TRSTimes

Volume 4. No. 5. - Sep/Oct 1991 - $4.00

LITTLE ORPHAN EIGHTY

From the letters and calls received here at TRSTimes, it is evident that most people use their computers to run application programs that make their lives easier or more fun. To them, DOS is that annoying 'thing' that happens immediately after the machine is turned on, and before the application program can perform whatever magic it does. The operating system is a necessary evil that is always used, but so little understood.

This certainly is the case in the MS-DOS world where users are encouraged to avoid doing anything from the feared C: > prompt by working directly from special menu programs. TRS-80 users, not having the luxury of an abundance of DOS shells (SHELL and DOSTAMER are the only ones I know of), have had to learn the simplest of DOS commands in order to survive.

It has been my experience that, generally, if the Model 4 is someone's first computer, they started with the best of intentions to read the DOS manual, but quickly became overwhelmed and quit. On the other hand, if the Model 4 is not a first computer, but a graduation from a Model I or Model III, the user is much more versed in DOS. Mind you, the old-timers didn't read the manual either, they have just learned from bitter experience to fake it a little better.

The bottom line is that hardly any of us master this complex series of programs called DOS, so to bring it down to earth, next issue will start a series on LS-DOS 6.3.1. Novices will, hopefully, find this somewhat easier reading than the manual, and the old-timers may learn a trick or two. I certainly plan on learning a little something myself in the process of writing this tutorial (nothing like having to do research).

If you use another DOS on your Model 4, do yourself a favor and buy LS-DOS 6.3.1 from Roy Soltoff at MISOSYS, Inc. P.O Box 239, Sterling, VA. 22170. It is now not only the standard for Model 4, but by far the finest DOS ever written for any TRS-80.

While I'm on the MISOSYS bandwagon, let me recommend LITTLE BROTHER version 2 to anyone looking for a Model 4 database. It is a dandy. I was present at a Valley TRS-80 Hackers Group meeting where Roy Beck demonstrated this latest version of LB. I was duly impressed. It manages data just about any way you could ever wish for, fast and efficiently. Even if you already have a database, you just might want to take a look at LB v2 (no, Roy Soltoff is not paying me for these words).

During the demonstration, Roy Beck uncovered a minor bug. When exiting from LB, you are asked to confirm the exit. This confirmation prompt is displayed in reverse video and, if the exit is confirmed, LB exits without turning off the reverse video. This shows up only if another program using graphic characters is run immediately after LB. In our case, Roy Beck ran AllWrite whose front screen now displayed 'garbage-graphics'. The immediate fix, until MISOSYS issues the official patch, is to type CLS from DOS after exiting LB. This little problem in no way detracts from the program, but it should be fixed.

More good news from MISOSYS. Roy Soltoff has just announced that Model III LDOS 5.3 will be upgraded to 5.3.1. The shocker, and the best news of all, is that Roy also says that Model I LDOS 5.1.4 will be upgraded to 5.3.1.

Imagine that... 1991... and we have new DOSes...and you Model I users thought you were abandoned long ago! These upgrades will make the Model I and III LDOS as compatible with LS-DOS 6.3.1 as the hardware allows. Another good reason to get rid of TRSDOS and settle down with LDOS.

On another note, I was a little sad to browse through TRSLINK #41. Here I found a file called REMOVE4/ARC and, as the title sounded familiar, I unarced it and ran the program called REMO4/CMD. It was, as I suspected, a program I had written some years ago and sold to 80 Micro under the name REMOV4/CMD.

Now, I have no problem with the fact that it found its way to TRSLINK; rather, it is flattering that other TRS-80 users feel that the program is of some benefit. What I do have a problem with, however, is that my name and copyright was removed from the screen header and replaced with '(c) 1989 REMOVE4 SOFTWARE'. I put my name on everything that I write, and I am just vain enough to want it to remain there. I am sure that the editor of this particular issue of TRSLINK was not aware of this change, as he did include a short doc file where I got full credit for the program, but someone did go through the trouble to change it. Why bother?

Speaking of TRSLINK, what is going on? I haven't seen a new issue for several months now. Have they ceased publishing? If so, that is unfortunate for our rapidly shrinking TRS-80 world. I hope the people taking over the responsibilities, after Luis Garcia-Barrio left TRSLINK, are just late with a new issue and haven't quit.

Finally, a big thank you to all the contributors of this issue of TRSTimes. You make it all worthwhile.

And now.....welcome to TRSTimes 4.5

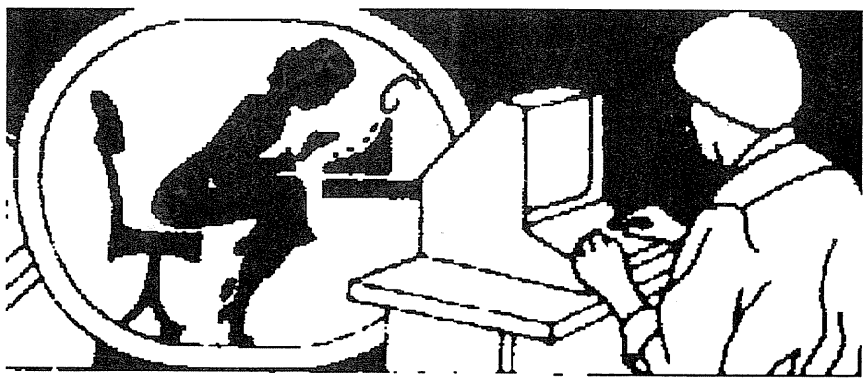
TRSTimes magazine

Volume 4. No. 4. - Sep/Oct 1991

PUBLISHER EDITOR
Lance Wolstrup
CONTRIBUTING EDITORS
Roy T. Beck
Dr. Allen Jacobs
Dr. Michael W. Ecker
TECHNICAL ASSISTANCE
Members of:
San Gabriel Tandy Users Group
Valley TRS-80 Users Group
Valley Hackers' TRS-80 Users Group

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

1991 subscription rates (6 issues):
UNITED STATES & CANADA: $18.00 (U.S. currency)
EUROPE, CENTRAL & SOUTH. AMERICA: $23.00 for surface mail or $29.00 for air mail. (U.S. currency only)
ASIA, AUSTRALIA & NEW ZEALAND: $25.00 for surface mail or $$32.00 for air mail. (U.S. currency only)
Article submissions from 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 is used.



THE MAIL ROOM

MODEL III QUESTIONS

I have a couple of questions for you. First, will the Model III play music and, if so, what will I need for that?

Second, what is the cheapest and best way to hook up a modem, and what will I need for that?

Finally, does System 1.5 work with the Model III and what does it do?

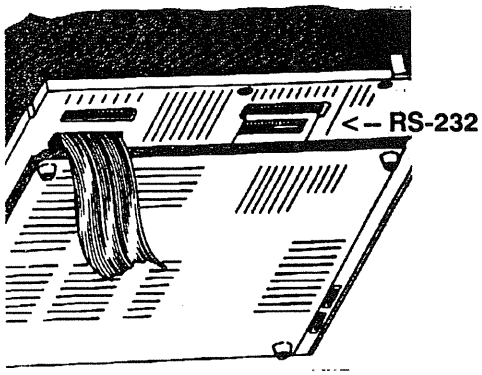
Any assistance you can give will be much appreciated..

Paul Saffell
Cedar Falls, IA

There is a very powerful program called Orchestra 90 which will turn your Model III into a full-fledged music machine with thousands of music files available in the public domain. The only problem with Orchestra 90 is that you need a hardware attachment to make it work - and they are getting scarce. However, you may be able to get both the program and the hardware interface from Computer News 80, P.O. Box 680, Casper, Wyoming 82602-0680.

The other way to make music is to program it yourself. However, this has to be done in assembly language. Years ago there were many articles on how to do this. Maybe TRSTimes will revive this concept in an article or two sometime in the next year.

Hooking up a modem is easy, but first make sure that your macine has an RS-232 interface (this was not standard equipment and some machines came without one). If you have one, it is located on the bottom side of the computer. In other words, you have to turn the Model III upside down to get to it. Assuming that you have now turned the machine upside-down you should see four slots, looking something like this:



You need a modem and a modem cable. You can buy a modem from Radio Shack or any other computer store. I recommend that you get either a 1200 or 2400 baud modem. The cable should be purchased from Radio Shack. Make sure you tell them that it is for a TRS-80. They may have to order it from for you. When you get the cable you will easily see which end plugs in to the modem and which end plugs in to the RS-232. Next plug in the modem to and electical outlet, connect the modem to the wall telephone outlet with a standard phone cable. Finally, connect your telephone to the modem with another standard phone cable.

The hardware is now hooked up. To use the modem you will have to get communications software. There are many good programs available in the public domain. For example, XT3, which is on our Model I/III Public Domain disk #3 is excellent and should do the job for you.

SYSTEM 1.5 is a disk operating system especially for Model III. It is an upgrade to TRSDOS 1.3 and it offers many enhancements. For example, if you have double sided disk drives you can access these with SYSTEM 1.5, thus doubling your disk space. If you are currently using TRSDOS 1.3, you should upgrade to SYSTEM 1.5.

-Ed.

TRS-80 ORIENTED PUBLICATIONS

It would be appreciated if you could advise whether there are any publications, other than yours and Computer news-80, which cater to the TRS-80 machines. It appears to me that there are - or were - others, but I was unable to ascertain from the articles I read, whether they were currently being published.

Gordon Symonds
Burwood, Australia

Other than CN-80 and ourselves, the only TRS-80 publication is The Misosys Quarterly from the wizard himself, Roy Soltoff. The address is: Misosys, P.O. Box 239, Sterling, VA 22170-0239, U.S.A.

All the others, 80 Micro, Basic Computing, Softside, Computer User, etc. have been out of business for years. Too bad, they were all good magazines.

-Ed.

PROGRAM LISTINGS

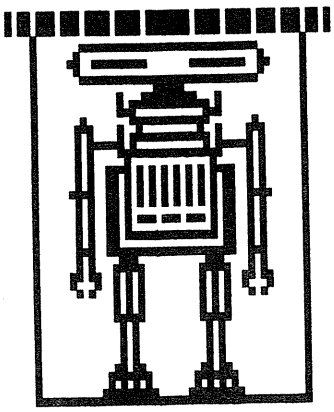
This is just to let you know that my favorite parts of TRSTimes are the program listings. I spend many hours typing in the listings and debugging my mistakes, learning a lot in the process. I enjoy seeing the different programming styles and techniques - and many of the programs have been outstanding. Keep up the good work.

Jerome Jensen
Rochester, MN

GRAPHICS-90 WITHOUT TRSDOS 1.3

Model III

by Gary W. Shanafelt



Some of you may remember a package developed several years ago by Software Affair for Radio Shack. It was called Graphics-90. The full package included utilities to draw circles and lines, move blocks, shrink and enlarge images, and compile frames of pictures together in BASIC to create amination effects. It was easily the fanciest graphics and animation program ever written for the TRS-80 Model 3/4 without a hi-res board. Graphics-90 was actually listed in a RS catalog, but it was never commercially marketed. The story is that by the time it was ready, the Shack decided the market for Model III software was no longer robust enough to justify trying to sell it. So, Software Affair made it available for downloading on Delphi.

Unfortunately, there were several "gotcha's" involved in acquiring the program modules. First, you had to have access to Delphi. Then, the manual was about a hundred pages long and could be printed only on a printer which supported the TRS-80 block graphics characters (the DMP120, etc.) And if that wasn't bad enough, the package only ran under TRSDOS 1.3. It was incompatible with any other operating system. As a result, few people ever got Graphics-90, or even heard of it. This is a shame, because even if you never used the program utilities, the demo that accompanied them was by itself enough to blow your eyes out.

The reason the program ran only under TRSDOS 1.3 is fairly simple. The core of the package was a program called ABASIC -- short for Animation BASIC. It resided in high memory and added several special animation commands to BASIC. When you ran it, it actually patched TRSDOS 1.3 Disk BASIC in memory to recognize the new commands. The patches would have no effect (at least, no positive effect) on any other BASIC because the relevant addresses would all be different.

It seemed to me that Graphics-90 was too good a package to be accessible only under TRSDOS 1.3. I wanted to make it work with my Model III system of choice, LDOS. I finally did. What helped me to do it was an old article in the July 1981 issue of 80 MICRO by Gil Spencer. Spencer noted that you can add commands to BASIC because there are a number of addresses or "latches" in RAM where ROM BASIC jumps to check for instructions provided by Disk BASIC. If Disk BASIC isn't present, they all jump back to BASIC and give you a "Level 3 error." If it is, those addresses are changed to jump to routines in Disk BASIC RAM. And Spencer listed the addresses.

Anyhow, I ran TRSDOS BASIC, then dumped the memory at those addresses; then, I ran ABASIC and rechecked them. Sure enough, two of them were different: the jumps at 4173 and 4194 hex. In other words, ABASIC changed those jumps to point to its new routines in high memory, then jumped back to the original addresses in TRSDOS Disk BASIC. The next step was to determine where those original addresses were. All BASICs have a section of jumps where they change the "latch" addresses to interface with themselves. I took the original values at 4173 and 4194, then found the addresses of those values in TRSDOS BASIC. Finally, I searched ABASIC for the TRSDOS BASIC addresses and found them in four places. In those four places, ABASIC changed the original "latch" instructions in TRSDOS BASIC to interface with itself.

The rest was easier to do than it is to explain. I found where LDOS BASIC interfaced with the jumps at 4173 and 4194 and then zapped those new values into the four places mentioned above in ABASIC. There was one final hitch: I had to add the password "BASIC" where ABASIC loads BASIC into memory. With LDOS, access was denied without the password. My final reward was watching the Graphics-90 demo program run flawlessly on my Model 4 under LDOS, and (equally gratifying) erasing TRSDOS 1.3 from another disk.

It goes without saying that this couldn't have been done without LS-FEDII, the Model 4 disk zapper which not only shows you the bytes in a disk file but actually computes their addresses and Z80 mnemonics as you move from one instruction to the next. To use it, of course, I had to CONV the TRSDOS 1.3 files to a Model 4 disk.

It seemed to me that the same technique could be used to make ABASIC work with other DOSes as well, now that I knew what addresses were involved. I'm not familiar with

Gary Shanafelt can be contacted at:
Dept. of History, McMurry University,
Abilene, TX 79697

any of them except LDOS, so I borrowed some copies and made versions for NEWDOS80 ver. 2, MULTIDOS ver. 1.7, and DOSPLUS ver. 3.4. After being suitably patched, ABASIC ran the demo under all these DOSes. So, if you want to see the demo program, you can now run it under just about any Model III DOS that you want. The requisite patches are at the end of this article.

Most of the other utilities in the Graphics-90 package run without alteration under LDOS. They probably run fine under the other DOSes too, though I haven't checked them out. The EDIT/CMD program required two patches. If you try to create a "key" file with it under LDOS, you'll get a "record number out of range" error. This seems to be because TRSDOS 1.3 and LDOS handle file records differently. It can be avoided simply by deleting the error check after the @POSN subroutine call with no negative side effects (as far as I can tell, though maybe someone can provide a more sophisticated way of doing this). Further, you'll need a second patch to make the directory feature work properly... and that's it.

Some other details for usage with LDOS: If you want to run SHAPES/BAS, change CMDCLEAR 3 in line 20 to CMDCLEAR 4. If you want to run the program demo, CARTOON/BAS, with the LDOS keyboard driver (KI/DVR) activated, you'll run out of memory unless you either enter ABASIC with the (F=0) prompt or change CMDCLEAR 122 in line 100 to CMDCLEAR 121. Incidentally, you can't run EDIT/CMD with KI/DVR active; the driver interferes with editor's use of the <CLEAR> key.

Of course, the patches are only good if you only have Graphics-90 to begin with. And except for the demo, you can't do much with the package without the 100-page manual. I'm hoping to assemble a whole version compatible with LDOS and to reprint the manual on my DeskJet printer, as well as creating LaserJet/DeskJet softfonts and drivers for the program. This could all be distributed at cost by TRSTimes so that anyone who wanted could easily acquire both the program disks and the manual for this outstanding package (and might write some programs that I could then run on MY copy of it!) I'm currently trying to locate the author, Larry Payne, who holds the copyright, to get his permission for this. Expect an update in a future issue. Finally, the patches:

LDOS 5.3 BASIC

PATCH ABASIC/CMD (D00,C8=22 4E:F00,C8=74 41)

PATCH ABASIC/CMD (DOO,EF=22 4E:F00,EF=74 41)

PATCH ABASIC/CMD (D00,CE=43 4E:F00,CE=95 41)

PATCH ABASIC/CMD (D00,D7=43 4E:F00,D7=95 41)

PATCH ABASIC/CMD (D04,47=2E 42 41 53 49 43 03:
F04,47=03 B7 C4 3B 6A EB 22)

NEWDOS80 Ver. 2 BASIC

PATCH ABASIC/CMD (D00,C8=DD 67:F00,C8=74 41)

PATCH ABASIC/CMD (DOO,EF=DD 67:F00,EF=74 41)

PATCH ABASIC/CMD (D00,CE=FE 67:F00,CE=95 41)

PATCH ABASIC/CMD (D00,D7=FE 67:F00,D7=95 41)

PATCH ABASIC/CMD (D04,47=2E 45 5A 38 30 03:
F04,47=03 B7 C4 3B 6A EB)

MULTIDOS Ver. 1.71 BASIC

PATCH ABASIC/CMD (D00,C8=42 4E:F00,C8=74 41)

PATCH ABASIC/CMD (DOO,EF=42 4E:F00,EF=74 41)

PATCH ABASIC/CMD (D00,CE=63 4E:F00,CE=95 41)

PATCH ABASIC/CMD (D00,D7=63 4E:F00,D7=95 41)

PATCH ABASIC/CMD (D04,47=2E 4B 54 41 32 03:
F04,47=03 B7 C4 3B 6A EB)

DOSPLUS Ver. 3.4 BASIC

PATCH ABASIC/CMD (D00,C8=07 63:F00,C8=74 41)

PATCH ABASIC/CMD (DOO,EF=07 63:F00,EF=74 41)

PATCH ABASIC/CMD (D00,CE=28 63:F00,CE=95 41)

PATCH ABASIC/CMD (D00,D7=28 63:F00,D7=95 41)

PATCH FOR EDIT/CMD with LDOS 5.3

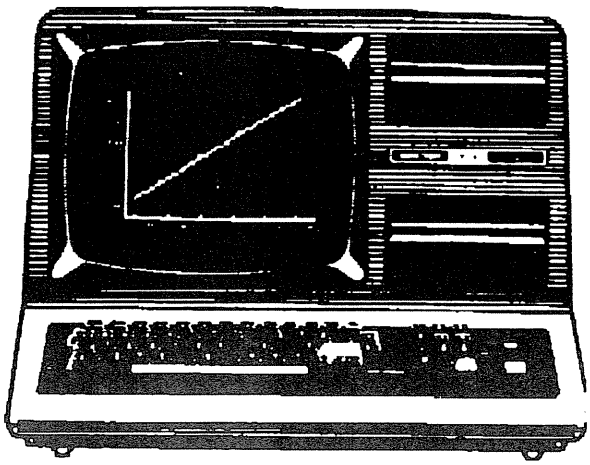
To make EDIT/CMD fully compatible with LDOS, do the following:
PATCH EDIT/CMD (D13,F3=00 00:F13,F3=20 18)

PATCH EDIT/CMD (D09,8A=06 00 00:
F09,8A=32 71 42)

HOW'S YOUR DCT

Model 4 - TRSDOS/LS-DOS 6.x

By Lance Wolstrup



Roy Beck's article from last issue (EXPLORING CONFIG/SYS - TRSTimes 4.4) intrigued me to no end as I was having a minor problem with my hard drive. This, along with the fact that M.C. Matthews from England had just submitted an article about patching the Newdos/80 version of EDTASM to work with Model 4 TRSDOS/LS-DOS 6.x., prompted me to sit down and write a program that will display the DCT (Drive Code Table) of each drive on the screen.

The program, called SHOWDCT/CMD, was written entirely on my Model 4 with the patched version of Newdos/80 EDTASM, which I have renamed NEDTASM/CMD to keep from confusing it with the Radio Shack Series I Editor/Assembler, also named EDTASM/CMD.

So, get busy and patch the Newdos EDTASM (the patches are listed elsewhere in this issue) and then type in SHOWDCT/SRC and assemble it to SHOWDCT/CMD.

SHOWDCT/SRC

```
00100 ;SHOWDCT/SRC - for TRSDOS/LS-DOS 6.x
00110 ;(c) 1991 by Lance Wolstrup
00120 ;all rights reserved
00130 ;
00140           ORG     7000H
00150 ;
```

Turn the cursor off, erase the screen and display the program name, copyright , etc.

```
00160 START     CALL    CUROFF      ;turn cursor off
00170           CALL    CLS         ;clear screen
00180           LD      HL,0        ;cursor at 0,0
00190           CALL    LOCATE
00200           LD      HL,MSG1     ;point to msg
00210           CALL    DSPLY       ;display msg
00220           LD      HL,0010H    ;cursor at 0,16
00230           CALL    LOCATE
00240           LD      HL,MSG2     ;point to msg
00250           CALL    DSPLY       ;display msg
00260           LD      HL,0111H    ;cursor at 1,17
00270           CALL    LOCATE
00280           LD      HL,MSG3     ;point to msg
00290           CALL    DSPLY       ;display msg
00300           LD      HL,0215H    ;cursor at 2,21
00310           CALL    LOCATE
00320           LD      HL,MSG4     ;point to msg
00330           CALL    DSPLY       ;display msg
00340           LD      HL,0300H    ;cursor at 3,0
00350           CALL    LOCATE
```

Draw line accross screen line 3

```
00360           LD      B,80        ;loop counter
00370 LINE03    LD      A,140       ;chr$(140)
00380           CALL    DSP         ;draw line
00390           DJNZ    LINE03
```

This is the main body of the program.

We need to check eight drives (0-7), so begin with drive :0 and display at line 9, column 20.

```
00400           LD      BC,0800H    ;B=counter,
                                    C=drive #
00410           LD      HL,0914H    ;cursor at 9,20
00420 DLOOP     CALL    LOCATE
00430           PUSH    HL          ;save cursor
00440           LD      HL,DRVMSG   ;point to msg
00450           CALL    DSPLY
00460           LD      A,C         ;drive # to A
00470           ADD     A,30H       ;make it ascii
00480           CALL    DSP         ;display it
```

Space over twice to seperate 'drive #' from table.

```
00490           LD      A,32        ;chr$(32)
00500           CALL    DSP         ;display space
00510           LD      A,32
00520           CALL    DSP         ;another space
```

Go check if drive is defined in DCT. Jump to DCGOOD if drive is defined.

```
00530           CALL    DCSTAT      ;is drive defined
00540           JR      Z,DCGOOD    ;jump if yes
```

Drive is not defined in DCT. Display 'not available' message and jump to NXTDRV.

```
00550           LD      HL,NODRV    ;point to msg
00560           CALL    DSPLY       ;display msg
00570           JR      NXTDRV
```

Drive is defined in DCT. By CALLing GTDCT we find the address of the DCT - it is stored in IY. Then display the 10 bytes starting at IY

```
00580 DCGOOD PUSH   BC              ;save counter
                                     and drive #
00590           CALL    GTDCT       ;find dct for drv
00600           LD      B,10        ;get 10 bytes
00610 DLOOP1 LD     C,(IY)          ;get chr
00620           CALL    HEX8        ;chr to hex ascii
00630           LD      HL,HEXBUF   ;point to #
00640           LD      A,(HL)      ;display msb
00650           CALL    DSP
00660           INC     HL
00670           LD      A,(HL)      ;display lsb
00680           CALL    DSP
00690           LD      A,32        ;chr$(32)
00700           CALL    DSP         ;display space
00710           INC     IY          ;next byte
00720           DJNZ    DLOOP1
```

Restore loop counter and drive number. Point to the next screen line, increment the drive number, and loop back to display the DCT for the remaining drives.

```
00730           POP     BC          ;restore counter
                                     and drive #
00740 NXTDRV POP    HL              ;restore cursor
00750           INC     H           ;next screen line
00760           INC     C           ;next drive
00770           DJNZ    DLOOP
```

The DCT for all available drives have been displayed. Position cursor at line 20, column 0.

```
00780           LD      HL,1400H    ;cursor at 20,0
00790           CALL    LOCATE
```

Turn cursor on and exit to DOS.

```
00800 EXIT      CALL    CURON       ;turn cursor on
00810           RET                 ;return to dos
00820 ;
```

Subroutine to erase screen.

```
00830 CLS       LD      A,28        ;home cursor
00840           CALL    DSP
00850           LD      A,31        ;erase to eof
00860           JR      DSP
00870 ;
```

Subroutine to turn cursor off.

```
00880 CUROFF LD     A,15            ;cursor OFF chr
00890           JR      DSP
00900 ;
```

Subroutine to turn cursor on.

```
00910 CURON     LD      A,14        ;cursor ON chr
00920           JR      DSP
00930 ;
```

Subroutine to determine if drive is defined in DCT. Enter with C = drive number

```
00940 DCSTAT LD     A,40            ;@DCSTAT svc
00950           RST     40          ;do it
00960           RET                 ;return to caller
00970 ;
```

Subroutine to display a character on the screen. If character is a control character, the appropriate action will be taken. Enter with A = character

```
00980 DSP       PUSH    BC          ;save
00990           PUSH    DE          ;BC and DE
01000           LD      C,A         ;copy chr to C
01010           LD      A,2         ;@DSP svc
01020           RST     40          ;display chr
01030           POP     DE          ;restore
01040           POP     BC          ;DE and BC
01050           RET                 ;return to caller
01060 ;
```

Subroutine to display a message line at current cursor position. Enter with HL pointing to message.

```
01070 DSPLY     PUSH    DE          ;save DE
01080           LD      A,10        ;@DSPLY svc
01090           RST     40          ;do it
01100           POP     DE          ;restore DE
01110           RET                 ;return to caller
01120 ;
```

Subroutine to convert a 1-byte number to hex ascii. Enter with C = number to convert. Exits with converted number in buffer.

```
01130 HEX8      LD      HL,HEXBUF   ;point to buffer
01140           LD      A,98        ;@HEX8 svc
01150           RST     40          ;do it
01160           RET                 ;return to caller
01170 ;
```

Subroutine to find the start of the DCT. Enter with C = drive number. Exits with IY pointing to start of DCT.

```
01180 GTDCT     LD      A,81        ;@GTDCT svc
```

```
01190           RST     40          ;do it
01200           RET                 ;return to caller
01210 ;
```

Subroutine to position the cursor.
Enter with H = vertical, L = horizontal

```
01220 LOCATE    PUSH    BC          ;save
01230           PUSH    DE          ;BC and DE
01240           LD      A,15        ;@VDCTL svc
01250           LD      B,3         ;function 3
01260           RST     40          ;do it
01270           POP     DE          ;restore
01280           POP     BC          ;DE and BC
01290           RET                 ;return to caller
01300 ;
```

Screen messages. Note: The text in MSG2 and MSG4 have been broken into two lines to fit in 2-column format of the magazine. You should type each message in as one line.

```
01310 MSG1      DEFM    'SHOWDCT'
01320           DEFB    3
01330 MSG2      DEFM    '(c) 1991 by Lance Wolstrup
                         - all rights reserved'
01340           DEFB    3
01350 MSG3      DEFM    'Inspired by Roy Beck'
01360           DEFB    39
01370           DEFM    's article in TRSTimes 4.4'
01380           DEFB    3
01390 MSG4      DEFM    'Written with Newdos EDTASM
                         for Model 4'
01400           DEFB    3
01410 DRVMSG    DEFM    'Drive :'
01420           DEFB    3
01430 NODRV     DEFM    '*** not available ***'
01440           DEFB    3
```

Buffers

```
01450 HEXBUF    DEFS    2
01460           END START
```

End of listing.

HOLLAND DAZE

Fractals for your Model 4

Program by Johan Volgers. Text by Pieter Plomp and Dr. Allen Jacobs

We all know that if computers can do anything better than a human, they can do repetitive calculations faster and more accurately. That is why Babbage, the father of computing, called his original mechanical computer a "mathematical engine". That ability, alone, is their greatest appeal. As individual users, however, we usually do not have any "practical" need for intense calculation, except in the area of graphics. The only catch to that is that we usually do not NEED to do graphics. Largely, we WANT to do graphics! That's why we bought graphics boards. There is no better "test" to which we can put our graphics boards than to calculate and visualize "the most complex figure in mathematics". That is the Mandelbrot Fractal.

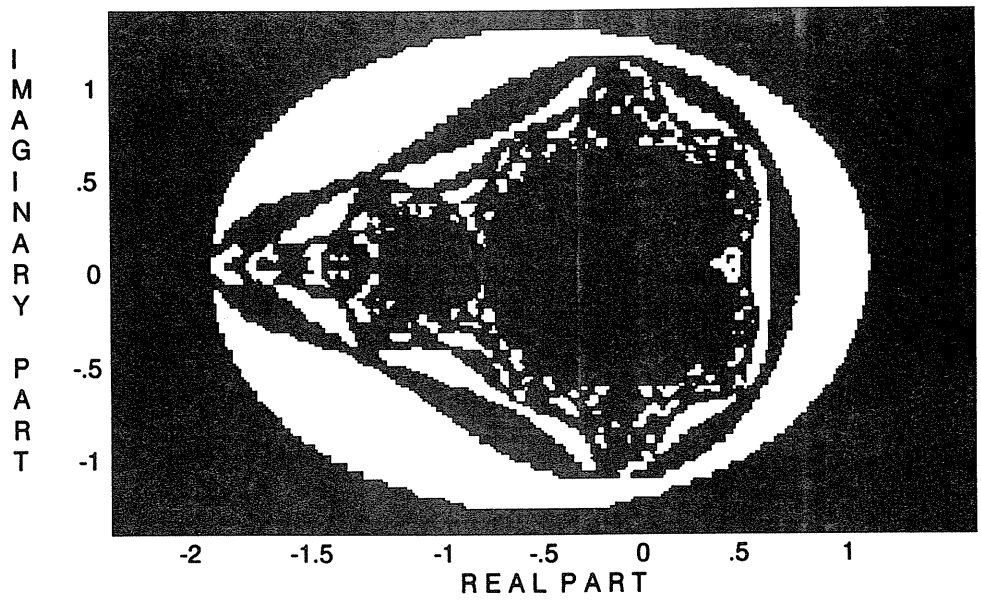
We have all seen views of this fractal, each being a plot of part of the Mandelbrot Set. They are often published in books, computer magazines, and as demonstrations on expensive MS-DOS PC's and graphical work stations. These views comprise an infinite number of graphical representations of a single calculation, recursively performed numerous times and subsequently displayed. The shapes thus generated are almost always beautiful and never repeat. The sheer beauty of the Set demands our attention.

I know a number of TRS-80 users who have defected, not only the TRS-80 world, but even slower PC's, by purchasing fast MS-DOS machines. They have done so, just to run one or another Mandelbrot fractal generating program faster, more colorfully, and in greater detail than they could on their last machine. They all say that they need the speed their fast machine provides, for other purposes. However, if you ask them specifically what these other purposes are, you get one of two answers. The most common answer is: "I need the bigger machine for word processing". The second is: "I need to be able to run Windows... so that I can do better word processing". We all know that we can do most "practical" word processing tasks on our Model 4's as well as we can on a machine with which we are not familiar. So, if you question most TRS-80 defectors closely, they will admit that they really just play Windows solitaire, do a little fancy word processing to prove that it works, try out Lotus for their home checking account,.....and then spend the rest of their time playing with graphics.

What graphics? You know! They like to let their machine run the MS-DOS shareware program called FRACTINT, for 29+ hours at a stretch while it calculates yet another view of the Mandelbrot Fractal. They then add that view to the subdirectory containing their Mandelbrot Fractal views and hope that they can get a still faster machine to plot the same fractal view in half the time or be able to "go deeper into" the fractal, by another level of magnitude in the same amount of time. Their collection of fractal files, all in their own subdirectory, treasured like high-tech shrunken heads, is second in size only to their program files. Sometimes it is even larger than that. We are talking about a primitive graphical addiction that is seemingly without end. While this quest is a somewhat mindless pursuit of form and color, admittedly, it is fun!

We TRSTimers need not be left out of the entire affair because John Volgers in Holland has provided us with a BasicG program that generates fractals on the Model 4 hi-rez screen. It does this by calculating the following equation for different values. The coordinates and number of iterations it requests in the plot mode provide the program with the location, magnification, and level of detail desired to plot a part of the mighty Mandelbrot Fractal. While you can't view them on the high-rez screen in color, you can print them that way if you have the right color printer. The program was written for a CGP-220 Printer but I don't know its brand name. I guess it doesn't matter unless you have one or want to buy one. In either case, your options become clear.

The program works in this manner. It displays the set of numbers that do NOT increase to infinity after a given number of recursive iterations of the function: Z^2 + C. It does this graphically, in three dimensions. The X dimension is the value of C, a real number. The Y direction is the value of an imaginary number Z^2, which starts at Z = (0) and is increased by the number of iterations required to increase the value of the expression to be in excess of 2. Remember, an imaginary number includes the value i. If you recall your mathematics, i is the square root of -1. In this program, the limit of iterations has been limited to the range of from 4 to 400. Each iteration is represented by alternating light and dark points which form corresponding characteristically band-like areas. The shade of a given point is the third axis. The number of shade shifts is therefore equal to the number of iterations selected, or less, depending on the calculated values in a given region. With multiple colors available, subsequent iterations are indicated by repeatedly rotating through the number of available colors (ie.: modulo, the number of available colors). The best way to understand this is to see a low resolution plot of the fractal. This is accomplished by specifying a large range of points in both the X and Y dimensions. Maybe Dr. Michael Ecker can provide a more understandable explanation of how the program works.



The included illustration gives a good look at the fractal and its accompanying coordinates.

When you run the program, the opening menu is pretty much self explanatory. What you are doing when you plot a given view is to select a region of the fractal that you wish to view. All views fill the entire window. Therefore, the closer together the values you select are, the greater will be the magnification, since the smaller area now fills the window. Just select a region, start the plot, and grow a beard waiting for it to finish. Feel satisfied that your computer is calculating its little heart out and you aren't. Save your favorites by returning to the Main Menu.

Personally, I find the theory and programming of fractals to be of more enduring interest than the plotting of them. However, being able to see a plot extending as far to infinity as any of us are likely to experience, (and in such a beautiful corner of our conceptual universe), is intriguing to say the least.

On a practical basis, this enterprise would otherwise be too arduous an undertaking for calculation by hand. Such a grand project, utilizing error fraught human calculation, without a computer, would never be contemplated by rational people. It's like trying to calculate the absolute value of pi.

My only contribution to this program is to smooth out Pieter Plomp's English translation of the program's prompts and comments. I did not otherwise alter the program. It's in BasicG so we can all see how it works. We should all feel free to enjoy any and all aspects of John Volgers' project. There are certainly enough different aspects to go around.

Here is the program.....

MDLBROT/BAS

```
100 ' ***** MDLBROT/BAS, Version 3, by Johan Volgers, June 1989.
102 '       Version for TRSDOS 6.2 / BasicG / Model 4.
104 '
106 OPTION BASE 0:DIM C1%(51)
108 SP$=STRING$(80,32):
U1$="+#.############":
U2$="+###":XR%=434:YR%=103
110 FOR TR% = 0 TO 51:
READ DA%:
C1%(TR%)=DA%:
NEXT TR%
112 PV%=-1:CV%=-1:LV%=-1:
ON ERROR GOTO 772
114 '
116 '     ----- MANDELBROT PROGRAM MENU -----
118 '
120 SCREEN 1:CLS:PRINT CHR$(15);:GOSUB 748
122 PRINT @  0," FRACTALS for Tandy Model 4 /
 Mandelbrot Formula / Johan Volgers, May 1989 ";
124 PRINT @ 177,"> Menu for Fractals .....";
126 PRINT @ 257,"_____";
128 PRINT @ 417,"- (L) - Load a Fractal File (using
FGLOAD/CMD).";
130 PRINT @ 497,"- (W) - Write a Fractal File (using
FGSAVE/CMD).";
132 PRINT @ 577,"- (P) - Plot a Fractal.";
134 PRINT @ 657,"- (C) - Enter Cursor Mode.";
136 PRINT @ 737,"- (Clear - P) - Toggle Plot to Cursor
Flag.";
138 PRINT @ 817,"- (Clear - C) - Toggle Cursor to Plot
Flag.";
140 PRINT @ 897,"- (X) - Clear HRG-screen (using
FGCLS/CMD).";
```

```
142 PRINT @ 977,"- (V) - View HRG-screen.";
144 PRINT @1057,"- (8) - Dump Screen to CGP-220
printer.";
146 PRINT @1137,"- (@) - End  Program.";
148 IF PV%=-1 THEN PRINT @1280,"> Plot to Cursor
Flag is OFF.";
150 IF PV%= 1 THEN PRINT @1280,"> Plot to Cursor
Flag is ON.";
152 IF CV%=-1 THEN PRINT @1360,"> Cursor to Plot
Flag is OFF.";
154 IF CV%= 1 THEN PRINT @1360,"> Cursor to Plot
Flag is ON.";
156 PRINT @1520,"* Cursor Mode can ONLY be
entered AFTER loading a MDLBROT$/$$$.!!";
158 PRINT @1600,"* When the Plot to Cursor Flag is
ON, the";
160 PRINT @1700,"given parameters will be copied to
the Cursor Mode.";
162 PRINT @1760,"* When the Cursor to Plot Flag is
ON, while in the";
164 PRINT @1860," Plot Mode, only the Iteration Count
will be requested.";
166 '
168 MN$="":MN$=INKEY$
170 IF MN$="L" OR MN$="l" THEN GOTO 552
172 IF MN$="W" OR MN$="w" THEN GOTO 622
174 IF MN$="P" OR MN$="p" THEN GOTO 198
176 IF MN$="C" OR MN$="c" THEN GOTO 364
178 IF MN$=CHR$(195) THEN GOTO 510
180 IF MN$=CHR$(208) THEN GOTO 518
182 IF MN$="X" OR MN$="x" THEN GOTO 526
184 IF MN$="V" OR MN$="v" THEN GOTO 534
186 IF MN$="8" THEN GOTO 698
188 IF MN$="@" THEN GOTO 756
190 GOTO 168
192 '
194 '    ----- ENTER COMPLEX PARAMETERS -----
196 '
198 SCREEN 1:CLS:PRINT CHR$(15);:GOSUB 748
200 IF CV%= 1 THEN GOTO 214
202 PRINT @1200,"* The Initial X Value must be in the
Range: -2.00 < = X < = +0.70";
204 PRINT @1280,"* The Final X Value must be in the
Range: -2.00 < = X < = +0.70";
206 PRINT @1360,"* The Initial X Value must be < the
Final X Value.";
208 PRINT @1520,"* The Initial Y Value must be in the
Range: -1.35 < = Y < = +1.35";
210 PRINT @1600,"* The Final Y Value must be in the
Range: -1.35 < = Y < = +1.35";
212 PRINT @1680,"* The Initial Y Value must be < the
Final Y Value.";
214 PRINT @1840,"* The Iteration Counter must be in
the Range: 4 < = Counter < = 400.";:PRINT CHR$(14);
216 '
218 IF CV%= 1 THEN GOTO 224
220 PRINT @  0,"";:
INPUT "> The Initial (Real Complex) X.....";XB#
222 PRINT @  80,"";:
INPUT "> The Final (Real Complex) X.....";XE#
224 GOSUB 328
226 IF CV%= 1 THEN GOTO 232
228 PRINT @ 160,"";:
INPUT "> The Initial (Imaginary Complex) Y...";YB#
230 PRINT @ 240,"";:
INPUT "> The Final (Imaginary Complex) Y...";YE#
232 GOSUB 340
234 PRINT @ 320,"";:
INPUT "> The Number of Iterations.....";TN%
236 GOSUB 352
238 SCREEN 1:CLS:PRINT CHR$(15);
240 PRINT @1760,"* Hit (Clear - M) for Menu.....";
242 FOR TD% = 0 TO 1999:NEXT TD%
244 '
246 '    %%%%% PRINT PARAMETERS
248 '
250 SCREEN 0
252 LINE(230,  0)-(639,  0),1:
LINE(233,  2)-(636,  2),1
254 LINE(233,204)-(636,204),1:
LINE(230,206)-(639,206),1
256 LINE(230,  0)-(230,206),1:
LINE(233,  2)-(233,204),1
258 LINE(636,  2)-(636,204),1:
LINE(639,  0)-(639,206),1
260 PAINT(XR%,YR%),1,1
262 GLOCATE( 0,  0),0:
PRINT #-3,"**** Plot Mode ****";
264 GLOCATE( 0, 10),0:
PRINT #-3,"Initial X = ";:
PRINT #-3,USING U1$;XB#;
266 GLOCATE( 0, 20),0:
PRINT #-3,"Final X = ";:
PRINT #-3,USING U1$;XE#;
268 GLOCATE( 0, 30),0:
PRINT #-3,"XC Step = ";:
PRINT #-3,USING U1$;XS#;
270 GLOCATE( 0, 50),0:
PRINT #-3,"Initial Y = ";:
PRINT #-3,USING U1$;YB#;
272 GLOCATE( 0, 60),0:
PRINT #-3,"Final Y = ";:
PRINT #-3,USING U1$;YE#;
274 GLOCATE( 0, 70),0:
PRINT #-3,"YC Step = ";:
PRINT #-3,USING U1$;YS#;
276 GLOCATE( 0, 90),0:
PRINT #-3,"Iterations = ";:
PRINT #-3,USING U2$;TN%;
278 '
280 '    %%%%% PLOT PROGRAM
282 '
284 FOR XT% = 0 TO 400 STEP 2:
PX%=XR%+(XT%-200)
286 LINE(PX%-2,207)-(PX%-2,211),0:
LINE(PX%,207)-(PX%,211),1
```

```
288 FOR YT% = 0 TO 200 STEP 1:
PY% = YR% + (YT%-100)
290 LINE(220,PY%-1)-(229,PY%-1),0:
LINE(220,PY%)-(229,PY%),1
292 CR# = XB# + XT%*XS#:
ZR# = CR#:
CI# = YB# + YT%*YS#:ZI# = CI#
294 FOR TI% = 0 TO TN% STEP 1
296 RK# = ZR# ^ 2:
IK# = ZI# ^ 2:
SM# = RK# + IK#
298 RN# = RK#-IK# + CR#:
IN# = 2*ZR#*ZI# + CI#:
ZR# = RN#:ZI# = IN#
300 IF SM# = > 4 THEN GOSUB 314:GOTO 304
302 NEXT TI%
304 IN$ = "":IN$ = INKEY$:
IF IN$ = CHR$(205) THEN GOTO 310
306 NEXT YT%:
LINE(220,PY%)-(229,PY%),0
308 NEXT XT%:
LINE(PX%,207)-(PX%,211),0
310 GOSUB 748:GOTO 120
312 '
314 KL% = TI% MOD 2
316 IF KL% = 0 THEN PSET(PX% + 0,PY%),0:
PSET(PX% + 1,PY%),0:GOTO 320
318 IF KL% = 1 THEN PSET(PX% + 0,PY%),0:
PSET(PX% + 1,PY%),1
320 RETURN
322 '
324 '  %%%%% CHECK GIVEN  COMPLEX X-VALUES
326 '
328 IF XB# < -2.1 OR XB# > .71 OR XE# < -2.1 OR
XE# > .71 OR XB# = > XE# THEN GOTO 332
330 XS# = (XE#-XB#)/400:RETURN
332 GOSUB 748:PRINT @  0,SP$;:
PRINT @  80,SP$;:GOTO 220
334 '
336 '    %%%%% CHECK GIVEN COMPLEX Y-VALUES
338 '
340 IF YB# < -1.351 OR YB# > 1.351 OR YE# < -1.351
OR YE# > 1.351 OR YB# = > YE#    THEN GOTO 344
342 YS# = (YE#-YB#)/200:RETURN
344 GOSUB 748:PRINT @ 160,SP$;:
PRINT @ 240,SP$;:GOTO 228
346 '
348 '   %%%%% CHECK GIVEN ITERATION COUNTER
350 '
352 IF TN% < 4 OR TN% > 400 THEN GOTO 356
354 RETURN
356 GOSUB 748:PRINT @ 320,SP$;:GOTO 234
358 '
360 '    ----- MICROSCOPE -----
362 '
364 IF TN% = > 4 THEN LV% = 1
366 IF LV% = -1 THEN PRINT @1920,"";
368 SCREEN 1:CLS:PRINT CHR$(15);:GOSUB 748
370 PRINT @ 176,"> Microscope Cursor Help .....";
372 PRINT @ 256,"_____";
374 PRINT @ 416,"- (Left  Arrow) - 1 position to Left.";
376 PRINT @ 496,"- (Right Arrow) - 1 position to Right.";
378 PRINT @ 576,"- (Up    Arrow) - 1 position Up.";
380 PRINT @ 656,"- (Down  Arrow) - 1 position Down.";
382 PRINT @ 736,"- (Shift Left  Arrow) - 20 positions
Left.";
384 PRINT @ 816,"- (Shift Right Arrow) - 20 positions
Right.";
386 PRINT @ 896,"- (Shift Up    Arrow) - 20 positions
Up.";
388 PRINT @ 976,"- (Shift Down  Arrow) - 20 positions
Down.";
390 PRINT @1056,"- (Clear U) - Update the previous
Complex Values.";
392 PRINT @1136,"- (Clear C) - Copy the previous
Values.";
394 PRINT @1216,"- (Clear M) - Menu.";
396 GOSUB 734
398 '
400 IF PV% = -1 THEN XP% = XR% + 96-20:
YP% = YR%-10
402 IF PV% =  1 THEN XP% = XR%-200:YP% = YR%-100
404 SCREEN 0:GOSUB 448
406 CP$ = "":CP$ = INKEY$
408 IF CP$ = CHR$( 8) THEN XP% = XP%-1
410 IF CP$ = CHR$( 9) THEN XP% = XP% + 1
412 IF CP$ = CHR$(10) THEN YP% = YP% + 1
414 IF CP$ = CHR$(11) THEN YP% = YP%-1
416 IF CP$ = CHR$(24) THEN XP% = XP%-40
418 IF CP$ = CHR$(25) THEN XP% = XP% + 40
420 IF CP$ = CHR$(26) THEN YP% = YP% + 20
422 IF CP$ = CHR$(27) THEN YP% = YP%-20
424 IF XP% < 234 THEN XP% = 234 ELSE
IF XP% > 595 THEN XP% = 595
426 IF YP% <  3 THEN YP% =  3 ELSE
IF YP% > 184 THEN YP% = 184
428 GET(XP%,YP%)-(XP% + 39,YP% + 19),C1%
430 PUT(XP%,YP%),C1%,PRESET:
FOR TD% = 0 TO 49:
NEXT TD%
432 PUT(XP%,YP%),C1%,PSET:
FOR TD% = 0 TO 49:NEXT TD%
434 IF CP$ = CHR$(213) THEN GOSUB 448
436 IF CP$ = CHR$(195) THEN GOSUB 494
438 IF CP$ = CHR$(205) THEN GOTO 502
440 GOTO 406
442 '
444 '    %%%%% UPDATE PREVIOUS VALUES
446 '
448 IF PV% = -1 THEN GOSUB 470
450 IF PV% =  1 THEN GOSUB 478
452 GLOCATE( 0,110),0:PRINT #-3,"**** Cursor Mode
****";
454 GLOCATE( 0,120),0:PRINT #-3,"Initial X = ";:
PRINT #-3,USING U1$;X8#;
456 GLOCATE( 0,130),0:PRINT #-3,"Final X = ";:
```

```
PRINT #-3,USING U1$;X9#;
458 GLOCATE( 0,140),0:
PRINT #-3,"XC Step = ";:
PRINT #-3,USING U1$;UX#;
460 GLOCATE( 0,160),0:
PRINT #-3,"Initial Y = ";:
PRINT #-3,USING U1$;Y8#;
462 GLOCATE( 0,170),0:
PRINT #-3,"Final Y = ";:
PRINT #-3,USING U1$;Y9#;
464 GLOCATE( 0,180),0:
PRINT #-3,"YC Step = ";:
PRINT #-3,USING U1$;UY#;
466 GOSUB 748:RETURN
468 '
470 UX# = 2.7/400:
X8# = -.14175 + (((XP%-510) + 1)*UX#):
X9# = X8# + 40*UX#
472 UY# = 2.7/200:Y8# = -.1485 + (((YP%- 93)
+1)*UY#):Y9# = Y8# + 20*UY#
474 RETURN
476 '
478 IF XB# > XE# THEN UX# = ABS(XB#-XE#) ELSE
UX# = ABS(XE#-XB#)
480 UX# = UX#/400:X8# = XB# + ((XP%-234)*UX#):
X9# = X8# + 40*UX#
482 IF YB# > YE# THEN UY# = ABS(YB#-YE#) ELSE
UY# = ABS(YE#-YB#)
484 UY# = UY#/200:Y8# = YB# + ((YP%- 3)*UY#):
Y9# = Y8# + 20*UY#
486 RETURN
488 '
490 '    %%%%% CURSOR VALUES TO  PLOT VALUES
492 '
494 XB# = X8#:XE# = X9#:YB# = Y8#:YE# = Y9#:
GOSUB 748:CV% = 1:RETURN
496 '
498 '    %%%%% TO MENU WHEN CLEAR - M
500 '
502 GOSUB 748:GOTO 120
504 '
506 '    ----- TOGGLE CURSOR > PLOT FLAG -----
508 '
510 CV% = -CV%:GOSUB 748:GOTO 120
512 '
514 '    ----- TOGGLE PLOT > CURSOR FLAG -----
516 '
518 PV% = -PV%:GOSUB 748:GOTO 120
520 '
522 '    ----- CLEAR HRG-SCREEN -----
524 '
526 SCREEN 0:CLR:GOSUB 748:GOTO 120
528 '
530 '    ----- MONITOR HRG-CARD -----
532 '
534 SCREEN 1:CLS:PRINT CHR$(15);:GOSUB 748
536 PRINT @1760,"* Hit (Clear - M)enu when
desired.....";
538 FOR TD% = 0 TO 1999:
NEXT TD%:
SCREEN 0
540 IN$ = "":IN$ = INKEY$
542 IF IN$ = CHR$(205) THEN GOSUB 748:GOTO 120
544 GOTO 540
546 '
548 '    ----- LOAD FILE -----
550 '
552 SCREEN 1:CLS:PRINT CHR$(15);:GOSUB 748
554 PRINT @1760,"> Directory of Drive (0 - 4) ?";
556 DN$ = "":DN$ = INKEY$
558 IF DN$ = "0" OR DN$ = "1" OR DN$ = "2" OR
DN$ = "3" OR DN$ = "4" THEN GOTO 562
560 GOTO 556
562 LD$ = "CAT :" + DN$:CLS:PRINT CHR$(14);:
SYSTEM LD$:GOSUB 748
564 PRINT @1680,"";:
LINE INPUT "> Filename as ...../...:D ? ";NM$
566 PRINT @1760,"* Loading File .....";NM$;
568 GOSUB 764:LD$ = "GLOAD " + NM$:
SYSTEM LD$:GOSUB 580
570 IF MID$(NM$,1,7) = "MDLBROT" THEN LV% = 1
572 GOSUB 748:GOTO 120
574 '
576 '    %%%%% READ CX,CY PARAMETERS
578 '
580 OUT 128,0:
OUT 129,236:
OUT 131,18:
L1 = INP(130)
582 OUT 128,0:
OUT 129,237:
OUT 131,18:
L2 = INP(130)
584 OUT 128,0:
OUT 129,238:
OUT 131,18:
L3 = INP(130)
586 OUT 128,0:
OUT 129,239:
OUT 131,18:
L4 = INP(130)
588 R1$ = "":R2$ = "":R3$ = "":R4$ = ""
590 FOR DT = 1 TO L1
592 OUT 128,0 + DT:
OUT 129,236:
OUT 131,18:
R1$ = R1$ + CHR$(INP(130))
594 NEXT DT:
XB# = VAL(R1$)
596 FOR DT = 1 TO L2
598 OUT 128,0 + DT:
OUT 129,237:
OUT 131,18:
R2$ = R2$ + CHR$(INP(130))
600 NEXT DT:XE# = VAL(R2$)
602 FOR DT = 1 TO L3
```

```
664 OUT 128,0:OUT 129,239:
OUT 131,66:OUT 130,L4
666 FOR DT = 1 TO L1
668 OUT 128,0+DT:OUT 129,236:OUT 131,66:
OUT 130,ASC(MID$(W1$,DT,1))
670 NEXT DT
672 FOR DT = 1 TO L2
674 OUT 128,0+DT:OUT 129,237:OUT 131,66:
OUT 130,ASC(MID$(W2$,DT,1))
676 NEXT DT
678 FOR DT = 1 TO L3
680 OUT 128,0+DT:OUT 129,238:OUT 131,66:
OUT 130,ASC(MID$(W3$,DT,1))
682 NEXT DT
684 FOR DT = 1 TO L4
686 OUT 128,0+DT:OUT 129,239:OUT 131,66:
OUT 130,ASC(MID$(W4$,DT,1))
688 NEXT DT
690 RETURN
692 '
694 '    ----- DUMP SCREEN TO CGP-220
(GPRT8/CMD) -----
696 '
698 SCREEN 1:CLS:PRINT CHR$(15);:GOSUB 748
700 PRINT @ 420,"* Choose from the following
colors.....";
702 PRINT @ 500,"_____
_____";
704 PRINT @ 660,"- 48 for Black.";
706 PRINT @ 740,"- 49 for Red.";
708 PRINT @ 820,"- 50 for Green.";
710 PRINT @ 900,"- 51 for Yellow.";
712 PRINT @ 980,"- 52 for Violet.";
714 PRINT @1060,"- 53 for Magenta.";
716 PRINT @1140,"- 54 for Blue.";
718 PRINT CHR$(14);:PRINT @1218,"";:INPUT "> ";KL
720 IF KL<48 OR KL>54 THEN KL=-1:GOSUB 748:
PRINT @1200,SP$;:GOTO 718
722 LPRINT CHR$(30);CHR$(13);CHR$(27);CHR$(84);
CHR$(KL);CHR$(18);
724 GOSUB 764:SYSTEM "GPRT8":FOR LF = 1 TO 7:
LPRINT CHR$(13);:NEXT LF
726 GOSUB 748:GOTO 120
728 '
730 '    ----- NEXT ? -----
732 '
734 GOSUB 748:PRINT CHR$(15);:
PRINT @1874,"> (N)ext ?";
736 IN$="":IN$=INKEY$
738 IF IN$="N" OR IN$="n" THEN RETURN
740 GOTO 736
742 '
744 '    ----- BLEEP -----
746 '
748 SOUND 7,0:RETURN
750 '
752 '    ----- END PROGRAM -----
754 '
756 SCREEN 1:CLS:PRINT CHR$(14);:
GOSUB 748:STOP
758 '
760 '    ----- LOCATE FOR ERROR -----
762 '
764 PRINT @1440,"";:RETURN
766 '
768 '    ----- ERROR HANDLING -----
770 '
772 SCREEN 1:CLS:PRINT CHR$(15);
774 PRINT @ 896,"* ERROR, Returning to MENU .....";
776 FOR A1 = 1 TO 5:GOSUB 748:FOR A2 = 1 TO 50:
NEXT A2:NEXT A1
778 GOSUB 734:RESUME 120
780 '
782 '    ----- DATA FOR CURSOR MATRIX -----
784 '
786 DATA 40,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

End of listing

GOODIES FROM ENGLAND

FROM NEWDOS TO MODEL 4

USE NEWDOS EDTASM ON YOUR MOD 4

By M.C.Matthews

The version of Edtasm modified by APPARAT for use with NEWDOS 80 v2 had certain disadvantages, the principal one being that the switches which controlled the line printer and stoppage on errors, among other things, did not work in Model 4. These weaknesses have now been rectified.

Operation is exactly as the NEWDOS version except that if an error is found during assembly it will stop, displaying the error message and the offending line. If BREAK is pressed it will exit from the assembly to the command mode for editing, while pressing any other key will continue the assembly.

When assembly is complete it will display the total errors and free memory, and again wait. Pressing BREAK will exit to command mode, while any other key will continue.

The amendments have been made by patching in the space that was devoted to the Tape routines. Accordingly, the program can no longer be used with Tape.

The following summary of operations is recorded for the benefit of anyone wishing to make further amendments. All addresses are in HEX.

Program entry is at 6F00. This uses a small piece of code which appears to put a flag at 4478, which is changed at exit. It probably does nothing now, but I have left it. The true start is at 588A, where the program is set up.

588ATrue start. Set start of source code at 7700, and put in 5815.
589CGet MEMEND and put in 5313 and set various stores.
58BBCLS.
58BEHome cursor.
58C6Turn on cursor.
58D1Display opening message.
58D7Do CRLF.

At this point we are ready to start entry.

The following is a summary of the assembly process so far as I understand it, with references to the points at which various steps are taken.

There are three stages.

1. The label table is constructed. This is stored at the top of memory, and works down from there.

2. The first assembly is done and displayed on the screen.

3 .The second assembly is done and is written to the disk file.

The procedures used are basically the same, to run through the program getting the required information on a line by line basis.

Assembly starts at 63E3.

6410Get filespec and open file.
643DCheck how many passes done (3 in all). If less then go on else do total errors and label table.
6496is the start of each round. It comes back here after every line is displayed, and at 64A2 tested for the WE flag. This test has now been removed,so that if there is an error it always stops.
64F1 61D6finds the start and end of each line in turn. It returns with the start in DE and the end in HL. This is a key routine. It then runs through each line in turn, finding Labels, and putting them in the label table. After this it starts on a second round, this time assembling the program on the screen and indicating errors.
6563Check for valid opcode.
6DDFThis assembles the line and displays it.
6E14-6E30display the address, opcode and operand during assembly.
6E38Displays the whole line.
6445Displays the total errors.
644EDisplays free memory.

At this stage there is no record of the assembly in memory.

Round 3 repeats round 2, but nothing is displayed. Instead each line is first entered in a buffer at 5324.

6DDFThis is the main subroutine. It evaluates the opcode, putting the result into 53B1.
6EA5 onThis gets the byte (opcode or operand), puts it in a temporary buffer at 5324+.
6ED3-6ED8When the line is finished this moves it from 5324+ to 72D4+.

This buffer ends at 73D3. I have not yet discovered what happens when it is full, but I think it is immediately filed on disk.

6BBB-6BC3Add the end instruction (0202) and the transfer address if there is one. I do not know where the sector addresses are created, but presumably as soon as the sector is filled to 252 bytes the necessary address is put in. It may do it by counting bytes from the start.
550CStarts the display of the label table.
5529actually displays the label.
52B2Exit from label table.

The label table is built up by a complete run-through of the source code for each label in turn. Since it is in alphabetical order it presumably does two runs, the first to find the next label and the second for the addresses.

5E3Ais the line counter for P.
(Display a screen-full.)

There are two further patches which can be inserted. The first removes the need for the rather awkward syntax for L(oad) and W(rite). The commands then become simply L filename and W filename.
It is a 1-byte patch as follows:

PATCH EDTASM/CMD (D1B,FB=2B:F1B,FB=05)

The second is makes it possible to use lower case for comments, but you need to remember to use upper case for commands and operational matters:-

PATCH EDTASM/CMD (D04,68=00 00 00:
F04,68=CD 5E 70

```
.EDTASM/FIX.  Patches the APPARAT version of EDTASM/CMD supplied with NEWDOS to
.run in Model 4 mode.
D00,20=3E 01 00;F00,20=3A 40 38
D03,19=2A 57 C3;F03,19=2D 40 CD
D03,23=3E 3C EF C9 C5 4F 3E 1A EF C1;F03,23=AF 32 F2 75 C3 F8 01 28 52 D3
D03,2D=C9 3E 43 EF C9 3E 3A EF C9 3E;F03,2D=FF 23 7D E6 FE 6F C9 B7 C8 FE
D03,37=3B EF C9 C5 4F 3E 4B EF C1;F03,37=05 28 8A 11 88 56 D5 FE 08
D03,40=C9 3E 44 EF C9 3E 16 EF;F03,40=28 BA FE 0A CA 81 57 FE
D03,48=C5 4F FE 0C 20 07 3E 01;F03,48=0D CA 81 57 FE 0E 28 92
D03,50=32 29 40 18 0B FE 0D 20;F03,50=FE 0F 28 95 FE 17 28 CC
D03,58=07 3A 29 40 3C 32 29 40;F03,58=FE 18 28 AC FE 19 28 BA
D03,60=3E 06 EF C1 C9 3E 64 06;F03,60=FE 1A 28 B1 FE 1B 28 B7
D03,68=00 21 00 00 EF C9 3E 08;F03,68=FE 1C 28 82 FE 1D 28 1B
D03,70=EF FE 80 C0 3E 01 C9 00;F03,70=FE 1E 28 37 FE 1F 28 3C
D03,78=00 7E 23 F5 E6 7F CD 39;F03,78=C9 77 23 3A 28 52 E6 08
D03,80=59 F1 07 30 F4 C9 4F 3E;F03,80=28 01 23 7C FE 40 C0 11
D03,88=02 EF C9 3E 01 EF C9 C5;F03,88=C0 FF 19 E5 11 00 3C 21
D03,90=4F 3E 02 EF C1 C9 3E 08;F03,90=40 3C C5 01 C0 03 ED B0
D03,98=EF C9 08 09 0A 0B 0D 18;F03,98=C1 EB 18 19 7D E6 C0 6F
D03,A0=01 1B 21 E4 5A CD 5E 57;F03,A0=E5 11 40 00 19 7C FE 40
D03,A8=CD 34 58 F5 CD 37 59 F1;F03,A8=28 E2 D1 E5 54 7D F6 3F
D03,B0=FE 59 32 7E 58 CA 27 64;F03,B0=5F 13 18 04 E5 11 00 40
D03,B8=3E 00 32 7E 58 C3 27 64;F03,B8=36 20 23 7C BA 20 F9 7D
D03,C0=0D 50 72 65 73 73 20 42;F03,C0=BB 20 F5 E1 C9 79 B7 28
D03,C8=52 45 41 4B 20 74 6F 20;F03,C8=40 FE 0B 28 0A FE 0C 20
D03,D0=73 74 6F 70 20 6F 72 20;F03,D0=1B AF DD B6 03 28 15 DD
D03,D8=61 6E 79 20 6B 65 79 20;F03,D8=7E 03 DD 96 04 47 CD EE
D03,E0=74 6F 20 67 6F 20 6F 6E;F03,E0=57 20 FB 3E 0A 32 E8 37
D03,E8=2E 8D F5 E5 D5 C5 21 A5;F03,E8=10 F4 18 18 F5 CD EE 57
D03,F0=57 CD 5E 57 CD 66 58 C1;F03,F0=20 FB F1 32 E8 37 FE 0D
```

```
D03,F8=D1 E1 F1 C9 CD 8D 70 CD;F03,F8=C0 DD 34 04 DD 7E 04 DD
D04,00=C5 57 C9 00 00 00 00 00;F04,00=BE 03 79 C0 DD 36 04 00 C9
D04,56=70 57 21 7E 57;F04,56=49 00 21 50 58
D04,5C=07;F04,5C=06
D04,6F=3E 09 EF C9 00 00 D5;F04,6F=08 09 0A 0D 18 01 D5
D04,76=CD 7B 57 D1 B7 C8 F8 FE 80;F04,76=CD 2B 00 D1 B7 C8 F8 FE 01
D04,BB=CD 4A 57;F04,BB=2A 11 44
D04,D9=FB 3E 69 EF 21 00 00;F04,D9=CD F8 01 FB 00 00 00
D04,E0=3E 0F 06 03 EF 0E 0E 3E;F04,E0=00 3E 1C CD 39 59 3E 1F
D04,E8=02 EF 00 00 00 00 00 00;F04,E8=CD 39 59 3E 0E CD 39 59
D05,52=CD 5E 57 00 00 00 00 00;F05,52=7E 23 CD 39 59 07 30 F8
D05,78=2D 57;F05,78=3B 00
D05,7C=74 57;F05,7C=33 00
D07,0B=50 52 49 4E 54 20 53 4F;F07,0B=52 45 41 44 59 20 43 41
D07,13=55 52 43 45 3F 8D;F07,13=53 53 45 54 54 C5
D07,4B=0B;F07,4B=5B
D07,85=80;F07,85=01
D07,8F=0B;F07,8F=5B
D0A,72=18;F0A,72=10
D10,9E=E1 57;F10,9E=8D 70
D10,F1=00 00 00 00 00 00 CD CF 57;F10,F1=3A 86 58 B7 28 08 CD 66 58
D1B,AD=87 57;F1B,AD=27 64
D1C,8C=1B 57;F1C,8C=24 44
D1C,AF=17 57;F1C,AF=20 44
D1C,B4=1B 57;F1C,B4=24 44
D1C,D8=74 57;F1C,D8=33 00
D1D,1D=DB;F1D,1D=DA
D1D,2F=C9;F1D,2F=32
D1D,68=13 57;F1D,68=36 44
D1D,71=13 57;F1D,71=36 44
D1E,40=1F 57;F1E,40=39 44
D1E,4D=1F 57;F1E,4D=39 44
D1E,5F=C8 00 00;F1E,5F=CA 08 57
D1E,8B=1F 57;F1E,8B=39 44
D1E,97=1F 57;F1E,97=39 44
D1E,9D=26 57;F1E,9D=3F 44
D1E,A3=13 57;F1E,A3=36 44
D1E,B1=08 57;F1E,B1=28 44
D1E,E8=0C 57;F1E,E8=09 44
D1E,F6=50 58;F1E,F6=40 00
.End of patch!
```

Editor's note:

To patch the NEWDOS/80 version of EDTASM, you must first transfer it to a Model 4 compatible disk. The easiest way to do this is to boot-up in NEWDOS and format a 35 track, single density disk in drive :1.
Use the following PDRIVE command to set drive :1 as a 35 track, single density drive:
PDRIVE,0,1, TI=A,TD=A,TC=35,SPT=10,TSR=3,GPL=2,A
Now format drive :1 with the command: FORMAT :1
Copy EDTASM/CMD to the 35 track sd-disk in drive :1 by issuing: COPY EDTASM/CMD:0 :1
Now boot-up with TRSDOS/LS-DOS 6. Make sure that EDTASM/CMD is closed by typing RESET EDTASM/CMD. Then type in the above patch listing into an ASCII text editor (TED is perfect for this type of job). When you have made absolutely sure that everything is correct, save it in a file called EDTASM/FIX. Exit the text editor and, from DOS ready, type: PATCH EDTASM/CMD USING EDTASM/FIX
If this is too much work, the patched version of EDTASM will be on TRSTimes on DISK #8.

HINTS & TIPS

IX3
A REVIEW
by Jim Savage

IX3 is a really neat, easy to use indexing program that runs in the Model III mode. Written by Brian D. Harney, who in 1986 was the Treasurer of the Kentucky Genealogical Society, so he and his associates could index genealogical reference found in all kinds of publications that were not indexed, or not indexed by the type reference needed. Example death notices in old newspapers or books about town or area histories.

I am a genealogy hobbyist and I use the program to index such things as correspondence received. When I see a mention in a newspaper, a talk show, or elsewhere, I contact these persons to see if they may have information I can use. When I receive a reply, I number it with the next sequential number in my correspondence file. The letter or whatever document I receive is then entered into my Ix3 correspondence file.

Anytime I wish to search for a particular name, or town, etc. I have only to look at the Index print out sorted by the item desired and I can immediately go to the source document(s). If there are more than one file with the same information all files are listed.

In 1986 the author sold the program for $20, and for a person on limited "fun money" it is a real jewel. I have about 900 entries in some 300 correspondence files and can locate anything there in less than 2 minutes.

IX3 is written and compiled in ZBasic and runs ONLY on TRSDOS 1.3. The "system" is really 5 modules:

1) A mini-editor that works like regular Scripsit, but no printing.

2) A Config program that allows you to enter/edit:
a)Header/Footer/Neither?
b)Header or Footer text (up to 51 chars)
c)Number of columns to print (3 or 4)
d)Column height
e)Page height
f)Left margin
g)Number of spaces between columns
h)Auto page numbering? Y/N?
i)Up to 3 sets of printer codes (in decimal)
and comments for each.

3) An Err/Stat program that performs a quality check on the data, and produces a report with counts based on the starting characters.

4) A Select/Sort program. This reads all index files and selects records based on a letter range that you supply. If the capacity of 987 records is not exceeded, they are sorted and written to a file named IXSORTED.

5) The print program that reads IXSORTED and formats the index into columns with hanging indents, etc.

You are allowed up to 32 characters per index entry, and 3 characters for the page numbers. This limits you to documents of 999 pages. A data file may span multiple data disk, usually between 5,000-10,000 per disk. The maximum limit is 32,766 per indexing application.

Harney has several keystroke saving features, the neatest of these is that when you are keying in several names that have the same Surname, you just enter a comma and that tells the program to use the previous Surname again for this entry. If you can number it you can index it with this program.

The author's address is:
Brian D.Harney, RT.2, Frankfort, KY 40681
(please be aware that this address is from 1986 and may no longer be current.)

PROGRAM UPDATES
Enhancements & Bugs
(Oops)

Dick Burley, faithful member of all the TRS-80 clubs in Los Angeles, reports an error in HAPPY4/BAS on page 20 of the Jul/Aug 1991 issue.

Line 40 should read: **40 IF X < 63 THEN 60**

Dick also suggests the following enhancements to the program:

Change line 70 to: **70 FOR X=7 TO 57 STEP 10**

Add line 135: **135 FOR Y=32 TO 44:SET(0,Y):NEXT**

Also from the Jul/Aug 1991 issue, we have encountered a very deceptive error in the assembly language listings BUFSECTR/ASM and BUFTRACK/ASM.

On page 10, BUFSECTR/ASM has a label called GTB010. It should be **GBT010**.

On page 11, BUFTRACK/ASM has a label called GTB040. It should be **GBT040**.

Gary Shanafelt writes regarding the GrafDISK article, also from the Jul/Aug issue: GrafDISK as currently written will only work on the Radio Shack board. Since the Micro-Labs board has less memory, the utilities abort since they expect more memory than they actually find. The following patches will make the programs compatible with the MicroLabs board:

```
PATCH GRAFDISK/DCT (D04,70=3E 12;F04,70,3E=15)

PATCH GDSAVE/CMD (D00,A7=01 00 4B;
F00,A7=01 00 7A)

PATCH GDLOAD/CMD (D00,D0=01 00 4B;
F00,D0=01 00 7A)
```

Now, it is possible that these patches may already be on the TRSLINK version, as I got my copy from a different source and so had solve the problems on my own.

One correction to the article: The MicroLabs board gives you only 20K additional memory, not 24K. I end up with a GrafDISK of 18 cylinders and 81K of storage area. If I'm doing something wrong and missing additional memory, somebody please let me know.

EXPANSION INTERFACE TEST SEQUENCES FOR MODEL I

The following program is one of the tests used by Radio Shack Repair Centers to check out the operations of the Expansion Interface.

```
10 CLS:
PRINT@10,"EXPANSION INTERFACE TEST
SEQUENCES."
12 PRINT@132,"1.  LOWER 16K MEMORY TEST."
14 PRINT@196,"2.  FULL 32K MEMORY TEST."
16 PRINT@260,"3.  PRINTER TEST SEQUENCE."
18 PRINT@324,"4.  PRINTER INTERFACE
DEBUGGER."
20 PRINT@388,"5.  FLOPPY CONTROLLER I.C.
CHECK."
22 PRINT@452,"6.  NOTES ON EXPANSION
INTERFACE."
24 PRINT@580,"INSERT DIGIT CORRESPONDING TO"
26 PRINT@644,"DESIRED MODULE & DEPRESS
'ENTER'."
28 PRINT@772,"TO EXIT ANY OF THE ABOVE,
DEPRESS 'X'."
29 PRINT@836,"TO STOP (AND THEN CONTINUE),
HIT SHIFT-@."
30 INPUT A1:
IF A1=6 THEN 100
40 IF A1<>1 THEN 50
45 CLS:
PRINT@10,"EXPANSION LOWER 16K MEMORY TEST.":
M=-16384:
GOTO 200
50 IF A1<>2 THEN 60
55 CLS:
PRINT@10,"FULL EXPANSION 32K MEMORY TEST.":
M=0:
GOTO 200
60 IF A1<>3 THEN 70
65 CLS:
PRINT@10,"PRINTER TEST SEQUENCE.":
GOTO 300
70 IF A1<>4 THEN 80
75 CLS:
PRINT@10,"PRINTER INTERFACE DEBUGGER.":
GOTO 350
80 IF A1<>5 THEN 100
85 CLS:
PRINT@10,"FLOPPY CONTROLLER I.C. CHECK.":
GOTO 400
100 CLS:
PRINT@10,"EXPANSION INTERFACE NOTES:"
102 PRINT@128,"32768 = 8000 HEX EXPANSION
INTERFACE RAM LOWER LIMIT."
103 PRINT@192,"49151 = 8FFF HEX EXP. 16K RAM
UPPER LIMIT ADDRESS."
104 PRINT@256,"65535 = FFFF HEX EXP. 32K RAM
UPPER LIMIT ADDRESS."
105 PRINT@384,"14304 = 370E HEX  DRIVE SELECT
LATCH (WRITE) & INTERRUPT BITS."
106 PRINT@512,"14308 = 37E4 HEX  80 IS CASSETTE
SWITCH (WRITE ONLY)."
107 PRINT@640,"14312 = 37E8 HEX  OUTPUT TO
PRINTER (WRITE) & PRIN.STAT. (READ)."
108 PRINT@768,"14316 = 37EC HEX  F.D.CMD
(WRITE) & STATUS (READ) REGISTERS."
109 PRINT@832,"FDC TRACK, STATUS & DATA REGS.
IN NEXT THREE ADDRESSES."
120 A1$=INKEY$:
IF A1$="X" THEN 10
130 GOTO 120
200 RANDOM:
DEFINT A-Z:
P=0:
Z=0:
I=1:
B=255
210 X=RND(255):
N=-32767:
P=P+I:
PRINT@128,"TEST NUMBER = ";P
215 PRINT@128,"RUNNING OK UNLESS ERROR
MESSAGES SPEWING."
220 X=X+1:
IF X>B THEN X=Z
222 A1$=INKEY$:
IF A1$="X" THEN 10
225 POKE N,X:
Y=PEEK(N):
IF Y<>X THEN 290
230 N=N+I:
```

```
IF N< =M THEN 220
290 PRINT"AT ADDRESS ";N+65535;" WAS ";Y;
" SHOULD BE ";X;
GOTO 230
300 A=10:
GOSUB 390:
A=31:
REM *** PRINTER TEST PATTERN ***
310 A=A+1:
IF A> =96 THEN 330
320 GOSUB 390:
GOTO 310
330 A=13:
GOSUB 390:
A=31:
GOTO 310
350 PRINT@128,"OUTPUTS TEST PATTERN TO
PRINTER INTERFACE"
351 PRINT@192,"LOGIC AT ADDRESS 14312
(37E8 HEX) FOR SIGNAL"
352 PRINT@256,"TRACING (W/OSCILLOSCOPE)."
353 A=1
360 FOR X=1 TO 8:
POKE 14312,A:
PRINT@512,PEEK(14312)
370 A=2*A:
NEXT X
380 A1$=INKEY$:
IF A1$="X" THEN 10
385 GOTO 353
390 IF PEEK(14312)=63 THEN 394
392 IF INKEY$="X" THEN 10 ELSE 390
394 POKE 14312,A:
RETURN
400 POKE 14316,3
405 PRINT@256,"FDC FD1771B-01 STATUS
REGISTER = ";PEEK(14316)
406 POKE 14317,18
407 PRINT@320,"TRACK REG.  SHOULD READ 18,
IS ";PEEK(14317)
408 POKE 14318,8
409 PRINT@384,"SECTOR REG. SHOULD READ  8,
IS  ";PEEK(14318)
410 POKE 14319,18
411 PRINT@448,"DATA REG.   SHOULD READ 18,
IS  ";PEEK(14319)
414 IF INKEY$="X" THEN 10 ELSE 405
420 GOTO 407
9999 END
```

End of listing

1987 to 2011

by Roy T. Beck

Sure, the title is cryptic; if George Orwell could use "1984" as a title, surely I can use a longer version of the same. Is there a message to go with it? In Orwell's case, it was a philosophical message. Mine is more technical and mechanistic. Please bear with me.

My message is about the TRSDOS dating structure, which has caused us a lot of worry in the past and will cause more for some of us in the future. As most of us know, the original TRSDOS V 2.3 for the Model I allowed for a range of dates from 1980 to 1987. Who set this limited range, and why?

THE CULPRIT

The who is easy; Randy Cook did it. Surely you all remember Randy? He is the original genius who took Radio Shack's proposed Z-80 computer and Microsoft's BASIC and wrapped an operating system around the whole works, which became TRSDOS for the Model I. Sure, we revile him for some of his bad decisions, but don't forget all the good ones he made. Example: He created the HIT table concept in the directory structure and made it work, beautifully! What, you don't know what I'm talking about? Shame on you. If it weren't for Randy's HIT table, we would still be logging on to a particular drive, ala MS-DOS and CP/M, to locate a program. On any version of TRSDOS, you can call a program located anywhere on up to 8 logical drives in a flash. I regard that as a minor miracle. Try to call a program on another drive under MS-DOS, and all you get is "Program not found". Not found, indeed! The stupid DOS doesn't know how to LOOK for the program, that's what's wrong. IT DIDN'T EVEN TRY TO LOOK FOR IT! But TRSDOS will look for it and will find it.

Sure, Randy made a few bad decisions in his day, and later came to a parting of the ways with Radio Shack. That's another whole story, but VTOS was the product of that schism. Seems to me VTOS had the version numbers 3.0 and 4.0 attached to it, which is of course the reason LDOS has the number 5.X as its family name. And of course LS-DOS for the Model 4 bears the family name of 6.X. These latter two DOSes were developed in the heyday of Logical Systems, Inc, and of course Roy Soltoff had much to do with both of them, to our great advantage. Where would we be today without his tenacity and genius?

PERMITTED DATE RANGES

To get back to the subject of this story. The limited date range of 1980 to 1987 was one of Randy's less good decisions: Why did he select such a limited range? Obviously one of his concerns was to limit the amount of directory space devoted to the date of the file. By restricting the date range to eight years, he was able to hold the year record to only 3 bits. But why only 8 years? The only explanation I ever heard was an apocryphal story that the whole Radio Shack computer venture wasn't expected to survive that long, and therefore 8 years was ample. We disproved that one, didn't we?

Many of us remember Jan 1, 1988. Suddenly our dutiful slaves wouldn't accept the current date. Had the world come to an end? No, just the range of years allowed for in the DOS. But Roy Soltoff to the rescue. During the previous summer, he and the people at LSI had been planning for this day (date). LSI had worked out a scheme to update TRSDOS 6.2.1 to 6.3.0 and include some needed fixes and improvements along the way, one of which was to extend the date structure to 1999. This postponed the day of reckoning for another 12 years, and at the same time made some desirable improvements in the DOS. (It also was the time of the infamous rumor about a secret, hidden copy protection scheme in 6.3.0, but that's another story for another day). At the same time, MISOSYS issued an updated version of LDOS 5.3 to parallel the development of TRSDOS.

By the way, LS-DOS = TRSDOS. LS-DOS is the DOS, as developed by LSI and carried on by Soltoff (MISOSYS). TRSDOS is the version of LS-DOS licensed to Tandy Radio Shack. So far as I know, they are absolutely identical, the distinction being purely legalistic. For my (our) purposes, they are identical.

LS-DOS 6.3.0

For some time, a year or two if memory serves me, all was well in the date structure of TRSDOS. And then Roy Soltoff acquired control/ownership of TRSDOS 6.3.X from LSI, and announced a new upgrade of TRSDOS, 6.3.1. This included some minor improvements, and also ex-

tended the date limit yet again, this time through the year 2011! So now TRSDOS has a date range of 32 years, from 1980 to 2011. Since 32 years can be stored in 5 bits, it was accomplished by assigning two extra bits to the date structure in addition to the original 3. But maintaining upward compatibility has always been a prime concern of Soltoff, and something else had to be given up in order to have the new date range and maintain a maximum of compatibility with that which has gone before.

To obtain the needed extra bits to extend the year range, something else had to be sacrificed. What was that? TRSDOS used to carry two passwords, which were the OWNER password and the USER password. Again, a feature defined by Randy Cook. LSI and Soltoff decided there really was not much use being made of two levels of passwords, and one could be foregone to extend the dating. At the same time, time stamping could be introduced. Accordingly, the two bytes in the directory, numbers 18 and 19 which formerly held the USER password were redefined to provide space for the new date and time features.

THE DIRECTORY STRUCTURE

Every directory entry for an LDOS or LSDOS file has the same format, 32 bytes in a directory structure which contains almost all the information required to locate and load a file into the RAM. I qualified this because the concept of "Extents" allows linking of multiple directory entries for highly fragmented files. I will not discuss this as it is not related to the dating situation.

AN EXAMPLE

I will show the directory entry of a file under both TRSDOS 6.2 and TRSDOS 6.3.1, and then will explain what changed. The 32 bytes will be presented on two lines of paired bytes, with a two line header giving the (decimal) byte numbers for ease of locating the bytes being discussed.

Below is the directory entry for a file named FOLK6 under both DOSes.

The bytes which store the dates under 6.2.X are numbers 01 and 02. Under 6.3.1 the bytes involved include 01, 02, 18 and 19.

So what happened to bytes 18 and 19? This pair of bytes formerly contained the "USER password". Bytes 16 and 17 continue to hold the "OWNER password". The deletion of the USER password made 16 bits available for reassignment. To make good use of this, LSI and Soltoff elected to add "time stamping" to the DOS' capability. Thus, if you are revising and saving multiple versions of a file, the time stamp now allows you to identify them chronologically in the directory.

Byte 01

Bits 3-0 hold the month in binary form. The 7 of 47 therefore represents July.

Byte 02

Bits 7-3 hold the day of the month, 1-31, in binary notation. C8 = 1100 1000; bits 7-3 = 25, the date.

Bits 2-0 formerly held the year offset, with 3 bits representing an 8 year range. For technical reasons, these bits could not be reassigned. They are now simply ignored by the date system. Don't try to use them for any purpose, they will still occasionally be written to by the operating system.

Byte 18 is now used as follows:

Bits 7-3 contain the hour. 5 bits allow for 32 values, so hours 0 through 23 are represented in binary. 53 = 0101 0011 = 10 AM.

Bits 2-0 contain the most significant bits of the minute, in this case 011.

Byte 19 is now used as follows:

Bits 7-5 of byte 19 contains the least significant bits of the minute. The combined 6 bits from byte 18 and 19 allows for values of 0 to 59 minutes. In this case bits 7-5 contain 011, and the combined 6 bits 011011 represents 27 minutes.

Example 1:

Byte	0001	0203	0405	0607	0809	1011	1213	1415	
	1617	1819	2021	2223	2425	2627	2829	3031	
TRSDOS 6.2.X									
	1047	C8E9	0046	4F4C	4B36	2020	2020	2020	.G##.FOLK6
	9642	9642	0800	2441	FFFF	FFFF	FFFF	FFFF	#B.K..$A########
TRSDOS 6.3.1									
	1047	CBE9	0046	4F4C	4B36	2020	2020	2020	.G##.FOLK6
	9642	536B	0800	2441	FFFF	FFFF	FFFF	FFFF	#B.K..$A########

Finally, bits 4-0 of byte 19 contains the year offset from 1980. 5 bits represents 32 years from 1980 to 2011. In this case, 01011 = 11. 1980 + 11 = 1991.

Combining all the pieces of information reveals the file FOLK6 was last modified at 10:27 AM on July 27, 1991.

MISCELLANEOUS PIECES OF INFORMATION

The DOS keeps its version number in the GAT sector at byte CB. This was 60 for version 6.0, and is now 63 for version 6.3. DOS 6.3 needs to know which dating system, old or new, is in use on a disk. Byte CD of the GAT table serves this purpose. Bit 3 is a flag; if set, the new dating is in use. Since the older versions 6.0 to 6.2 did not anticipate the change in dating, the bit is meaningless to them; Version 6.3.X looks for and understands the bit.

CONSEQUENCES OF THESE CHANGES

If the user is careful not to mix DOSes, everything works fine. The only real fly in the ointment occurs when a disk with the new style dating is used with a 6.2 or earlier DOS. Since bytes 18 and 19 were formerly the User Password, the earlier DOS will interpret whatever is there as a User Password and deny access to the file. By zapping these two bytes back to 9642, all will be well under 6.2, but returning the disk to 6.3.1 will then produce peculiar hour and year information.

Moral: don't use disks with new dates under old DOSes. Roy has provided a utility function to convert a new style disk back to the earlier date structure. Be sure to use this if you MUST use the earlier DOS. The best procedure is to retire 6.2 to pasture and DATECONV all your data disks to the new date system under 6.3

LDOS 5.X

Many times, Roy Soltoff has said in the pages of The Misosys Quarterly that he would never do an upgrade of LDOS 5.1.4 for the Model I because there was simply not enough market to justify his spending his limited time upon it. (MISOSYS is pretty much a one-man operation, and Roy is the man). I understood his position, and never took issue with it. But now, what do you know, Roy has suddenly announced an upgrade of both LDOS 5.3 for the Model III AND LDOS 5.1.4 for the Model I! Both will be upgraded to the 5.3.1 level to bring the three different DOS' to the same level designation, namely 5.3.1 for Models I and III and 6.3.1 for the Model 4. This update extends the dating structure to 2011 for both the Model I and Model III, and Roy also says he has made all the commands and functions as nearly identical as the hardware differences between the three models permits. I promptly ordered both 5.3.1's so I can keep au courant as to what Roy is doing. I urge every one with either or both of the Models I and III to upgrade to this latest level. Roy is remaining the 'good guy', so let's support him.

DZZ

ANOTHER DATA BASE SYSTEM

for Model 4 - Multidos

by Jim E. King, MSEE, C10

When I first began writing my Data Base program, I needed to create and maintain telephone and address lists. I still have this need, so DZZ/BAS is my latest rewrite of that data base started long ago. As an aside, an earlier version was published on Vol. 7 (12/86) of the Valley Hackers TRS-80 Users Group disk library.

For a couple of years now, I have been using MultiDOS 2.1 for Model 4, so naturally DZZ is written to use some of the special features of that DOS. However, with some modifications the program should run under any of the others, even 6.3 and Model I.

DZZ uses the machine language two-dimensional sort, CMD"Q", by Vernon Hester in line 16, but I believe that the other DOS's have something similar, though with a slightly different syntax.

I use no PRINT@'s, but do PEEK & POKE 16425, which is the printer line counter during LPRINT, to get the spacing that I want. Before printing it checks if INP(248) = 61 to find out if my 8510 is ready (line 800); your printer may return a different INP number. This will not work on the other Model 4 Dos's, so it will probably be best to eliminate this check.

DZZ loads the entire sequential file into memory. It displays one record on a line, twenty records on a Model 4 screen, about fifteen on a Model III.

You can sort on any column. There is binary search on column 2 (very quick), and sequential INSTR search on all columns.

Editing permits you to rewrite any record, field by field. If a field is OK, press enter (<E>), else rewrite it.

Deleting a record is done by changing the field to a CHR$(133), sorting, and then decrementing the length of the data by 1. I first did it by moving all records up by 1, and the garbage was immense. The binary search displays all copies of your search $tring, not just the one it lands on.

I have a need to know the approximate date that I input or update a record, thus I use the last field to store the month and the year. I use the following format: the last digit of the year, followed by a letter indicating the month: Jan, Feb, Mar, Apr, maY, Jun, juL, Aug, Sep, Oct, Nov, Dec. Press the '.' & <E>: 1L = July 91.

Operation

The program first asks how many Cols. (fields per record) are wanted (line 991); the default is 4.

Lines 990-999 is the setup area.

Main Menu (line 99): The program checks whether there is data in the array. If the array is empty the menu says:

Model 4 DBZ; # Cols <I>nput <L>oad ?

You then do either. After there is data in the array the Menu says:

Model # DBZ: dispLAyB Edit Format sOrt SeaRch = blanK saVe Print

The available one-key commands are the capital letters; some are obvious.

B starts the display at the beginning.

L enables you to start display at the line of your choosing.

If you should want to continuously scroll through the rest of the data, press A.

Set the tabs for the fields with Format.

S is the binary search.

R is the sequential search.

= displays records that are the same in the sorted field.

K goes through the data and changes " " to "".

The display scrolls 10 lines then shows the Edit Menu (line 350):
---- <E>dit <D>elete < >.

Pressing the up arrow resets the line counter to about 40 lower so that scrolling starts earlier.

The program starts at line 0, which says GOTO990.

I put general comments & reminders to me in lines 1 and 2.

990-999 is the setup area: DEF, etc., then GOTO 135 (Load Data).

All lines from 3 to 90 are subroutines.

Line 91 is one of the error recoveries.

The Menus are followed by Inkey$, GOSUB 8, and followed by IF tests for the selections, lines 99 - 199, and 350 - 390.

200-240 is new input that calls:

250-290 the input subroutine, also used for edit.

300-320 is display.

350-395 is the Edit Menu & IF tests.

440-465 is the display formatting Sub called in 310.

500-535 is the binary search, etc.

570-575 is the sequential search.

600-630 is the formatting input.

700-705 is Save

730-735 is Load

800-822 is LPrint

840-865 is the LPrint format, same as 440-465 + the capability of an offset to the right

Along with DZZ, I am including some data files which may be of interest to the TRS-80 community, such as:

List of Computer Companies

Micro80, 1st. 2 years.

Local ZIP codes and where they are.

Indices for MultiDOS, LSDOS, etc. that have page #, Command/ & a short description.

Indices for TRSTIMES, partial TRSNews & OCTUG

I believe that Micro80, ErrMod4, ErrMulti, HackrLib, LSDOS631, MultDOS2, MultDS17, Prefx213, Prefx818, TRSTimes, and Visi4 are complete.

One of my pet peeves is editors trying to be cute with titles to articles. Sometimes they ARE cute, but then most of the time the title gives no idea what the article is about, making filing by title almost impossible. So I file by some keyword.

Feel free to edit, correct spelling, syntax, grammar - but try not be cute with the title.

DZZ/BAS

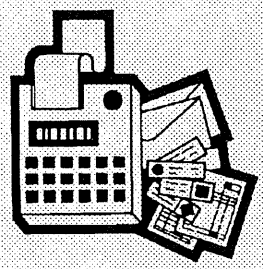
```
0 CLEAR21000:DEFINTC,H-V:DEFSTRD,W-Z:W="J":
GOSUB6:Y="Model 4 MultiDOS 2.1 Version; Printout
Wraps":GOSUB3:PRINT:POKE16410,40:GOTO990'-
DZZ
1 'MaxCLEAR=24500;
Check LPrint start at 0; Change to LINK?; ReDo
seaRch;
2 '/DA2 SNAFU;
Fix: upper Rt corner heading 804-810; ? load routine at
720;
Split ZH into ZH & ZP; Merge a datafile;
Load a narrower array,Z=width;
Saves Caps(KC) instead of ZD, restore ZD;
3 PRINTTAB(38-LEN(Y)/2)Y;:
RETURN
4 FORH=0TO0+K:
PRINTCHR$(7);:
NEXT:
K=0:RETURN
6 PRINTTIME$;:
Y="DZZ Z$(I,"+W+") Data Base":
GOSUB3:PRINT:
RETURN
7 IF LEN(Z) THEN FOR L=1 TO LEN(Z):
M=ASC(MID$(Z,L,1)):
MID$(Z,L,1)=CHR$(M+32*(M>96)):
NEXT:
RETURN:
ELSE RETURN
8 Z=INKEY$:IFZ=""THEN8ELSE
IFZ=CHR$(31)THENENDELSEGOSUB7:
RETURN
9 PRINTCHR$(29)STRING$(JU+1,27)CHR$(31);:
JU=0:
RETURN
10 IFKDTHENY="Close"+D:
KD=0
ELSE Y="OPEN"+D
11 GOSUB3:
RETURN
12 PRINT"CAPITALIZE"ZC"("KC+1")?";:
GOSUB8:
KC=VAL(Z)-1:
PRINTKC+1:
RETURN
13 PRINTZF"/DA"W" "FNZR(N)"/"FNZ3(MM)"="
FIX(N*100/MM+.5)CHR$(8)"%";:
RETURN
14 X=ZD+STR$(KC)+XF+YF:
RETURN
16 PRINTCHR$(29)CHR$(30)"Sorting on"C+1;:
CMD"Q",N,Z(0,0),C:
PRINT"-SORTED":
RETURN
17 FORI=0TON:
IFINSTR(Z(I,LW),Z)THENGOSUB440:
NEXT:RETURN
ELSENEXT:RETURN
18 KE=0:FORJ=0TOLW:KE=KE+(LEN(Z(0,J))>0):
NEXT:RETURN
```

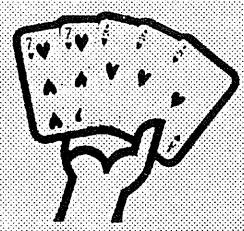
```
19 IFLEN(X)>16THENXF=MID$(X,17,LEN(X)-16):
RETURNELSERETURN
20 PRINT" Heading: ("ZH")?";:LINEINPUTZ:
IFLEN(Z)THENZH=Z
21 PRINTZF;ZH", Is a Space needed before the
Heading? (Y/N)":GOSUB8:
IFZ="Y"THENZH=" "+ZH
22 RETURN
26 FORI=0TON:
FORJ=0TOLW:I
FZ(I,J)=" "THENK=K+1:Z(I,J)=""
27 NEXTJ,I:
PRINTTAB(55)CHR$(27)K"Blanks Removed":
GOTO99
30 PRINTZF"!"ZH" N="N"Sort Col."C:
RETURN
32 CLS:GOSUB30:I=0:PRINT" I Z(I,0)     Z(I,1) --etc."
33 PRINTI;:
FORJ=0TOLW:
PRINTCHR$(133)Z(I,J);:
NEXTJ:
PRINT:
IFZ<>"A"ANDIANDI/10=FIX(I/10)THENGOSUB350:
IFZ<>""THEN105
34 I=I+1:GOTO33
50 GOSUB8:
IFVAL(Z)=0ORVAL(Z)>LW+1THENRETURN
ELSEC=VAL(Z)-1:PRINTSTRING$(6,24)ZC;C+1;:
RETURNELSERETURN
51 STOP:PRINTSTRING$(11,24)CHR$(30)C+1"For? ";:
LINEINPUTZ:
RETURN
55 L=0:GOSUB50:
FORI=1TON-1:
K=K+1:
IFZ(I,C)=Z(I+1,C)THENGOSUB440:PRINT:
I=I+1:GOSUB440:
IFKANDK/5=FIX(K/5)THENGOSUB350
56 NEXT:
GOTO99
70 PRINT" Data Filename ("ZF") Without /DA"W;ZM;:
LINEINPUTZ:GOSUB7:
IFLEN(Z)>8THENGOSUB9:GOTO70
ELSEIFLEN(Z)>1THENZF=Z:RETURNELSERETURN
71 ZD=LEFT$(TIME$,14):RETURN
75 IFKYPRINT"Testing File SNAFU (Col.1=0)":P=0:
FORI=0TON:
IFVAL(Z(I,0))=0 THEN P=1:
GOSUB440 ELSEELSERETURN
76 NEXT:
IFPTHENPRINT"There Are 0s in Col.1, CONTINUE?
Y/<N>":K=6:GOSUB4:GOSUB8:IFZ="N"THEN99
77 RETURN
80 PRINT"Test for Line Too Long to LPRINT on 1 Line":
K=0:
FORI=0TON:
L=0:
FORJ=0TOLW:
L=L+LEN(Z(I,J)):
NEXT:
IFL>(76-LW)THENK=K+1:
GOSUB440:IFK/5=FIX(K/5)THENGOSUB350
81 NEXT:
RETURN
88 IFLEN(Z(I,0))=7THEN
LPRINTLEFT$(Z(I,0),3)"-"RIGHT$(Z(I,0),4)" ";:
ELSELPRINTZ(I,0)" ";
89 RETURN
90 PRINTTAB(59)"("FNZR(MM)","W")"MEM"Memory":
RETURN
91 K=9:GOSUB4:
PRINTERR"ERROR-Line"ERL;':
CMD"E":IFERR=106THEN200
92 RESUME105
99 CLOSE:K=0:KD=0:GOSUB4:GOSUB18:
PRINT"Model 4 DBZ";:IFKETHENGOSUB10:GOSUB90:
GOSUB13:
PRINT": dispLAyB Edit Formats sOrt SeaRch = blanK
saVe Print"
ELSEPRINT", "W" Cols <I>nput <L>oad?";:
GOSUB90
101 GOSUB8
105 IFZ="?"THEN32 ELSEIFZ="C"THENGOSUB12:
GOTO99 ELSEIFZ="H"GOSUB20
110 IFLEN(Z)THENIFASC(Z)>47ANDASC(Z)<52
KD=1:GOSUB10:CMD"DIR "+Z
120 IFZ="D"GOSUB380
125 IFZ="A"ORZ="B"KD=1:I=0:CLS:
PRINTCHR$(28):GOTO300
130 IFZ="I"THEN200
135 IFKE=0THEN KD=1:
GOSUB10:GOSUB4:GOSUB90:
PRINT"Load";:GOSUB70:IFZ="M"THEN99
ELSEIFLEN(Z)=1THEN105 ELSE730
140 IFZ="L"ORZ="Y"I=0:INPUT"Start at line #";I:
GOTO300
145 IFZ="E"GOSUB370
150 IFZ="O"KD=1:GOSUB10:PRINT:
PRINT"Sort "ZF" on"ZC"("C+1")?";:
GOSUB50:GOSUB16
155 IFZ="S"THENPRINT"Binary UPPERCASE
Search"ZC;C+1"For? ";:
LINEINPUTZ:GOSUB7:GOTO500
160 IFZ="R"THENPRINT"Sequential Lowercase
Search"ZC"("C+1")?";:GOSUB50:
IFZ<>"<"AND Z<>"M"
THENPRINTSTRNG$(11,24)CHR$(30)C+1"for? ";:
LINEINPUTZ:GOTO570
165 IFZ="="PRINT"Test for Identical Fields in"ZC;C+1:
GOTO55
170 IFZ="K"PRINT"Removing ' ' in all Fields":GOTO26
175 IFZ="T"PRINT"Search for Records with Date";:
INPUTZ:GOSUB17
180 IFZ="F"PRINT"Format: ";:GOTO 600
```

```
190 IFZ="V"KD=1:ONERRORGOTO710:GOSUB10:
PRINT:GOSUB16:GOSUB75:ZT="1":GOTO700
199 IFZ="P"THEN800ELSE99
200 I=(N+1)*(-1*(N>0)):IFKETHEN210 ELSEZF="":
PRINT"Input <9 Letters";:GOSUB70:GOSUB9:
PRINT"New File Called:"ZF;:
GOSUB20:GOSUB9:PRINTZF;ZH:I=0:
PRINT"Enter: Phone #s Without '-', '.'to End Input,
'<'if Error":GOSUB12
210 IFI>MMTHENPRINT"ARRAY FULL!":K=9:
GOSUB4:GOTO99
220 GOSUB250:GOSUB18:
IFZ="."ORZ="v"ORZ="o"THENI=I-1:
GOSUB7:GOTO105
230 IFI>N THEN N=I
240 I=I+1:GOTO210
250 PRINTTAB(59)".EOF <ErrCol"KC+1"CAPS"
CHR$(29)"NEXT:";:FORJ=0TOLW:
IFJ=1 PRINTTAB(5);
255 PRINTICHR$(8)","J+1CHR$(8);:
IFJ=KCTHENPRINT"CAPS";
260 PRINT">"Z(I,J)">";:LINEINPUTZ:
IFJ=0ANDZ="."ORZ="v"ORZ="o"THENGOSUB9:
RETURN ELSEIFIANDJ=0ANDZ="<"THENI=I-1:
GOSUB9:GOTO250 ELSEIFZ="<"THENGOSUB9:
GOTO250 ELSEIFJ=KCANDLEN(Z)GOSUB7
270 REMIFJ=LWANDLEN(Z)=2
THENGOSUB7'capitalizeDate
280 JU=-(J<>0):
IFJ=LWANDZ="/"ORZ="."THENZ(I,LW)=ZA:
Z=""ELSEIFLEN(Z)THENZ(I,J)=Z
290 GOSUB9:GOSUB440:NEXTJ:
GOSUB71:J=0:RETURN
300 O=20:R=0:
PRINTZD;N;ZF;ZH:I=-I*(I>0)'R=Display counter
310 GOSUB440:
IFRANDZ<>"A"ANDR/O=FIX(R/O)GOSUB350:
IFZ<>"A"ANDZ<>"["ANDZ<>""THEN105
320 IFI<N THENI=I+1:R=R+1:GOTO310ELSE
PRINT"EoF ";:GOSUB350:
IFZ="["ORZ=""THENI=I+1:GOTO310 ELSE105
350 PRINT;:GOSUB13:
PRINT" <E>dit <D>elete <[> "XF;ZM:GOSUB8
360 IFZ="["THENCLS:I=I-33:I=FNI0(I):I=I*(-(I>0)):
O=20:PRINTZD;N;ZF;ZH:RETURN
370 IFZ="E"ORZ=","PRINT"Edit";:GOSUB395:
GOSUB9:IFI<0ORI>NTHEN350
ELSEGOSUB440:GOSUB250:I=FNI0(I):
O=10:R=0:Z="[":
PRINTZD;N;ZF;ZH:RETURN
380 IFZ="D"PRINT" DELETE from";:GOSUB395:
IFI<0ORI>NTHEN350
ELSEL=I:GOSUB440:
PRINT"Thru";:GOSUB395:IFI<0THEN350
ELSEM=I:FORI=LTOM:Z(I,C)=CHR$(143):NEXT:
GOSUB71:GOSUB16:N=N-M+L-1:
I=FNI0(L):O=20:Z="[":PRINTZD;N;ZF;ZH:RETURN
390 O=10:R=0:GOSUB9:RETURN
395 PRINT" Line #("I")";:INPUTI:RETURN
440 PRINTFNZ3(I)" "TAB(VAL(LEFT$(XF,2)));'Col.1 tab
450 IFRIGHT$(X,1)="F" AND LEN(Z(I,0))=7 THEN
PRINTLEFT$(Z(I,0),3)"-"RIGHT$(Z(I,0),4)" ";:
GOTO470'fone#
460 PRINTZ(I,0);'Col.1
470 PRINTTAB(VAL(MID$(XF,3,2)))Z(I,1);:
IFLW<2THEN490
475 FORH=2TOLW:IFVAL(MID$(XF,2*H+1,2))
THENPRINTTAB(VAL(MID$(XF,2*H+1,2)))Z(I,H);
ELSE PRINTCHR$(133)Z(I,H);
480 NEXT
490 PRINT:RETURN
500 J=1:PRINTTAB(39)CHR$(27)CHR$(30)Z:
ZT=Z:LO=0:HI=N
505 I=(HI+LO)/2:IFZ>Z(I,C)THENLO=I+1
ELSEHI=I-1'binary
510 IFINSTR(Z(I,C),ZT)<>1ANDLO<=HITHEN505
'endBinary
515 IFINSTR(Z(I,C),ZT)THENI=I-1:IFI>0THEN515
ELSEI=I+1ELSEI=I+1
520 IFLO>HIANDINSTR(Z(I,C),ZT)<>1 PRINTZT
" would be near line"HI
525 IFINSTR(Z(I,C),ZT)<>1 PRINT"EoS ";:I=I-1:
GOSUB350:IFZ<>"["THEN105
530 GOSUB580:IFJ/20=FIX(J/20)GOSUB350:
IFZ<>"["ANDZ<>""THEN105
535 IFI<NTHENI=I+1:GOTO525 ELSE
PRINT"EoS&F";:GOSUB350:
IFZ="["THEN525 ELSE105
570 ZT=Z:J=1:L=0:FORI=0TON:GOSUB580:
IFJ/20=FIX(J/20)GOSUB350:
IFZ<>""ANDZ<>"["THEN105
575 NEXT:PRINTJ-1"Records":GOTO99
580 IFINSTR(Z(I,C),ZT)THENGOSUB440:
J=J+1:RETURNELSERETURN
600 YF="":GOSUB19:
PRINTXF"; Tab values OK<Y>/N?":GOSUB8:
IFZ="Y"THEN99
610 K=0:PRINT"Enter Tab Values, or For a \ Delimeter
Enter 0 or any non #:":Z="04":
PRINT"Tab for Column 1 (04)";:INPUTZ:
GOSUB635:IFKGOSUB9:GOTO610
615 XF=Z:Z="13":PRINTXF"; Tab for Column 2 (13)";:
INPUTZ:GOSUB635:IFKTHEN615
620 XF=XF+Z:FORJ=2 TO LW:Z="00":
PRINTXF"; Tab for Column "J+1"(00)";:INPUT Z:
GOSUB635:IFKTHEN620
622 ' put Offset command, ZO, here
625 XF=XF+Z:NEXT:
PRINT"Is this a Telefone File? <Y>/N":
GOSUB8:IFZ="Y"YF="F
630 GOSUB14:I=0:GOTO300
635 IFZ="."ORZ="<"THEN99
640 IF LEN(Z)>2 THEN GOSUB9:K=1
645 IF LEN(Z)=1 THEN Z="0"+Z
650 RETURN
666 FORI=0TON:PRINTI;:
```

```
IF MID$(Z(I,0),3,1) = "0"THEN MID$(Z(I,0),3,1) = " "
670 NEXT:GOTO99
700 PRINTZD" Saving"N;ZF"/DA"W":"ZT",";:GOSUB14:
OPEN"O",1,ZF + "/DA" + W + ":" + ZT:
PRINT#1,N;C;X","ZH:FORI = 0TON:FORJ = 0TOLW:
PRINT#1,Z(I,J):NEXTJ,I:PRINT"-SAVED
705 CLOSE:
IFVAL(ZT) < 3THENZT = RIGHT$(STR$(VAL(ZT) + 1),1):
GOTO700 ELSE99
710 K = 9:GOSUB4:PRINTERR"ERR-Line"ERL;:
CMD"E":RESUME705
730 OPEN"I",1,ZF + "/DA" + W:INPUT#1,N,C,X:
LINEINPUT#1,ZH:PRINT"Loading"N;ZF;ZH:
IFN > MMTHENPRINT"Data"N-MM"Longer than Array
DIM, CLEAR 24600":CLEAR 24600:
DEFINTC,H-V:DEFSTRD,W-Z:GOTO990
735 FORI = 0TON:FORJ = 0TOLW:LINEINPUT#1,Z(I,J):
NEXTJ,I:CLOSE:GOSUB75:I = 0:KE = 1:
ZD = LEFT$(X,14):KC = VAL(MID$(X,16,1)):
PRINT"Capitalize Col."KC + 1:GOSUB19:GOTO300
777 PRINT"Load File/DA3 into /DA4; Name
Without/DA3";:INPUTZ:GOSUB7:ZF = Z:W = "3"
780 OPEN"I",1,ZF + "/DA" + W:INPUT#1,N,C,X:
LINEINPUT#1,ZH:PRINT"Loading"N;ZF;ZH:
IFN > MMTHENPRINT"Data"N-MM"Longer than Array
DIM, CLEAR 24600":CLEAR 24600:DEFINTC,H-V:
DEFSTRD,W-Z:END
785 FORI = 0TON:FORJ = 0TO2:
LINEINPUT#1,Z(I,J):NEXTJ,I:CLOSE:
GOSUB75:I = 0:KE = 1:ZD = LEFT$(X,14):
KC = VAL(MID$(X,16,1)):XF = MID$(X,17,2*LW):
PRINT"Capitalize Col."KC + 1:W = "4":GOTO300
800 IFINP(248) < >61THEN
PRINT"Turn on Printer"ZM;:Z = INKEY$:
GOSUB7:IFZ = "M"THENGOSUB9:GOTO99
ELSEPRINTCHR$(29);:GOTO804 ELSELN = 0:
PRINTCHR$(30)"LPrint Line #s? <Y>/N;"ZF;ZM:
GOSUB8:IFZ = "Y"THENLN = 1 ELSEIFZ = "M"THEN99
804 PRINT"Offset All Print Tabs ("IT") Spaces?";:
GOSUB8:
IT = VAL(Z):PRINTIT
806 K = 1:L = 0:
INPUT"Start at Line # (Default = 0)";L:
IFL < 0THEN99
808 INPUT"# of Copies";K:
IFK < 0THEN99
810 FORG = 1TOK:
PRINTN;ZF;ZH"/DA"W:
POKE16425,1:LPRINTZD;N;ZF;ZH"/DA"WTAB(72)ZF:
FORI = LTON:
IFLNTHENLPRINTFNZ3(I)" ";
812 GOSUB845:
IFI = 0THENLPRINTTAB(75)LEFT$(Z(0,C),2)"-";:
IFI + 61 < NTHENLPRINTLEFT$(Z(61,C),2);
ELSELPRINTLEFT$(Z(N,C),2);
814 IFPEEK(16425) = 1THEN
LPRINTTAB(75)LEFT$(Z(I-1,C),2)"-";:
IFI + 62 < NTHENLPRINTLEFT$(Z(I + 62,C),2); EL-
SELPRINTLEFT$(Z(N,C),2);
816 IFPEEK(16425) > 0THENLPRINT
818 IFI > 1ANDPEEK(16425) = 0THEN
LPRINTTAB(72)ZF
820 IFPEEK(16425) > 63THENLPRINT:
LPRINT:POKE16425,0'NextPage
822 NEXTI:
POKE16425,PEEK(16425) + 1:
LPRINTCHR$(12):LPRINT:
NEXTG:GOTO99
840 IF LI LPRINTTAB(IT)FNZ3(I)" ";' formats follow
845 LPRINTTAB(VAL(LEFT$(XF,2)) + IT);'Col.1 tab
850 IFRIGHT$(X,1) = "F" AND LEN(Z(I,0)) = 7 THEN
LPRINTLEFT$(Z(I,0),3)"-"RIGHT$(Z(I,0),4)"";:
GOTO860'fone#
855 LPRINTZ(I,0)" ";'Col.1
860 LPRINTTAB(VAL(MID$(XF,3,2)) + IT)Z(I,1);:
FORH = 2TOLW:
IFVAL(MID$(XF,2*H + 1,2))
THENLPRINTTAB(VAL(MID$(XF,2*H + 1,2)) + IT)Z(I,H);
ELSE LPRINTCHR$(92)Z(I,H);
865 NEXT:
RETURN
900 FORI = 0TON:
IF LEN(Z(I,LW)) = 2 AND ASC(RIGHT$(Z(I,LW))) < 91
THENM = ASC(RIGHT$(Z(I,LW))):
MID$(Z(I,LW),2,1) = CHR$(M-32*(M < 91))
910 NEXT:
GOTO99' does?
990 DEFFNZ(I) = RIGHT$(STR$(I),2):
DEFFNZ3(I) = RIGHT$(" " + STR$(I),3):
DEFFNZR(I) = RIGHT$(STR$(I),LEN(STR$(I))-1):
DEFFNI0(I) = FIX(I/10)*10:ONERRORGOTO91
991 KC = 8:KY = 0:PRINT"# of Columns/Fields(4)?":
GOSUB4:GOSUB8:W = Z:IFVAL(Z)THEN LW = VAL(W):
IFLW < 2THENLW = 2:
W = FNZR(LW) ELSEELSELW = 4:W = "4":ZF = "BUS
992 IFVAL(TIME$)THEN
ZA = MID$(TIME$,8,1) +
MID$("jfmayuLgsond",VAL(LEFT$(TIME$,2)),1)
993 D = " DiskDrive":ZC = " Field #":ZM = " <M>enu?
994 JU = 2:GOSUB9:GOSUB6:
MM = FRE(Z)/11/LW:LW = LW-1:
DIMZ(MM,LW):PRINT"'"ZA"' will be put in the last field
for the year/month when '.' is entered.
995 IFLW = 2THENZF = "TOLL"
ELSE IFLW = 4THENZF = "GASDATA"
ELSEIFLW = 5THENZF = "DRUGS
996 'C = SortCol? ZD = FileDate760 JC = Col#Yelected
KC = Col#Capitalized J$ = LW + 1KD:line10
ZL(777,780)? ZT = Ztemporary KP = YaveFlag93,93
O = #LinesDisplay
999 GOTO135:REMPRINT"SNAFU Check <Y>/N?";:
GOSUB8:IFZ = "Y"THENKY = 1:
PRINT" Always Enter a # in Col.1
```

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.

TRS-80 NOSTALGIA

Thinking guilty little thoughts about getting a 386?

Then you need:

WHAT I DID WITH MY TRASH

(Ten years with a TRS-80)

A new book by Eric Bagai, bound in mercedes silver an filled with essays, parodies, weird rumors, mythic hacks, and TRS-graphics. The way it was when love was a warm Z-80 - the way it should have been. Send $5.95 (non US: $6.95) now to:

Flaming Sparrow Press
Box 82289
Portland, OR 97282

Add five cents for autograph.
Order yours today!
See? You're feeling better already.

TRSTimes on DISK #7

Issue #7 of TRSTimes on DISK is now available, featuring the programs from the Jan/Feb, Mar/Apr and May/Jun 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 are still available at the above prices

PUBLIC DOMAIN GOOD GAMES FOR MODEL I/III.

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

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

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

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

LDOS 5.3.1: the support continues

- ☆ The DATE command, "Date?" prompt on boot, and the @DATE SVC now support a date range of 32 years; from **January 1, 1980 through December 31, 2011;** time-stamping, too.
- ☆ **Double-density BOOT support for Model I** with embedded SOLE and FORMAT (SYSTEM). Supports mirror-image backup, too. Reworked FDUBL driver eliminates PDUBL and RDUBL and takes less memory; enhanced resident driver eliminates TWOSIDE.
- ☆ Model III version auto-detects Model 4 for installation of KI4 keyboard driver; supports CAPS, CTRL, and function keys.
- ☆ SYSTEM command supports removable and reusable BLINK, ALIVE, and UPDATE memory modules.
- ☆ The TED text editor now has commands to **print the entire text buffer**, or the contents of the first block encountered. Obtain directories from TED, too!
- ☆ The SPOOL command offers Pause, Resume, and Clear parameters. (OFF) attempts to reclaim memory used.
- ☆ **Alter the logical record length** of a file with "RESET filespec (LRL=n)"
- ☆ Specify "RESET filespec (DATE=OFF)" to restore a file's directory entry to the old-style dating of pre-5.3 release. Specify "RESET filespec (DATE=ON)" to establish a file's directory date as that of the **current system date and time.**
- ☆ Both Model I and Model III support similar commands: all features of Model III 5.3.0 are in Model I 5.3.1. That includes such facilities as DOS and BASIC help files, SETCOM and FORMS library commands, TED text editor, BASIC enhancements, etc. All DOS commands have been groomed for Model 4 LS-DOS 6.3.1 syntax where possible.
- ☆ Best of all, **a 5.3.1 diskette is available as a replacement for your 5.3.0 diskette for $15** (plus $3 S&H in US and Canada, $4 elsewhere). <u>There's no need to return your current master</u>.
- ☆ The 5.3.1 diskette(s) come(s) with a 30-day warranty; written customer support is available for 30 days from the purchase date. Versions for the Model I and Model III are available. **If you do not already have an LDOS 5.3.0, order the 5.3.1 Upgrade Kit with 30 days of customer support for $39.95** (+$4 S&H). Some features require lower case or DDEN adaptor.

LB Data Manager: for your data base needs

LB Version 2: A Flat File Data Manager with more powerful and easy to use features in this latest enhancement of Little Brother Data Manager!

We've added many features asked for over the past few years by LB users; yet LB is still just about the easiest, most flexible data manager you can use for managing your data. Absolutely no programming is needed to create a database with numerous fields, construct input screens for adding and editing data, and create your own customized report. Quickly you define your data fields in response to LB's prompts, and then draw your data input screen using simple keystrokes. In no time at all, you're entering data. Customize your printed reports with user-definable print screen definitions. LB is just what you need in a data manager!

- ☆ **Data capacity:** up to 65,534 records per data base; 1,024 chars (64 fields) per record; and up to 254 characters per field.
- ☆ **Field types:** ten field types for flexibility: alphabetic, calculated, date last modified, dollar , floating point , literal , numeric right-justified numeric, upper case alphabetic, and upper case literal.
- ☆ **Data entry:** ten input screens; selectively protected fields; selectively required data entry. Global search and replace with wild-card character match and source string substitution.
- ☆ **Selection and sorting:** Maintain ten index files. Select records using: EQ, NE, GT, GE, LT, and LE; on up to eight fields with AND and OR connectives.
- ☆ **Automatic operation:** LB can be run in an *automatic* mode, without operator intervention. Frequently used procedures can be saved by LB's built-in macro recorder for future use.
- ☆ **Report generation:** is customized through print formats which you define on screen. Truncate field data or strip trailing spaces; control where you want each field to appear; provide headers and footers. Reports may be sent to a printer, the display screen, or to a disk file. Formatting allows for multiple across mailing labels, multiple copies of the same record, or even printing one record per page for sales books. Generate mail/merge files of address or other data for your word processor.

We provide two utility programs for managing your database, a shell to DOS so you can perform other DOS commands, on-line help available, and a 200-page User Manual.

Specify MS-DOS or TRS-80 Model 4 version. LB is priced at $99 + $5 S&H (US; $6 Canada; $7 Europe; $9 Asia, Pacific Rim, and Australia). To upgrade from version 1.0, send Table of Contents page and $40 +S&H. Remit to:

MISOSYS, Inc.
PO Box 239
Sterling, VA 22170
703-450-4181