TRSTimes

Volume 3. No. 1. - Jan/Feb 1990 - $4.00

Follow us in 1990!
Our third year of exclusive coverage
for TRS-80 Models I/III & 4

LITTLE ORPHAN EIGHTY

Welcome to the first installment of the 1990 edition of TRSTimes. As you can see, we have taken on a new look, more magazine style. I hope you like it.

I wanted to use this format from our very first issue, but had neither the knowledge, equipment nor the money to do it. However, as our subscriber base grew, we paid off the initial startup loan and began investing in better equipment. Last year brought the HP series II laser printer; this year we blew the profits on a 2-megabyte memory upgrade to the laser and some specialized software.

Yes, TRSTimes made a profit in 1989. Oh, it was not the kind of profit you'd brag about in publishing circles. It was small - BUT - it was a profit. That is something the big guys from 80 Micro could not do (or would not be satisfied with), so they left us at the end of 1987. TRSTimes, on the other hand, is now starting its third year of publication, and we are satisfied with small profits. Heck, we'd been ecstatic just to break even.

The memory upgrade is mind-boggling. Can you imagine, my printer now has 20 times more memory than my 128K Model 4! It was much needed, though. Fonts, especially the ones of the large headline variety, take up an enormous amount of memory. The HP series II originally comes with 512K of memory, which is barely adequate for a page of text and medium sized headlines. Try to add a graphic or two, and the memory is used up. This limitation made it necessary to paste headlines and graphics with scotchtape, glue, or any other sticky substance readily available. Obviously, this was messy, prone to error, and took up a lot of time.

Now, with the added 2-megabytes, I can now do full page graphics, large headlines, and still have plenty of room for normal text. Just view the front page of this issue; it was done as one file, everything residing in the printer memory at one time. The headline is 96 point Americana, the graphic is large with a lot of black, and the remaining text is much larger than standard size text. It was wonderful to be able to do it all at once.

There are many other upgrades I would like to have, such as a full page text scanner, a full page graphics scanner, etc. Alas, those will have to wait for a while!

The exciting news in TRS-80 land, is that Roy Soltoff seems to be making a move to consolidate (and make available) software from vendors who are no longer in the TRS-80 market. Not only is his company (MISOSYS) producing great software for our machines, he just bought up the rights to the Powersoft software. He got it all, lock-stock and barrel, which includes the famous hard disk drivers, Super Utility, PowerMail Plus, the Powersoft hard disk utilities, Trscross, Backrest, and every one of the other goodies from this fine company. Roy has also managed to get license to the Cornsoft Groups fun games for Model III; that is: Frogger, Scarfman, Bouncezoids, Crazy Painter and Space Castle.

According to 'The MISOSYS Quarterly', Roy has made a deal with Randy Cook for MISOSYS to publish Double-Duty, and is looking to get the rights to other programs if the legal owners can be found.

Finally, Roy reports the following newsworthy item:

"It is probably safe to report that for many months now I have been discussing with Tandy Merchandising, a means to ensure that all software owned by Tandy Corporation that has ever been published for the TRS-80 Models III and 4 will continue to be made available in some method of distribution. I have a verbal acknowledgement that contrary to some folks opinion of Tandy, they do listen; the wheel has been turning."

A very interesting development and I wish Mr. Soltoff every possible success with this venture. Having software available again will certainly help to keep our machines churning for some time to come.

This issue brings quite a bit of variety. Charles Harris tells us how to interface the Model 100/102 with Multidos and Lazywriter. Andrew Bruns concludes his three part series on Multidos, this time focusing on the very powerful enhancements to Basic. Michael Ecker (of REC) gives us some more computer puzzles and games. The HINTS & TIPS section was fun to do, as it proves that fine minds are still working with the TRS-80. As you may recall, we issued a challenge to convert DIGITIME/BAS to the Model 4. Our four winners, David Goben, William Bowman, Lloyd Evans and Don Bergthold, provided the best (and most unique) answers to the problem. To all who sent a non-winning solution, we thank you very much. Try your hand at the next contest. Yes, we have one in the works, which will appear in an upcoming issue.

Dale Parsons shares TOP SECRET, a crytography gem for Model 4, with us. It is a toughie. I am told by a certain eastern-power leader that the code produced by this program was so tough that it made them not only climb the wall --- they tore it down. (*well!!*)

TRSDOS 1.3. Corner and Assembly 101 returns and, for the hi-res fans, Frank Slinkman provides a wealth of information in the response to the review of his programs by Allen Jacobs in the Nov/Dec 1989 issue of TRSTimes. Roy Beck tells of his trials and tribulations getting 8 different DOSes on his 35 meg hard drive. Yes, 8 different DOSes! He hasn't yet succeeded, but knowing his dedication, he will accomplish his goal, and we will be informed of his progress, blow-by-blow.

And now.....Welcome to TRSTimes 3.1.

TRSTimes magazine

Volume 3. No. 1. - Jan/Feb 1990

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

TRSTimes magazine is published bi-monthly by TRSTimes publications. 20311 Sherman Way, suite 221. Canoga Park, CA. 91306. U.S.A. (818) 716-7154.
Publication months are January, March, May, July, September and November.

1990 subscription rates (6 issues):
United States & Canada: $18.00 (U.S.)
All other countries: $23.00 (U.S.) for surface mail, or $29.00 (U.S.) for air mail.

Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible, a disk with the material saved as ASCII files. Any disk format is acceptable, but please note on disk label which format is used. Also, please write your name and address on both hardcopy and disk label.

LITTLE ORPHAN EIGHTY 2
Editorial

THE MAIL ROOM 4
Reader mail

MULTIDOS, LAZYWRITER AND MODEL 100/102 6
Charles Harris, MD

RECREATIONAL & EDUCATIONAL COMPUTING . . . 7
Michael W. Ecker, Ph.D.

A GUIDED TOUR THROUGH MULTIDOS - PT. 3 9
Andrew Bruns

HINTS & TIPS 12
Goben, Bowman, Evans, Bergthold, Wolstrup

TOP SECRET 15
Dale Parsons

TRSDOS 1.3. CORNER 17
Jacobs, Wolstrup

HI-RES MYSTERIES EXPLAINED 21
Frank Slinkman

A LITTLE HARD DISK PROBLEM 25
Roy Beck

THE SWAP MEET 27
Classifieds

ASSEMBLY 101 28
Lance Wolstrup

THE MAIL ROOM

MORE TECHNICAL ARTICLES

I subscribe to both TRSTimes and CN80 and appreciate their different approaches. So far TRSTimes has tended to go into more detail about how a program works and ways of adapting it, as well as some technical details of the guts of operating systems and of Basic. This, rather than games, is what I am looking for. The series of articles "LDOS: How it works: and "Roy's technical corner" in the old LDOS Quarterly were just about my ideal of a useful and absorbing article. Anything of the sort would be most welcome. In the current issue (Vol.2. #6) "Peeking and Poking Model III" was my first choice, but when will people start using hex for addresses? The decimal system just does not make sense for non-numerical information.

John Shepherd
Mountainhome, PA.

We will do our best to present more articles catering to the tecnical crowd. However, as we have readers with different degrees of expertise and interests, we will also bring games and other fun things. From the very beginning, the focus of TRSTimes has always been geared to the hobbyist. As for hex versus decimal, I think that most of us use a mixture of both. Certainly, even when writing Assembly language, I will use hex when convenient, and decimal when that is easier.

Ed.

MORE GENEOLOGY

Another good geneology program aid is a a PD program, available through THE FILE CABINET, called FAMILY/BAS. I like CLAN better as it contains charts that can be printed out - - but it does have some errors that I haven't been able to correct yet. FAMILY/BAS can be found on File Cabinet Disk MD4 BUS 14.

Also, LOANCALC/BAS line 70 has a syntax error in it, and I don't know how to correct it. This program was published in ther most recent TRSTimes 2.6. Nov/Dec 1989 Page 18. Can you help me out?

Robert D. Lutes
Mountain Home, AR

Thanks for the FAMILY/BAS info. Line 70 of LOANCALC/BAS is correct. However, do keep in mind that the line is one contiguous line, rather than, because of our two-column format, the listed three split lines. Also, make sure that the separators are semicolons and not colons. As a last resource, retype the line as Basic will on rare occasions store erroneous keystrokes invisibly

By strange coincidence, both John and Robert live in towns with the same rather uncommon name. Yes, I did check the subscriber database to make sure I had not fouled up.

Ed.

USERS GROUP

I am interested in a users group. If you make up a list of names, please include mine. I have a Model III.

Karen Redding
306 Robinall Dr.
Easley, SC. 29640
(803) 855-1645

REVERSE VIDEO

I loved that inverse video for Mod 4 in 3 mode (Hidden Video Fun by Donald G. Shelton. TRSTimes 2.3.)

Be prepared for the return of 'NX' ('NX2')? Keep up the great magazine!!!

Danny C. Mullen
APO Miami, FL.

Thanks for the kind words, and we look forward to 'Son of NX'.

Ed.

BACKUP4

Thank you for the continuation of your fine publication, TRSTimes. Sometimes you run across a program that is just great. One such program I obtained sometime ago from 80 Micro called "BACKUP4". It has performed very well. Other backup programs are OK, but they hang when they encounter a bad sector on the disk, and you have to reboot the system and start over, which can be a real pain if you only have two disk drives. Your program simply locks out that sector and continues with the copying. Let me say again, A JOB WELL DONE.

Lenox C. Smith
Hamburg, IL.

BACKUP4 was on the 'UNPUBLISHED GEMS' disk from 80 Micro, the last disk they published (is there a message there?) I am glad you find the program useful, however, I must make one correction. BACKUP4 does not lock out sectors, as this would cause loss of data; instead it goes back and reformats the destination track, reads the

source track, and writes the data to the destination track. If an error is encountered during any of these three steps, the program will retry the process. It will repeat a maximum of five times (or is it 6, I can't remember!). If the track copy at this point is still bad, an error message is displayed and you are returned to the source and destination prompts.
Ed.

BUG REPORT

Hate to mention this but my "Hasher" program developed a bug some where between here and printing the Aug/Sep issue of TRSTimes. Lines 130 and 140 should read:

```
130 IF L>15 THEN L=L-16:R=R+1
140 IF R>15 THEN R=R-16:L=L+1
```

Can't wait for the Model 4 "101" series. Believe I can translate EDTASM verbiage to MISOSYS EDAS. I have the Model 4 for free article but never had the 3 assembler. A request please, can you perhaps add a description of where and how TRSDOS handles keyboard input. I have "The Source" and can identify the prompts but what happens next escapes me.
Dale Parsons
Cross Lanes, WV.

Sorry about the bugs - definitely our fault. After three unsuccessful versions of a translation program, an equally worthless attempt to patch 'Ventura Publishing' to use other characters than < and > for its internal use, I gave up. I am now inserting the deleted characters manually.

TRSDOS does some very tricky things in the keyboard routine. I will go through 'The Source' to see if I can make it out. If so, I will present an article in an upcoming issue.
Ed.

DEBUG

While reviewing the "Passwords for Model 4" article in the latest issue (Vol. 2.6.) if TRSTimes a thought occured to me on which I am sure you could provide enlightenment. It seems that it is always suggested to use a ZAP program to look at directory entries and/or change hash codes. Is it not possible to use the TRSDOS6/LS-DOS6 DEBUG utility to accomplish this? If so, I would appreciate seeing a detailed article in TRSTimes illustrating how this can be accomplished. I have done some experimenting with the DEBUG utility for this, as well as other purposes, but have not always been successful.

I am most grateful that TRSTimes will continue for another year, as the articles contained therein have enabled and facilitated Model 4 programming feats that I could not have accomplished otherwise.
Elton L. Wood
Morgan, UT.

It is, indeed, possible to use DEBUG as a disk-zapper. To answer your question, we included a short article about the subject in the HINTS & TIPS section in this issue.
Ed.

COPYRIGHT

I have a question maybe you can answer for me. Where does one draw the line on copyrighted material? For example, you're writing a program and you come to a place where a routine is needed to do a math function. Now, you've seen in lots of other programs how it was done and you put that routine into your program (something like A=B:C=A+B). Now somewhere I know this is in somebody's copyrighted program. If the same technique is used in a different program, doing something else, is it infringement on copyrights? After all, isn't 2+2=4 the same to everyone? This may be an over simplification, but I hope you get what I am driving at. Can one use modules or routines from copyrighted material in their own programs without stepping on somebody's toes? I'm really not clear on this.
Don Bergthold
Fort Madison, IA.

Not being an attorney, I cannot give you exact answers. My personal view, for whatever that is worth, is that the copyright of a computer program covers the concept AND code AS A WHOLE. I believe, if you take the concept of a program and write it with enough of your own original code, OR take portions of code from a program and use it to create a reasonably different concept, you may be stepping on toes, but you are not violating copyrights. If this is not correct, we invite our readers to set us straight?
Ed.

WHAT'S WITH MULTIDOS 80/64 v2.1?

The directory on my working MULTIDOS 80/64 v2.1. crashed, and after I backed up the master onto a blank I found that the **TAB(#)** in Basic LPRINT has become relative, as if it were **IND(#)**. I tried again, using the master disk, and it does the same thing, so the glitch is there.

Printout of my datafiles with this condition is impossible. I would guess that, since it is only LPRINT, there is some sort of incorrect instruction being sent to the printer. I use 'down-arrow' i.e. linefeeds, in my Basic programs in order to put many statements on a line #, but the linefeeds DO NOT LLIST. The video display for TAB(#) and linefeeds is perfect. I am using a C.ITOH Model M-8510 that performs correctly on 1.71, DOSPLUS & TRSDOS 6.2. Could something be wrong in the codes that are sent to the printer (FORMS, PRT)? If someone has a patch for this, please let me know.
Jim King
Topanga, CA

MULTIDOS, LAZYWRITER and MODEL 100/102

By Charles Harris, MD.

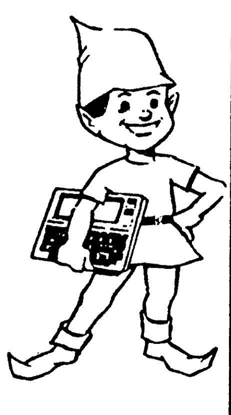
The article on Multidos in TRS Times (September 1989) should be amplified to let the Model 3/4 crowd know the unique advantages of using the Model 100/102 in conjunction with the Model 3/4 and a wordprocessor called Lazywriter on Multidos.

Lazywriter, written by David Welsh, became one of the most popular word processors for the Model III. Multidos was written by Vernon Hester to take advantage of the Model 4 utilities, and still be compatible and be able to read Model 3 DOSes. Both programs were sold by Alpha-Bit Communications out of Dearborn Michigan. Thus, Welsh adapted Lazywriter to Multidos, and wrote a 3/4 version which could be used either with TRSDOS 1.3, MULTIDOS or TRSDOS 6.X/LSDOS 6.3.

Lazywriter is a magnificent word processor, run with simple toggles. It is compact, easy to use and facile. It can roam through a text for editing with greater facility than many of the more "advanced" and prolific word processors including the klutzy ones on MSDOS, that do everything but make your bed. The Multidos version of Lazywriter allows the user to change printer fonts and create his own stationery. The printer menu is direct and easy, and permits overriding soft margin commands. You can do anything with Lazywriter except multiple columns.

But the great advantage of Lazywriter is that it interacts with the Model 100/102. David Welsh wrote a very simple COMM module into the Multidos version which permits Lazywriter to tap into the COMM file of the Model III.

Those fortunate enough to own a Model 100/102 are familiar with both the TEXT and TELCOM files, the first for writing, and the second for communications. Thus by connecting the Model 3/4 to the Model 100/102 with a null-modem cable, communications between the two is a cinch.



TEXT in the Model 100/102 is convenient, but includes no formatting commands. When uploaded into Lazywriter via the null-modem communication system, the file is formatted with the appropriate terminators. The only thing to be added is that paragraphs must be demarcated by either indenting or skipping lines.

Here's how it's done. When in Multidos Lazywriter, go into the Utility menu <F1> and press <E>.

Write the word <COMM>.

The screen will clear, and the program can be made ready to receive <UP ARROW> or send <RIGHT ARROW>.

If the <UP ARROW> is pressed, a graphic block appears at the left of the screen and Lazywriter is ready to receive at 300 Baud.

Meanwhile to upload a file from Model 100/102 the sender enters TELCOM on the Model 100/102, puts in the correct stat, in this case 38N1D, and then the name of the file to upload (following Model 100/102 TELCOM instructions. If done correctly, your file will flows across the Model 3/4 screen in ASCII. When finished press <SHIFT> <@>, and the file is in Lazywriter, formatted except for the indented paragraphs.

Thus the Model 100/102 becomes a traveling typewriter, whih in conjunction with Multidos and Lazywriter creates a terrific partnership that simplifies the life of the writer and liberates him or her from the confines of the computer desk.

Multidos has another great advantage over other Model 3/4 DOSes, an extremely simple MEMDISK. All you have to do to create this is write <MEMDISK> and the system is installed. Then the VFU utility can be used to create an IDO file (the BUILD and DO files of LSDOS).

It just happens that Lazywriter is compact enough to fit into the Multidos Memdisk, if BASIC and other unneeded files are eliminated. This can be done using VFU after the Memdisk is installed. Then the LW can be installed with an AUTO commando followed by the IDO file which erases the superfluous Multidos files and installs Lazywriter.

Once done, LW works at the speed of a ram disk. A small calc program called SUMUP can also be installed with Lazywriter.

Milliken has a marvelous shell for the Model 4 which, unfortunately, does not work with Multidos. However, I can use SUPER UTILITY to copy Multidos files to the Model 4 Lazywriter (LS-DOS), then use the shell to make a description of each file, which becomes equivalent to an index, when used with the Lazywriter FIND function.

The combination of Multidos, Lazywriter and Models 100/102 is quite terrific, and I can safely advocate it for any serious writers who wants to enjoy the wizardry of word processing with the convenience of a portable typewriter which can save files to the infinite memory of Multidos formatted diskettes. This interaction is well worth exploring and enjoying.

REC
RECREATIONAL & EDUCATIONAL COMPUTING

Greetings again, everybody! We've got a lot of goodies this issue, so let's jump right in with these brain-teasers. Can you solve these with a program? Straight math?

Teaser No. 1): Suppose there were a million people to whom a government distributed money as follows:
$1 to first person, $2 to each of the next two people, $3 to each of the next three, and so on. How much does the millionth person get?

Teaser No. 2): There are three crates of fruit, but the labels have been switched around so each crate bears an incorrect label. One is labeled oranges, another apples, and apple & oranges for the third. Tell how to identify the contents solely by picking only one piece of fruit from the crates and from no other cues (such as peeking, weighing, shaking, etc.).

Teaser No. 3): If [k434k0] (positional notation) is divisible by 36, then k=?

We also have some correspondence.

Query from Jorge Ferry, N. Miami, FL: How do you round numbers to, say, the nearest 50?

Reply: For a variable E, in BASIC (the language Mr. Ferry uses), try E=50*INT(E/50+.5)

What this does essentially is find out how many 50s are in E from the E/50, and then, by adding a .5 and taking the greatest integer in the result, we round this up or down. Now we have the appropriate closer number of 50s, so we multiply by 50 to get the actual nearest multiple of 50. (This will extend to other multiples. Just replace 50 in all places by whatever else you wish.)

Crypto-Calc

Program by Ray McClanahan and Mike Ecker

In the game of Krypto, five cards are dealt out, along with a target value. The object is to use just the operations of arithmetic and each of the five cards exactly once so as to achieve the designated target value.

The card game is fun to play mentally. However, Ray McClanahan has written a program in several BASICs (including PC, TRS-80, including Model 100, and Commodore) to implement the game. Here is a version that I've edited and touched up a tad.

Ray's commentary on the topic included discussion of the number of operational possibilities the program will check as a function of the number of cards. I would just invite readers to beat the computer. You will find, given the intuitive capabilities you have as a bright human, that you may beat the machine in many cases. Comments on the programming and improvements are invited, therefore.

```
10 CLS: PRINT: PRINT "Crypto-Calc, copyright 1989-90, Ray
McClanahan
20 PRINT: PRINT "and Dr. Mike Ecker / Recreational Mathe-
magical Software / REC
30 FOR DL=1 TO 2000: NEXT: PRINT: PRINT STRING$(80,"*")
40 INPUT "Would you like instructions (Y for yes, anything else
for no)"; INS$
50 IF INS$="Y" OR INS$="y" THEN GOSUB 780
60 PRINT: INPUT "How many cards (default=5)";C:W=C-1
70 IF C<=0 THEN C=5: W=4
80 INPUT "Target value";V
90 F(1)=1:FOR A = 2 TO C:F(A)=A*F(A-1):NEXT A
100 INPUT "Who will pick the cards? You (press Y & ENTER) or
computer (default)"; PIK$
110 PRINT: IF PIK$<>"Y" AND PIK$<>"y" THEN 870
120 FOR A = 1 TO C:PRINT"card ";A:INPUT I(A):NEXT A: PRINT
130 FOR U = 1 TO F(C):D=C:J=U:GOSUB 480
140 FOR A = 1 TO C:M(A)=I(R(A)):NEXT A
150 FOR E = 1 TO F(W):D=W:J=E:GOSUB 480
160 FOR A = 1 TO W:Q(A)=R(A):NEXT A
170 FOR A = 1 TO W:Z(A)=1:NEXT A
180 FOR A = 1 TO C:N(A)=M(A):NEXT A
190 FOR A = 1 TO W:P(A)=Q(A):S(A)=Z(A):NEXT A
200 FOR A = 1 TO W:L=C+1-A
210 ON S(P(A)) GOTO 220,230,240,250
220   N(P(A))=N(P(A))+N(P(A)+1):GOTO  270
230   N(P(A))=N(P(A))-N(P(A)+1):GOTO  270
240   N(P(A))=N(P(A))*N(P(A)+1):GOTO  270
250 IF N(P(A)+1)=0 THEN 390
260   N(P(A))=N(P(A))/N(P(A)+1)
270 IF A=W THEN 370
280 IF P(A) = L-1 THEN 330
290 FOR B = P(A)+1 TO L-1
300  N(B)=N(B+1):NEXT B
310 FOR B = P(A) TO W-1
320  S(B)=S(B+1):NEXT B
330 FOR B = A+1 TO W
340 IF P(A)>P(B) THEN 360
350 P(B) = P(B) - 1
360 NEXT B
370 NEXT A
380 IF ABS (V-N(1))< .001 THEN 580
390 FOR A = W TO 1 STEP -1
400 IF Z(A) = 4 THEN 420
410 Z(A) = Z(A) + 1:GOTO 470
420 Z(A)=1
430 NEXT A
440 PRINT "no luck with operation order permutation number
```

```
";E:NEXT E
450 PRINT "no luck with card order permutation number
";U:NEXT U
460 PRINT "all possibilities checked - no match":STOP
470 GOTO 180
480 FOR H = 1 TO D:T(H)=H:NEXT H
490 FOR X = 1 TO D-1
500 Y = D-X
510 G = INT ((J-1)/F(Y)+.00001)
520 R(X) = T(G+1)
530 J = J - G*F(Y)
540 IF G = Y THEN 560
550 FOR K = G+1 TO D-1:T(K) = T(K+1):NEXT K
560 NEXT X
570 R(D) = T(1):RETURN
580 FOR A = 1 TO C:O$(A) = STR$(M(A)):NEXT A
590 FOR A = 1 TO W:P(A)=Q(A):S(A)=Z(A):NEXT A
600 FOR A = 1 TO W:L = C+1-A
610 ON S(P(A)) GOTO 620,630,640,650
620 O$(P(A)) = "(" + O$(P(A)) + " +" + O$(P(A)+1)+
")":GOTO 660
630 O$(P(A)) = "(" + O$(P(A)) + " -" + O$(P(A)+1)+ ")":GOTO
660
640 O$(P(A)) = "(" + O$(P(A)) + " *" + O$(P(A)+1)+ ")":GOTO
660
650 O$(P(A)) = "(" + O$(P(A)) + " /" + O$(P(A)+1)+ ")"
660 IF A = W THEN 760
670 IF P(A) = L-1 THEN 720
680 FOR B = P(A)+1 TO L-1
690 O$(B) = O$(B+1):NEXT B
700 FOR B = P(A) TO W-1
710 S(B) = S(B+1):NEXT B
720 FOR B = A+1 TO W
730 IF P(A)>P(B) THEN 750
740 P(B) = P(B) - 1
750 NEXT B
760 NEXT A
770 PRINT O$(1):END
780 CLS: PRINT "Crypto-Calc is modeled after a card game
called Krypto, put out"
790 PRINT "by MPH Games Co. The object of the game is to
achieve a target value"
800 PRINT "using only the other cards dealt out and +,-,*, /
(operations)."
810 PRINT: PRINT "In this game, you choose how many cards,
the value of each,"
820 PRINT "and the target value. The program searches for
solutions... "
830 PRINT: PRINT "And now, get ready to test your computer's
mettle..."
840 PRINT: INPUT "Touch <ENTER> to begin..."; X$: PRINT
850 RETURN
860 END
870 TT=10000*RND MOD 2 + 1
880 FOR A=1 TO C:
I(A)=1+(INT(VAL(RIGHT$(TIME$,TT)))*250*RND    MOD    25)
890 IF I(A)=I(A-1) THEN I(A)=I(A)+4 MOD 25
900 FOR DL=1 TO 1000*RND
910 NEXT DL
920 PRINT "Card "; A; "is "; I(A)
930 NEXT A
940 PRINT: GOTO 130
```

Krypto-Teaser: Given the five cards 17, 3, 23, 10, 2, can you achieve a target value of 1? It took me perhaps a half-minute, which easily beat the program. Can you do even better?

If you'd like a disk with the program, send $4 to the address below. Specify your computer brand.

Next column we'll have one of my favorite tricks, what I call the **Triangle Number Trick**.

Do you like this column? It was excerpted and mildly edited from issue #27 of my own publication, REC. If you wish to subscribe, you may use the form below (or better, a photocopy).

Reminder: If you would like an acknowledgement or reply to a question, please enclose a self- addressed, stamped envelope (SASE). (The address appears below.) Many thanks!

REC Subscription

- ☐ Dear Mike, I like your column here in TRS Times. Please enter my subscription for your newsletter for 1990, eight issues; $24 in USA, $25 to Canada, $34 abroad.
- ☐ Dear Mike, Let me try a trial sub instead, three recent or new issues for $7.50, $7 of which is creditable later towards a regular subscription.
- ☐ Dear Mike, Please send more information. SASE enclosed.

Rather than mutilate this issue, just send your courtesy title, name, and full address with payment in US funds drawn on a US bank. Send it and/or inquiries, comments, solutions to:

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

Until next time, happy recreational computing!

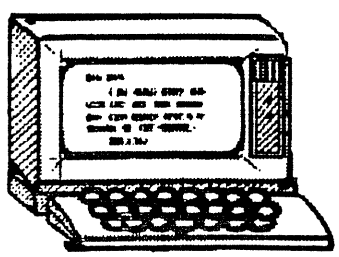
Dr. Michael W. Ecker is a mathematics professor and well-known computer columnist / reviewer.

Recreational & Educational Computing is a personal, two-way communication of challenging "mathemagic" and mathematics, recreation, education, and computer activity between subscribers and Dr. Ecker, assisted by contributors.
REC is published 8 issues per year at 129 Carol Drive/ Clarks Summit, PA 18411. Editor's chat line: (717) 586-2784.
Annual subscriptions by calendar-year, $24 in U.S. and Mexico. (All payments in US funds on a US bank: $25 to Canada, $34 abroad.)
Back issues: Vol. 1 (1986), $20; Vol. 2 (1987), $20; Vol. 3 (1988), $24; higher abroad.

A guided tour through MULTIDOS

by Andrew J. Bruns

Part 3: SuperBasic and other Goodies



SuperBasic is the Basic interpreter supplied with MULTIDOS 2.1. and is a fine product for both novice and experienced user.

The number and type of I/O buffers can be set when loading and protected memory can also be specified for machine language code so you can access machine code routines already loaded. For we lazy people, the defaults make it all automatic.

Those who are using tape systems or are running games from tapes that require level II basic, can drop from SuperBasic into Level II to run these programs and come back to the Disk basic level with a single command when you are finished.

For a lot of enhanced debugging functions there is a slightly bigger version of SuperBasic called BBasic. With this version you can trace program operation and use the "@" key as a control key to activate special tracing functions. You can also single step line-by-line or even instruction-by-instruction and you can inject delays in the trace so you can see what happens at a slower speed. There is also a powerful variables review capability while running so you can review and/or modify variables and continue to run the program to see what happens.

If you enjoy little gems, there is a Basic program "stacking" function which allows you to stack Basic routines in high memory as independent modules and then merge them with the code you are developing any time you want. This works like a MEMDISK and is limited only by the free memory and your ingenuity. It is faster than a MEMDISK, however, in that a couple keystrokes are all that is necessary to make a stored Basic module part of your current code.

There are a number of single stroke functions to help you work on that Basic program as well. They include listing or editing the current line, listing the previous or next line, and moving to the first or last program line. If your program aborts, you can force the listing of the offending line with just two key strokes, no typing required.

In addition to these functions, there are also several single letter function keys for routine basic functions like AUTO, CONT, DELETE, LIST, LOAD, KILL and EDIT.

Now for the good part! The user can move lines and duplicate lines to anywhere in the program and you can page the listing of a program from any line. A lot of Basics won't support these capabilities.

There are also a number of commands that are included in SuperBasic program coding to simplify the programming task, many associated with arrays.

You can zero numeric arrays or sort string arrays in 1 or 2 dimensions and you can also delete an array and then reDIMension it.

For the adventurous there are functions which are used to compress your code by removing spaces and linefeeds, and Basic programs can be packed to improve speed. While normal Basic limits lines to 255 characters, this command will allow you to pack those lines to more than 1000 characters which speeds execution. You can also just pack a portion of the program that is exceptionally slow for you,speeding up that area, or pack several different routines.

There is a function here that shortens the code by removing REMark statements too. By eliminating non-executable code the interpreter has to wade through, your program will certainly run much faster. When deleting remarks, you do need to make sure that your GOTOs are not referencing remark lines since this function won't compensate for that. It will tell you, however, if there are open GOTOs as a result of REMARK deletion (very handy).

The best way to avoid this problem, though, is to program without using REMark statements as the destination of GOTOs or GOSUBs. I'd recommend that, if you use this compressing function and remove the comments, that you keep a copy with the comments for your documentation and then create a compressed version as your runtime program.

For documentation or debugging, there is also an unpacker that maximizes the number of lines to simplify following the logic of a program. This function can actually break your program down to the point where each instruction has it's own line. If you have an obtuse section of a program, this can sometimes make a poor logic loop visible when it may escape your view in a larger line or section where the code is packed or combined.

The mathematically inclined are given a DEF FN routine which allows defining a math function for use later in the program. This can be any function you chose to define and allows the transfer of variables into the routine and a result will then be recovered to the basic program that called it. This works like a basic routine that is used as a USR.

As before, defining machine code entry points with DEFUSR is fully supported, but if you are into Basic, the

DEF FN capability lets you create functions without learning assembly language to do it.

For we string manipulators, there is some extra power here too. In addition to finding, splitting and extracting strings as before, it is now possible to replace portions of strings without recreating a new string from components with a function called MID$.

And if you need the time, the TIME$ function returns a 17 byte string of both the date and time in memory. This is an exact copy of the new time stamp and can be useful for dating reports, memos, listings and other output. And while the DOS function CLOCK will display the time, this function can be used inside a program to display both time and date.

That would seem like a lot of power and I would be content right here, but that isn't all folks. SuperBasic also boasts a Global Editing function GEDIT. This powerful function will allow the programmer to change all or part of any variable names, constants, data list items or strings. Graphic codes, as CHR$(x), can be turned into packed strings and space compression codes, as CHR$(x), into packed strings.

There is also a function to merge adjacent line numbers into one long line or, conversely, split a long line into two shorter ones. And, of course, you can also exchange any KEYWORD throughout the program with any other. That means if you want to change all the "PRINT" words for "LPRINT" everywhere they occur in the program, it can be done quickly and easily with GEDIT.

"What about renumbering the program lines?", you ask. No problem! You can open up or close down the number sequence of program lines in a portion or all of the program in memory. So while the best way to number a basic program for speed is to start with 0 or 1 and sequence through single digit increases, most of us number lines in increments of ten or more. This is so we can add some missing function or instruction later if we have the need (do any of us hackers ever do this?) without having to renumber the whole program. Now you can create your program numbering anyway you want and then, with a couple keystrokes, renumber the whole program in the most compact line numbering sequence.

If you find a problem in a line and have to add a new line between 12 and 13, no hassle. Just renumber the program with a 5 line increment and you have created a 5 line cavern where you need it. When you are done, just renumber with single increments and Voila! What is real nice,too, is that this function accounts for all control transfer points and when lines are renumbered, the GOTOs and GOSUBs all have their line number references changed in the process as well (how convenient).

A new function has been added to SuperBasic that will delight Basic programmers. "NAME" is a function that allows you to chain Basic programs without loosing variable data from one program segment to another. This allows very long programs to be split and executed in modules on your Tandy without the storage and reloading of variable data. Previously, this was only possible with machine language programs!

As before, the disk functions LOAD, RUN, SAVE and KILL are available and Basic programs are mergable in memory and , as with all the disk basics, the normal file access commands are available for handling both sequential and Random files.

If you should run into any Disk errors while in Basic, then use the CMD"E" function to get a listing of the disk error without leaving Basic. This provides the user with ALL the error messages in Basic and DOS from within Basic, a real frustration controller for me.

The creator of this program has also done a fine job of creating and sharing ZAPS that help the user customize MULTIDOS 2.1.

There are ZAPS to help you set speed at bootup automatically for speed modifications you have installed and unique zaps for allowing unusual filenames.

A number of ZAPS are discussed to increase disk access performance by altering head load delays, the number of disk I/O tries and improve BACKUP/CMD speed by changing the verify function to "off" or "only on if DOS verify is on".

You can also alter the interleave with double density disks, especially with the model 4! In any case, a CPU clock speed greater than 2.5 Mhz can handle an interleave of 2 under MULTIDOS instead of the typical value of 3. This speeds up disk access but only works with MULTIDOS !!

While SuperBasic sets the default number of buffer files to 3, you can always change this to a smaller or larger number. This too can help disk file loading time if you're clever.

For those who have been frustrated by the incompatibility of VisiCalc's print command in MULTIDOS 1.7 or later, the simple fix is available and included in MULTIDOS 2.1.

And for those wanting to read and write NEWDOS80 diskettes and still have them work with NEWDOS80, there are some fixes available.

This release continues to support a "one byte to any port" transmission capability on bootup. By providing the entry points for model I/III speed mods, Holmes boards and the model 4 in 3 mode, it is possible to address clock speeds in any model by this direct ZAP method. (I said this DOS was uniform from machine to machine!)

As mentioned earlier, the focus for this DOS is on model I/III/IV compatibility and uniform user interface for normal functions between the machines with an applications emphasis on Basic language well supported for programmer and user alike. The "MULTI" part of the name has historically stood for being able to read multiple DOS formats on disks and this capability is still maintained. It does everything well, but as with other DOSes, there are a few applications programs out there that try to clobber the DOS kernel resident in memory. While applications can be written to be DOS tolerant, most programmers write for a specific DOS and what comes out is modified for whatever DOSes the programmer has available in his shop. When Model I's first came out there were 10 or more systems, but now there are only a few true survivors: MULTIDOS, LDOS (LS-DOS),TRSDOS and DOSPLUS. Only the first two are still supported to my knowledge.

If you used older versions, MULTIDOS used to have the smallest Basic/DOS kernel of any of the DOSes, which gave you an extra 2-4K of memory to hold and run Basic programs. MULTIDOS 2.1. uses this extra memory to expand its built in capabilities, but, now with the incorporation of the new NAME function, use of this extra memory just isn't critical. And compared to other DOSes, there is no difference in storage capability at all.

One thing non-MULTIDOS users are going to hear when they switch to this DOS is the change in disk drive operation. When a file is accessed, it sounds like the heads are cycled through the cylinders twice for each new access. This was a change in release 1.7 and has continued into version 2.1. Part of this may be due to the need to be able to adjust for foreign diskette reads as well as being able to access a file. You also get to put the directory on any cylinder/track you want so during an access, the DOS has to first, find the directory, and then locate and access the file. I'm not sure of this function, but it doesn't really seem to affect the drives on my machine (they have been doing this for years now and I'm still running the original drive 0)!

On the positive side of things, LDOS users will be pleasantly surprised at the fact that MULTIDOS doesn't hang up on drives running close to a 300 rpm spin, especially Model I users. (We know who we are, but nobody else does, I guess)

MULTIDOS is commercially available from Hypersoft which continues to support the Models I/III/IV with software and compatibility utilities for TRS-80/PC transfers. They can be called on their order phone (919) 847-4779 or write to: PO Box 51155, Raleigh, NC 27609. They have ads in TRSTIMES and COMPUTER NEWS 80. These are the trusted folks who have brought you Hyperzap, the Super disk repair/analyze diskette utilities system and Hypercross, the file copy utility to move files from TRS-80 to other systems including CP/M, MS-DOS and others, but that is really another story.

MULTIDOS 2.1 is supplied for models I/III at $79 and for the Model 4(3), there is the 64/80 version for $89. Quite a reasonable price for a state of the art Disk Operating System.

Andrew Bruns can be reached at: RR#1, Box 226, Marble Hill Rd. Great Meadows, NJ. 07883.

TRS-80 PUBLIC DOMAIN SOFTWARE BONANZA

We have bought collections of software from people leaving the TRS-80 world. As fast as we can, we are weeding out the good Public Domain and Shareware from the Commercial programs and the junk. So far, we have come up with 6 disks for the Model I & III, and 3 disks for the Model 4.

Model I & III.

PD#1: binclock/cmd, binclock/doc, checker/bas, checker/doc, chomper/bas, cls/cmd, dduty3/cmd, dduty3/doc, 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, diskrnam/bas, menu/cmd, menu/doc, ripper3/bas, sky2/bas, sky2/his, space/cmd, stocks/bas, trs13pat/bas, vidsheet/bas

Model 4

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

M4GOODIES#2: arc4/cmd, arc4/doc, cia/bas, etimer/cmd, index/cmd, index/dat, mail/bas, mail/txt, trscat/cmd, trscat/txt, util4/cmd, xt4/cmd, xt4/dat, xt4help/dat

M4GOODIES#3: convbase/bas, dates/bas, dctdsp/cmd, dmu/cmd, dmu/doc, dskcat5/cmd, dskcat5/doc, editor/cmd, editor/doc, fedit/cmd, fkey/asm, fkey/cmd, fkey/doc, hangman/cmd, m/cmd, m/src, membrane/bas, miniop2/cmd, miniop2/src, move/cmd, move/doc, otello4/bas, scroll4/cmd, scroll4/src, setdate6/cmd, setdate6/doc, setdate6/fix, spaceadv/bas, taxman/bas, utilbill/bas, utilbill/doc

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

TRSTimes PD-DISKS
20311 Sherman Way. Suite 221.
Canoga Park, CA. 91306

ZAPPING WITH DEBUG

By Lance Wolstrup

Model 4 DEBUG is a very powerful (and little understood) utility. It is usually employed to trace assembly language program flow, but it is also possible to use it as a disk-zapper. Using it in such a manner you must be extremely careful. This short article will give a step-by-step description on how to use DEBUG to 'ZAP' the passwords from a file; that is, removing a file's password protection.

First, to avoid possible dissaster, make a backup of the disk you are going to work with. Now, assuming you wish to strip SYS6/SYS of its password follow these directions explicitly:

Place a system disk in drive :0. Place the disk (the backup) you are going to zap in drive :1.

Enter DEBUG by typing **DEBUG** <ENTER>. Then press <BREAK>.

You now have the DEBUG display, so carefully, type:

1,14,2,R,3000,1 <ENTER>

You have told DEBUG that you want drive :1, track 14H (20 - directory track), sector 2, the R means READ, use memory locations starting at 3000H, and read just 1 sector. Now, you'll see a full screen display of memory starting at 3000H. It is a display of drive :1, track 14H, sector 2. If you look to the top right of the display, you'll recognize the entry for BOOT/SYS, followed by the entry for SYS6/SYS.

We now enter the second stage of our quest, modifying first the protection status of SYS6/SYS, then the actual password.

The protection status of SYS6/SYS is now located at memory location 3020H. It says: 5F. It needs to be changed to 10, which is the status of an unprotected, visible, non-system file.

Type: **H3020** <spacebar>

Make sure you press the spacebar and NOT the enter key. The highlighted bars will surround the 5F, and at the bottom left of the screen you'll see:

```
3020
5F -
```

You can now change the protection status by typing:
10 <ENTER>

Now we need to change the actual password. The password for SYS6/SYS is now located at 3030H and 3031H in memory, so type:

H3030 <spacebar>

Again, the highlighted bars will surround the displayed contents of the chosen memory location (3030H), which is F6. (3031H should be 37). Also, at the bottom left, the display should now be:

```
3030
F6 -
```

At this point type: **96** <spacebar>

By hitting the spacebar, instead of the enter key, you moved the highlighted bars to 3031H. The bars now surround 37, and the bottom left of the screen displays:

```
3031
37 -
```

Now type: **42** <ENTER>

So far, so good. We have changed the protection status of the file to 'no protection', and we have removed the password - IN MEMORY. In the last leg of our journey, we need to write this portion of memory back to drive :1, track 14H, sector 2.

CAREFULLY, type this command:

1,14,2,*,3000,1 <ENTER>

The command told DEBUG to write one 256 byte segment, starting at memory location 3000H, to drive :1, track 14H, sector 2. The write command, in this case, is the asterisk (*), instead of 'W'. The asterisk forces DEBUG to write system (directory) sectors.

Leave DEBUG by typing: **O** <ENTER>

SYS6/SYS is now completely free of any protection status and passwords.

One last thing, the syntax READ and WRITE is:
READ:
drive #, track #, sector #, R, mem location, # of sectors
WRITE normal sectors:
drive#, track#, sector#, W, mem location, # of sectors
WRITE system (directory) sectors:
drive#, track#, sector#, *, mem location, # of sectors

DIGITIME for Model 4 Version 1

By David Goben

Regarding the challenge made on the last page of issue 2.6. for DIGITIME/BAS, the reason that the Model 4 messes up the screen display has to do with Microsoft's going to virtual screen formatting. Although not fully developed on the Model 4 (more so in GWBASIC), it checks string LENGTHS and crops it to the start of the next line if it goes over 80 characters from the current position. It does not recognize the fact that our strings have backspacing and the like. It simply considers it a continuous FORWARD-moving string.

My solution was to make use of a small machine language subroutine that will not check for position and line lengths; it will simply print out the string. This 17-byte routine could be used in other programs as well, to straighten up a formatted display.

Only line 10, 520 and 525 of the Model I/III version have been altered. In addition, lines 501, 504, 505 and 515 have been added.

A further Model 4 enhancement to fix up the 'jumpy-8' display (the 8 character flickers a bit) is to change the: **C=RIGHT$(TIME$,8)** at the start of line 510 to: **IF C=LEFT$(TIME$,5) THEN 510 ELSE C=LEFT$(TIME$,5):** *rest of line goes here.*

Thanks for the challenge. I'd sure be interested in the solution submitted by others.

Changes:

```
10 CLEAR:DEFINT L,T

520 PRINT@240,;:CALL TIME(C(H1)):PRINT@255,;:
CALL TIME(C(H2)):PRINT@430,;:CALL TIME(CB)

525 PRINT@274,;:CALL TIME(C(M1)):PRINT@289,;:
CALL TIME(C(M2))
```

Additions:

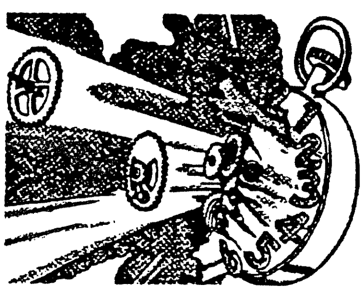
```
501 READ A$:IF A$< >"END" THEN
CP=CP+CHR$(VAL("&H"+a$)):GOTO 501

504 DATA 46,23,7E,23,66,6F,04,18,05,4E,23,3E,02,
EF,10,F9,C9,END

505 CB=CQ+STRING$(3,24)+STRING$(2,26)+CQ

515 TIME=PEEK(VARPTR(CP)+1)
+256*PEEK(VARPTR(CP)+2)-65536!
```

Enhancement:

```
510 IF C=LEFT$(TIME$,5) THEN 510 ELSE
C=LEFT$(TIME$,5):H1=VAL(MID$(C,1,1)):
H2=VAL(MID$(C,2,1):M1=VAL(MID$(C,4,1)):
M2=VAL(MID$(C,5,1))
```

DIGITIME for Model 4 Version 2

By William R. Bowman



Here is a Model 4 version of DIGITIME/BAS presented in TRSTimes 2.6. The changes required to run the program are in lines 10, 520 and 525. Line 10 cures the problem of string handling in Model 4 Basic. The undocumented WIDTH 255 statement makes Model 4 Basic handle strings just like Model I and III. The other changes are just the print locations for the larger screen.

There was a slight problem with the figure 8 display. This was fixed by adding line 125, and changing line 250.

Of course, I could not resist a slight enhancement to make the colon blink. The code for this is in the added lines 455 and 527.

I really could not understand why Mr. Scott went to the trouble to null out most of the string variables, unless possibly he felt that this would prolong the coming of 'garbage collection'. I did not change any of the original coding, except as noted above.

I might add that this could be a real screen burner if someone was to run run it continuously for any length of time, but I doubt that anyone would do that.

May I remind you that, regardless of whose Model 4 version is selected for publication, Mr. Scott did all the hard work and deserves another mention as the author of the original version of this program.

Changes:

```
10 CLEAR:WIDTH 255

250 C(8)=CC+CS+CHR$(27)+CR+CE

520 PRINT@(7,8),C(H1);:PRINT@(7,23),C(H2);:
PRINT@(9,38),CQ;STRINFG$(3,24);STRING$(2,26);CQ;

525 PRINT@(7,42),C(M1);:PRINT@(7,57),C(M2);
```

Additions:

```
125 CU=CHR$(162)+P(12)+CHR$(145)+CB
CS=CY+CU:CR=CU+CX
```

Enhancements:

```
455 CT=" "

527 PRINT@(9,38),CT;STRING$(3,24);
STRING$(2,26);CT;
```

DIGITIME for Model III & 4 Version 3

By Lloyd Evans

Here is a DIGITIME/BAS that will run on the Model III or 4. Five lines have been changed and 4 added. Lines 20 amd 30 define a new variable for the print positions. Line 33 reads the PRINT@ positions for the Model III in the DATA line 600. Line 35 checks which machine is running and changes the variable, if necessary. It also issues the WIDTH 255 command, which will make the Model 4 screen behave like the Model III. Lines 520 and 525 replace the hard print positions with the new variables.

Also, I didn't like the 8 that was displayed, so line 75 and 250 fixes this problem.

```
20 DEFSTR B,C,P

30 DIM B(12),P(15),X(5)

33 FOR X=1 TO 5:READ X(X):NEXT

35 IF PEEK(42)< >64 THEN FOR X=1 TO 5:
READ X(X):NEXT:WIDTH 255

75 CR=CHR$(162)+CHR$(187)+P(10)+CHR$(183)
+CHR$(145)+CB

250 C(8)=CC+CY+CR+CX+CE:CR=""

520 PRINT@X(1),C(H1);:PRINT@X(2),C(H2);:
PRINT@X(3),CQ;STRING$(3,24);STRING$(2,26);CQ;

525 PRINT@X(4),C(M1);:PRINT@X(5),C(M2);

600 DATA 192,207,350,226,241,568,583,758,602,617
```

DIGITIME for Model 4 Version 4

By Don Bergthold

TIMELY FUN FOR THE MODEL III by J. D. Scott sure hit the spot with me. Sometime ago, I created a KSM file (Key Stroke Multiply) that contained common strings and other things that I hated to type since I don't type well and complex strings is usually where I find all my typing errors. DIGITIME/BAS had just enough CHR$('s, STRING$('s and VAL(MID$('s to unleash the wrath and power of my Model 4 KSM/FLT routine. What a joy to type in a program listing and have the computer do most of the typing! And here is the solution I came up with:

Since Model 4 BASIC allocates string space dynamically, line 10 is not needed and should be deleted.

I added line 15 because Model 4 BASIC will attempt to keep the string on one line if it ca. BASIC takes the current cursor position and ADDS it to the length of the string. If the sum is greater than 80 (default screen width) the string will be displayed at the beginning of the next line. To prevent this I changed the WIDTH to 255, See WIDTH in Model 4 BASIC Manual. Line 527 was added to help prevent screen burn-in if the clock were to be left on for any length of time. The first timing loop controls how long the clock is displayed, and the second loop controls how long the clock is not displayed.

```
DELETE LINE 10

15 WIDTH 255

520 PRINT@(7,4),C(H1);:PRINT@(7,22),C(H2);:
PRINT@(9,39),CQ;STRING$(3,24);STRING$(2,26);CQ;

525 PRINT@(7,46), C(M1);:PRINT@(7,64),C(M2);

527 FOR J = 1 TO 1000:NEXT J:CLS:
FOR J = 1 TO 500:NEXT J
```

I really enjoy TRSTimes and look forward to every issue. As for the contests, I think it would be of interest to many. Gee, with the sound capabilities of the Model 4, one could make DIGITIME/BAS chime on the hour, or play a tune at given intervals. I can see the little wheels and cogs turnin' in lots of minds; who knows, maybe the contests will continue thru 1990 yet!

DIGITIME for Model 4 Version 5 (the final version?)

By Lance Wolstrup

Having the benefit of the above four contest winning solutions, I couldn't leave well enough alone. Add the following lines to William Bowman's code (version 2).

```
502 PRINT@(5,5),CHR$(151);STRING$(68,131);
CHR$(171)

503 FOR V=6 TO 14:PRINT@(V,5),CHR$(149):
PRINT@(V,74),CHR$(170):NEXT

504 PRINT@(V,5),CHR$(181);STRING$(68,176);
CHR$(186)

505 PRINT@(V+1,37),STRING$(5,191):
PRINT@(V+2,37),STRING$(5,191)

506 PRINT@(V+3,10),STRING$(60,191)

507 PRINT@(4,10),STRING$(4,188);STRING$(4,32);
STRING$(4,188);
```

TOP SECRET
Cryptography for Model 4

By Dale Parsons

The computer makes possible addition of a third dimension to the art of cryptography. "Xlate" goes beyond traditional methods of transposition and substitution, it translates from a 26 alphabetic language to one of fewer characters. The method is adapted from an article "Call Me 10DD29" by Norton C Richard that appeared in the November 1986 issue of 80 Micro, a program for number base conversion within a range of 2 to 201. "Xlate" restricts the range to the 26 alphabetic characters (plus a single invisible one) to conform with classical cryptography, and adds a few bells and whistles.

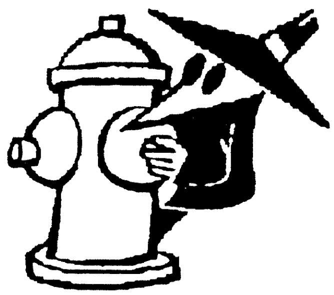
The basic algorithm is at lines 140,170 and 190 for encipherment, or lines 330, 350 and 370 for the inverse operation. Lines 110 (310) convert the alphabetic input into internally digestible numbers which are reconverted by lines 190 (390) for human readability. The source base is fixed at 27. The Program prompts for the output base and an increment (more about this later), then asks if you want to encode or decode and finally to input the message group. You may have to wait quite a few seconds depending upon the message length, don't panic too soon. The program then gives you a chance to check your input, go for more or simply to end.

Destination base D is limited to 25 characters maximum, with A to Y representing 0 (zero) thru 24. Z in this case is reserved for a special function. The source base S assigns values 1 thru 26 to A thru Z respectively. A psuedo zero utilizes the the space(bar) charactor for zero. This last greatly simplifies input and reading. Reading a continuous string of random letters is extremely difficult, line 210 chops the encoded output into 5 letter blocks conforming very nicely with classical methods. Upon decoding, type these spaces as part of the message. These blanks are then filtered out by line 240.

If 0 is entered for increment G then then letter A always has the value zero in the encoded output. If this bothers you there are ways around it. First, A string D of length 14 for example will fit anywhere within the limit 0 to 24. Inputting G as 1, B now represents zero and A is absent. Error checking in line 70 guards against illegal sums of D+G, but other input error either blows or produces garbage. A check routine helps in the latter case.

BASIC double precision allows up to 16 characters. Beyond that the value of N in line 140 (300) will become ambiguous. Line 30 allows setting a not to exceed limit NX. Both clear text and decode input will arrive at the identical value less than NX but the D multiple will not necessarily be greater than NX. The D+1-th character is reserved to signal this event

The clear text routine in line 140 says stop, restore the last N value, and send a D+1 marker Line 170 then processes what it has, generating an "internal word". Line 200 follows up with a "go back for more if not done" order. R array is dimensioned 11, the maximum possible number of digits Xlate can generate without danger of violating the 16 digit double precision rule.An alternate to the NX limit is to vary the R parameter, with some loss of randomness. BASIC does not recognize numerical quantities with leading zeroes. Lines 110 (310) will generate this condition so it is necessary to force acceptance following line 170 (350) by use of counter C1 and flag C2.



The capability to vary D and G provides near infinite distinct variations. But there is more, a relatively small change in NX can output an entirely different group of characters. The example value is for year 1934 5th month 10th day with 1200 hours thrown in, commemorating my first computing device - a 12th birthday gift of a 12 cent drug store slide rule. The output can further be scrambled by combining substition somewhere in the process. Retyping the following lines is one way:

```
50 K$="KEYWORDINGABCFHJLMPQSTUVXZ"
60 FOR J=1 TO 26:K=ASC(MID$(K$,J,1))-E:
KI(J)=K:KD(K)=J
130 A=KI(A)
370 FOR J=M TO 1 STEP -1:A=R(J):A=KD(A)+E
```

Note that maximum keyboard input length is 254 characters. Using array input and storage appears as one possibilty for longer message lengths but adds complexity.I would greatly appreciate hearing from anyone who successfully adapts it or other method to extend Xlate's meassage length capability. A final note: standard transmittal practice is to fill out the final block with null characters if needed. You cannot readily know the enciphered length and retyping may over or undershoot. However, you can add nulls of a sort, the internal word terminating character D+1. Xlate thoughtfully provides this prior to beginning input. Additional nulls if needed can be other characters.

Xlate's security lies in masquerading as a substitution-transposition or other scheme and an inherent unfriendliness to solution attacks for such schemes. It does run counter to traditional military demands that no character depend upon a previous one, else the remaining message will be garbled. However, such garbling will normally be localized within the offending internal word plus rechecking and computer accuracy further minimize this problem. Also, the military either ignored or accepted the risk of this error in multiple substitution encryption, inadvertent use twice or skipping of the next substitute string outputs gibberish.

XLATE/BAS

```
10 CLS:DEFINT A-M:REM SAVE "XLATE",A
20 DEFDBL N,Q:
DIM KI(26),KD(26),R(11):
S=27:E=64:F=65
30 NX=193405101200#
40 ON ERROR GOTO 420
50 '
60 '
70 INPUT"destination base, increment";D,G:
PRINT"** D+1 = ";CHR$(D+66+G):
IF D+G>24 THEN 420
80 INPUT"1 to encode or 2 to decode";B:
INPUT "message group";N$
90 ON B GOTO 100,270
100 N=0:X$="":L=LEN(N$):C=0:C1=0
110 C=C+1:C1=C1+1:A=ASC(MID$(N$,C,1))-E:
IF A=-32 THEN A=0
120 IF C1=1 AND A=0 THEN C2=1
130 '
140 N=N*S+A:IF N>NX THEN N=(N-A)/S:
C=C-1:M=1:R(M)=D+1:GOTO 170
150 IF C=L THEN M=0:GOTO 170
160 GOTO 110
170 M=M+1:Q=INT(N/D):R(M)=N-Q*D:N=Q:
IF N>0 THEN 170
180 IF C2=1 THEN M=M+1:R(M)=0:C2=0
190 FOR J=M TO 1 STEP -1:
A=R(J)+F+G:X$=X$+CHR$(A):NEXT
200 IF C<L THEN C1=0:N=0:GOTO 110
210 N$="":FOR I=1 TO LEN(X$):
N$=N$+MID$(X$,I,1):
IF I MOD 5=0 THEN N$=N$+" "
220 NEXT:PRINT:PRINT N$
230 INPUT "1 to check,2 more input ,3 to end";B1:
ON B1 GOTO 270,80,410
240 '
250 'The deciphering section starts next
260 '
270 X$="":FOR J=1 TO LEN(N$):
IF J MOD 6=0 THEN 290
280 X$=X$+MID$(N$,J,1)
290 NEXT
300 C=0:C1=0:N=0:L=LEN(X$):N$=""
310 C=C+1:C1=C1+1:A=ASC(MID$(X$,C,1))-F-G:
IF A=D+1 THEN M=0:GOTO 350
320 IF C1=1 AND A=0 THEN C2=1
330 N=N*D+A:IF C=L THEN M=0:GOTO 350
340 GOTO 310
350 M=M+1:Q=INT(N/S):R(M)=N-Q*S:N=Q:
IF N>0 THEN 350
360 IF C2=1 THEN M=M+1:R(M)=0:C2=0
370 FOR J=M TO 1 STEP -1:A=R(J)+E
380 IF A=E THEN A=32
390 N$=N$+CHR$(A):NEXT:
IF C<L THEN C1=0:N=0:GOTO 310
400 PRINT N$:
INPUT"1 to check, 2 more input, 3 to end";B1:
ON B1 GOTO 100,80,410
410 END
420 PRINT"Error !!! - Try Again":RUN 70
```

MORE GOODIES FOR YOUR TRS-80

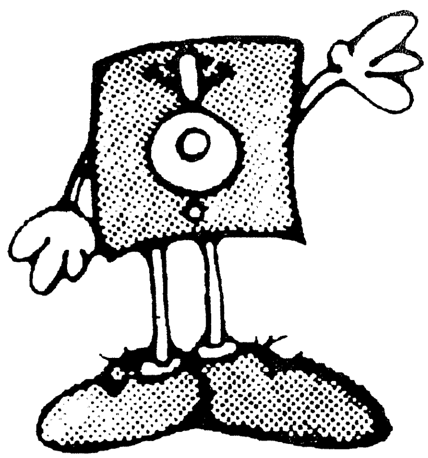
Get the latest issue of TRSLINK

TRSLINK is the new disk-based magazine dedicated to providing continuing information for the TRS-80. A new issue is published monthly, featuring Public Domain programs, "Shareware", articles, hints & tips, nationwide ads, letters, and more. TRSLINK can be obtained from your local TRS-80 BBS, or download it directly from:

8/n/1 #4
(215) 848-5728
(Philadelphia, PA.)
Sysop: Luis Garcia-Barrio

Believe it or not:
TRSLINK is FREE

TRSDOS 1.3. CORNER



More Patches

By Dr. Allen Jacobs

In an earlier issue of TRSTimes, it was mentioned that a TRSDOS 6.X command can be repeated by pressing <CONTROL><R> simultaneously. If you enjoy this feature, you will find that this convenience is not restricted to TRSDOS 6.X. Roy Beck, (of CP/M and hard drive fame in this very publication), reminded me that DOSPLUS 4 uses the / <ENTER> keystrokes at the beginning of a blank command line to accomplish the same task. MULTIDOS requires just the <ENTER> key for this purpose.

As I recalled that Andy Levinson coded a patch for TRSDOS 1.3 that would also allow pressing the <ENTER> key to repeat the last command, I gave him a call about it. He informed me that the article containing this and other patches to TRSDOS 1.3 can be found in the August, 1985 issue of '80 MICRO, on pages 72-75. The article is titled "A Patchwork Revisited".

The patch actually does three things.

1. It allows you to re-execute the last command by pressing the enter key.

2. It automatically converts all keyboard input to upper case. This is useful because most DOSes and many programs only read upper case commands.

3. It also allows you to place remarks in your DO files by adding a period to beginning of each remark line. The DOS will ignore the comment and go on to the next line in the file.

Andy reminded me to mention that you must enter the patch commands to this patch in the exact order as they are shown, and in one session. This is because you are altering the command interpreter, as you are using it! Andy said that the Multidos command repeat feature inspired this patch. It is easy to see, however, that Andy cannot resist improving on any project he undertakes.

When I told him that I was going to write about this subject, he also informed me that NEWDOS has a repeat command. It is: R <ENTER>.

Andy's elegant patch is as follows:

```
PATCH *0 (ADD=4CD1,FIND=204E53,
CHG=4E530D)

PATCH *1 (ADD=4E32,FIND=212542112642,
CHG=180B773E1BCD)

PATCH *1(ADD=4ED9,FIND=E5C52A,
CHG=C3BB4E)

PATCH *1 (ADD=4EA0,FIND=2040E548060009,
CHG=1A28917EFE6138)

PATCH *1 (ADD=4EA7,FIND=014000B7ED4222,
CHG=03D620772310F5)

PATCH *1 (ADD=4EAE,FIND=20403E1ECD3300,
CHG=D1E17E12FE2ECA)

PATCH *1 (ADD=4EB5,FIND=E1222040C1E1,
CHG=4A4EFE0D288F)

PATCH *1 (ADD=4E9B,FIND=28ADC3BB4E,
CHG=11D44CE5D5)
```

..end of patch.

SPEEDING UP MODEL 4
(in Model III mode)

By Lance Wolstrup

I originally wrote this article a little better than a year ago in response to a discussion at the San Gabriel Tandy User Group (SAGATUG) meeting. The article was published in SAGATUG's newsletter, called The Interface, as well as in the English National Amstrad/Tandy General User Group newsletter, NATGUG NEWS. As Bill Bermudez and several other readers, have asked the very same questions that initiated the article in the first place, I think it worthwhile to repeat the information.

First, let's understand that when you operate in Model III mode of the Model 4, the physical hardware is STILL Model 4. This means, if you know how, that you are able to use the faster Model 4 clock speed (4mhz) while in Model III mode.

Accessing the faster clock speed can certainly be beneficial. Your application program will execute faster, an otherwise sluggish program will now move along quite nicely. On the other side of the coin, a game or other program that uses delay loop routines so as not to run away from the user, may now execute entirely too fast. What we need is a way to speed up the machine when we want it to run fast, and slow it back down again (2 mhz) when that suits our purposes. We'll do just that.

The clock speed is handled through PORT &HEC (237 decimal). Bit 6 of this port determines whether the machine will run fast at 4mhz, or slow at 2mhz. If bit 6 is set (ON or 1) we run in fast mode. If bit 6 is reset (OFF or 0) slow mode is indicated. This is the hardware aspect of the clock speed. Now we move to the software portion.

As we probably all know by now, Model III has an abundance of different DOSes (Disk Operating Systems). They are as follows:

TRSDOS 1.3.	(the standard)
SYSTEM 1.5.	(the ultimate upgrade to 1.3.)
LDOS 5.1.x	(the forerunner of TRSDOS 6.x.x)
DOSPLUS 3.x.	
MULTIDOS 1.x. and 2.x.	
NEWDOS/80 v2.	

The last five were all designed to give the user an alternative to TRSDOS 1.3., which was always viewed as a fairly limited DOS. Unfortunately, except for SYSTEM 1.5., the standard set by TRSDOS 1.3. was not adhered to, and the result was a series of DOSes that were either not compatible with each other, or just somewhat compatible:

TRSDOS 1.3.	Compatible only with SYSTEM 1.5.
SYSTEM 1.5.	Compatible with TRSDOS 1.3. Can also read and copy files from LDOS and DOSPLUS (also TRSDOS 6).
NEWDOS/80 v2.	Unique - Not compatible with any other DOS.
LDOS-DOSPLUS & MULTIDOS	Somewhat compatible with each other.

The good news is that all the above DOSes use memory location &H4210 (16912 decimal) to set an 'image' of PORT &HEC. Numerous times per second this 'port image' is read, and the result is sent to PORT &HEC.

The bad news is that that this 'image' varies from DOS to DOS. This means that you cannot POKE a specific value, such as 249 into &H4210 on all DOSes. We have to be careful not change anything other than the bit that handles the clock speed (bit 6). Before we actually change the speed, let's see what we are up against:

LDOS and MULTIDOS - recognizes the Model 4 hardware and comes up in fast mode automatically.
DOSPLUS and NEWDOS/80 comes up in slow mode.
SYSTEM 1.5. allows the speed to be set directly from the DOS ready prompt.
TRSDOS 1.3. - comes up in slow mode - has a potential 'killer' problem if speed-up is on a 4P.

Changing the clock speed can be done several ways. First, let's do it from Basic: Boot up your favorite Model III DOS and get in Basic.

Type: POKE &H4210,PEEK(&H4210) OR 64 <ENTER>

This sets (turns ON) bit 6 of the '&HEC port image' memory location WITHOUT disturbing any of the other bits. At the next machine 'INTERRUPT' (within a fraction of a second) the port image is sent to PORT &HEC and the Model III mode now functions at high speed.

To effectively slow the machine back down to 2 mhz, type:

POKE &H4210,PEEK(&H4210) AND 191 <ENTER>

This resets (turns OFF) bit 6 while leaving all other bits alone. As above, the port image is sent to the port and, since bit 6 is OFF, we are back at 2 mhz.

Getting in Basic to POKE the fast or slow values can be inconvenient, especially if the program you eventually want to execute is a /CMD file. For those of you who own an Editor/Assembler, and know how to use it, write the following Assembly language programs:

LISTING 1.

```
         ORG  0E000H
START    LD   A,(4210H)    ;get image
         OR   64           ;set bit 6
         LD   (4210H),A    ;put new value back
         OUT  (0ECH),A     ;send to port
         RET               ;back to DOS
         END  START
```

Assemble listing 1 as FAST/CMD.

LISTING 2.

```
         ORG  0E000H
START    LD   A,(4210H)    ;get image
         AND  191          ;reset bit 6
         LD   (4210H),A    ;put new value back
         OUT  (0ECH),A     ;send to port
         RET               ;back to DOS
         END  START
```

Assemble listing 2 as SLOW/CMD

Now, whenever you wish to operate in fast mode, from DOS type: FAST <ENTER>

To slow down, type: SLOW <ENTER

The above PEEKs and POKEs and Assembly language programs work marvelously on ALL Model III DOSes on a DESKTOP Model 4. Some Model 4P's, however, will have a severe problem with disk I/O if we try to speed up while using TRSDOS 1.3.

We can solve this problem by writing this Basic program:

LISTING 3.

```
10 CLS
20 POKE &H4210,PEEK(&H4210( OR 64
30 FOR X=17408 TO 17415
40 POKE X,227
50 NEXT
60 PRINT"TRSDOS 1.3. IS NOW IN FAST MODE"
70 END
```

SAVE listing 3 as FAST/BAS

LISTING 4.

```
10 CLS
20 POKE &H4210,PEEK(&H4210) AND 191
30 FOR X=17408 TO 17414 STEP 2
40 POKE X,245
50 POKE X+1,241
60 NEXT
70 PRINT"TRSDOS 1.3. IS NOW IN SLOW MODE"
80 END
```

SAVE listing 4 as SLOW/BAS

The above can be written in Assembly language as follows:

LISTING 5.

```
         ORG  0E000H
START    LD   A,(4210H)    ;get image
         OR   64           ;set bit 6
         LD   (4210H),A    ;new value back
         OUT  (0ECH),A     ;send to port
         LD   B,8          ;loop to double
LOOP     LD   HL,17408     ;TRSDOS 1.3.'s
         LD   (HL),227     ;internal delay
         INC  HL           ;routine before
         DJNZ LOOP         ;disk I/O.
         RET               ;back to DOS
         END  START
```

Assemble listing 5 as FAST13/CMD

To slow TRSDOS 1.3. back down to 2 mhz, we need to restore the original delay loop before disk I/O:

LISTING 6.

```
         ORG  0E000H
START    LD   A,(4210H)    ;get image
         AND  191          ;reset bit 6
         LD   (4210H),A    ;put new value back
         OUT  (0ECH),A     ;send to port
         LD   B,4
         LD   HL,17407     ;TRSDOS1.3.'s
LOOP     INC  HL           ;original delay
         LD   (HL),245     ;routine before
         INC  HL           ;performing any
         LD   (HL),241     ;disk I/O.
         DJNZ LOOP         ;continue until B=0
         RET               ;back to DOS
         END  START
```

Assemble listing 6 as SLOW13/CMD

You are now able to speed up and slow down your various Model III DOSes (on a Model 4) to suit your needs. Happy speeding, and don't worry about that flashing red light in your rearview mirror.

ATTENTION TRSDOS 1.3. USERS!

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

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

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

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

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

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

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

ORDER YOUR COPY TODAY!

Send $39.95 (U.S. funds) to:
TRSTimes - SYSTEM 1.5.
20311 Sherman Way. Suite 221
Canoga Park, CA. 91306

HI-RES MYSTERIES EXPLAINED

comments on the hi-res reviews from issue 2.6.

By Frank Slinkman

I received my advance copy of Dr. Allen Jacobs' combined review of my three programs, SLOT4MOD4, VIDPOKR4 and GIF4MOD4, which I am very pleased with, and grateful for.

I thought I should address one minor technical inaccuracy (possibly a typo – or keyboard-o) which appeared in the review, as well as respond to some of the points Allen raised.

Addressing the technical inaccuracy, both the Radio Shack and Micro-Labs boards are addressed through ports 128-131 (80H-83H), as follows:

Port	Function
128 (80H)	X (horizontal) coordinate (write only)
129 (81H)	Y (vertical) coordinate (write only)
130 (82H)	Graphics data (read/write)
131 (83H)	Control register (write only)

In addition, the RS board utilises a couple of other ports to implement text overlay and scrolling around it's full 1024 x 256 image area. There has been some confusion over these port assignments in the past, as they are listed incorrectly in the Model 4 Technical Manual.

I hope Allen knows someone with a Micro-Labs board and tries SLOTMOD4 on that. One of the differences between the two boards is that the uLabs board accepts data from port 130 when either the horizontal or vertical blanking is on, while the RS board normally only permits writes during vertical blanking.

Since the Model 4 monitor is horizontally blanked 20% of the time and vertically blanked 1/16th of the time, the uLabs board can support a data transfer rate theoretically four times that of the RS board. In most applications, the apparent speed difference is only 30-40%, since only writes to the board are faster.

However, in a heavy animation application like SLOTMOD4, which sends a *huge* amount of data to the graphics board, the speed difference is quite dramatic. A typical SLOTMOD4 game cycle is about 2.5 times faster on the uLabs board than the RS board, which results in much smoother and more realistic animation.

The prototype for the video poker machine is in a friend's collection. Knowing the gaming industry as well as I do (I once lived in Las Vegas), I am not surprised that machines with the "double-or-nothing" option are getting hard find. This option, if correctly used in the manner described in the video poker strategy included with the program, actually gives the player a slight edge over the machine. Obviously, someone in the industry figured this out (or, more likely, observed some *players* who had figured it out), and took the appropriate action to "correct" the "problem." After all, they're not in the casino business for their *health* – just ask them.

As Allen stated, the differences between the graphics configurations of the various types of computers do present problems for any kind of hardware-independent graphics exchange format such as GIF.

The "IBM standard" is an screen aspect ratio of 4:3. That is, the screen is 75% as high as it is wide. This is doubtless due to the fact that this is the standard for television. All TRS-80's comply with this standard. Steven Jobs, however, took a different approach on the Mac, turning the CRT on its side, and adopting an aspect ratio of 4:5, the standard for photography.

Most IBM graphics are done in CGA (320 x 200 in color or 640 x 200 monochrome), EGA (640 x 350) or VGA (640 x 480). We are starting to see more SVGA (super VGA) images with 800 x 600, 1024 x 768, and even higher resolutions.

Note: Contrary to Allen's statement, GIF is well able to handle all these resolutions. In fact, GIF is so flexible, the only limit to image size is the number of bits (16) used to describe the image dimensions. Specifically, GIF can handle image sizes up to 65535 x 65535 x 256.

Under the proposed new standard, this will be increased to 65535 x 65535 x 16,777,216! (The third number is the number of colors.)

Achieving a 4:3 screen aspect ratio at these different resolutions requires different *pixel* aspect ratios. For example, on the Model 4, our 640 x 240 screen has a pixel aspect ratio of 1:2 (twice as high as wide):

$$(640 * 1) / (240 * 2) = 640/480 = 4/3.$$

I have not seen the image, BALLS.GIF, in which Allen found the distortion. But I've seen the problem in other images. Here's the reason for it.

The MacIntosh always uses pixels which have a 1:1 aspect ratio ("square" pixels, in the vernacular of us hi-res junkies). Some generous Mac users, when they find or create an especially good image, want to share it with others with screen ratios of 4:3 [as God clearly intended screen aspect ratios to be ;-)].

Now if a Mac user, knowing that CGA (320 x 200) is the most common screen size in the IBM world, converts his image to that resolution with his square pixels, it is going to be distorted when viewed on an CGA machine, because CGA pixels are *not* square -- they have a 5:6 pixel aspect ratio.

(320 * 5) / (200 * 6) = 1600/1200 = 4/3.

Thus, a 35mm diameter circle (or ball, in this case) will be 20% longer along the vertical axis, and would measure 35mm wide x 42mm high.

Similarly, EGA (640 x 350) pixels are not square. Their aspect ratio is about 3:4. Thus a 35mm circle would display as a 35mm x 47mm ellipse.

The GIF file can be patched to correct this. The values at file locations 00,08 and 00,09 will probably be C8 00 (lsb-msb hex for 200 decimal). Patching the C8 to F0 (240) will trick GIF4MOD4 into treating the pixels as square, thus properly scaling the image. If the values are 5E 01 (350), change the 5E to E0. If the values are other than those stated, it's a non-standard image, and you will have to experiment.

Bear in mind that, in either case, the decoded image will no longer fill the entire screen (5/6ths in the first instance, about 2/3rds in the second).

The proposed new GIF standard now under consideration (which, due to my status as a CompuServe GIF Developer, I have had a small *(very* small) hand in, and for which I emphasise the word "proposed") has provision for pixel aspect ratio information within the file which will completely eliminate the problem as soon as encoder and decoder software can be upgraded to implement this and other features of the proposed new standard, once it is finalised and adopted.

The original version of GIF4MOD4 (never released in that form) did display the screen and image dimensions. I removed that feature mainly because, while that information is interesting, it has no practical *use*, and pausing to display it only increased processing time. Besides, the image dimensions are what they are, and there's nothing the average user can *do* about it.

I chose to display screen size information for only those images which, due to their non-standard screen (and, presumably, non-standard pixel) aspect ratios, are beyond the program's ability to scale them to exact proportions.

The exact number of colors in a GIF image is not available, as such, until after the image has been decoded. There is a 3-bit field in every GIF file which contains the number of bits needed to describe the *maximum* possible number of colors in the image.

The contents of this field are base zero, which means the value therein must be incremented by one to get this number. For example, a value of 000 (binary) becomes one, which is the number of bits needed to describe a monochrome image, where 0 is probably (but not necessarily) black and 1 is probably white. Similarly, a value of 111 (binary) becomes eight, which is the number of bits needed to describe a 256-color image.

This field is used to tell the decoder how many three-byte fields will be used in the "color map." However, just because a map can *hold* 256 colors does not mean the image has 256 colors. I have seen images which have 256-size color maps, but only have 2, 4, 8 or 16 colors actually mapped, and actually *use* fewer than are mapped [e.g., only 13 colors actually in the image when 16 (presumably a "standard palette") are described in a color map which is 16 times too large].

There is a similar 3-bit field which is supposed to describe the color resolution in the image. However, this was inadequately described in the GIF specs; so many GIF encoder programs either don't use it, or use it incorrectly.

If the contents of the CR field were reliable, it *would* be possible to inform the user in advance of roughly how many colors are in the image. It's intended purpose is to tell the decoder whether or not the image requires dithering and, if so, give it the opportunity to automatically choose the best type of dither. Unfortunately it is *not* reliable; so cannot be used.

Thus Allen's complaint that the advice about which dithering methods are best for which images based on the number of colors present is understood -- and I sympathise. Unfortunately, for the reasons stated above, there is no practical alternative to the trial-and-error method I suggested in the docs.

The actual colors are described in the color map, which contains the RGB (red-green-blue) components of each color, one byte for each component, with the intensities of each expressed in 255ths. The typical color map for a black and white image would be six bytes long, namely 00 00 00 FF FF FF hexadecimal.

The first three bytes mean that the intensity of red, blue and green of color #0 are 0/255, 0/255 and 0/255, respectively. This translates to black. Color #1 has 255/255, or full, intensities of each of the three primaries, which translates to white. There is, however, no reason this map couldn't be reversed to FF FF FF 00 00 00, so long as the raster data reflected the fact the color #0 is white and color #1 is black.

A typical 8-color map would be something like (all values in hex):

Color#	R G B	Description
0	00 00 00	Black
1	00 00 FF	Blue
2	00 FF 00	Green
3	00 FF FF	Cyan (blue-green)
4	FF 00 00	Red
5	FF 00 FF	Magenta (reddish-purple)
6	FF FF 00	Yellow
7	FF FF FF	White

A typical map for a monochrome image with 4 grey-scales would be:

Color #	R G B	Description
0	00 00 00	Black
1	55 55 55	Dark Grey
2	AA AA AA	Light Grey
3	FF FF FF	White

Allen's suggestions that the program should give a color scale, describe which dithering patterns are used to represent each color and/or a listing of the colors are, I'm afraid, based on assumptions which are in turn based on outmoded methodology. I've seen the kind of "pattern dithering" he described in his review in papers dated 1977. The latest techniques, which are far superior, are known as "error distribution dithering."

An "error" is the difference between what is supposed to be displayed and what is actually displayed. Error distribution dithers use what is called a "filter" to distribute these errors. Good filters take into account the fact that light intensity decreases in proportion to the square of the distance, while minimising the amount of time-consuming computation required.

```
For example, Burke's filter is:     X 4 2
                                1 2 4 2 1    16ths.
```

What this means is that 4/16ths of the error from pixel "X" is to be distributed to each of the pixels indicated by "4." Two sixteenths of the error goes to each of the pixels indicated by "2," and 1/16th to each of the pixels indicated by "1."

As an example, lets look at an area which is supposed to be displayed in medium green, which has an equivalent black & white display intensity of 100/255.

The value of pixel "X" is 100. Since 100 is closer to 0 than it is to 255, 0 (black) will be displayed, and the error is 100 (the true value minus the displayed value, or 100 - 0 = 100). The pixel to the right and the pixel below will receive 4/16ths of this value; so they will each become 125. The 2nd pixel to the right, and the two diagonally lower pixels will each receive 2/16ths of this value (12.5 rounded to 13); so they will each take the value of 113. The last two pixels will each receive 1/16th of the error, and each become 106.

Now we advance one pixel to the right, which now has the value of 125, which is also closer to 0 than to 255, and which therefore will also be displayed as 0 (black) leaving an error of 125. This error is now distributed in the same manner, relative to the new location. One fourth of this value (31) will be added to the pixel to the right and below, each of which have the value of 113, giving them new values of 144. The rest of the error is disbursed as described in the paragraph above.

The next pixel will be displayed as 255 (*white*), because its value, 144, is closer to 255 than to 0. The error for this pixel is -111 (144 minus 255). In the case of the first two pixels the error was positive, and therefore adds to the neighboring pixels, brightening them. This error is negative, and is therefore *darkens* its neighbors.

Normally, this process continues left-to-right and top-to-bottom. However, for some images it is better to alternate between left-to-right and right-to-left (using the reflection of the filter) on alternating lines, which gives a different result.

The net result, when the whole process is completed, is that the area which is supposed to be displayed with each pixel at 100/255ths of white will instead have 100 of each 255 pixels white, and 155 of each 255 black, giving the *area* an intensity of 100/255ths.

A little randomness should be introduced to perturb the disbursed errors to break up what are called "dithering artifacts" -- the sudden "dumping" of accumulated errors which usually take the form of dotted lines which were not in the original image, and which detract from the rendition of the image.

As can be seen from the above, there *are* no set dithering patterns or colors to list. That's the beauty of good dithering filters like Burke's -- they automatically adjust to *any* combination of colors and intensities.

Frankly, I have not been completely happy with the image quality produced by either Burkes' or Floyd and Steinburg's filters, even though these are generally considered the best. These filters were designed for use with square, or nearly square pixels. The Model 4's pixels, however, are *not* square.

Therefore, I have developed a new filter (the "Slinkman filter", of course) which is designed for use with these decidedly oblong pixels.

```
The Slinkman filter:    X  10  6
                        2  36 32    32nds.
```

This new filter gives vastly superior results on the Model 4, and is being incorporated into GIF4MOD4 Version 2, which has a planned 1/2/90 release date. (Customers please note: if your order was received in December, shipment is being held up for the release of Version 2; so you won't have to send your disk back in for an upgrade, or pay the upgrade fee. Previous customers will soon be advised of the upgrade procedure.)

It is obvious from his comments that a combination of curiosity and some assumptions based on previous dithering techniques left Dr. Jacobs more than a little bit frustrated. With computer graphics technology advancing so fast, *nobody* can keep up with *all* the changes!

In my defense, let me say that the program docs were never intended to be a tutorial on dithering or any other technical aspect of computer graphics.

Few users share Dr. Jacobs' admirable scientific curiosity. In fact, he is the first person to ever ask me, albeit indirectly, *how* GIF4MOD4 works! Most users just want to see as pleasing and as accurate an image on their monitor screen as possible, and couldn't care less *how* it's accomplished -- and that's to be expected, and REspected.

I have no objection whatsoever to providing such information -- witness the fact that I am doing so here. It's just that it's hard enough to get people to read documentation as it is. The more difficult the docs are to read and understand -- and the more intimidating they appear -- the less they're going to be read. For this reason, I subscribe to the KISS philosophy when it comes to program documentation: give the user what he needs to know to get the most out of the program, and don't bother him with details.

To at least partially satisfy Dr. Jacobs' curiosity -- a curiosity hopefully shared by at least *some* other TRSTimes readers -- the following BASIC code will display most of the descriptive information contained in a GIF file.

```
10 '/* CHECKGIF/BAS
20 '    routine to gather screen size, color and image
30 '    size information from a version 87a GIF file */
40 '
50 CLS:DEFSTR A:DEFINT B-Z:GOTO 200
100 PRINT ASC(INPUT$(1,1))+256*ASC(INPUT$(1,1));:
RETURN
200 INPUT "Input file name: ",ANAME:
FOR X=1 TO LEN(ANAME):A=MID$(ANAME,X,1)
210 IF (A>="a" AND A<="z") THEN
MID$(ANAME,X,1)=CHR$(ASC(A) AND 95)
220 NEXT X:IF INSTR(ANAME,"/") = 0 THEN
ANAME=ANAME+"/GIF"
230 OPEN "I",1,ANAME
300 CLS:PRINT"Analysing";TAB(31);ANAME
310 PRINT"File Signature:";TAB(31);INPUT$(6,1)
320 PRINT"Screen width:";TAB(30);:GOSUB 100:PRINT
330 PRINT"Screen height:";TAB(30);:GOSUB 100:
PRINT
340 X=ASC(INPUT$(1,1)):CR=(X AND &H70)/16:
PIXEL=X AND 7
350 PRINT"Color Resolution:";TAB(30);2^(CR+1)
360 PRINT"Colors available:";TAB(30);2^(PIXEL+1)
370 PRINT"Background is color #";ASC(INPUT$(1,1))
380 A=INPUT$(1,1):
'               /* skip screen descriptor terminator */
390 FOR X=1 TO 3*2^(PIXEL+1):A=INPUT$(1,1):
NEXT X:'     /* skip color map */
400 IF INPUT$(1,1)<>"," THEN 400

500 PRINT:PRINT"Image starts";:GOSUB 100:
PRINT"pixels from left and";
510 GOSUB 100:PRINT"pixels from top"
520 PRINT"Image is";:GOSUB 100:PRINT"x";:
GOSUB 100:PRINT"in ";
530 IF (ASC(INPUT$(1,1)) AND 64)=0 THEN
PRINT"normal"; ELSE PRINT"interlaced";
540 PRINT" order":A=INPUT$(1,1):' /* skip code size */
600 X=ASC(INPUT$(1,1)):IF X>0 THEN
A=INPUT$(X,1):GOTO 600:'/*skip raster data*/
700 IF INPUT$(1,1)="," THEN 500:
'                       /* go if multiple image */
710 PRINT:PRINT"Analysis of ";ANAME;" completed."
720 END
```

"70 INCOME TAX PROGRAMS"

For Filing By April 15, 1990
TRS-80 MODELS I, III and 4/4P
For the Tax Preparer, Lawyer, C.P.A. and the Individual.
Buy only the disks you'll use.
Our 11th Year of TRS-80 Income Tax Programming.

Last year there were 5 disks for Personal Taxes, and 3 disks for Business Taxes, for the Models III and 4/4P.
There may be another Personal Disk this year, depending on how many new Forms are necessary.
There are twice as many disks for the Model I.

The Personal Series includes the 1040, 1040A, 1040X, 1040ES, all Schedules and Forms 2106, 2119,2210,2441, 3468, 3800, 3903, 4136, 4137, 4562, 4684, 4797, 4835, 4868, 4972, 6251, 6252, 8027, 8283, 8396, 8582, 8606, 8615, and 8814.

The Business Series includes the 1120, 1120A, 1120X, 1120S, 1041, 1041S, 1065, 2220 and Schedule D, 1120S, K-1, 1120S, K-1, 1041, 8656, 7004, Schedule D, 1065, K-1, 1065.
"Signature Forms" (1040, 1040A, 1120, etc.) are for use with Overlays; all Forms and Schedules are Computer Generated.
Write for Listings and Prices

MasterCard

GOOTH SOFTWARE
931 SO. BEMISTON
ST. LOUIS, MO. 63105

VISA

TRSTimes on DISK #4

Issue #4 of TRSTimes on Disk is now available, featuring the following programs from the July, September and November 1989 issues:

MAXIT/BAS	4	All
MACHRELO/BAS	III	All
TRSLABEL/BAS	III	All
TRSLABEL/CMD	III	All
HASHCODE/BAS	4	TRSDOS/LS-DOS 6
WDCOUNT/BAS	I/III/4	All
EPSONSET/BAS	4	All
HEXCHART/BAS	4	All
BASTEST3/BAS	III	All
DIGITIME/BAS	I/III	All
VARILIST/BAS	I/III/4	All
LOANCALC/BAS	I/III/4	All
WORDPUZL/BAS	4	All

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
20311 Sherman Way, Suite 221
Canoga Park, CA. 91306. U.S.A.

TRSTimes on DISK #1, 2 & 3 are still available at the above prices.

A LITTLE HARD DISK PROBLEM

By Roy T. Beck

I am fortunate to have two working hard disks, one at home (for pleasure), and one at my office (for business). Sounds great, and under most circumstances, it is. But leave it to me to find (or create) a problem. In this case, the problem is incompatibility of the drives.

The one at home is essentially a stock 35 Meg Radio Shack unit, with 8 heads and 512 tracks, and a Western Digital controller. Actually, the only part of this unit that ever came from Radio Shack is the hard disk controller (HDC), which is a R/S Cat. No. 26-1138, supplied by Dave Dalager of Texas. (*Thanks again, Dave!*) The remainder of the unit started out as a Xebec drive running on an Apple II. How, you may ask, can that be called a Radio shack unit? Simple, sez I; I pulled the Xebec S-1410 controller out of the Apple package, mounted the 26-1138 HDC (which is the WD-1010 small unit used in the later RS packages) in its place, and Hey Presto! my Model 4's think they are seeing a stock RS unit. I am using the PowerSoft drivers to partition by head, and have TRSDOS 6.3 and LDOS 5.3 working on the first two heads, respectively.

I have part of head #6 operating under CP/M, utilizing Montezuma Micro's driver. Actually, I am cheating a little bit there, as the driver thinks the bubble is a 15 Megger with 6 heads and 306 tracks, and only one working partition on track 6. It thinks heads 1 thru 5 are not assigned, and doesn't even know heads 7 and 8 exist! And it also doesn't know there are 206 more tracks on head #6. But we won't tell it, will we?

I created this setup because the Montezuma Micro drivers are very inflexible, and this was about the only arrangement I could find which would not try to gobble up unnecessarily large chunks of my 35 Meg drive. With this arrangement my CP/M Drive A is 2.5 Megs, amply large for my CP/M purposes, and I still have 7 other heads for other purposes. Eventually I hope to have a different DOS on every head, just for the versatility of it! Now, this is all well and good for my home machine, and it all works reliably. But, my office machine is different. Oh, how different!

First off, it never saw Radio Shack in its life. It was made by VR DATA, a still existing company who now prefers to forget they ever made anything for Radio Shack. The unit was identified as a VR HARD DISK III, designed to go with the Model III, and had a 5 Meg Tandon bubble, a Xebec S1410 HDC, a linear power supply, and a host adapter to convert the Mod III interface to SASI, because the Xebec HDC is a SASI device. (SASI is a subset of the present SCSI standard). When I acquired the unit, the bubble was long gone, but the other components were still there, and I was told it had worked properly under DOSPLUS. I even received a DOSPLUS drive disk with it.

Careful exploration revealed the box was actually set up to house TWO bubbles, even to the extent of having the second power supply cable already installed.

On one of my dusty shelves I had two 5 Meg Tandon drives, left over from a previous venture with my IBM clone (boo, hiss, etc). Since both of those drives had operated happily on the clone, and had no bad sectors, I expected I could use them in the VR DATA box. (The two drives were retired from years of commercial lease service, and cost me only $15 each; I give Tandon high marks for a good product).

Good old Roy Soltoff of MISOSYS had the necessary drivers for running the VR DATA unit under LDOS 5.3 and TRSDOS 6.3. With only a little experimentation, I was able to get one of the drives operating under TRSDOS 6.3, and it is now working very well for me in my office. I have Electric Webster, AllWrite and a few other odds and ends on it.

But the second bubble is sitting there keeping itself warm as it goes around, and is not presently in use. Since I have dBASE II, I would like to put it to work on the second bubble. But there's the rub. I don't have a suitable driver for the VR DATA unit under CP/M.

I called Monte at Montezuma Micro and asked for help. Actually, the first person I talked to didn't know what a VR DATA was; but he turned me over to the manager, who knew what I wanted, but said they did not have the driver I needed. As an alternative, I ordered their Aerocomp 5 Meg driver, since I knew that was also a SASI/SCSI unit, hoping I might be able to patch port numbers, etc, to make it work on the VR DATA box. They were pessimistic about my chances, and warned me they would not guarantee anything, and would not accept return of the driver under the circumstances. I said I understood their reservations, and this was strictly at my own risk. (For $30, I didn't really see much risk, but they are conservative people).

A few days later the disk arrived, and when I looked at it there was the 5 Meg Aerocomp driver, as requested, but Lo and Behold!, there were two more drivers on the disk identified as VR Data 17 Meg drivers, one for partition by head offset and the other for partition by cylinder offset.

I tried all three. The Aerocomp driver just flat wouldn't do anything. Complete zero. I then tried the two 17 Meg VR DATA drivers. Both seemed to work, but when I got to the Format with verify stage, both drivers claimed to have successfully formatted 17 Megs of my 5 Meg bubble. No

Way! I then tried to load the system files. The driver which partitioned by head offset just reported continuous errors when it tried to read a file from the HD. The driver which partitioned by cylinder offset did load the DOS alright, and would run CP/M from the hard disk. However, when I tried PIP A:=B:*.*, it got only a little way along and then reported I/O errors.

So far, the drive is unusable for CP/M, but I am greatly encouraged by the fact that the partition by cylinder offset worked partially, including running CP/M from it. This proves the ports are correct, and likely the entire driver structure is correct, except for the following factors.

Total track count is wrong.
(Should be 153 instead of 480)

Write Current Reduction is probably occurring at track 240 instead of 128 where it should occur.

Write precommpensation is probably also occurring at track 240 instead of 128 where it should be.

The formatter may be putting the directory track in the wrong place.

What to do? The fix is obviously to patch the 17 Meg driver to adjust the above parameters. But which parameters, and where?

ANALYSIS

My starting point in analysis of the CP/M driver is to study the inner workings of the MISOSYS driver which runs correctly on the same VR DATA drive. I am now in the throes of disassembling and annotating the MISOSYS driver. My tools are DSMBLR, by MISOSYS, DISASSEM out of NEWDOS80 V2, and The Programmer's Guide to LDOS/TRSDOS Version 6 by Soltoff.

I cannot recommend too highly the MISOSYS Programmer's Guide, as it covers not only the philosophy but also the nuts and bolts of the DOS, especially as to how floppies and hard disks are interfaced with the DOS.

The use of DSMBLR is obvious, but why did I use DISASSEM from ND80V2? One of the nice features of that disassembler is the construction of a back reference table which shows where every CALL and JP comes from. If you see a subroutine at a given location, just check the reference table and it tells you every instruction which calls that particular subroutine. A very powerful feature when analysing code. I don't know any other disassembler which provides this particular feature. Actually, MISOSYS' DSMBLR is better in other ways, but I run both, and by giving the NIP command to DISASSEM, I suppress all output except that back reference table, so it takes only a few minutes to run. Note that I am doing all of this under LDOS in the MOD III mode. The reasons are that the ND80 disassembler only runs under MOD III, and because LDOS does not use SVC calls, I believe I can more easily determine what is happening in the LDOS HD driver as opposed to the TRSDOS version. Also, I haven't yet purchased the Mod 4 version of DSMBLR. Because the disk formats are the same for both, I can disassemble either version of the MISOSYS HD driver from the same disk with the same LDOS disassembler.

As I write this, I haven't yet found all the answers in the MISOSYS driver, but I am close.

Looking ahead, the next step is to disassemble and study the Montezuma Micro 17 Meg driver for the same bubble. Here I have a problem, as I don't have all the nice tools under CP/M that I have under TRSDOS. Specifically, I don't have a disassembler for CP/M which will give me Z-80 opcodes. The DDT debugger will give me a printed disassembly, no problem there, but it is all in 8080 opcodes, and I don't want to spend the time necessary to become fluent in them. So I played a little trick. I used DDT to load and access the 17 Meg driver. I found the code began at 100H as usual, and ran up to 1280H, about 4k worth. Knowing this, I used DDT to block move a copy of the code to 8100-9380H. I then rebooted the machine under TRSDOS 6.3 and DUMPed a copy of the memory region from 8100H to 9380H to disk under a suitable file name, and now I have the 17 Meg CP/M driver in a vulnerable position on my TRS "operating table"! DISASSEM allows you to offset the address by any amount when it runs, so I just offset it by 8000H to produce my reference table with the correct locations in it. DSMBLR is not so smart, so that listing is 8000H too high. But it's usable. The only problem is absolute addresses are screwed up. But it should allow me to annotate the listing.

When I am finished with that, I should have all the pertinent locations staked out and hogtied so that I can use DU to go into the 17 Meg driver and make such surgical revisions as appear necessary to transform it into a 5 Meg VR DATA HD driver for the VR HARD DISK III box.

Incidentally, both Montezuma Micro and Aerocomp are owned by John Lancione (Monte), who is staying with us TRS types, many kudoes to him. By the way, JBO is Jesse Bob Overholt, who is Monte's Software Wizard. Many of you will remember JBO for his numerous articles in Northern Bytes and the Alternate Source. Monte is proud of his line of TRS drives, including floppies and HD's and points out that his equipment meets the FCC limits on emissions for home computers. Not every vendor does. Thank you, Monte, John, and JBO!

The next episode of this saga should (I hope) reveal how to transform the size of a 17 Meg VR DATA CP/M driver to any desired combination of heads, tracks, WRC and Precomp. Stay tuned, and witness my success (or failure).

— *Roy*

THE SWAP MEET

WANTED: To complete collection I am looking for any issues from the first 2 years of Softside Magazine, any issues from the 1st year of 80 U.S., Softside disks & Softside Adventure disk series.
Lance Wolstrup. 20311 Sherman Way. Suite 221.
Canoga Park, CA. 91306

WANTED: I am looking for back issues of the Radio Shack Computer Catalog to complete my collection. I specifically need the following issues: RSC-1, 3, 5, & 19. Any assistance would be greatly appreciated.
Roy Beck
2153 Cedarhurst Dr.
Los Angeles, CA. 90027

WANTED: Used Model 100 or 102 in good condition with at least 24K. Also need single/double sided disk drive for above. Must be priced reasonably.
Lance Wolstrup. 20311 Sherman Way. Suite 221.
Canoga Park, CA. 91306

FOR SALE: PD GOOD GAMES for Model III.
GAMEDISK#1: amazin/bas (maze), blazer/cmd (arcade), breakout/cmd (break down the walls), centipede/cmd (elect/bas (a simulation of the 1980 election), madhouse/bas (adventure), othello/cmd (board), poker/bas (almost better than going to las vegas), solitr1/bas (great solitaire card game), towers/cmd (puzzle game).
GAMEDISK#2: crams2/cmd (chase), falien (arcade), frankadv/bas (adventure), iceworld/bas (adventure), minigolf/bas (putt-putt on the trs-80), ping pong/cmd (1 or 2 player arcade game), reactor/bas (simulation), solitr2/bas (another good solitaire card game), stars/cmd (2 player race game), trak/cmd (maze game).
GAMEDISK#3: ashka/cmd (d&d), asteroid/cmd (arcade), crazy8/bas (card game), french/cmd (space invaders in french), hexapawn (board game), hobbit/bas (adventure game), memalpha/bas (adventure), pyramid/bas (good solitaire card game), rescue/bas (arcade), swarm/cmd (arcade game).
Price per disk: $5.00 (U.S)
or get all 3 disks for $12.00 (U.S.)
TRSTimes - PD DISKS. 20311 Sherman Way. Suite 221
Canoga Park, CA. 91306

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

FOR SALE: TRS-80 Model III, Dual standard drives. Some software. Good condition. $90.00.
Call/write: Javier Rodriguez. 13611 Gaines Circle.
Garden Grove, CA. 92643. (714) 638-3328

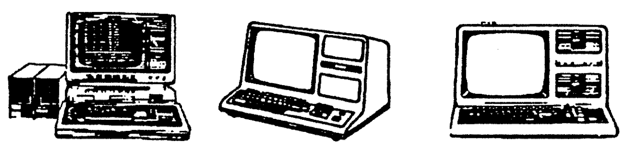
WANTED: Any software/hardware to allow the Model III or IV to communicate with H.P. xy recorders, analog or digital type.
John Greenland. Box 171,
Kelligrews Field, New Foundland. A0A 2T0 Canada

FOR SALE: Model 4 with dual floppy, 128 K, green monitor; keyboard needs work; will include software with original docs. Make a reasonable offer.
Paul Guglielmotti, 14512 Pacific Ave.
Baldwin Park, CA. 91706. (818) 962-8233

WANTED: (1.) Need CP/M boot disk for Ampro Little Board (note: NOT Little Board Plus). I am looking for general information on that machine, including articles, manuals, etc.
(2.) Need a HD driver for DOSPLUS IV for use on a Model 4 with a Radio Shack hard disk, any size. I'm not sure this exists, but I'm asking, anyway.
(3.) Need a HD driver for Montezuma Micro CP/M on a Model 4 with a VR DATA Hard Disk III, any size. I'm not sure this exists either, but I'm asking.
(4.) Need complete dBASE II V2.43* (CP/M) disks for Montezuma Micro, but any format is OK, as I can convert CP/M formats. I'm sure this exists!
Roy T. Beck. 2153 Cedarhurst Dr.
Los Angeles, CA. 90027 (213) 664-5059

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

WANTED: LISP, PROLOG, APL, ADA C++, etc. for Model III/4. Also RS Service Repair Manuals for Mod III/4, Forth Robot Arm for Mod III/4. Wish to contact folks who are in to Robotics, AI, Neural Networks, Nanotechnology.
R. Yves Breton. C.P. 95, Stn. Place D'Armes
Montreal, Quebec, Canada H2Y 3E9



ASSEMBLY 101

Z-80 without tears

Programming with SVC's

By Lance Wolstrup

Before we move on, let's briefly review the points made in our last installment.

Assembly language for Model 4 is identical to Model I & III with one major exception: No longer can we make ROM or DOS CALLs to specific addresses; instead, we must use the built-in SuperVisor Calls (SVC) made available by the authors of TRSDOS/LS-DOS 6.

Each SVC has an identity number. This number must be loaded into register A.

Some SVCs need other registers loaded with appropriate values to work correctly.

We learned to use the three following SVCs: @CLS, @DSP and @DSPLY.

@CLS is used to clear the screen. Its identity number is 105, and to use it, simply:

```
        LD      A,105
        RST     40
```

@DSP will send one character to the screen. The identity number is 2. This SVC needs the value of the character to be displayed stored in register C. Write this code to display an upper case A:

```
        LD      C,65
        LD      A,2
        RST     40
```

@DSPLY sends a string of characters to the screen. The SVC number is 10. In order for the SVC to find the string, register HL must point to the first character of the string. Thus:

```
        LD      HL,MSG1
        LD      A,10
        RST     40
```

OK, that ought to take care of the recap. Now, let's get on with the business at hand for this issue.

MORE SVCs

Putting text, such as prompts, on the screen is not particularly useful unless we can get a response from the user. In Basic we have the INPUT and INKEY$ commands to do this. INPUT allows the user to type a string of characters. Each character is echoed the screen, and the command is terminated by pressing the <ENTER> key. The INKEY$ function records a single keystroke. It is terminated whenever a key is pressed, and the key is not echoed to the screen. For now we will skip INPUT and concentrate on the INKEY$ function.

You have, undoubtably, seen Basic programs containing code that looked something like this:

```
100 PRINT"Press C to continue"
110 I$=INKEY$:IF I$=<>"C" THEN 110
```

The Model I & III AL version of INKEY$ is a simple call to 49H. The program would be written in this manner:

```
        ORG     7000H
START   LD      HL,MSG      ;point HL to MSG
        CALL    21BH        ;display MSG
INKEY   CALL    49H         ;@KEY
        CP      67          ;is it "C" ?
        JR      NZ,INKEY    ;no-back to INKEY
        RET                 ;continue with program
                            ;in this case, goto DOS
MSG     DEFM    'Press C to continue'
        DEFB    0DH
        END     START
```

Model 4 does not have the @KEY routine at 49H. It is located somewhere else in memory. We don't need to know where, because it is accessed by loading register A with the @KEY identity number, which is 1, followed by a RST 40. The value of the key pressed is returned in register A. Thus, the Model 4 version can be written as:

```
        ORG     3000H
START   LD      HL,MSG      ;point HL to MSG
        LD      A,10        ;@dsply
        RST     40          ;display MSG
INKEY   LD      A,1         ;@key
        RST     40          ;wait for keystroke
        CP      67          ;is it "C" ?
        JR      NZ,INKEY    ;no-back to INKEY
        RET                 ;continue with program
                            ;in this case, goto DOS
MSG     DEFM    'Press C to continue'
        DEFB    0DH
        END     START
```

As you can plainly see, the Model I/III version is identical to the Model 4 version, except the CALLs are translated to SVCs.

To demonstrate how the AL version of the INKEY$ routine works, we will write a short, relatively simple program that displays the directory of the drive chosen by the user. For the sake of convenience, we will call this program 'D'.

I know – we already have two such programs: DIR and CAT, so why do we need another? No other reasons than WE will write THIS one and, hopefully, learn something in the process.

However, before we begin coding, let's lay out exactly what we want the program to do.

1. Erase the screen.
2. Prompt the user for a drive number.
3. Allow the user to press a key.
4. Check if the key is a legal drive number (0 to 7).
5. Check if drive is ready.
6. Display directory.
7. Go back to step 2 and prompt for a drive number.

We will expand the key check in step 4 to also include "Q" and "q". Pressing either will terminate the program and return to DOS.

D/SRC

```
00050 ;D/SRC
00060 ;Model 4
00070 ;
00080 ;
00090 ;
00100          ORG     3000H
00110 ;
00120 ;step 1 - erase screen
00130 START    LD      A,105        ;@cls
00140          RST     40           ;clear screen
00150 ;
00160 ;step 2 - prompt for drive number
00170          LD      HL,MSG       ;point HL to MSG
00180          LD      A,10         ;@dsply
00190          RST     40           ;display prompt
00200 ;
00210 ;step 3 - allow user to press a key
00220 INKEY    LD      A,1          ;@key
00230          RST     40           ;wait for keystroke
00240 ;
00250 ;step 4 - check for legal characters (Q,q,0-7)
00260          CP      'Q'          ;is it "Q" ?
00270          JR      Z,QUIT       ;yes-goto QUIT
00280          CP      'q'          ;is it "q" ?
00290          JR      Z,QUIT       ;yes-goto QUIT
00300          CP      '0'          ;is it "0" ?
00310          JR      C,START      ;smaller-start over
00320          CP      '8           ;is it "8" ?
00330          JR      NC,START     ;if > "7"-start over
00340 ;
00350 ;step 5 - drive number ok - check if drive is ready
00360          SUB     30H          ;strip ascii value
00370          LD      C,A          ;copy drive # to C
00380          LD      A,33         ;@chkdrv
00390          RST     40           ;check if drive ready
00400          JR      NZ,ERROR     ;not ready-goto error
00410 ;
00420 ;step 6 - drive is ready - so display directory
00430          LD      A,34         ;@dodir
00440          LD      B,0          ;dodir function 0
00450          RST     40           ;display directory
00460 ;
00470 ;step 7 - issue prompt to press key to continue
00480 ANYKEY   LD      HL,ANYMSG    ;point HL to prompt
00490          LD      A,10         ;@dsply
00500          RST     40           ;display prompt
00510 ;
00520          LD      A,1          ;@key
00530          RST     40           ;allow key stroke
00540          JR      START        ;back to start
00550 ;
00560 ;exit to DOS - here if "Q" or "q" was pressed
00570 QUIT     RET                  ;return to DOS
00580 ;
00590 ERROR    LD      HL,ERRMSG    ;point HL to ERRMSG
00600          LD      A,10         ;@dsply
00610          RST     40           ;display error message
00620          JR      ANYKEY       ;go display ANYMSG
00630 ;
00640 ;
00650 MSG      DEFM    'Enter drive number: '
00660          DEFB    3
00670 ;
00680 ANYMSG   DEFB    10
00690          DEFM    'Press any key to continue'
00700          DEFB    13
00710 ;
00720 ERRMSG   DEFB    10
00730          DEFM    'Drive is not ready'
00740          DEFB    13
00750          END     START
```

In addittion to @KEY, the program used two new SVCs: @CKDRV and @DODIR. As the name implies, @CKDRV will check if a drive is in the system and a formatted diskette is in place. This SVC, number 33, needs to know which drive to check and register C is expected to contain that information.

```
        LD      C,drivenumber
        LD      A,33
        RST     40
```

On exit, the Z flag is set if the drive is in the system with a formatted diskette in place; otherwise, the NZ flag is set.

The other SVC, @DODIR, is a multifunction routine, each function relating to directory information. The SVC number is 34, and it also needs register C to be loaded with the drive number. In addition, register B must contain the number of the function to perform. These functions are as follows:

B=0 - displays the directory of the visible, non-system files on the disk in the specified drive. The filenames are displayed in columns, 5 filenames per line.
On exit, the Z flag is set if successful. AF is altered.

B=1 - the directory is written to memory. The HL register must point to a buffer to receive information.
On exit, the Z flag is set if successful. HL points to beginning of the buffer. AF is altered.

B=2 - a directory of the files on the specified drive is displayed for files that are visible, non-system, and match the extension partspec pointed to by HL. Register HL must contain a valid 3-character extension, padded with dollar signs ($). For example, to display all visible, non-system files that have the letter 'C' as the first character of the extension, HL should point to the string "C$$".
On exit, the Z flag is set if successful. AF is altered.

B=3 - a directory of the files on the specified drive is written to the buffer that is specified by HL for files that match the extension partspec pointed to by HL.
On exit, the Z flag is set if successful. HL points to beginning of buffer. AF is altered.

B=4 - the disk name, original free space, and current free space is read. HL must point to a 20-byte buffer to receive information.
On exit, the Z flag is set if successful. HL points to beginning of buffer. AF is altered. The disk name and free space information is stored in the format:

Bytes 0-7	Disk name. Disk name is padded on the right with blanks.
Bytes 8-15	Creation date, in the format MM/DD/YY.
Bytes 16-17	Total K originally available in binary LSB-MSB format.
Bytes 18-19	Free K available now in binary LSB-MSB format.

If any of the above functions are unsuccessful the NZ flag is set, and the error code is returned in register A.
Note that the directory information buffer consists of 18 bytes per active, visible file: the first 16 bytes of the directory record, plus ther ERN (ending record number). An FFH marks the buffer end.

Now let's move on to the @KEYIN, which will accept a line of input until terminated by <ENTER> or <BREAK>. The @KEYIN SVC acts like the Basic INPUT command, the one exception being that we are able to control the maximum allowable characters.
@KEYIN has the SVC number 9, and it needs several other registers to contain specific values before it can be executed.
Register B must contain the maximum characters allowed for input.
Register HL should point to a buffer that is one byte larger then the maximum characters allowed.
Register C is always 0.

```
        LD      A,9
        LD      B,15
        LD      C,0
        LD      HL,BUFFER
        RST     40
```

Let's write a little something to demonstrate the use of @KEYIN. The program, called NAC (for name, address & city) will prompt for three pieces of information which, upon completion, will be sent to the line printer.

NAC/SRC

```
00050 ; NAC/SRC
00060 ; for Model 4
00070 ;
00080 ;
00090 ;
00100          ORG   3000H
00110 START    LD    A,105      ;@cls
00120          RST   40         ;clear screen
00130          LD    HL,PRMPT1  ;point HL to 1st prompt
00140          LD    A,10       ;@dsply
00150          RST   40         ;display prompt
00160          LD    A,9        ;@keyin
00170          LD    B,20       ;max chrs=20
00180          LD    C,0        ;C is always 0
00190          LD    HL,BUF1    ;point HL to input buffer
00200          RST   40         ;get input
00210          LD    HL,PRMPT2  ;point HL to 2nd prompt
00220          LD    A,10       ;@dsply
00230          RST   40         ;display prompt
00240          LD    A,9        ;@keyin
00250          LD    B,30       ;max chrs=30
00260          LD    C,0        ;C is always 0
00270          LD    HL,BUF2    ;point HL to input buffer
00280          RST   40         ;get input
00290          LD    HL,PRMPT3  ;point HL to 3rd prompt
```

```
00300          LD    A,10        ;@dsply
00310          RST   40          ;display prompt
00320          LD    A,9         ;@keyin
00330          LD    B,40        ;max chrs = 40
00340          LD    C,0         ;C is always 0
00350          LD    HL,BUF3     ;point HL to buffer
00360          RST   40          ;get input
00370          LD    HL,PRMPT4   ;point HL to prmpt
00380          LD    A,10        ;@dsply
00390          RST   40          ;display prompt
00400          LD    A,1         ;@key
00410          RST   40          ;wait for key press
00420 ;
00430          LD    HL,BUF1     ;point HL to input
00440          LD    A,14        ;@print
00450          RST   40          ;print 1st input
00460          LD    HL,BUF2     ;point HL to input
00470          LD    A,14        ;@print
00480          RST   40          ;print 2nd input
00490          LD    HL,BUF3     ;point HL to input
00500          LD    A,14        ;@print
00510          RST   40          ;print 3rd input
00520          RET               ;return to DOS
00530 PRMPT1   DEFM  'Name: '
00540          DEFB  3
00550 PRMPT2   DEFM  'Address :'
00560          DEFB  3
00570 PRMPT3   DEFM  'City, State & Zip: '
00580          DEFB  3
00590 PRMPT4   DEFM  'Turn printer on - press any key '
00600          DEFB  3
00610;
00620 BUF1     DEFS  21
00630 BUF2     DEFS  31
00640 BUF3     DEFS  41
00650          END   START
```

This was a simple and straight-forward program. We did, however, use another new SVC: @PRINT.

The @PRINT SVC has the number 14. Register HL must be pointed to the start of the text to be sent to the printer. In the above case, since the text to be printed came from a user input and was therefore stored in buffers, register HL pointed to BUF1 for the first text, BUF2 for the second text, and BUF3 for the final text.

```
LD     A,14
LD     HL,BUFFER
RST    40
```

On exit, if successful, the Z flag is set; otherwise the NZ flag is set and the error number is stored in register A.
Registers AF and DE are both altered.

There is one other way to get information to the line printer. Whereas @PRINT sends a string of text, the @PRT SVC will output one byte to the printer.

@PRT is number 6, and it needs register C to be loaded with the character to print.

```
LD     A,6
LD     C,65
RST    40
```

On exit, if successful, the Z flag is set; otherwise the NZ flag is set and the error number is stored in register A.
AF and DE are both altered.

This SVC is useful for sending printer codes, so let's finish up for this time with an example that will switch the Radio Shack DMP series of printers into boldface mode.

```
        ORG    3000H
START   LD     HL,CODES     ;point HL to printer codes
        LD     C,(HL)       ;get code to C
        LD     A,6          ;@PRT
        RST    40           ;send code to printer
        INC    HL           ;point to next code
        LD     C,(HL)       ;get code in C
        LD     A,6          ;@PRT
        RST    40           ;send code to printer
        PUSH   HL           ;save code pointer
        LD     HL,MSG       ;point HL to text
        LD     A,14         ;@PRINT
        RST    40           ;send text to printer
        POP    HL           ;restore code pointer
        INC    HL           ;point to next code
        LD     C,(HL)       ;get code in C
        LD     A,6          ;@PRT
        RST    40           ;send code to printer
        INC    HL           ;next printer code
        LD     C,(HL)       ;get code in C
        LD     A,6          ;@PRT
        RST    40           ;send code to printer
        RET                 ;return to DOS
CODES   DEFB   27           ;27 and 31 turn the
        DEFB   31           ;boldface mode on
        DEFB   27           ;27 and 32 turn the
        DEFB   32           ;boldface mode off
MSG     DEFM   'This should print in boldface'
        DEFB   13
        END    START
```

In the next issue we will put all of this together and write the Model 4 version of TRSLABEL/CMD, maybe even adding some features missing from our Model I/III version from last year.

Until next time..................................KEEP PRACTICING!

If you don't have a hi-res board for your Model 4, get one NOW!

GIF4MOD4 VERSION 2 IS HERE

If you DO have a hi-res graphics board, GIF4MOD4 is the REASON you bought it!
There are literally thousands of BEAUTIFUL photographic and original art images of subjects ranging from Disney characters to 'XXX' rated".
These wonderful GIF images are available FREE from the information services and from BBSs nationwide.
But you'll never see ANY of them on YOUR Model 4 without GIF4MOD4!
You've seen the "rave" reviews of the original GIF4MOD4 --
-- but GIF4MOD4 VERSION 2 is up to 60+% faster and 100% better!
New features like brightness and contrast controls, interlaced image display, distortion correction, and a brand-new color-to-B&W conversion technique not found in ANY other program make GIF4MOD4 VERSION 2 one of the most advanced GIF programs for ANY hardware at ANY price.
Speaking of price, that's about the only thing that hasn't changed. GIF4MOD4 VERSION 2 is only $37.95 + $2 S&H. Outside North America add $2 for airmail.
VA. add 4 1/2% tax.

J.F.R. "Frank" Slinkman
4108-C Fairlake Lane
Glen Allen, VA. 23060

NEW PROGRAMS

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

Send S.A.S.E. for annotated list
Sample disk $5.00 (U.S.)

VTHG
BOX 9747
N. HOLLYWOOD, CA. 91609

STORAGE POWER

Models I, III, 4, 4D, 4P

HARD DISK DRIVES

5 Meg Hard Disk			$345.00
10 Meg $425.00	15 Meg $495	20 Meg	$545.00
30 Meg/up $ Call	Bare hard drive bubbles available		$ Call

Hard disk drivers for

LDOS, LS-DOS, TRSDOS 6.X	$ 49.95
MULTIDOS 80/64, III	$ 39.95

10 Meg. Removable Hard Disk Drive

External (complete)	$1,145

III/4 INTERNAL DISK DRIVE KITS

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

2 FH Drives	$149.95	4 HH Drives	$159.95
FD Controller only			$ 89.95

EXTERNAL DISK DRIVES

Complete w/case, power supply and cable

2-40 trk HH DS DD	$249.95	2-80 trk HH DS DD	$269.95
1-40 & 1-80 trk	$259.95	2-3.5" 80 trk	$299.95
1-80 trk FH DS DD			$119.95

BARE DRIVES

40 trk DS DD FH refurb. 360k (replacement for SS Mod III & 4)	$ 69.95
40 trk DS DD HH 360k	$ 89.95
80 trk DS DD HH 720k	$ 99.95
80 trk DS DD FH 720k	$ 49.95
3.5" 80 trk 720k	$114.95

DRIVE CASES w/Power supply

Hard drive - 1 FH or 2 HH w/fan	$109.95
Floppy - 1 FH or 2 HH	$ 59.95

MOD 4 MEMORY SETS

8 4164 - 200ns new	$19.95	Pulls	$ 15.95
8 4164 - 150ns new	$24.95	Pulls	$ 19.95
Pal chip for non Gate/array			$ 14.95

MOD 4 SPEED UP KITS

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

DISKETTES

w/sleeves & labels

Pkg of 10 (5.25")	$ 4.25	Pkg of 10 (3.5")	$ 11.95
Pkg of 25 (5.25")	$ 9.95	Pkg of 25 (3.5")	$ 25.95
100 - 5.25" Disk storage w/lock			$ 11.95
70 - 5.25" Disk storage w/lock			$ 9.95
40 - 3.5" Disk storage w/lock			$ 8.95
80 - 3.5" Disk storage w/lock			$ 12.95

MISCELLANEOUS

65w Power supply	$29.95	CRT Tube green/amber	$ 79.95
Model I Double Density Board			$ 89.95
Cables:Printer 6ft	$14.95	Printer 12ft	$ 19.95
Disk drive - 2 drives 3ft			$ 9.95
Custom cables			$ Call

We can supply most of the parts (new and used) that you will need in repairing & upgrading your Mod I, III & 4. Call or write for availability & price.

STORAGE POWER

10391 Oakhaven Dr. Stanton, Calif. 90680
(714) 952-2700 (9:30 am - 8:00 pm PST)
All C.O.D. orders are cash only.
Prices are F.O.B. and subject to change and availability.
Calif. residents require 6.25% sales tax.
NEW BBS (714) 952-8666 8/1/N