fill the bucket, so we were very fortunate.

The rains also ended another drought (I hope). For the last few months I have had 'writer's-block'; that is, I have had absolutely no ideas for articles or programs. At times I felt as if I had been computer-lobotomized. However, being stuck in the office for two days must have been an inspiration because, while listening to the rain outside, I turned on my Model 4 and began coding a new program.

This program is now just finished; it has been debugged, and is now in the hands of some good friends from the Valley Hackers' TRS-80 Users Group for further testing - if any bugs remain, they'll find them.

The program is called DR. PATCH and I am really excited about it. In essence, it is a utility program that modifies (patches) LS-DOS 6.3.1 to allow you to customize the DOS to your liking.

I know that many patches to TRSDOS 6.2 and LS-DOS 6.3.0 have been published in 80-Micro and TRSTimes, but as Misosys rewrote a large portion of the DOS to bring it up to version 6.3.1, they no longer work. DR. PATCH allows these patches to be implemented on 6.3.1, and there are several new ones. For example, since version 6.3.0, the DOS command 'KILL' was completely removed from the LIBrary. You no longer had the option of patching in the 'K' to enable it - the command simply ceased to exist. DR. PATCH gives you back the 'KILL' command - and it retains the 'REMOVE' command. You now can have both commands. As a third option, you can rename 'REMOVE' to 'DEL', which is very useful if you also work with MS-DOS machines. There are several other options for command-name changes that will make life easier.

All password protection can be disabled - from the DOS file protection to the irritating protection schemes in FORMAT, BACKUP, and PURGE. You can also bypass the AUTO command - even if it is protected with the * parameter. There are other modifications available to FORMAT, BACKUP, and PURGE, as well as several possible changes to DIR and CAT. You can also replace the 'LS-DOS Ready' DOS prompt with your own custom prompt, the cursor character can be substituted with whatever character you might like better, the DATE and TIME prompts can be disabled, if you so prefer, and error messages can be extended to give you additional information when these nasties occur.

These are just a few of the things you can do with the program; in all, there are 64 patches - 32 modifications and 32 restorations. Yes, each patch can be effortlessly restored to normal DOS operation.

DR. PATCH does not resemble the patch programs written for TRSDOS 6.x or LS-DOS 6.3.0. It is not a shell that simply executes a /JCL file which, in turn, makes the patches using the 'PATCH' command from DOS. This is entirely too cumbersome and slow. Rather, DR. PATCH is a fully self-contained program that works directly with the DOS files - and it is very careful; it constantly checks if the correct disk is in place, and if the correct file(s) are on that particular disk - and if the correct series of code is in the correct places in the file(s).

The program is completely menu-driven, so it is easy to use. Simply type the number of the modification you wish to make, press <ENTER> and, voila, it is done.

The greatest fun I had writing the program was using a few of the tricks from the 'HUNTING FOR BURIED TREASURE' series of articles in the 1988 issues of TRSTimes. For example, while the program is running, the screen is only 22 lines long (instead of 24). DOS is not aware that the last two lines even exists. These two lines are used to display, among other things, the date and the time. Yes, that's right; the real time clock is running, not on the top screen line, but on the bottom screen line. Neat!

DR. PATCH is purposely written in BASIC - straight BASIC - not even one imbedded assembly language routine. Why? Because I was told by a couple of my good friends that it couldn't be done - and if it somehow could be - that it would be too slow.

Well, they were wrong on both counts. It is done - and the program is fast; much, much faster than using the 'PATCH' command directly from DOS, and scores of times faster than 'PATCH' using a /JCL file. I thought about rewriting it in assembly language but, as it would not significantly speed up the program, I decided against it.

Now, why am I telling you about this program? Mainly, because I am going to market it, and I am trying to convince you that it will do wonderful things for you. See the ad elsewhere in this issue for the details. All proceeds from DR. PATCH will go into the TRSTimes account to help defray the ever rising cost of publishing.

With this sales pitch done - let's get on with the immediate business at hand......... Welcome to TRSTimes 5.2

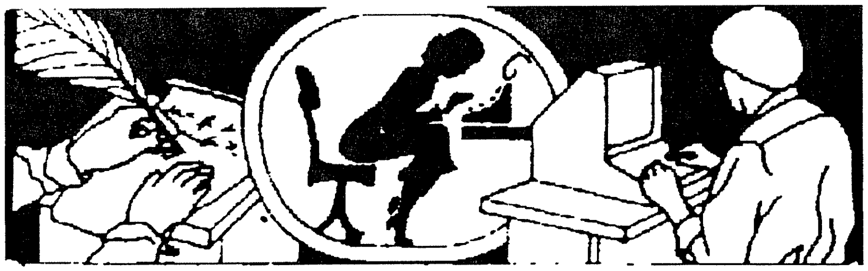
TRSTimes magazine

Volume 5. No. 2. - Mar/Apr 1992

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

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

1992 subscription rates (6 issues):
UNITED STATES & CANADA: $20.00 (U.S. currency)
EUROPE, CENTRAL & SOUTH AMERICA: $24.00 for surface mail or $31.00 for air mail.
(U.S. currency only)
ASIA, AUSTRALIA & NEW ZEALAND: $26.00 for surface maile or $34.00 for air mail.
(U.S. currency only)
Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible, a disk with the material saved in ASCII format. Any disk format is acceptable, but please note on label which format is used.



THE MAIL ROOM

GRAPHICS-90 UPDATE

Thought you would like the latest on Graphics-90. Pieter Plomp has been corresponding with me about Model I versions, and I finally got some time to convert all the files to 'universal' versions and even to add a few new ones, using Matieu Simons' patches in the latest TRS-Times as a starting point. I even did a version for Model I TRSDOS 2.3. There's a problem with that last version, though (aside from the fact that I don't have a Model I to test it on). The problem is that I could read the original Model I disk only by using the LDOS REPAIR utility on it; and if I remember right from my Model I days, once a disk is REPAIRed, it is no longer readable by TRSDOS 2.3. So, you end up with a disk of Model I TRSDOS utilities which can't be read on a Model I! I seem to remember that The Alternate Source once listed a program called UNRE-PAIR/CMD, which would return a Model I disk to its original state after it was examined under LDOS. If anyone has this utility, please send me a copy. I suppose Super Utility might be able to do it, too, or MultiDos (ie, read and transfer files to a Model I disk without first repairing it).

I've sent a disk with the new versions of the G-90 utilities to Pieter, who does have a Model I and a ton of software, for him to test them out. If they work okay, I'll incorporate them into the whole package and make the upgrade available via TRSTimes.

A minor correction to Allen Jacobs' otherwise excellent review: You don't need AllWrite to print Graphics-90 on a LaserJet or DeskJet printer. The HP fonts can be used with any word processor -- even Scripsit -- as long as you set it to print in Elite pitch. And you can print a screen without needing any word processor at all. What I did with my article was to create the title heading with the Graphics-90 editor, saved it to disk, then loaded the file into AllWrite and added the text. Many of the utilities in the package can be used in conjunction with other programs, but no other programs are required for them to be used.

Here are two thing we might watch for next: 1) Pieter would like an Epson printer driver for Graphics-90. I have no idea how to write one, but maybe one of the new users can come up with one. 2) If the new users really get into the package, we could be seeing some really creative animation programs in a few months.

Finally, Roy Beck's article on the history of the Model I was great (after all, I'm a history teacher).

Gary W. Shanafelt, Ph.D.
Department of History
McMurry University
Abilene, TX 79697

MORE ON GRAPHICS-90

Graphics-90 came yesterday, slightly bent! Mailman stuffed it in the 7" wide (normal) mail slot. At first, one disk would not read the whole directory, but after some diligent straightening, I was able to make the backups. I suppose that even a cardboard stiffener could get bent. Mail slots should be 9" wide!

At first look (no in-depth review) the program looks fine so far. The manual is very good, but still could use some refinements. For instance: Run CARTOON/BAS. Ok, how to stop? BREAK, and maybe have an open file? Not everybody knows, besides, I hate to BREAK with a cluttered screen, prefer a positive END with a CLS. An index (directory) of the different files with an explanation what they are would be nice. Also an alphabetical index.

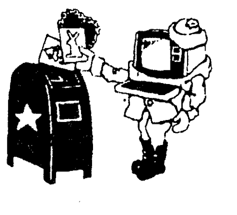
This is not to take anything away from Gary's and Larry's "non-reward" work, which is really astounding.

Henry H. Herrdegen
LaSalle, Ontario
Canada

LOVE MY TRS-80

I just read the article about booting from the hard drive, as PC's do. My two cents on the subject is: I think one of the TRS-80 world's big strengths is that the boot disk determines how things are configured. With my hard drive my wife can boot with a disk that puts two subdisks on line for her high school English schoolwork. With another disk a couple of BASIC directories are on line. With a third, Rembrandt is on line with a differently shaped cursor. Yet a fourth disk brings a couple of subdisks on line for everyday correspondence. And as long as a directory, be it on a subdisk or a floppy, is on line, any DOS in the TRS-80 world will find any file without a path instruction. I have macros that will bring any of several subdisks on line with a keystroke. There is even a menu for subdisks not used every day; the cursor is placed on the subdisk name and the user presses <enter>. I love my TRS-80!

Henry A. Blumenthal
Jacksonville, FL



THE FURTHER ADVENTURES OF MEMDISK

Model 4 with 128K - LS-DOS 6.3.1 and DISKDISK

By Lance Wolstrup

At the urging of Jim King, the MEMDISK/GRAFDISK initialization procedure was speeded up considerably. This was documented in the 'MEMDISK/GRAFDISK UPDATE' article from the last issue.

Since this month's article will deal with just MEMDISK, let me briefly recap what is going on, so far. I have a 128K desktop Model 4 with three floppy drives, along with a 15 meg hard drive and a 20 meg slave drive. After I boot the machine from the floppy in drive :0, the drive setup is as follows:

Drive :0 - floppy - 40 track, double-sided
Drive :1 - floppy - 40 track, double-sided
Drive :2 - floppy - 80 track, double-sided
Drive :3 - hard drive - half of 15 meg drive
Drive :4 - hard drive - second half of 15 meg drive
Drive :5 - hard drive - half of 20 meg slave drive
Drive :6 - hard drive - second half of 20 meg slave drive

This leaves drive :7 free for DISKDISKs. Except for rare occasions, I never use the 80-track drive, so I consider drive :2 expendable. The article from our last issue explained how to create a DISKDISK that contained the /SYS files (less SYS5/SYS, SYS9/SYS, and SYS13/SYS), along with FORMAT/CMD and BACKUP/CMD. This DISKDISK, called MEMBOOT/DSK, was placed on the drive :3 hard disk partition. Next, a /JCL file to automate the MEMDISK initialization was written. The file is called MEMBOOT/JCL, and it looks like this:

```
SYSTEM (DRIVE=2,DRIVER="MEMDISK:3")
D
D
Y
DD :7 MEMBOOT
BACKUP :7 :2
SYSTEM (SYSTEM=2)
SYSTEM (DRIVE=1,SWAP=2)
```

Whenever I want to setup the MEMDISK, I just type DO MEMBOOT. My drive configuration is now:

Drive :0 - MEMDISK
Drive :1 - floppy - 40 track-double-sided (formerly drive :0)
Drive :2 - floppy - 40 track-double-sided (formerly drive :1)
Drive :3 - hard drive - half of 15 meg drive
Drive :4 - hard drive - second half of 15 meg drive
Drive :5 - hard drive - half of 20 meg slave drive
Drive :6 - hard drive - second half of 20 meg slave drive
Drive :7 - the MEMDISK DISKDISK (this is available for other DISKDISKs)

To speed things up further, I presented four patches to MEMDISK/DCT that would eliminate the format verification. At this point, the time for setting up MEMDISK and configuring the drives was approximately 37 seconds. This is where I left off in the last issue.

Though the patches to skip the format verification work well, I was not particularly happy with them. To be honest, they were somewhat sloppy. Now, using 37 seconds to initialize MEMDISK and set up the new drive configuration is not bad, considering the standard procedure normally takes well over a minute. But, to paraphrase Jim King once again, "There's got to be a better way!"

I thought about 'better ways' for a couple of days. Finally, it came to me: I always use the exact same configuration, so why not patch MEMDISK to do exactly what I want without the prompts?"

With that in mind, I disassembled MEMDISK/DCT and, after some studying, came up with the following four patches (do note that they are for MEMDISK/DCT v6.3.1 ONLY):

to select banks 1 & 2 (option 3):
```
PATCH MEMDISK/DCT (D04,80=3E 03 18 0E:
F04,80=21 6B 37 3E)
```

to select double density format:
```
PATCH MEMDISK/DCT (D04,CC=3E 03 18 64:
F04,CC=21 50 38 3E)
```

to skip the format verification (make this patch regardless of whether or not the patches from last issue were made):
```
PATCH MEMDISK/DCT (D06,68=C9:F06,68=21)
```

to skip the format prompt:
```
PATCH MEMDISK/DCT (D06,D6=AF C9:F06,D6=21 94)
```

This certainly made the MEMDISK setup procedure a little faster, and a lot more elegant. However, using a /JCL file to automate the setup made everything still seem slow. /JCL's are by nature tortoises; I wanted a rabbit!

There was no alternative, so I wrote an assembly language program that takes the place of the /JCL procedure. I call it MEMBOOT/CMD, and the source code is listed below.

The program is simple and straight forward. It uses the CMDR supervisor call, which simply goes to DOS, issues a command, and then returns to the program. CMDR is employed to initialize the MEMDISK and open the DISK-DISK.

Since CMDR will not invoke BACKUP/CMD to copy the /SYS and other files from the DISKDISK to the MEMDISK, I had to write my own backup routine. It turned out that this is much faster than calling on BACKUP/CMD.

Also, rather than using CMDR to swap drives :0 :2, and :1 :2, I wrote my own swap routine. Again, this sped things up by a couple of seconds.

Keep in mind that this code is 'hard-wired' for my system; that is, it initializes MEMDISK as drive :2, opens DISKDISK on drive :7, and then performs a track-to-track backup from drive :7 to drive :2. There are no prompts, the program simply does its job.

If your configuration is different than mine, and I assume that it is, be SURE to modify the code to reflect your desired drive configuration. Do NOT use the program if you are not sure how to do this, as it is possible to destroy data in the process. For example, if one of your hard drives is drive :2, and you fail to alter the code, you will overwrite the first 18 sectors of the first 14 tracks on that drive - not a real fine move. I assume no responsibility for your mistakes on this - so MAKE SURE you understand how and where to make the changes.

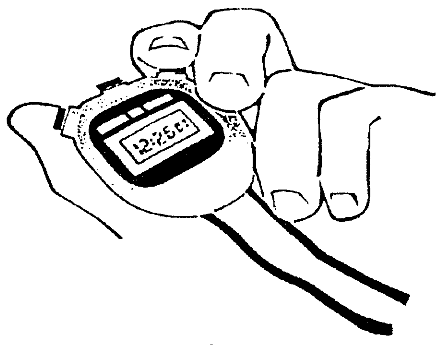
Having gotten this caveat out of the way, let me end this article by saying that MEMDISK and your drive configuration is now ready to go; we have reduced the loading and initializing time to only 15 seconds.

Finally, after careful consideration, I have come to the conclusion that the MEMBOOT/CMD file should NOT be released on TRSTimes-on-DISK. As it is quite unlikely that anyone will have the same drive configuration as my machine, I feel it will serve no purpose - and be dangerous to boot. Rather, only the source code will be on TTOD. You can then modify it to match your system.

MEMBOOT/ASM

```
;memboot/asm - assembles to memboot/cmd
;for TRS-80 Model 4 - 128K - LS-DOS 6.3.1
;must have DiskDisk
;(c) Copyright 1991 by Lance Wolstrup
;All rights reserved
;
;
        ORG     9000H
;
; initialize MEMDISK by using CMDR
;
START   LD      HL,MEMBUF  ;point to command
        LD      A,25       ;@cmdr
        RST     40
;
;open DiskDisk by using CMDR
;
        LD      HL,DDBUF   ;point to command
        LD      A,25       ;@cmdr
        RST     40
;
;display "Moving SYSTEM files to MEMDISK"
;
        LD      HL,BUMSG   ;display message
        LD      A,10       ;@dsply
        RST     40
;
        LD      D,0        ;track 0
        LD      B,14       ;loop for 14 tracks
LOOP    PUSH    BC         ;save trk counter (B)
        LD      HL,TRKBUF  ;point to buffer
;register C is set to read drive :7. This is the DiskDisk
;containing the system files, etc. in my system.
;Change register C to reflect the drive containing the
;DiskDisk with the system files in your system.
;
        LD      BC,1207H   ;18 sectors - drive :7 (C)
        LD      E,0        ;sector 0
RDTRK1  PUSH    BC         ;save sector counter
                           ;& drv #
        LD      A,D        ;trk # to A
        CP      1          ;is it DIR trk?
        JR      NZ,RDTRK2  ;no - jump
        LD      A,85       ;@rdssc
        JR      RDTRK3     ;skip @rdsec
RDTRK2  LD      A,49       ;@rdsec
RDTRK3  RST     40
        LD      BC,256     ;move buffer pointer
        ADD     HL,BC      ;to next position
        INC     E          ;next sector
        POP     BC         ;restore sector counter
                           ;& drv #
        DJNZ    RDTRK1     ;repeat for 18 sectors
;
        LD      HL,TRKBUF  ;point to buffer
;
;Register C is set to drive :2. This is the MEMDISK drive
;on my system.
;Make SURE you change register C to reflect the drive
;where MEMDISK is located in your system. Do NOT
;forget to do this, or you will write to whatever disk is
;designated as drive :2 - even your hard disk - disaster!!
;
        LD      BC,1202H   ;18 sectors - drive :2 (C)
        LD      E,0        ;sector 0
WRTRK1  PUSH    BC         ;save sector counter
                           ;& drv #
        LD      A,D        ;track # to A
        CP      1          ;is it DIR track?
        JR      NZ,WRTRK2  ;no - jump
        LD      A,54       ;@wrssc
        JR      WRTRK3     ;skip @wrsec
WRTRK2  LD      A,53       ;@wrsec
```

```
WRTRK3 RST    40
       LD     BC,256       ;move buffer pointer
       ADD    HL,BC        ;to next position
       INC    E            ;next sector
       POP    BC           ;restore sector counter
                           ;& drv #
       DJNZ   WRTRK1       ;repeat for 18 sectors
       POP    BC           ;restore track counter
       INC    D            ;next track
       DJNZ   LOOP         ;repeat for 14 tracks
;
;Now we swap drives.
;Register C holds the drive number.
;First, load drive :0 dct into the buffer
;
       LD     A,81         ;get dct
       LD     C,0          ;for drv :0
       RST    40
       PUSH   IY           ;copy drv :0 dct
       PUSH   IY           ;address to
       POP    HL           ;both hl
       POP    IX           ;and ix
       LD     DE,DCT0      ;copy dct :0
       LD     BC,8         ;to dct0 buffer
       LDIR
;
;Get drive :1 dct and save the address in IY
;
       LD     A,81         ;get dct
       LD     C,1          ;for drv :1
       RST    40
       PUSH   IY           ;save drv :1 dct address
;
;Get drive :2 dct (MEMDISK with system files) and copy
;it to the drive :0 slot - thus making MEMDISK the SYS-
TEM drive (drive :0).
;
       LD     A,81         ;get dct
       LD     C,2          ;for drv :2
       RST    40
       PUSH   IY           ;copy drv :2 dct
       POP    HL           ;address to hl
       PUSH   IX           ;copy drv :0 dct
       POP    DE           ;address to de
       LD     BC,8         ;copy memdisk dct
       LDIR                ;to slot 0
;
;Copy drive :2 dct address to DE.
;Get drive :1 dct address into HL - save it again for later
;use - and then copy drive :1 dct address to the drive :2
;dct slot, thus changing drive :1 to drive :2.
;
       PUSH   IY           ;copy drv :2 dct
       POP    DE           ;address to de
       POP    HL           ;restore drv :1 dct
                           ;address
       PUSH   HL           ;save drv :1 dct address
       LD     BC,8         ;copy drv :1 dct
       LDIR                ;to slot 2
;
;Put old drive :1 dct address into DE.
;Point HL to the buffer where we stored the old drive :0
;dct, and copy it to the drive :1 dct slot, thus changing
;old drive :0 to drive :1
;
       POP    DE           ;restore drv :1 dct
                           ;address
       LD     HL,DCT0      ;point to dct0 buffer
       LD     BC,8         ;copy drv :0 dct
       LDIR                ;to slot 1
;
;Since we no longer need the DiskDisk in drive :7, we
;disable the drive by writing 0C9H to drive :7 dct byte 0.
;
       LD     A,81         ;get dct for
       LD     C,7          ;drive :7
       RST    40
       LD     A,201        ;disable
       LD     (IY),A       ;dct :7
       RET                 ;return to dos
;
;Change the '2' in 'drive=2' to reflect the drive you wish
;to use as the MEMDISK.
;mdisk is actually mdisk/dct. It is the patched version of
;MEMDISK/DCT. In my case, this file resides on drive :3.
;Change the number to reflect the drive on which it
;resides on your system.
;
MEMBUF DB     'system (drive=2,driver="mdisk:3")'
       DB     13
;
;Change 'mboot :7' to reflect the name of the DiskDisk
;you have set up with system files, and the number of
;the drive where that DiskDisk is located on your system.
;
DDBUF  DB     'dd :7 memboot'
       DB     13
BUMSG  DB     0AH,
       DB     'Moving SYSTEM files to MEMDISK'
       DB     13
DCT0   DS     8
TRKBUF DS     4608
;
;
       END    START
```



BASIC RESCUE

Model 4

Basic 01.00.00 & 01.01.02

By M.C. Matthews

If you accidentally hit the <RESET> button, or your BASIC program somehow crashes and dumps you back into DOS, WARM00/CMD will warm-start BASIC v.01.00.00.

WARM02/CMD does the same for Basic 01.01.02. For this version of BASIC, in addition to WARM02/CMD, you also need an overlay, WARM/OVL. The overlay is created directly from DOS by typing the following commands:

LOAD BASIC/CMD <ENTER>

When it is loaded, type:

DUMP WARM/OVL (START=X'4000',END=X'43FF') <ENTER>

To create WARM/CMD for your version of BASIC (either 01.00.00 or 01.01.02), type in the appropriate BASIC program from the listings below. RUNning the program will write the data from lines 10 through 30 (and 40 in WARM02/BAS) as WARM/CMD to your disk in drive :0.

The data, of course, contain the machine language instructions that make the program work. To minimize the chance of 'data-typos', line 50 figures the sum of the data values, and then compares it to the correct value (WARM00/BAS should be 5985 - WARM02/BAS should be 5196.). If the data values do not add up to the predetermined checksum, the program will stop. Should this happen, check each piece of data and correct the errors.

Warning: It is possible to type in errant data and still match the correct checksum value, so be very careful. Remember, these are machine language instructions and, if the data values are incorrect - but still match the checksum, a faulty WARM/CMD will be written to disk. Using this faulty program will most likely just hang up the computer. However, looking at the worst-case scenario, you could have inadvertently issued instructions to write over the directory track of your main hard disk. You don't want to do that, so I repeat, be careful - type the correct data values.

Assuming that you heeded the warning, and the program is typed in correctly, WARM/CMD will act as your guardian angel. Now, whenever you find yourself staring at the DOS prompt after an unwilling exit from BASIC, simply type WARM <ENTER> and you will be returned to BASIC with the program and variables intact.

At this point it is just as well to save the program, do a proper reset and re-enter BASIC, although I have continued to run programs quite satisfactorily.

When you first re-enter BASIC, you may get an OUT OF MEMORY message on your first command. Should this happen, simply re-enter it, and everything should then function normally.

In case you wish to create WARM/CMD with your editor/assembler, the listings for WARM00/ASM and WARM02/ASM are also presented below.

WARM00/BAS for BASIC v.01.00.00

```
5 REM Basic warm start for Basic 01.00.00. Written by
M.C.Matthews
10 DATA 1,55,197,95,33,238,95,17,181,95,213,126,18,
        35,19,254,13,32,248
20 DATA 209,62,76,239,40,29,33,227,95,62,10,239,62,
        21,239,67
30 DATA 97,110,39,116,32,108,111,97,100,32,87,65,82,
        77,47,79
40 DATA 86,76,13,195,239,126,2,2,197,95
50 FOR I=1 TO 61:
READ A:
TT=TT+A:
NEXT:
IF TT< >5985 THEN PRINT:
PRINT"Checksum error: "TT:
STOP
60 RESTORE:
OPEN"O",1,"WARM/CMD:0"
70 FOR I=1 TO 61:
READ A:
PRINT #1,CHR$(A);:
NEXT:
CLOSE
80 PRINT:
PRINT"WARM/CMD for Basic 01.00.00 now ready"
```

WARM00/ASM for BASIC v.01.00.00

```
;WARM/CMD FOR RESET. 08.09.90 BASIC 01.00.00
;M.C.Matthews
        ORG     5FC5H
START   LD      HL,MESG1    ;"WARM/OVL"
        LD      DE,5FB5H    ;Buffer for filename
        PUSH DE
MORE    LD      A,(HL)      ;Copy file name to buffer
        LD      (DE),A
        INC     HL
        INC     DE
        CP      0DH          ;All done?
        JR      NZ,MORE
        POP     DE          ;Recover buffer address
        LD      A,4CH       ;Load file
        RST     28H
        JR      Z,DONE      ;Z= OK so go on
        LD      HL,MESG2  ;else load
                          ;"Can't load WARM/OVL"
        LD      A,0AH     ;and display it.
        RST     28H
        LD      A,15H     ;then Abort.
        RST     28H
MESG2   DEFM    'Can'
        DEFB    27H
        DEFB    74H
        DEFM    ' load '
MESG1   DEFM    'WARM/OVL'
        DEFB    0DH
DONE    JP      7EEFH     ;go restart Basic
        END     START
```

WARM02/BAS for BASIC v.01.01.02

```
10 DATA 1,55,57,113,33,98,113,17,57,112,213,126,18,
        35,19,254,13,32,248
20 DATA 209,62,76,239,40,29,33,87,113,62,10,239,62,
        21,239,67
30 DATA 97,110,39,116,32,108,111,97,100,32,87,65,82,
        77,47,79
40 DATA 86,76,13,195,43,128,2,2,57,113
50 FOR I=1 TO 61:
READ A:
TT=TT+A:
NEXT:IF TT<>5196 THEN PRINT:
PRINT"Checksum error: "TT:
STOP
60 RESTORE:
OPEN"O",1,"WARM/CMD:0"
70 FOR I=1 TO 61:
READ A:
PRINT #1,CHR$(A);:
NEXT:
CLOSE
80 PRINT:
PRINT"WARM/CMD for Basic 01.01.02 now ready"
```

WARM02/ASM for BASIC v.01.01.02

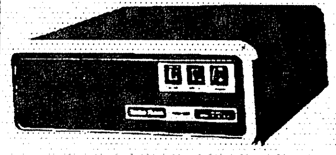
```
;WARM/CMD FOR RESET. 08.09.90. WARM02/ASM
;M.C.Matthews
        ORG     7139H
START   LD      HL,MESG1   ;"WARM/OVL"
        LD      DE,7039H   ;Buffer for filename
        PUSH    DE
MORE    LD      A,(HL)     ;Copy filename to buffer
        LD      (DE),A
        INC     HL
        INC     DE
        CP      0DH        ;All done?
        JR      NZ,MORE
        POP     DE         ;Recover buffer address
        LD      A,4CH      ;Load file
        RST     28H
        JR      Z,DONE     ;Z= OK so go on
        LD      HL,MESG2   ;else load "Can't load WARM/OV
        LD      A,0AH      ;and display it
        RST     28H
        LD      A,15H      ;then Abort.
        RST     28H
MESG2   DEFM    'Can'
        DEFB    27H
        DEFB    74H
        DEFM    ' load '
MESG1   DEFM    'WARM/OVL'
        DEFB    0DH
DONE    JP      802BH      ;Now go and restart Basic.
        END     START
```

HARD DRIVES FOR SALE

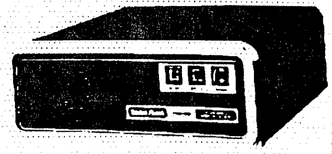
Genuine Radio Shack Drive Drive Boxes with Controller, Power Supply, and Cables.
Formatted for TRS 6.3, installation JCL included.
Hardware write protect operational.
Documentation and new copy of MISOSYS RSHARD5/6 included.
90 day warranty.

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

Shipping cost add to all prices

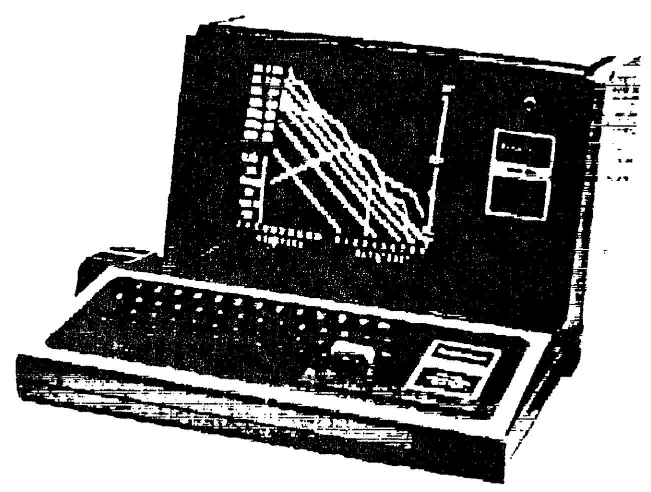


Roy T. Beck
2153 Cedarhurst Dr.
Los Angeles, CA 90027
(213) 664-5059



My Recollections of the Model I part 2

By Roy T. Beck



Did you know the RESET button on the Model I (and a heck of a lot of other machines, including CTL-ALT-DEL on the IBM) is not a RESET at all? On the Model I, the RESET button was actually wired to the NONMASKABLE INTERRUPT of the Z-80. Now you know why some lock-ups could not be overcome by hitting the RESET button; it never was a RESET! What was supposed to happen was the Z-80 would jump to a location (66D, I believe), and execute the recovery code chain which began there. But most of the code chain was in RAM, loaded there at BOOT time, and if the software bomb which locked up the machine altered any of this code, then you could not recover control with the RESET button. When such an unrecoverable lockup occurs, the only solution if you don't have a real RESET button is to turn the power off, wait awhile, and turn it on again, which of course loses everything you had in memory. To overcome this little trick, I wired an extra push button to the actual RESET line of my Z-80, and as long as I did not hold it down more than momentarily, I got an honest RESET which would overcome ANY software problem. Some knowledgeable IBM types do this today to their machines, for the same reasons. The RESET terminals exist on the IBM motherboard, they just don't connect to anything.

The Model I had an optional RS-232-C board which mounted in the top structure of the Expansion Interface. It suffered greatly from erratic contact between pressure contacts on the board and in the well of the EI which housed it. Due to poor design, the contacts often failed to make good contact, and the RS-232 would become erratic. Several tricks were developed by users to solve this problem, including use of the pink pearl as a spacer cum spring, mentioned earlier. None of the tricks were really neat, in my opinion. The best was to eliminate the spring contact system and solder a piece of cable in place of them, which yielded a solid, reliable system.

By this time, Radio Shack realized they had a good thing going in the Model I, but it had glaring deficiencies. The most stringent was its inability to meet a newly imposed set of FCC radio frequency interference (RFI) limitations. In fact, the Model I is notorious for the RFI crap it puts out. The FCC rules going into effect were so tough that the existing Model I could not meet them without major repackaging, and all of the foregoing narrative shows its other weaknesses. The solution was to redesign and repackage the Model I into the Model III, keeping the same logical layout for upward compatibility of software, fix all the worst features, including lack of lower case, lack of double density, lack of a decent RS-232, excess RFI, add built-in power supplies, more reliable and higher speed cassette operation, make provision for a hard drive, but still no double sided operation. The Model III is quite a good machine. And it was a one piece machine with almost no external cables for people to complain about.

Hard drive operation on a Model I was provided for, but only as an afterthought. Initially, the hard drive package was only made available for the Model III. Later, an adapter kit (No 26-1132) was offered which would permit operation of the 5 Meg hard drive package on the Model I. The package included LDOS 5.1.3 from MISOSYS, the adapter, and the TRSHD driver software.

One of the Model I's quirks was its behavior when a print operation was called for with no printer connected, or if the printer was connected and was non-functional. If you hit LPRINT or LLIST without a working printer, the Model I would wait forever before doing anything else. You had to reboot it to recover, which usually meant losing something you were working with. I once built a little connector which went on the printer edge card connector instead of the printer. This connector had a couple of pieces of wire on it so it acted as if a printer in the ready mode was always connected. I simply grounded the Paper out and BUSY signal, as I remember. This made the non-existent printer seem to be always READY, and never out of paper. The computer would then send all its data to the port, and quickly come back to Ready. I thought this was a nice little device.

As you know, the original Model I video display was a factory modified RCA television receiver, and it came with the short persistence, blue-white TV tube in it. Tough on the eyes. A company named Langly-St. Clair offered amber and green replacement tubes which you installed yourself. The amber tube had longer persistence than the the original TV tube. The green tube was even longer. In fact, the green was so slow that some people complained about the effect it had on the animation of some of the graphics games!

The Model I had a built-in relay on the keyboard to turn the tape cassette motor on and off. Unfortunately, the relay contacts tended to weld together, probably because the motor current was higher than the relay could safely handle. The result was the motor would continue running even when the software told it to stop. The temporary cure? Pick up the right end of the keyboard and bang it down onto the desk top, HARD. This would usually jar the relay loose so it would work again, for awhile. A fellow in the Los Angeles area came up with the permanent fix. He cast a germanium transistor and a couple of resistors into a small potted shape (I think he used his refrigerator's ice cube tray for a mold) with two cables hanging out. One cord plugged into the cassette player, the other mated with the cable from the computer. This allowed the relay in the keyboard to switch only the base current of the transistor, and the transistor handled the motor current. This completely solved the sticking relay problem, and was cheaper than replacing the relay, to boot. I never did replace my sticking relay, it never stuck again!

Another curious accessory for the Model I (offered by Radio Shack) was the Screen Printer. This was a device which would give you a Polaroid-like snapshot sized copy of the screen image on a piece of aluminum foil coated paper, accompanied by a burning smell. The printer used an electric arc to selectively burn away the foil, leaving the black paper showing through. It did this remarkably rapidly, the whole print cycle requiring only about 3 seconds, if I remember correctly. The concept of the printer was remarkable, as it actually took control of your Model I away from the Z-80 and did a DMA scan of the screen memory of the Model I, then transferred control back to the Z-80. Because of the DMA scan, the "Screen Printer" port on the Model I is actually just the bus of the machine. Probably RS would not have made the bus accessible to us users if it had not been for the Screen Printer! However, the Screen Printer wasn't compatible with other devices, gave only a small piece of paper, and soon fell out of favor.

Another curiousity was the Exatron Stringy Floppy. This was a cassette tape drive using tiny cassettes which were endless loops. This scheme included a 1K ROM which was addressed into the Z-80 memory map where RS had left a blank area. If memory serves, the blank area in memory was 3000H to 37FFH, except for a few locations used by the printer and the floppy disk controller. To get started in the Exatron ROM, you did a CALL in BASIC to a location in the ROM. Because 3000H is 12288D, they put their entry point at 3045H, which was 12345D. You did a CALL 12345 in BASIC, and you were into the String Floppy system, Clever mnemonic!

Their Tape Operating System (TOS) linked with the Model I's ROM. Because their storage media (the tape) was endless, you had to go all the way around to load a program which you had just saved. No big deal, as the tape was short. The Exatron system died because the original tape cassette fabricator was a sole source, and when that company quit, no one else could produce usable cassettes, although Exatron made a valiant effort to find a second source.

I remember another cute piece of hardware by another vendor which occupied the same part of the unused memory map where EXATRON located their EPROM. This other piece of equipment was basically just a power supply, a 1K memory chip, and an address decoder. When activated, it provided a small monitor which could take over the machine and allow you to do research in and about the machine under control of the monitor instead of the RS ROMS. Since it was addressed to the same memory space as Exatron's EPROM, you could not have both in the machine simultaneously.

Another branch of the development of the Model I was by Holmes Engineering in Utah which built a sort of mother board/expansion interface, into which you plugged whichever accessory module you were interested in. I believe they offered memory, a clock, 8" drives, and I don't remember what all else. I think it was good equipment, but the market wasn't strong enough to support them. The name "Chipmunk" comes to mind as one of their product names, but I'm not sure of anything.

One of the complaints about the Model I, especially by owners of Apples, etc, was the multiplicity of wires and cables draped about a fully loaded Model I. It sure did take a lot of cable to interconnect everything! However, when I look at the back of a loaded IBM or clone these days, I feel right at home again. Wires all over the place! Oh, well.........

Some weird and wonderful aftermarket devices were created for the Model I. One of the less well-known arrangements was a board by a company named HUH Electronics which contained about a half dozen S-100 sockets. The intent was to allow use of S-100 boards on the Model I. The idea was not bad, as the HUH board plugged into the 40 pin Screen Printer socket, and tried to convert each bus over to the other, as required. The failing was that the clock line and some other important S-100 signals were not available on the 40 line TRS bus, and so any S-100 card requiring them had a problem.

A major name then, and still around, I believe, is Alpha Micro Technology. This outfit offers a motherboard and a line of accessory cards for it, including analog and digital I/O cards. Their relay cards can operate larger loads by means of the relay contacts. They had an interface card which would plug into their bus (named the A-Bus) and into the Model I, allowing the Model I to control whatever was plugged into the A-Bus. I am sure they have interface cards for other machines, including Apple, IBM, etc.

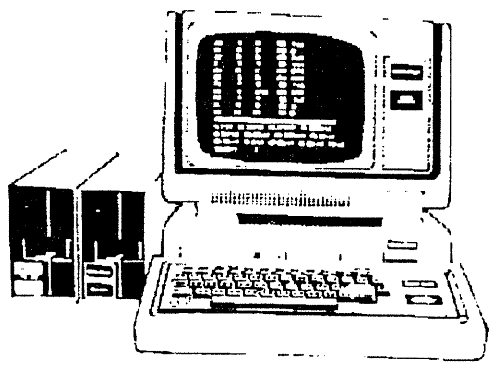
Alpha also offered a speech synthesizer. I remember a comment by a club member once that some one's speech synthesizer spoke with a Scandinavian accent, although I don't remember if it was Alpha Technology or Radio Shack's own Speech Synthesizer board! Yes, Radio Shack offered one, also.

Alpha also offered extension cords and cables of various sorts, such as floppy drive cables, hard disk cables, wye power cables to run two half height floppies on one power supply etc. They also offered the NEWCLOCK for the Models I, III and 4. This contained a clock chip and a small battery to keep it ticking when the computer was shut down. This avoided the need to answer the DATE and TIME question every time you booted up, and also put the correct time and date on your files.

Ever here of the Electric Crayon? It was an auxiliary board for the Model I which allowed you to create color images on a color monitor. I saw one once, but I don't know how it functioned. Must have had a block of memory plus a scanning circuit in it.

Another weird and wonderful device was a Rube Goldberg contraption which squatted on top of an electric typewriter, (most any brand), and by means of solenoids, struck the typewriter keys below it. This allowed any computer to interface with any electric typewriter, in theory. I once saw one driven by a Model I, and it ACTUALLY WORKED! Marvelous ingenuity to get it to function at all, but I hae me doots about its ultimate reliability.

Another problem was operation of CP/M, which was then big, on the Model I. At least two, maybe more companies created adapters for this purpose. One device was known as the Shuffleboard, which logically "shuffled" the TRS memory around in 16K blocks so CP/M could have RAM starting from 0000H, as required by CP/M. Another company, OMIKRON, offered a similar device which I bought, and which lived up to most of its promises. The most severe limitation was that only 48K of RAM was available, which did pinch a little as far as CP/M was concerned. But the device worked, and I enjoyed using it. OMIKRON also had a second adapter to allow the use of



8" floppies on the Model I, but I never tried it. Another source of CP/M for the Model I was Lifeboat Associates, who offered a modified form of CP/M which began loading at 4000H instead of 0000H, but this was incompatible with normal CP/M, and was very limiting to the user.

I also discovered another use for the OMIKRON mapper never intended by its manufacturer. The mapper had a 2K EPROM on board to give it enough intelligence to take control of the Model I at boot time, and also to contain the BIOS of the CP/M. By changing a jumper, I could prevent the EPROM from taking control, but could still engage it under control of RSM, a widely used monitor program of that time. Initially this just allowed me to disassemble the onboard EPROM to see how it was doing its tricks. The next step was to replace the OMIKRON EPROM with any 2K EPROM I wanted to read. Finally I figured out how to read a 4K EPROM in two halves by jumpering an address line. At about this time I bought an EPROM burner card for my IBM clone, and so have stopped playing with the OMIKRON mapper. But it was a useful tool in my learning process.

One of my friends, to this day, is making effective commercial use of two Model I's. He uses them to produce the paper tapes which control numerically controlled machine tools for his company. He keeps one Model I at work and has one at home, so if he has to take a problem home, all he carries is one floppy disk.

One of these machines is identified as a Model 1P (for portable); it is assembled with side brackets, a hinge, and a suitcase carrying handle. A couple of floppy drives are similarly packaged, so the whole machine is portable as two pieces of luggage, with only 3 cables to connect when he sets it up. Very effective and neatly done. We still see this machine at every monthly meeting of the San Gabriel Valley TRS User's Group, of which I was president in 1991.

The Model I can play music via a small addon board named Orchestra-80. This was a mono version, and later a stereo version identified as Orchestra-85 was offered. Still later there was an Orchestra-90 for the Models III and 4 and an Orchestra-90 CC for the Color machine. The

Orchestra-90 produces fairly decent sounding music with up to 5 "voices" through an audio amplifier.

The radio amateurs developed an interface for the Model I which allowed it to transmit and receive frequency shift keyed (FSK) Morse code, presenting both the incoming and outgoing messages on the screen and saving them to disk files, if desired. This hardware worked well and was fairly popular. Considering the variability of incoming code (duration of dots, dashes and spaces) the software for this setup was really excellent.

Probably more different DOSes have been written for the Model I than any other machine that ever existed. I will list the ones I know of, although there are probably others.

TRSDOS came with the disk versions of the machine. Version numbers include 1.0, 2.0, 2.1, 2.2, 2.3, 2.3a, 2.7DD and 2.8DD. Version 1.0 was atrocious, unusable and short-lived. 2.0 was buggy and barely usable. 2.1 was better, but still buggy. 2.2 was a general cleanup of troubles, but included a new, fatal error of its own. It was so bad that 2.3 came out within a few days. I don't know 2.3a, but I have been told it is incompatible with 2.3. Art McAninch has discovered a V2.7S, single density that was released with a software program. This too seems incompatible with almost all else! 2.7DD and 2.8DD were double density DOSes, incompatible with all the foregoing.

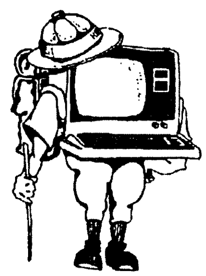
A genius named Randy Cook was the author of the early versions, up through V2.3 if my rumors are correct. Randy gave TRSDOS its architecture and flavor, and it is really a great credit to Randy that our DOSes have so many nice features. Randy had a falling-out with Radio Shack, and went his separate way.

The Model II/12/16 family also had a DOS with version numbers 2.0 and 2.0a and also a TRSDOS 4.x series; these had nothing to do with the Model I, but the similarity of names sometimes led to confusion as to which was for what.

NEWDOS 2.1 was a cleaned-up version of TRSDOS V2.1 done by APPARAT, Inc, of Denver Colorado. Primarily it fixed up the errors of TRSDOS V2.1. NEWDOS + was NEWDOS 2.1 with added utilities, including the famous Superzap. NEWDOS80 V1.0 was a completely rewritten DOS for the Model I. NEWDOS80 V2.0 was an improvement over NEWDOS80 V1.0, and is highly regarded and still used by many persons today, as a version of it was also written for the Model III.

There also were patches for NEWDOS 2.1 which made it into NEWDOS 77 for use with Percom's 77 track drives. My guess is the patches were by Percom.

There is also a NEWDOS80 V2.5 for both the Models I and III. These are the hard drive versions of NEWDOS80 V2.0. They consist of some additional files and an additional appendix and some ZAPS to the V2.0 manuals.



There are also versions of NEWDOS80 identified as NEWDOS86 and NEWDOS90 BY Warwick Sands in Australia. These exist in both Model I and III versions, and the latest version is being offered at the present time. Newdos80 and its successors was probably the most flexible DOS ever written (except perhaps MULTIDOS) when it came to configuring it for different floppy (and hard) disk formats. The most difficult thing about it was configuring it to read and write those selfsame formats. Many is the hour I have spent struggling with the rather terse manual, my machine and a strange disk, trying to make them all compatible. The formatting structure was controlled by the infamous PDRIVE command.

In fact, the PDRIVE was so difficult, especially if someone gave you a disk and didn't tell you what the PDRIVE was, that there was a real need for a routine to read the disk and determine what PDRIVE had been used to write the disk. I saw such a utility (called PDIR, if my memory is good) some years ago; Lance Wolstrup recently wrote another such device as a "finger exercises"! He writes good code just as easily as I drive a car. Phenomenal!

Art also has discovered a TRSDOS V3.0 for the Model I, apparently never officially released. It is single density, and when it boots up it, a title screen similar to V1.3 for the Model III appears on the Model I. Its commands and utilities are similar to V1.3 for the Model III, but the file formats are incompatible.

And now reenter Randy Cook! He brought out a DOS of his own and named it VTOS. In fact there was a VTOS 3.0 and a VTOS 4.0. Why 3 and 4? They were successors to TRSDOS V2, and so he simply went to 3 and 4 as the logical successors to TRSDOS V2.

Still another DOS was TRSDOS for the Model III. This DOS is not very compatible with anything, and has limitations such as single side only operation and no provision for hard drives. This went through V1.0, V1.1, 1.2, and 1.3. Still later, an aftermarket person developed it through V1.4 and 1.5. The V1.5 is now sold as SYSTEM 1.5 through TRS-Times, and now has the ability to operate double sided disks. It still has no hard drive capability, but otherwise is an excellent DOS. It also includes an Owners Manual, a rare feature these days! You have to print the manual from a file on the disk, but you do get a manual.

A DOS named DOUBLEDOS (DBLDOS) was written by Percom and was issued with or as a supplement to their doubler. Art has several versions of this DOS and also another Percom product named MICRODOS which is totally menu controlled from BASIC, and which he says gives you the feeling of operating an APPLE II. WHEE! Percom also offered an OS-80 at one time.

A company named LOBO International (I think) also developed an improved version of the Model I hardware named the MAX 80, and needed a DOS to go with it. Somehow this effort and a company named Galactic Software came together and the result was LDOS V5.x.

Later, Galactic Software became Logical Systems Inc., with Bill Schroeder and Roy Soltoff as two of the principals. There were other people, also, but I don't remember their names at this late date. Notice the Version number of LDOS, 5.x. It was the successor to Randy Cook's VTOSes, so up went the count. The Model III also received a version of LDOS, again with the V5 designation. LDOS went through version numbers 5.0.0, 5.0.1, 5.0.2, and 5.0.3 before it went to 5.1.0. From there it went through 5.1.1, 5.1.2, 5.1.3, and 5.1.4. The Model III version then went to V5.3.0 and lately 5.3.1. The Model I went from 5.1.4 directly to 5.3.1 to regain version number consistency with the Model III and 4 versions.

Still later, Radio Shack developed the Model 4, and needed a DOS for it. LSI produced LS-DOS (also known as TRSDOS) V6.x. This DOS for the Model 4 went through a number of upgrades, fixes and revisions, including V6.0, V6.1, V6.2, and V6.3, with 6.3.1 being the latest. All of these versions underwent minor revisions indicated by a third numeral. V6.2.1 was especially long-lived. LSI and MISOSYS, between them, put out versions of LS-DOS 6.2, 6.3 and 6.3.1 for the Model 2/12/16 family.

Another widely used DOS is DOSPLUS, done by Micro-Systems, Inc. in Florida, and it went through versions (that I know of) of V3.3, 3.4, 3.5 and 4.0 for both the Model I and Model III. They also had a version IV for the Model 4. Good DOSes, widely used and respected. The company was originally located in Hollywood FL and later moved to Boca Raton, where it may still be in business.

Another DOS still under sporadic development even today is MULTIDOS by Vern Hester. This DOS is known for its ability to automatically recognize and accommodate disk formats of almost every other Radio Shack DOS for the Models I, III and 4.

For further information on MULTIDOS and Vern Hester see Art McAninch's article "DOUBLE YOUR DENSITY? The Model I in Double Density, Part 2" in Computer News 80, Volume 4, Number 4, pages 4-6. In that article, Art relates the history of MULTIDOS, as told to him by David Welsh, the author of LazyWriter and a close personal friend of Vern Hester. MULTIDOS was a follow-on to ULTRADOS. ZDOS, MDOS, and LAZYDOS were all "kernels" of MULTIDOS. MULTIDOS 2.1 is still available for the Models I, III, and 4 from Alphabit Communications, Inc., Box 20067, Ferndale, MI 48220.

Art and I both have heard of SUPERDOS, but have no other information on it. Art would like a copy for his archives if anyone has it.

These are all the DOSes Art and I know about, but there may well be others!

Another area of interest is the software which was written for the Model I. Kim Watt, a well known genius in the Model I, III and 4 era wrote a marvelous utility known as SuperUtility which allowed editing of disks and memory on a byte, sector or other basis. It had the ability to copy much protected software (except itself!). It's major failing was its inability to access hard drives. Later, Kim brought out other versions of the program which could access hard disk files, known as Tool Belt and Tool Box. These and SuperUtility were all acquired by and are now available through MISOSYS.

An interesting trick developed by Kim Watt was his technique for writing both single density AND double density sectors on the same track of a disk at the same time. This allowed him to offer only one disk which would autoboot on either a Model I in single density or a Model III in double density. This allowed him to stock only one disk for both machines. A darn clever trick!

HINTS & TIPS

Using Backup/cmd

By Karl Mohr

How many of us 'create' a booting system disk by copying another system disk and then deleting the files not needed. There are several ways of creating a new booting system disk by using the Backup/cmd utility program. The first step is to format the disk either single or double sided, then the command: Backup :s :d (s=source, d=destination) may be given, but this will create a 'mirror' image of the source if both source and destination disks are formatted the same (single or double sides). And will only copy visible files if they are not formatted the same, so another solution must be found.

The first step would be to backup ALL /sys files to the new disk, this is accomplished by giving the command:

Backup /sys:s :d (s)

The /sys 'partspec' will cause ALL files with that extension to be transferred from the :s (source) drive to the :d (destination) drive, therefore, all systems files from 0 to 13, if resident on the source drive, will be transferred to the destination drive. The (s) parameter gives the command to backup the SYSTEM files. When this backup is complete, you will have a booting system disk with only the systems files on it, the next step being in backing up other required utilities. This may be accomplished by the following command:

Backup :0 :1 (i,q), or Backup :0 :1 (inv,q=y)

The (i), or (inv) parameter gives the command to backup invisible files, and the (q), or (q=y) parameter will 'question you' about each file before it is backed up. Some of the files usually backed up are: Format/cmd, Backup/cmd, Log/cmd, Shell19/ovr (if shell is used), the 3 Basic files; /cmd, /ov1, /ov2, if Basic is required. To backup a file using this command, pressing 'Y' will cause the file to be backed up, pressing <ENTER> will skip that file and move on to the next file on which you will be queried.

A 'partspec' is part of a file, that may either be the beginning such as Basic/ or it may be the extension such as /sys, or /cmd. In either case, ALL partspec files with that beginning or extension will be transferred providing that the correct parameters are also given. The (s), or (sys) parameter pertains only to System files, the (i), or (inv) parameter pertains to Invisible files.

ELECTRIC WEBSTER REFERENCE GUIDE

from the TRSTimes vault

If you do any amount of word processing you need a way of correcting your bad spelling habits. ELECTRIC WEBSTER will do just that for you. Here is another 'quick reference guide to help you get the most out of this fine program.

INTRODUCTION

The following programs are included with Electric Webster:

EW/CMD	Proof Reads Text
M/EW	Links Electric Webster
M/CLW	to Appropriate Word
M/NEW	Processing Program.
PENCIL05/SYS	
DICT1/EW	Holds 50,000 Word
DICT2/EW	Dictionary
DICT3/EW	Data File for added words
*CORRECT1/EW	Displays misspellings and
*CORRECT2/EW	provide an opportunity to
/LEW	make corrections.
/NEW	
/PEW	
ADDTODIC/EW	Adds new words to Dictionary
EXAMPLE	Sample text file
*PATCH PROGRAM	integrates EW into your word processor
*PRINTDIC/EW	Utility for editing dictionary.

PREPARING A WORKING DISK

TRS-80 Model III (or Double density Model I) - You will first need prepare two working disks. To prepare working disk #2, the Dictionary Disk, transfer the dictionary files, DICT1/EW, DICT2/EW, and DICT3/EW onto a separate disk. Single drive users will need to copy DICT3/EW onto the disk with their program files, instead of this Dictionary Disk.

To prepare working disk #1 start with a disk with an operating system on it in drive #0 and transfer to this system disk your word processing program plus the following Electric Webster files.

Selecting appropriate CORRECT2/EW and M/EW files:

scripsit or superscript	Use original CORRECT2/EW and M/EW files.
Newscript	Copy CORRECT2/NEW AND M/NEW onto your working disk. Then rename them to CORRECT2/EW AND M/EW
Lazy Writer	Copy CORRECT2/LEW and M/CLW to your working disk then rename CORRECT2/LEW to CORRECT2/EW.

Electric Pencil	Copy CORRECT2/PEW and PENCIL05/SYS to your working disk, then rename CORRECT2/PEW to CORRECT2/EW

After you have made the appropriate changes for your word processor as instructed above, all CORRECT2 files without the EW extension may now be Killed from your working disk.

PATCHING INSTRUCTIONS

SCRIPSIT OR SUPERSCRIPT	Make sure SPATCH/EW is on one of the disks in your drives. Type SPATCH/EW and press <ENTER>. The disk drive will begin to turn. When they stop you'll see SCRIPSIT PATCHED or SUPERSCRIPT PATCHED. Electric Webster has been integrated into your word processing program.
ELECTRIC PENCIL (V2 only)	Make sure PPATCH/SYS is on one of the disks in your drives. Type PENCIL followed by a space followed by PPATCH = "PENCIL PPATCH". The disk drives will begin to turn. When they stop your Pencil program will appear on the screen, just as if you had loaded it normally. Hit any key to enter the TEXT ENTRY mode then press the <clear> and <k> keys simultaneously to enter the SYSTEM MENU mode.

Now, from the SYSTEM MENU type "SETUP:0". Electric Webster will then be integrated.

USING ELECTRIC WEBSTER

1. Insert the disk with your word processing program and Electric Webster programs on it in drive #0.

2. Type out or load from disk a document to be proofed.

3. The next step is to initiate the proofing process. If you are using Scripsit or Superscript press <BREAK> and type E followed by the document name<cr>. For Lazy Writer, press both the <clear> and <break> keys. Electric Pencil press <CONTROL>. For Newscript return to the main menu and type 3 for spelling checker.

If you are using the independent correcting version, from DOS READY, type EW<cr> then the document name when prompted.

4. Electric Webster will now process and proof your document, giving you a count of the number of multi-letter words in your document, and of the number of words in your document not counting duplication. When Electric Webster finishes proofing the document, a list of potential errors will be displayed on the screen and you will see the prompt, "PROOFING COMPLETE PRESS <ENTER> TO CONTINUE." After you have had the opportunity to look at the list, press <cr>.

ELECTRIC WEBSTER COMMANDS

Correct Misspelled Word	Type correct word
Leave word as is	Hit <ENTER>
Display Word in context	<?>
Display Dictionary	<@>
Scroll Dictionary up	<+>
Scroll Dictionary down	<->
Return to menu	<ENTER>
Add word to Dictionary	<+>
EXIT	<!>

WORDSTAR Command Summary

from the TRSTimes vault

WORDSTAR has long been a favorite in the CP/M world. It is a fine, full-featured word processor, but it uses an inane, difficult to remember command structure. For all the people who just happens to have a copy of WORDSTAR, but has mislaid the manual, the following may be of help.

Cursor Controls

^S	Character left
^D	Character right
^E	Up line
^X	Down line
^A	Left word
^F	Right word
^K 0-9	Set/hide 0 to 9
^QS	Left end of line
^QD	Right end of line
^QE	Top of screen
^QX	Bottom of screen
^QR	Beginning of file
^QC	End of file
^Q 0-9	Mark 0 to 9

Scroll

^Z	Scroll down one line
^R	Scroll down one screen
^W	Scroll up line
^C	Scroll up screen

Delete

^Y	Delete line
^T	Delete right word
^G	Delete character right
[DEL]	Delete character left
^Q[DEL]	Delete to left side of line
^QY	Delete to right side of line

Margins

^OL	Set left margin
^OC	Center text
^OF	Set file margin
^OR	Set right margin
^OX	Release margin

Special Function

^OW	Word wrap on/off
^B	Reform paragraph
^QP	Cursor to previous position
^OT	Ruler display off/on
^V	Insert on/off
^OD	Dot command display
^O	Status

Moving Blocks

^KB	Mark/hide block beginning
^KK	Mark block end
^KV	Move block
^KC	Copy block
^KY	Delete block
^KW	Write block
^KR	Read file
^KJ	Delete file
^KH	Hide/display marked
^QB	Cursor block beginning
^QK	Cursor block end

Save

^KD	Done edit
^KS	Save and re-edit
^KX	Save and exit
^KQ	Abandon edit

Print Functions

^PS	Underscore
^PT	Superscript
^PB	Boldface
^PH	Overprint
^PD	Double strike
^PX	Strikeover
^PV	Subscript
^KP	Print

Spacing

^OS	Line spacing

Find

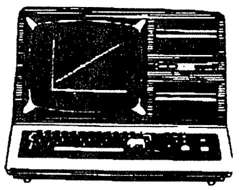
^QF	Find string
^QV	Cursor at starting point
^QA	Find and replace
^L	Find and replace again

Miscellaneous Commands

^QQ	Repeat next command
^U	Interrupt
^J	Help menu
^JH	Help level

Dot Command Summary

.LH	Line height
.CW	Character width
.PL	Paper length
.PO	Page offset
.MT	Margin top
.HM	Heading margin
.HE	Heading
.MB	Margin bottom
.FM	Footing margin
.FO	Footing
.PC	Page number column
.PA	New page
.CP	Conditional page
.OP	Omit page numbers
.PN	Page number
.IG(..)	Comment



NEW @BANK ROUTINE FOR THE RAI MEMORY BOARD

by J.F.R. "Frank" Slinkman

Back in the mid '80s, an outfit call RAI, in Hampton, Virginia, sold a one megabyte memory expansion board for the TRS-80 Model III and Model 4 which was similar to the Anitek board. However, there were enough differences between the RAI and Anitek boards where the same software will not work on both.

The driver supplied with the RAI board works fine on the Model 4 using TRSDOS 6.2.x, and with Model III LDOS, but will not work properly under LS-DOS 6.3.x. It can't handle the new dating scheme. Anitek's HyperDrive works with this board after a fashion, but the RAMdisk it creates cannot be used as the system drive.

A local (Richmond, Virginia) user brought this problem to me, and asked if I could solve it. The accompanying assembly language listings and patches are the solution, which I present here on the assumption that there must be other Model 4 users out there who have the RAI memory expansion board.

What these patches do is allow the LS-DOS 6.3.x system to recognize the existence of all but one of the thirty-two 32K "banks" on the RAI board via the @BANK supervisor call (SVC). Then, any software which obeys the rules about how to correctly access extra memory will operate.

In addition to the patches here, RAI memory board users will also have to obtain DiskNotes 3.2 from Misosys, Inc., P.O. Box 239, Sterling, VA 22170-0239. I believe Misosys wants $10 for the disk plus $3 S&H, but I suggest you check with Roy Soltoff before ordering. Misosys, Inc., can be reached at (703) 450-4181 (information) or 1-800-MISOSYS (orders ONLY).

DiskNotes 3.2 contains several programs written by Michel Houde' (pronounced "you-day"') of France, who wrote them to support the XLR8er board on the Model 4. The files you need are PEXMEM/CMD, which sets up the @EXMEM SVC (SVC 125); ERAMDISK/CMD, which is a RAMdisk driver which will support up to thirty 32K RAM banks; and ERAMLD/CMD, which allows you to save and load images of RAMdisks to a physical drive, facilitating the initiation of various different RAMdisk configurations.

One small patch to PEXMEM/CMD is desirable, although not necessary. PEXMEM gives it's low memory module the name "PXM." But the program ARC4V2, for example, looks for a module with the name "$XM," and will not operate unless it finds it. The optional patch to PEXMEM/CMD merely changes the "P" in the module name to "$". This does not affect the performance of ERAMDISK/CMD or ERAMLD/CMD which are equally happy with modules named either "$XM" or "PXM."

This patch is:

```
PATCH PEXMEM/CMD (D00,99='$';F00,99='P')
PATCH PEXMEM/CMD (D01,27='$';F01,27='P')
```

It must be applied before PEXMEM is invoked.

Of my patches, the patch to SYS0/SYS is the simpler; so let's look at it first. All it does is, upon bootup, change one invocation of the @BANK SVC to a CALL, and load the system's BUR$ and BAR$ bitmaps with the value 0FEH in each. These are the values a 64K machine would normally have. The reason for making the machine look like a 64K machine is to thwart attempts by nonstandard software to do illegal bank switching. Hopefully, if such software thinks it's dealing with a mere 64K machine, it will not attempt to play with the memory configuration.

The patch to BOOT/SYS contains the code which really does all the work. This code is based on the work done by Michel Houde', as modified by me back in 1989 to work with Richard King's 512K hardware mod, then modified by David Goben to reduce code size, and then modified by me again to further reduce code size and correct a "problem" involving Goben's use of BUR$ and BAR$ which, although perfectly legal, caused several popular programs to fail because of their attempts to manage extra banks in an illegal fashion.

Bank switching on the RAI board is rather simpler than the standard Model 4 method. All you have to do to switch banks is send the desired bank number out port 43H. However, you cannot determine the current bank number by reading that port; so you have to keep track of which bank is resident separately.

This is done by using the system's LBANK$ at 0202H. In addition, sending 00H and 01H out port 43H both result in the selection of RAM bank 0. This is due to the basic design of the Model 4. Therefore, even though there are 32 banks on the RAI board, only 31 can be accessed. As far as the system is concerned, these are numbered 0 through 30. However, for the sake of simplicity, the code in NEWBANK2/ASM numbers them 1 through 31.

To understand the NEWBANK2/ASM code, we need to go through the listing step by step.

This code starts at 0877H, replacing the @BANK SVC servicing routine.

Upon entry, the B register holds the @BANK SVC function code. Code 0 is "select bank;" Code 1 is "release bank;" Code 2 is "test bank;" Code 3 is "reserve bank;" and Code 4 is "return number of currently resident bank."

The C register holds the desired bank number. In addition, if B = 00H and Bit 7 of C is set, then a jump is to be made to the address in HL once the new bank has been switched in.

The first instruction saves the HL register pair on the stack.

Note the second instruction: LD A,C.

Normal SVC handling carries the byte destined for the A register in the C register, since the SVC number must be in the A register. After the SVC code has determined the address of the SVC handling routine, the value in C is moved into A, and the jump then made to the desired code.

This LD A,C instruction permits the @BANK routine to be CALLed, as well as be accessed as SVC 102. This speeds up RAMdisk operation by eliminating the overhead of SVC address decoding, and makes it possible to correct a situation in system code which results in inappropriate error handling.

The next instruction strips the "jump" bit from the bank number, and then a test is made to make sure the bank number is in the legal range of 0 to 30. Any value greater than 30 results in the return of Error 43, "Parameter error," this being accomplished by the system code starting at 0DEDH.

Assuming the bank number is legal, the function code in B is incremented to enable its value to be tested via highly efficient DJNZ instructions.

Thus, at SELBANK, if the function code in B was anything other than 00H, a jump is made to NOTSEL, and the value in B reduced by one when it gets there.

If the code was 00H (select bank), control falls through to the next instructions, which are a test to make sure the stack is not in an area where it would be destroyed by bank switching. If stack integrity is in danger, a "Parameter error" is returned, in accordance with the @BANK SVC specifications.

Assuming the stack is safe the bank number in A is incremented to bring it to a base 1 value rather than the system's base 0 scheme, and the requested bank is switched in by the write to port 43H.

Then the value is decremented back to the system's base 0 scheme, and written to LBANK$ so the system can know which bank is resident.

The remainder of the code above NOTSEL updates the bank number in C, preserving the "jump bit," returning to the caller if the jump bit is reset, or making the jump to the address in HL if the jump bit is set.

All other function codes, except Function Code 4, require addressing the 32-bit bitmap at XBUR$. A set XBUR$ bit means the corresponding bank is in use, or has been reserved. A reset XBUR$ bit means the bank is available.

Upon entry at NOTSEL, the bank number is in the A register, and HL has already been saved on the stack; so can be altered at will. All the other registers, however, must be preserved. The code from NOTSEL down to but not including RSTBUR creates a mask in the A register to be compared to an XBUR$ byte, and creates a pointer to the correct XBUR$ byte in HL.

At RSTBUR, the function code in B is tested via a DJNZ instruction. If the code was anything greater than 01H, control passes to TESTBUR. If the function code is 01H (release bank) the bitmask in A is complemented, then ANDed with the XBUR$ byte pointed to by HL, which leaves the masked bit reset. At SETMSK, the XBUR$ byte is replaced, indicating that bank is now free.

At TESTBUR, control passes to SETBUR if the function code was anything greater than 02H (test bank). To test whether or not this bank is available, the bitmask in A is ANDed with the XBUR$ byte. If the bank is free (i.e., the XBUR$ bit reset) the Z flag will be set. If the bank is in use or reserved, the Z flag will be reset.

Then, at DNA, the A register is loaded with 08H, the code for a "Device Not Available" error. Then the HL register pair is restored, and return is made to the caller. Thus, if the bank is available, the Z flag will be set upon return. If not available, the Z flag will be reset indicating an error, with "Device Not Available" indicated.

At SETBUR, if the function code in B was anything greater than 03H (reserve bank), control passes to GETBANK. To reserve a bank, we must first determine whether the bank is available. The OR (HL) and XOR (HL) instructions will set the Z flag if the bank is NOT available. If not available, the value in A will be 00H; so incrementing A will reset the Z flag, and a branch is made to DNA, where a "Device Not Available" error is generated.

If the bank is available, control passes to SETBUR1, where the XOR (HL) instruction restores the correct bits in A (i.e., the masked bit plus all the bits which were previously set in the XBUR$ byte) whereupon a jump is made to SETMSK, to update the XBUR$ byte and return to caller via BANKX.

At GETBANK, if the function code in B is greater than 04H, then the function code is illegal; so control passes to PERR, which restores the HL register pair, with return made to the caller via the system code @ 0DEDH, which generates a "Parameter error."

If the code was 04H, "Return current bank number," the A register is loaded with the value from LBANK$, the HL register is restored, the Z flag set to indicate "No error," and return is made to the caller.

Next is the 4-byte (32-bit) XBUR$ bitmap, which is initiated with all bits reset, indicating all banks are free and available for use.

The final section of code has to do with managing the shadow RAM which is used for keyboard and video display access. The RAI memory board allows any bank to be shadowed, but the system code starting at ENADIS_DO_RAM (0817H) only assumes the existence of Banks 0, 1 and 2, and always makes a write to the OPREG port with the Mem1 and Mem0 bits reset to make sure Bank 0 is switched in before any attempt to switch in the keyboard and video shadow RAM.

This write to OPREG interferes with our RAM bank management scheme, since BOTH port 43H and the OPREG port affect RAM bank selection; so system attempts at RAM bank selection must be intercepted and made to conform with our scheme.

Thus, at VDCTL, the contents of LBANK$ are examined. If Bank 0 is resident, this satisfies the system's requirements, since our scheme always leaves the Mem1 and Mem0 bits reset; so normal system access to the shadow RAM is permitted via the jump to @_VDCTL.

If the resident bank is other than Bank 0, then the number of the currently resident bank, in the A register, is saved on the stack via a PUSH AF instruction, Bank 0 is switched in by writing 00H to port 43H, and system access to shadow RAM is performed via a CALL to @_VDCTL. We can get away with this because the system code at ENADIS_DO_RAM only looks at the OPREG$ image of the OPREG port, which our scheme always leaves reset, no matter which bank is actually resident.

Upon return, the HL register is saved, and the original bank number is put into the H register, via the EX (SP),HL instruction. Then the AF register is saved, the original bank number moved from the H to the A register, incremented to conform with our offset 1 bank numbering scheme, and the original bank restored by writing this value to Port 43H. Then the AF and HL registers are restored from the stack; and return made to the calling routine.

The last code segment puts the address of our VDCTL intercept routine in the JP address @ 0B9AH, enabling our VDCTL shadow RAM handling intercept code.

Now we need to talk about the operation of the programs on the DiskNotes 3.2 disk you will be ordering from Misosys, Inc.

The PEXMEM module, as it uses SVC 125, is SYSGEN-able, and MUST reside in low memory. The installation code will tell you if installation fails due to lack of low memory. In other words, if PEXMEM "takes," you should immediately SYSGEN the new configuration.

However, if it does not "take," you need to reconfigure your system from scratch, making sure that both the PEXMEM module and any other modules, such as a hard disk driver, which must reside in low memory can and do fit there.

Once your system is reconfigured (if necessary), then you can SYSGEN.

ERAMDISK can support one, and only one, RAMdrive at a time. Therefore, it makes sense to make that drive as large as possible. The best practice would be to use the 28 RAM banks 3 through 30, which keeps banks 0, 1 and 2 free for the software which uses them, and creates an 896K RAMdisk.

To accomplish this, you would enter, at LS-DOS Ready:

```
ERAMDISK (D=d,T=4,B=3,S=28)
```

where:"d" is drive number you want to give to the RAMdisk, which must be a number between 0 and 7, and must not be the number of a drive already in use, which precludes ever using "D=0."

The "T" parameter specifies the "Type" of RAMdisk you want, where Type 2 creates a configuration of two sectors (0.5K) per granule; Type 4 sets up four sectors (1.0K) per granule; and Type 5 sets up six sectors (1.5K) per granule.

However, there are limitations. A Type 2 RAMdisk can hold a maximum of 25 banks, or 800K, and a Type 5 RAMdisk can hold a maximum of 13 banks, or 416K.

Of course, a Type 2 RAMdisk will store files more efficiently, but not efficiently enough to make up the 160K difference in size between its maximum size and the maximum size of a Type 4 RAMdisk.

A Type 5 RAMdisk has the minor advantage that you can do a mirror image BACKUP to a 5-1/4" floppy drive.

Thus, if you want the largest possible RAMdisk, you should specify Type 4.

However, if you are using a program which makes use of expanded RAM, such as David Goben's new spreadsheet and word processing programs, you may wish to set up a smaller, Type 2 RAMdisk, thereby letting the program use some of the extra RAM and let your RAMdisk use the rest.

The "B" parameter specifies the starting bank number and the "S" parameter specifies the number of banks to use. Thus, if you wanted to set up the largest possible RAMdisk -- a 960K drive using the 30 banks from 1 through 30, you would enter:

```
ERAMDISK (D=d,T=4,B=1,S=30)
```

Typing ERAMDISK, with no parameter specified, gives you a list of all the parameters, their functions and uses. The most important of these being the "V" (verify) and "F" (format) parameters.

The "V" parameter forces all RAM to be tested and verified. This should be used occasionally to make sure all your RAM is good, or whenever you suspect you may have a problem with your RAM.

The "F" parameter allows you to restore a RAMdisk after an accidental reboot or program crash. Adding "F = N" to the parameter string suppresses RAMdisk formatting, enabling you to recover a RAMdrive, assuming nothing has happened to destroy the contents of the RAM.

To recover a 896K Type 4 RAMdisk, for example, you would enter:

```
ERAMDISK (D=d,T=4,B=3,S=28,F=N)
```

If something HAS happened to destroy the RAM, ERAMDISK will tell you so, and abort.

ERAMLD/CMD allows you to easily populate an ERAMDISK, and is very useful at bootup time.

Say you want to have a special RAMdisk for using VisiCalc, and you want this RAMdisk to be the system drive.

First you would use ERAMDISK to create the RAMdisk. Next, change the way the system looks at the drive by executing the following LS-DOS command:

```
MEMORY (A="A",B=0)
```

This forces the most compact possible storage of disk files on the drive.

Now BACKUP the system files you need to the RAMdisk (you don't need SYS0/SYS, and you probably don't need SYS5/SYS, SYS9/SYS or SYS13/SYS). Thus, if your RAMdisk is Drive 2, at LS-DOS Ready you would enter:

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

and answer the prompt for each file with a "Y" or "N" as appropriate.

Then you would backup VC/CMD to the RAMdisk.

Once the RAMdisk is set up the way you want, then you can invoke ERAMLD to save the contents to a physical drive.

This is a little involved, and requires some memorization on your part. In fact, it is best of you start all your RAMdisk configurations with the same bank number so you won't have to remember which /RAM file starts at which bank.

To save a RAMdisk image, you need to do a DIR :d (I,S,N) from LS-DOS Ready to see the total size of all the files on the RAMdisk. If the total size is, for example, 76K, then you need to save three banks, since 2 banks is only 64K, while 3 banks is 96K. You should also do a FREE :d command to make sure all the granules in use are contiguous.

If you can't remember the starting bank number, the MEMORY command generates a map of all banks in use; so you can simply count from left to right, starting at zero, until the first "+," indicating a bank in use.

For example, a map of:

```
"---+++++++++++++++++++++++
+++++++"
```

indicates Bank 3 is the lowest in use, since Banks 0, 1 and 2 all show minus signs.

Armed with this knowledge -- the number of banks to save and the lowest bank number in use (both "3" in this case), we can invoke ERAMLD/CMD via:

```
ERAMLD filename:d (DUMP,B=3,S=3)
```

In this example, since we're saving the VisiCalc configuration, "filename" would probably be "vc." Obviously, ":d" should specify a physical drive, not the RAMdisk itself. Using the suggested name, the command will dump the RAMdisk image to a file named VC/RAM on the specified drive.

Then, whenever you wish to work with VisiCalc, you would go through the following sequence of commands:

```
ERAMDISK (D=d,T=4,B=3,S=28)
ERAMLD VC (LOAD,B=3,FORCE)
SYSTEM (DRIVE=0,SWAP=d)
```

The first command above creates the RAMdisk as Drive "d." The second loads the image from physical disk to the RAMdisk, and the third command makes the RAMdisk the system drive.

These commands could easily be built into a /JCL file, the use of which will eliminate the need to remember the starting bank numbers, since that information can be "remembered" in the JCL commands. In other words, once you set up your RAMdrive, save its image to disk, and correctly build the /JCL file, all you'd ever have to remember after that is the name of the /JCL file.

RAI is still in business, but they no longer handle TRS-80 products. Therefore, if you have a RAI memory expansion board, the patches and software described in this article are the only way you now have to get the board to work properly under LS-DOS 6.3.x.

BOOTSYS/FIX

```
.BOOTSYS/FIX - New @BANKer to recognize RAI Memory Board (31-Dec-91)
.apply via:  PATCH BOOT/SYS.SYSTEM6 [using] BOOTSYS/FIX
.must also PATCH SYS0/SYS.SYSTEM6 [using] SYS0SYS/FIX
.@BANK beginning @x'0877'
d06,77=E5 79 E6 7F FE 1F 30 4D 04
f06,77=E6 7F FE 03 D2 ED 0D 05 FA
d06,80=10 19 21 05 80 39 38 44 3C D3 43 3D 21 02 02 46
f06,80=B3 08 0E 86 28 19 0E 46 05 28 14 05 28 09 05 C2
d06,90=77 A9 B0 4F A8 06 00 E1 C8 BF E9 3C 67 2E D4 3E
f06,90=ED 0D 3A 02 02 BF C9 47 CD 9F 08 C0 78 0E C6 E6
d06,a0=80 07 30 01 2D 25 20 F9 26 08 10 05 2F A6 77 18
f06,a0=07 07 07 07 B1 32 B0 08 AF 3E 08 E5 21 00 02 CB
d06,b0=18 10 05 A6 3E 08 E1 C9 10 0A B6 AE 20 03 3C 18
f06,b0=46 E1 C9 E5 21 05 80 39 E1 DA ED 0D FE 01 17 47
d06,c0=F3 AE 18 EA 10 06 3A 02 02 E1 BF C9 E1 C3 ED 0D
f06,c0=3A 01 02 A0 20 C9 78 1F 3F CE 00 07 07 07 07 47
d06,d0=00 00 00 00 3A 02 02 B7 CA 42 0D F5 AF D3 43 CD
f06,d0=3A 78 00 E6 8F B0 32 78 00 D3 84 3A 02 02 47 79
d06,e0=42 0D E3 F5 7C 3C D3 43 F1 E1 C9
f06,e0=E6 7F 32 02 02 A9 B0 4F CB 79 06
.@VDCTL @x'0b9a' JP to fix -- ensure RAM shadowing
d09,9a=D4 08;f09,9a=42 0D
.eop
```

SYS0SYS/FIX

```
.SYS0SYS/FIX - To use RAI Memory Board (31-Dec-91)
.apply via:  PATCH SYS0/SYS.SYSTEM6 [using] SYS0SYS/FIX
.must also PATCH BOOT/SYS.SYSTEM6 [using] BOOTSYS/FIX
.fix call to modified @bank instead of SVC @x'1a16' (disk I/O)
d09,74=CD 77 08
f09,74=3E 66 EF
.set BUR$ & BAR$ to indicate 64K machine to thwart illegal bank switching
.@'1e79' & x'219A'
d0d,07=CD 9A 21
f0d,07=00 00 00
d10,34=21 FE FE 22 00 02 C9
f10,34=00 00 00 00 00 00 00
.eop
```

PEXMEM/FIX

```
. PEXMEM/FIX - change module name from 'PXM' to '$XM' (31-Dec-91)
. apply via: PATCH PEXMEM/CMD [using] PEXMEM/FIX
PATCH PEXMEM/CMD (D00,99='$';F00,99='P')
PATCH PEXMEM/CMD (D01,27='$';F01,27='P')
.eop
```

NEWBANK/ASM

```
;         NEWBANK2/ASM
;
BUR$      EQU     200H
LBANK$    EQU     202H
@_VDCTL   EQU     0D42H          ;addr of DOS @vdctl routine
@PERR     EQU     0DEDH          ;generate 'parameter error'
;------------------------------------------------------------------
          ORG     0877H          ;in BOOT/SYS
;
BANK      PUSH    HL
          LD      A,C            ;p/u requested bank #
          AND     7FH            ;strip "jump" bit
          CP      30+1           ;in legal range?
          JR      NC,PERR        ;error if not
          INC     B              ;in case function code = 0
;
SELBNK    DJNZ    NOTSEL         ;go if code < > 0
          LD      HL,8005H       ;are there 3 levels of
          ADD     HL,SP          ;stack below x'8000'?
          JR      C,PERR         ;error if not
          INC     A              ;change range from 0-30
                                 ;to 1-31
          OUT     (43H),A        ;switch in requested bank
          DEC     A              ;change range back to 0-30
          LD      HL,LBANK$
          LD      B,(HL)         ;current bank #
          LD      (HL),A         ;write new bank# to LBANK$
          XOR     C              ;isolate "jump" bit in A
          OR      B              ;merge in old bank #
          LD      C,A            ;old bank # w/"jump" bit
          XOR     B              ;set Z if no jump to bank
          LD      B,0            ;B = 0 for return
          POP     HL
          RET     Z              ;if no bank jump
          CP      A              ;set Z for return
          JP      (HL)           ;go to jump address
;
NOTSEL    INC     A              ;in case bank 0
          LD      H,A            ;counter = bank# +1
                                 ;(range 1-31)
          LD      L,.LOW.XBUR$+4
          LD      A,80H          ;bit mask
BLOOP     RLCA                   ;shift mask bit
          JR      NC,BLOOP1      ;cross byte boundary?
          DEC     L              ;for banks 0, 8, 16 & 24
```

```
BLOOP1 DEC     H
       JR      NZ,BLOOP
       LD      H,.HIGH.XBUR$
                          ;complete XBUR$
                          ;pointer
;
RSTBUR DJNZ    TESTBUR    ;go if code < > 1
       CPL                ;reverse mask
       AND     (HL)       ;reset masked bit
SETMSK LD      (HL),A     ;replace XBUR$ byte
       JR      BANKX
;
TESTBUR DJNZ   SETBUR     ;go if code < > 2
       AND     (HL)       ;is masked bit set?
DNA    LD      A,8        ;'dev not avail' if NZ
       POP     HL
       RET
;
SETBUR DJNZ    GETBANK    ;go if code < > 3
       OR      (HL)       ;combine mask &
                          ;XBUR$ bits
       XOR     (HL)       ;set Z if bit was already
                          ;set
       JR      NZ,SETBUR1;go if XBUR$ bit was
                          ;not set
       INC     A          ;else reset Z flag and
                          ;return w/NZ
       JR      DNA        ;and Device Not
                          ;Available error
SETBUR1 XOR    (HL)       ;restore A to merged
                          ;mask & XBUR$ bits
       JR      SETMSK
;
GETBANK DJNZ PERR         ;go if code > 4
       LD      A,(LBANK$) ;p/u res bank #
BANKX  POP     HL
       CP      A
       RET
;
PERR   POP     HL
       JP      @PERR
;
XBUR$  DW      0
       DW      0          ;bitmap for banks-in-use
;
VDCTL                     ;ensure @_vdctl RAM
                          ;shadowing
       LD      A,(LBANK$) ;p/u bank #
       OR      A          ;is it bank 0?
       JP      Z,@_VDCTL  ;OK if so
       PUSH    AF         ;else save current bank #
       XOR     A
       OUT     (43H),A    ;switch in bank 0
       CALL    @_VDCTL    ;do *DO processing
       EX      (SP),HL    ;save HL, get orig bank
                          ;# in H
       PUSH    AF
       LD      A,H
       INC     A          ;A = orig bank # + 1
       OUT     (43H),A    ;restore original bank
       POP     AF
       POP     HL
       RET
;
LAST1  EQU     $-1
;----------------------------------------------------------------
       ORG     0B9AH      ;in BOOT/SYS
       DW      VDCTL      ;intercept @vdctl I/O
;----------------------------------------------------------------
       ORG     1A16H      ;in SYS0/SYS
       CALL    BANK
;----------------------------------------------------------------
       ORG     1E79H      ;in SYS0/SYS
       CALL    SETUP      ;initialize BAR$ & BUR$
;----------------------------------------------------------------
       ORG     2194H      ;in SYS0/SYS
;
SETUP  LD      HL,0FEFEH  ;64K machine values to stop
       LD      (BUR$),HL  ;non-SVC bank fiddling
       RET
;
SIZE3  EQU     $-SETUP
;
       END
```

ITEMS FOR SALE

RADIO SHACK 5 meg hard drive for sale;
only one available for $150.

Tom Wass
P.O Box 1735
Beverly Hills, CA 90213

MODEL II SOFTWARE WITH MANUALS FOR SALE

VISICALC	RS 26-4511	$25
SCRIPSIT 2.0	RS 26-4531	$20
SCRIPSIT UTILITY PACK	RS 26-4532	$20
COMPILER BASIC	RS 26-4705	$25
PROFILE II	RS 26-4512	$25
PROFILE FORMS	RS 26-4556	$20
TRSDOS 2.0	RS 26-4910	$20

Lance Wolstrup
c/o TRSTimes magazine
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

Boot your Hard Drive in LS-DOS and LDOS without a floppy! Part II

By Roy T. Beck

This is the continuation of the Autoboot article which began in the last issue of TRS Times. Since then, I have learned several things which I will pass along.

First of all, and most important, M.A.D. Software is NOT going out of business. That story began (erroneously) with a small item in a recent issue of Computer News 80.

I have since been in touch with M. A. D. Software, and the problem actually began when someone else misinterpreted a flyer sent by M.A.D. to a customer. M.A.D.'s flyer said certain items would not be available after a date in early 1992, and apparently this got to CN-80 in a garbled form, which caused them to report the impending demise of M.A.D. TAIN'T SO! In fact, M.A.D. has just announced they will have some new utilities and games available in 1992. Hurray, I'm glad they are staying with us.

By the way, I incorrectly reported M. A. D.'s ZIP code in the last article. The correct code is 76163. Mike said several orders came through to them even with the wrong code, but don't tempt fate, use the correct ZIP!

M. A. D. also sent me a BETA test version of two utilities they are preparing to offer. One utility named LOOK looks at a disk, and tells you what DOS wrote it (even including CP/M or MS-DOS), and presents a tabulation of the tracks and grans, complete with the filespec to which every individual gran belongs. Of course this can be sent to the printer. Looks like an attractive utility to have at hand.

The other utility is MAPMEM which shows the names and addresses of all the drivers and other modules resident in your memory at the moment. This will undoubtedly be of assistance to those doing software development.

By the way, do you know who M.A.D. Software really is? My best information is that the proprietor is Mike A. Durda, son of Frank Durda IV, the author of our Model 4P ROM! All in the family, you might say.

The next item of business is to relate my experiences in combining a SmartWatch with M.A.D.'s improved 4P EPROM. The SmartWatch will work nicely in the 4P with the original RS ROM, but requires some curious jumpers to do so. Three jumpers are required, one to get 5 Volts to the Smartwatch, one to supply a ground to it, and one to bring the A11 address signal to the RS PROM mounted on top of the SmartWatch.

When I went to install M.A.D.'s new EPROM with a SmartWatch, I found a supplementary instruction sheet in the package covering its installation with a SmartWatch. It clearly stated the 5 volt jumper should be removed, and implied the ground jumper was also not needed. It said nothing about the A11 jumper. I installed the M.A.D. EPROM, its adapter and the SmartWatch in my gate array 4P without the power jumpers. Since nothing was said about the A11 line, and since it went to the RS PROM, I assumed it was still required and installed it per the original SmartWatch instructions from Duane Saylor. The machine booted OK, so I did not pursue the matter, although a nagging doubt did remain.

Later, I tried the builtin diagnostics in the M.A.D. EPROM, and got strange results. Then I tried to boot a floppy in the 4P, but no dice. I wondered if there could be a problem in the EPROM, and swapped it with a different one (I have three of them for my three 4P's), but kept the A11 jumper. No luck, all three misbehaved exactly the same way. Obviously the trouble was not in the M.A.D. EPROM. I then removed the A11 jumper and inserted the EPROM pin into the SmartWatch socket. Every thing works! Neither the power, ground nor A11 jumper is required when you install the M.A.D. EPROM with the SmartWatch, as the M.A.D. adapter has been designed to correct all the compatibility problems. Anyone who was afraid of the SmartWatch because of the required jumper connections can forget all of that if he also installs the new M.A.D. EPROM in his 4P. Since M.A.D. supplies the EPROM adapter in order to be able to use ordinary EPROMS, I expect the Model 4 and 4D EPROMS will also have the adapter sockets, and this should allow the same simple installation of the SmartWatch in the Model 4 and 4D.

Now my 4P autoboots reliably, brings up the SmartWatch with correct date and time, and will boot from a floppy when I choose to do so. Thank you, Mike!

Incidentally, CN-80 is now offering the SmartWatch for $29.95 plus $3.00 S&H. Stan Slater, the publisher of CN-80, tells me his SmartWatch package includes the necessary TRS-80 software, as written by David Goben.

If anyone already has the SmartWatch, but lacks the software, I can supply a copy of Duane Saylor's shareware programs on a disk for $5.00, postpaid.

Mike Durda also informed me there have been some necessary price increases in his product line, due to price increases by Radio Shack and the supplier of the EPROM adapter. Check with M. A. D. Software for the latest prices.

The third area of discussion last time was the BOOT5/CMD by Adam Rubin which allows booting up LDOS from LS-DOS without need for a floppy.

Lance Wolstrup has solved all the problems of interfacing BOOT5 with both LDOS 5.3.0 and 5.3.1, and has altered BOOT5/CMD so it recognizes which version of LDOS it is calling up and patches it correctly. My hat is off to you, Lance, for a neat piece of detecting!

TRSTimes would like to make this version of BOOT5 available on an upcoming TRSTimes disk, but we need permission from Adam Rubin to do so, as his copyright notice specifically requested that no altered versions of the program be circulated. Can someone supply a good address and/or telephone number for him? Without his permission, TRS-Times cannot distribute the modified version of the program.

Duane Saylor's SmartWatch shareware package also includes a version of his software for use with LDOS, so when BOOT5 brings in LDOS 5.3.x, the auto command can also bring up the SmartWatch to work with it. GREAT!

And talk about speed. When I ask the 4P to run under LDOS, three things have to occur. The first is to load the ROM image (about 14K, I believe), the second is to load LDOS itself, and finally to enable the SmartWatch. When I give it the BOOT5 command, the elapsed time to LDOS ready is 4.5 seconds. That is astonishing! Of course, HDs are fast, but that is really moving. A Model 4 ought to be still faster, because it doesn't have to load the ROM image, as the ROM is already in the Model 4 and 4D. Further, if you don't have a SmartWatch, then you can eliminate the AUTO command and the time for it to go get the clock software. It would be interesting to try this combo, but I don't have a working Model 4 or 4D. Any takers?

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

YOU NEED 'DL'

'DL' will automatically read your TRSDOS6/LDOS compatible disk and then print a neat label, listing the visible files (maximum 16). You may use the 'change' feature to select (or reject) the filenames to print. You may even change the diskname and diskdate. 'DL' is written in 100% Z-80 machine code for efficiency and speed. Don't be without it - order your copy today.

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

'DL' for Model 4 only $9.95

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

HI-RES FUN

Model 4 and BasicG

By Lance Wolstrup

While calling a couple of the local Southern California BBS', I found some fun hi-res programs written in GW-Basic for the IBM-PC. Since GWBasic is nearly identical to Model 4 BasicG, it didn't take me long to do the translations.

Now, I don't claim to be even efficient writing code for the hi-res board, so I am just presenting the programs for you to have fun with. I am sure that many of you out there can spruce up the listings. I am particularly interested to see what you can do with HRCLOCK/BAS. Send your version to TRSTimes.

HRCLOCK/BAS

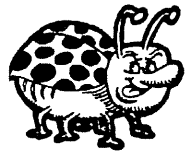
```
10 CLR
20 R=50:PI=3.14593
30 MINUTE=VAL(MID$(TIME$,4,2))
40 HOUR=VAL(MID$(TIME$,1,2))
50 CIRCLE(320,90),R
60 CIRCLE(320,90),2
70 LINE(380,60)-(380,220)
80 LINE(260,60)-(260,220)
90 LINE(260,60)-(380,60)
100 LINE(260,220)-(380,220)
110 LINE(260,120)-(380,120)
120 LINE(260,122)-(380,122)
130 GOSUB 440
140 PAINT(369,65),1,1
150 LINE(321,122)-(321,190)
160 LINE(320,122)-(320,190)
170 LINE(319,122)-(319,190)
180 CIRCLE(320,199),10
190 CIRCLE(320,199),4
200 PAINT(320,195),1,1
210 SCREEN 0
220 R1=R*.5:R2=R1*3/4
230 GOSUB 490
240 MHOLD=MINUTE
250 ANGLE=(PI/2)-(MINUTE*(PI/30))
260 ANGLE2=(PI/2)-(HOUR*(PI/6))-(MINUTE*(PI/360))
270 X1=R1*COS(ANGLE)
280 Y1=(5/6)*R1*SIN(ANGLE)
290 LINE(320,90)-(320+X1,90-Y1)
300 X2=R2*COS(ANGLE2)
310 Y2=(5/6)*R2*SIN(ANGLE2)
320 LINE(320,90)-(320+X2,90-Y2)
330 GOSUB 370
340 IF MINUTE=0 AND SECOND=0 THEN
GOSUB 490:GOTO 350 ELSE 360
350 IF HOUR>12 THEN HOUR=HOUR-12
360 GOTO 240
370 MINUTE=VAL(MID$(TIME$,4,2))
380 SECOND=VAL(MID$(TIME$,7,2))
390 HOUR=VAL(MID$(TIME$,1,2))
400 IF INKEY$="" THEN 410 ELSE SCREEN 1:END
410 IF MINUTE=MHOLD THEN 370
420 LINE(320+X1,90-Y1)-(320,90),0
430 LINE(320+X2,90-Y2)-(320,90),0
440 LINE(320,66)-(320,70)
450 LINE(320,109)-(320,113)
460 LINE(272,90)-(280,90)
470 LINE(360,90)-(368,90)
480 RETURN
490 GLOCATE(290,36),0:PRINT#-3,DATE$;:RETURN
```

HRSTAR/BAS

```
10 CLR:SCREEN 0
20 FOR I=200 TO 1 STEP -6
30 CIRCLE(320,120),I
40 NEXT
50 FOR I=1 TO 75
60 CIRCLE(320,120),I,,,,5/22
70 CIRCLE(320,120),I,,,,22/5
80 CIRCLE(320,120),I,0,,,5/22
90 CIRCLE(320,120),I,0,,,22/5
100 NEXT
110 IF INKEY$="" THEN 110 ELSE SCREEN 1:END
```

HRSTAR2/BAS

```
10 CLR:SCREEN 0
20 FOR I=110 TO 50 STEP-4
30 CIRCLE(320,120),I
40 NEXT
50 FOR I=1 TO 55
60 CIRCLE(320,120),I,,,,1/15
70 CIRCLE(320,120),I,,,,15/1
80 NEXT
90 IF INKEY$="" THEN 90 ELSE SCREEN 1:END
```



GIF4MOD4 AND HR2GIF

for the Model 4/4P/4D

They say a picture is worth 1,000 words. This picture was converted from GIF to TRS-80 format using GIF4MOD4. Until now, Model 4 users had no way to view GIF images or to send their own hi-res graphics creations to other types of computers. GIF4MOD4 will decode any GIF image up to 640 x 480 x 256 (VGA)

and put it on your hi-res screen. If you have no hi-res board, GIF4MOD4 puts it in an /HR disk file so you can dump it to your dot-matrix printer. HR2GIF converts /HR, /CHR and /BLK files to GIF format.

ORDER FORM -- Mail to
J.F.R. Slinkman, 1511 Old Compton Rd., Richmond, VA 23233
Please send me GIF4MOD4 & HR2GIF for my TRS-80 Model 4. I am enclosing $37.95 + $2.00 S&H (add $2.00 outside N. America. Virginia residents please add 4-1/2% [$1.71]).
Name________________________________
Address______________________________
City__________________State__________Zip__________

MISOSYS, Inc.

Aerocomp Hardware is now available from MISOSYS

Model I DDen Controller (DDC)	$45 + $6S&H
Model III/4 FDC board	$45 + $6S&H
Model III/4 RS232 board	$45 + $6S&H
Model III/4 RS232 Kit	$50 + $6S&H
Aerocomp 5 Meg HD	$250 + S&H
Aerocomp 20 Meg HD	$400 + S&H
Aerocomp 40 Meg HD	$500 + S&H
MM CP/M 2.2 HD drivers	$29.95 + $3S&H

***The MISOSYS Quarterly* subscriptions**

Keep up to date with the latest information on MISOSYS products, programs, patches, and articles in a professional magazine format. A subscription to *TMQ* will provide you with information, news, and announcements concerning our entire product line and related machine environments. **As a special for new subscribers, we'll start you off with issue VI.i and send you a complete past volume of four issues at no charge.** That's eight issues for the price of four! Subscriptions are: $25 US; $30 Canada; $35 Europe; $40 Australia

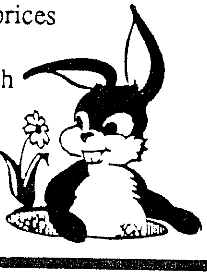
The Cornsoft Group
Model I/III Action Games Special

All five games on a single disk for one low price! You get Bounceoids, Crazy Painter, Space Castle, Scarfman, and The Official Frogger; all five for $20 + $3S&H

PRO-WAM Year-End Price Plunge

If you are not using PRO-WAM on your 128K Model 4, you're not using your 4! You'll get a pop-up desktop manager with ADDRESS, BRINGUP, CAL, CARD, CALC, PHRASE, and more. Export|Import across windows. PSORT your data files. $37.48 + $6S&H

Sale prices good through Jan 31st.



MISOSYS, Inc.
P. O. Box 239
Sterling, VA 22170-0239
703-450-4181
orders: 800-MISOSYS

DoubleDuty doubles your 128K Model 4
Now on sale at half price!

DoubleDuty divides your Mod4's memory into three complete and independent partitions. Two operate as they were each their own 64K computer. Get the best task switcher you can buy. Our 2.6 release also works with extended memory. If you thought you needed a second computer, think again. At $24.98 + $3S&H, you can't afford to not have DoubleDuty.

ATTENTION TRSDOS 1.3. USERS!

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

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

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

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

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

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

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

ORDER YOUR COPY TODAY!

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

TRSTimes - SYSTEM 1.5.

5721 Topanga Canyon Blvd., Suite 4
Woodland Hills, CA. 91367

Model 4
Public Domain Disks

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
M4GOODIES#4: WORD WIZARD disk #1 of 3
anima/bas, archi/bas, autos/bas, battuere/bas, captus/bas, contvert/bas, curro/bas, dico/bas, ducere/bas, eulogos/bas, facere/bas, fluere/bas, gradi/bas, jacere/bas, kata/bas, male/bas, metron/bas, naus/bas, startup/bas, startup/jcl, stig/bas, tangere/bas, wordmenu/bas
M4GOODIES#5: WORD WIZARD disk #2 of 3
cognos/bas, frangere/bas, juris/bas, medius/bas, mittere/bas, monos/bas, numbers/bas, orare/bas, pandemos/bas, para/bas, pathos/bas, pendere/bas, philanth/bas, phongrap/bas, polynom/bas, prefix1/bas, prefix2/bas, premere/bas, sal/bas, startup/bas, startup/jcl, statuere/bas, wordmenu/bas
M4GOODIES#6: WORD WIZARD disk 3 of 3
bible/bas, french1/bas, french2/bas, french3/bas, italian/bas, latphras/bas, lit1/bas, lit2/bas, myths/bas, piaces/bas, plicare/bas, spanish/bas, stagnare/bas, stare/bas, startup/bas, startup/jcl, synpath/bas, televid/bas, tenere/bas, vaco/bas, valere/bas, vox/bas, wordmenu/bas
M4GOODIES#7: calendar/cmd, castladv/bas, civilwar/bas, crimeadv/bas, dctdsp/cmd, ed6/cmd, ed6/doc, edittext/bas, fedit/cmd, mail/bas, mail/txt, scramble/bas, states/bas, textpro/cmd, time4/bas, wizard/bas, wizard/doc, worldcap/bas
M4GOODIES#8: books/bas, books/doc, dmu/cmd, dmu/doc, hamcalc/bas, hamhelp/bas, network/bas, network/doc, pirate/bas, pirate/doc, vmap/bas, vmap/doc, vmap2/bas, vmap2/doc, zork1/doc, zork2/doc, zork3/doc
M4GOODIES#9: ft/cmd, ft/doc, pterm/cmd, pterm/doc, r/cmd, r/doc, scrconv/bas, scrconv/doc, video4/asm, video4/cmd
M4GOODIES#10: checker/cmd, crossref/cmd, crossref/doc, ddir/cmd, diskcat/cmd, diskcat/doc, division/bas, division/doc, getput/bas, getput/doc, host/cmd, hv/bas, maszap4/cmd, maszap4/doc, park/cmd, profile4/doc, protect/bas, protect/doc, rename/bas, replace/bas, restore/bas, rm/bas, scrndump/bas, scrndump/doc, super/hlp, vers/cmd
M4GOODIES#11: benchmrk/bas, bigcal/bas, bigcal/doc, birthday/bas, dearc4/cmd, dezip2/cmd, dname/cmd, docufile/bas, docufile/doc, docufile/mrg, escape/bas, mem4/cmd, million/bas, nomad/bas, password/bas, password/dat, password/doc, password/jcl, roman/bas, sixtymin/bas, startrek/bas, trekinst/bas
M4GOODIES#12: awari/bas, buyimg/bas, crasher/bas, curvfit2/bas, gradebk/bas, mortcost/bas, mortcost/doc, print/bas, print/doc, reiman/bas, square/bas, starlane/bas, staybus/bas, sunrise/bas, synonym/bas, timezon1/bas, timezon2/bas, travel/bas, vmap2/bas, vmap2/doc, weekday/bas
M4GOODIES#13: calndr1/bas, calndr2/bas, calndr3/bas, formltrs/bas, invloan/bas, limerick/bas, martian/bas, mission/bas, moneymkt/bas, munchmth/bas, numbrfun/bas, smith/bas, smith/doc, star2000/bas, starfind/bas, starfind/dat, starfind/doc, starfind/jcl, states/bas, wallst/bas
M4GOODIES#14: alphahex/bas, bowlchng/bas, bowlcrea/bas, bowldetl/bas, bowlfinl/bas, bowling/doc, bowlmenu/bas, bowlprnt/bas, bowlrcap/bas, bowlrecd/bas, bowlrecp/bas, bowlschd/bas, bowlscor/bas, bowlsort/bas, buscheck/bas, calculat/bas, chekform/bas, deprec/bas, futrdate/bas, membrain/bas, minimath/bas, normalz/bas, numconv/bas, pcbdest/bas, pcbdest/doc, pcform/bas, pcpm/bas, pcpm/doc, pcpm/jcl, utscan/bas, yagibeam/bas, zeller/bas
M4GOODIES#15: laughs/bas, laughs/dat, laughs/doc, laughs1/dat, laughs2/dat, laughs3/dat, laughs4/dat, laughs5/dat, laughs6/dat, laughs7/dat, laughs8/dat, laughs9/dat, laughs10/dat, laughs11/dat, laughs12/dat, laughs13/dat, laughs14/dat, laughs15/dat
M4GOODIES#16: trivia/bas, trivia/doc, trivia1/dat, trivia2/dat, trivia3/dat, trivia4/dat
M4GOODIES#17: acrs/bas, amorloan/bas, clokmod/bas, compound/bas, dcform/bas, decide/bas, easyword/bas, editno/bas, epslabel/bas, esckey/bas, expect/bas, funct1/bas, funct2/bas, gasform/bas, hexprint/bas, hexsay/bas, lostgold/bas, mathfunc/bas, mpgcalc/bas, neclabel/bas, nicelist/bas, nonlin/bas, nonlin/rem, payback/bas, peekprnt/bas, percent/bas, prntcall/bas, proverbs/bas, randseed/bas, savings/bas, speech/bas, tasklist/bas, tempconv/bas, weightfm/bas

Each disk is $5.00 (U.S.)
or get any 3 disks for $12.00 (U.S.)
please specify the exact disks wanted.

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

Model I & III Public Domain Disks

PD#1: binclock/cmd, binclock/doc, checker/bas, checker/doc, chomper/bas, cls/cmd, dduty3/cmd, driver/cmd, driver/doc, drivtime/cmd, mazeswp/bas, minibase/bas, minitest/dat, mx/cmd, piazza/bas, spdup/cmd, spdwn/cmd, vici/bas, vid80/cmd, words/dic.

PD#2: creator/bas, editor/cmd, maze3d/cmd, miner/cmd, note/cmd, poker/bas, psycho/cmd, supdraw/cmd, vader/cmd

PD#3: d/cmd, trsvoice/cmd, xmodem/cmd, xt3/cmd, xt3/txt, xthelp/dat

PD#4: cobra/cmd, disklog/cmd, flight/bas, flight/doc, narzabur/bas, narzabur/dat, narzabur/his, narzabur/txt, othello/bas, vid80x24/cmd, vid80x24/txt

PD#5: eliza/cmd, lu31/cmd, sq31/cmd, usq31/cmd

PD#6: clawdos/cmd, clawdos/doc, cocoxf40/cmd, dskrnam/bas, menu/cmd, ripper3/bas, sky2/bas, sky2/his, space/cmd, stocks/bas, trs13pat/bas, vidsheet/bas

PD#7: cards/bas, cities/bas, coder/bas, eye/bas, heataudt/bas, hicalc/bas, life/bas, moustrap/bas, ohare/bas, slots/bas, stars/cmd, tapedit/bas

PD#8: craps/bas, fighter/bas, float/bas, hangman/bas, jewels/cmd, lifespan/bas, varidump/bas, xindex/bas, xor/bas

PD#9: bublsort/bas, chess/bas, finratio/bas, homebudg/bas, inflat/bas, mathdril/bas, midway/bas, nitefly/bas, pokrpete/bas, teaser/bas

PD#10: ltc21/bas, ltc21/ins, lynched/bas, match/bas, math/bas, message/bas, message/ins, portfol/bas, portfol/ins, spellegg/bas, storybld/bas

PD#11: alpha/bas, caterpil/cmd, cointoss/bas, crolon/bas, cube/cmd, dragon/cmd, fastgraf/bas, fastgraf/ins, lunarexp/bas, music/bas, music/ins, planets/bas, volcano/cmd

PD#12: baccarat/bas, backpack/bas, backpack/ins, doodle/bas, dragons/bas, dragons/ins, king/bas, sinewave/bas, snoopy/bas, wallst/bas, wallst/ins

PD#13: atomtabl/bas, boa/bas, chekbook/bas, conquer/cmd, dominos/bas, morse/bas, mountain/bas, quiz/bas, signbord/bas, sketcher/bas

PD#14: autoscan/bas, checkers/bas, craps/bas, ducks/bas, isleadv/bas, nim/bas, rtriangl/bas, sammy/cmd, typing/bas, wordpuzl/bas

PD#15: budget/bas, corp/bas, corp/ins, fourcolr/bas, fullback/bas, grapher/bas, illusion/bas, jukebox/bas, ledger/bas, maze/cmd, reactest/bas, shpspree/bas, states/bas, tapecntr/bas, tiar/bas, tiar/ins

PD#16: amchase/bas, constell/bas, filemastr/bas, foneword/bas, geometry/bas, heartalk/bas, hidnumbr/bas, lgame/bas, marvello/bas, powers/bas, scramble/bas, speed/bas, subs/bas

PD#17: conundrm/bas, eclipse/bas, esp/bas, esp/ins, hustle/bas, jacklant/bas, mindblow/bas, othello/bas, pleng/bas, rubik/bas, trend/bas, ufo/bas, veggies/bas

PD#18: backgam/bas, chess/cmd, cosmip/cmd, distance/bas, hexpawn/bas, music/cmd, stokpage/bas, texted/bas, texted/ins, trex/bas, twodates/bas, wanderer/bas

PD#19: banner/bas, cresta/cmd, lander/bas, medical/bas, moons/bas, par/bas, parchut/bas, pillbox/bas, readtrn/bas, replace/bas, ship/cmd, solomadv/bas, space/cmd, survival/bas

PD#20: bomber/bas, bumbee/cmd, ciaadv/bas, dice31/bas, dice31/ins, diskcat1/bas, firesafe/bas, flashcrd/bas, hitnmiss/bas, mazegen/bas, mazescap/cmd, roulette/bas, seasonal/bas

PD#21: aprfool/bas, catmouse/bas, d/cmd, escape/bas, header/bas, kalah/bas, mathwrld/bas, nameit/bas, note/cmd, photo/bas, read/cmd, syzygy/bas, timeshar/cmd, timeshar/doc, trace80/cmd, trsdir/cmd, worm/bas, yatz80/bas

PD#22: arcade/bas, cube/cmd, eclipse/bas, lcd/bas, leastsqr/bas, medical/bas, million/bas, pwrplant/bas, round/bas, subway/bas, tapeid/bas

PD#23: artil/bas, artil/ins, baseconv/bas, crushman/bas, dissert/bas, huntpeck/bas, jungle/bas, jungle/ins, messages/bas, monitor/bas, monster/bas, moons/bas, ohmlaw/bas, stockpage/bas, tictacto/bas

PD#24: baslist/asm, baslist/cmd, baslist/doc, cleaner3/cmd, cleaner3/doc, difkit1/bas, difkit1/doc, dirpatch/asm, dirpatch/cmd, e/cmd, ei/doc, i/cmd, newmap/bas, newmap/doc, varlst/asm, varlst/cmd, varlst/doc

PD#25: copy/bas, copy/doc, dirpw/asm, dirpw/cmd, dirpw/doc, dskfmt/bas, dskfmt/doc, himap/asm, himap/cmd, huricane/bas, hv/bas, hv/doc, keydemo/bas, keyin/bas, keyin/doc, lazyptch/asm, lazyptch/doc, salvage/bas, salvage/doc,wpflt/asm, wpflt/flt

PD#26: constell/bas, divisor/bas, frame/bas, heatfus/bas, heatfus/doc, hicalc/bas, mathlprt/bas, mathquiz/bas, molecule/bas, morscode/bas, phyalpha/bas, phyalpha/doc, remaindr/bas, usa/bas, wiring/bas

PD#27: engine/bas, fraction/bas, geosat/bas, grades/bas, julian/bas, lunarcal/bas, mailist/bas, metaboli/bas,musictrn/bas, perindex/bas, potrack/bas

PD#28: chainfil/bas, citoset/bas, convnum/bas, cursors/bas, cursors/doc, datamkr/bas, deprec/bas, gmenuii/bas, ledger12/bas, menui/bas, menuii/bas, minives/bas, ninteres/bas, refinanc/bas, regdepo/bas, rembal/bas, rndbordr/bas

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

please specify the exact disks wanted.

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

*** DR. PATCH ***

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

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

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

DISABLE PASSWORD CHECK IN FORMAT/CMD
FORMAT DOUBLE-SIDED AS DEFAULT
FORMAT 80 TRACKS AS DEFAULT
DISABLE VERIFY AFTER FORMAT
CHANGE 'DIR' TO 'D'
CHANGE 'CAT' TO 'C'
VIEW DIR/CAT WITH (I) PARAMETER AS DEFAULT
VIEW DIR/CAT WITH (S,I) PARAMETERS AS DEFAULT
CHANGE 'REMOVE' TO 'DEL'
CHANGE 'RENAME' TO 'REN'
CHANGE 'MEMORY' TO 'MEM'
CHANGE 'DEVICE' TO 'DEV'
DISABLE THE BOOT 'DATE' PROMPT
DISABLE THE BOOT 'TIME' PROMPT
DISABLE FILE PASSWORD PROTECTION
ENABLE EXTENDED ERROR MESSAGES
DISABLE PASSWORD CHECK IN BACKUP/CMD
BACKUP WITH (I) PARAMETER AS DEFAULT
BACKUP WITH VERIFY DISABLED
DISABLE BACKUP 'LIMIT' PROTECTION
DISABLE PASSWORD CHECK IN PURGE
PURGE WITH (I) PARAMETER AS DEFAULT
PURGE WITH (S,I) PARAMETERS AS DEFAULT
PURGE WITH (Q=N) PARAMETER AS DEFAULT
IMPLEMENT THE DOS 'KILL' COMMAND
CHANGE DOS PROMPT TO CUSTOM PROMPT
TURN 'AUTO BREAK DISABLE' OFF
TURN 'SYSGEN' MESSAGE OFF
BOOT WITH NON-BLINKING CURSOR
BOOT WITH CUSTOM CURSOR
BOOT WITH CLOCK ON
BOOT WITH FAST KEY-REPEAT

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

***** DR. PATCH *** $14.95**

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

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

DON'T LET YOUR LS-DOS 6.3.1 BE WITHOUT IT!

GRAPHICS-90

for Model III TRSDOS 1.3 or LDOS

The definitive graphics program that Radio Shack SHOULD have released is now, through the courtesy of author Larry Payne and the hard work of Gary Shanafelt, available from TRSTimes. Our agreement is to sell this wonderful package at COST. Neither Larry Payne, Gary Shanafelt nor TRSTimes will make a dime on this venture. The price charged will cover the cost of reproducing the manual, disks, and mailing only. The only one profiting is YOU!!

GRAPHICS-90 $6.75

Shipping: U.S. & Canada: add $3.00
Europe: add $6.25 air mail or $4.25 surface mail
Asia, Austrailia & New Zealand: add $7.25 air mail or $4.25 surface mail
Please specify TRSDOS 1.3 or LDOS format

TRSTimes magazine - dept. GR90
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

RECREATIONAL & EDUCATIONAL COMPUTING

REC is the only publication devoted to the playful interaction of computers and 'mathemagic' - from digital delights to strange attractors, from special number classes to computer graphics and fractals. Edited and published by computer columnist and math professor Dr. Michael W. Ecker, REC features programs, challenges, puzzles, program teasers, art, editorial, humor, and much more, all laser printed. REC supports many computer brands as it has done since inception Jan. 1986. Back issues are available.

To subscribe for one year of 8 issues, send $27 US or $36 outside North America to REC, Att: Dr. M. Ecker, 909 Violet Terrace, Clarks Summit, PA 18411, USA or send $10 ($13 non-US) for 3 sample issues, creditable.

TRSTimes on DISK #8

Issue #8 of TRSTimes on DISK is now available, featuring the programs from the Jul/Aug, Sep/Oct and Nov/Dec 1991 issues:

TRSTimes on DISK is reasonably priced:

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

Send check or money order to:

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

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