NORTHERN BYTES

Volume 5 Number 6

GREETINGS! Welcome to the late summer/early autumn edition of NORTHERN BYTES. It seems strange to realize that we've already published six issues of NORTHERN BYTES this year. For an "irregular" publication, we've been getting out pretty regularly!

Now that 80-Micro is beginning to look more and more like MAD Magazine (see their September, 1984 issue), it may be that we are the only serious publication left for the TRS-80 Model I/III/4 user. How long can we keep doing it? Well, suppose we decided to offer subscriptions. To paraphrase the TV announcer, "How much would YOU pay for six issues of this fine newsletter?" While you're thinking about it, keep in mind that so far, we haven't had much advertising, and that almost all of the material appearing in NORTHERN BYTES can be considered to be in the public domain, meaning you can use it as you see fit (for non-commercial purposes, anyway).

I hate to say it, but the day may come when NORTHERN BYTES will either have to commit itself to becoming a subscription item, or discontinue publication. Editors (and Publishers) have to eat, too.

I've received a few complaints about the content of NORTHERN BYTES - such as, "Why don't you publish more programs in BASIC instead of assembly language" or "Why don't you publish some programs for non-disk users (cassette, Stringy-Floppy, TC8, etc.)" The reason is simple - I can't publish what I don't have. Most of the material in NORTHERN BYTES has been contributed directly to us by folks like YOU. If I don't get any articles on a given subject, I can't publish them. At this point we "reject" only a very small percentage of the articles we receive. If you sent us something and you haven't seen it published yet, you may want to check and make sure we received it (unless you sent it in the last month or two, in which case it may appear in the next issue of NORTHERN BYTES).

I will point out that I know of at least two instances where someone mailed us a disk, and it never arrived here. One disk was traced as far as Detroit, where it suddenly disappeared into a "black hole" (or maybe someone's overcoat), never to be seen again. When you send diskettes to NORTHERN BYTES, if you have easy access to United Parcel Service you might be better off to send them that way. Note that MOST diskettes sent to us DO arrive safely, so don't put off sending us something for fear that it might get lost.

On a related note, if you are sending something short and don't want to use a diskette, PLEASE consider sending it electronically via MCI Mail. Folks, I am NOT a touch-typist. The three paragraphs that took you five minutes to type will probably take me twenty minutes to re-type (no, I don't have secretarial help!). It costs you ABSOLUTELY NOTHING to register for MCI Mail, and there are NO monthly service charges or connect-time charges. An article elsewhere in this newsletter gives the procedure for registering with MCI Mail.

What about personal replies from the editor? On this point I have to apologize, but I'm afraid that I am going to have to curtail those. Unfortunately, I have only so much time, and I can either spend it answering mail, or working on NORTHERN BYTES and similar projects. So, effective immediately, the following policy is in effect: 1) If you need to contact me, you may telephone me, at your expense, and with the understanding that if I'm not home and you get my wife, you may leave a message asking me to call you back COLLECT (or you may try again later). I'm usually up until at least 11:30 Eastern Time, sometimes later. My number is (906) 632-3248. 2) You may write to me, but if you want a reply you MUST send a SELF-ADDRESSED, STAMPED ENVELOPE, and even then I do not guarantee a reply. Exceptions: Canadians send a self-addressed envelope and a Canadian quarter if you don't have U.S. postage. From other countries, if all you want is a simple one or two page reply then don't worry about sending an envelope or return postage (unless you get in the habit of writing often), it's too much hassle to cash in International Reply Coupons and the like. Again, please note that I do NOT guarantee a reply to every letter I receive, even if you DO send the SASE, but I will NOT reply without the SASE. It's a case of "You pays your money and you takes your chances." Please be assured that I do READ every piece of mail I receive, and if you're sending a question for which I do not have an answer, it may appear here in NORTHERN BYTES to give our readers a chance to help.

"What brought on this policy?", I hear you ask. Well, it was just that earlier this summer, I moved - sort of. My address is still the same, though. What happened was that I sold my old mobile home, and got a larger one which was placed on the same lot that my old one had been on (that's why my address hasn't changed). Unfortunately, during this "move" I had only very limited use of my computer for about a month and a half. The problem was that my computer area is not physically located in the mobile home itself, but rather in a 12'x12' added room which we retained. In order to be attached to our new mobile home, the added room had to be physically moved forward about twenty feet, and it took the company that was going to do the job nearly a month to get around to doing it (despite many broken promises that the job would be done sooner). We finally wound up getting someone else to finish the job, after the delay (and the lies) became intolerable.

During most of this period my system was effectively "down", but the mail just kept on pouring in, and most of it was from folks that wanted at least some sort of answer. Since, as I have mentioned, I am not a particularly fast typist, it would have taken me a couple of months to personally answer each letter once I got my system up and running again. So, what happened was that many of these folks got a "form letter" reply. However, I'm sure that some pieces of mail got misplaced during the move, so if you wrote and didn't hear anything at all from me, you might want to try again.

In any case, I felt totally "swamped" for a while, and decided that unless and until I get into a position where I can afford a secretary to answer mail, I'm just not going to promise replies to anyone. Since I don't have to type replies to phone conversations, I don't mind phone calls. I will also point out that if I owe you anything (for example, a diskette) it might be wise for you to drop me a postcard and remind me, since a lot of things got moved around here, never to be seen again (we STILL can't find our pencil sharpener)!

Well, enough of the editorial comments. I hope that you and yours had a nice summer (or nice winter, if you live in the Southern Hemisphere). By the way, it is NOT true that Sault Ste. Marie only has two seasons (winter and the Fourth of July). We also have an Indian Summer in August (usually around the second weekend). So there.

THE EXTERMINATOR - Summer's almost gone, but the BUGS linger on. Laurie Shields of Chesterfield, England passed along the information that the NEWDOS/80 PDRIVE settings that we published in Volume 5, Number 3, that supposedly would read LDOS double density disks were all wrong. As Laurie put it, "Granted they allow you to read the first 10 sectors of the directory but as for getting the files they would make a right mess of a disk." I'd be interested to know how anyone else made out with those PDRIVE settings, but for now, I wouldn't use them with anything that hasn't been properly backed up first!

Paul Snively's article on "BACKING UP MORE RECENT VERSIONS OF SUPER UTILITY PLUS" had a slight problem - his method wouldn't work on a Model III or 4! Paul explains:

"Here is the listing to a program that was written at the request of Mike Davis from San Diego, California. It seems that he tried my 'SU+ as a /CMD file' technique from Northern Bytes Volume 5, Number 4 and it made the /CMD file OK, but shortly after executing the thing, it'd crash! Well, it seems that he was getting a serial number checksum of zero. Not only that, he got zero for every copy of SU+ that he tried! Well, he was using a Model III, so I tried using my technique on a pristine SU+ that I happen to have here, and I did so on the Model 4. Sure enough, zero! And the /CMD file crashed just like Mike said. Nuts.

"Well, I assumed that the III (and 4) alter the contents of the I register upon boot-up, thereby making it impossible to get an accurate serial number checksum by using the method that I mentioned (which works just fine on a Model I.) So, here is a program which generates a checksum of the serial number given to it. NOTICE THAT THE CORRECT SERIAL NUMBER OF YOUR COPY OF SU+ CAN ONLY BE GOTTEN BY HOLDING DOWN THE 'S' AND THE <CLEAR> KEYS FROM AN SU+ MENU!!!

```
5200            00100           ORG     5200H           ;As good a place as any
5200 CDC901     00110 START     CALL    01C9H           ;Clear the screen
5203 216452     00120           LD      HL,PROMPT       ;Get prompt address
5206 CD6744     00130           CALL    4467H           ;Display the prompt
5209 218E52     00140           LD      HL,BUFF         ;Point to buffer for input
520C E5         00150           PUSH    HL              ;Save address for later
520D 118F52     00160           LD      DE,BUFF+1       ;Point to address after BUFF
5210 010F00     00170           LD      BC,15           ;# of bytes minus one
5213 3620       00180           LD      (HL),20H        ;Byte to fill with
5215 EDB0       00190           LDIR                    ;Bombs away!
5217 E1         00200           POP     HL              ;BUFF back to HL
5218 0610       00210           LD      B,16            ;16 characters max. in serial#
521A CD4000     00220           CALL    0040H           ;Get a line from keyboard
521D 38E1       00230           JR      C,START         ;START over on <BREAK>
521F 48         00240           LD      C,B             ;Get actual count to C
5220 0600       00250           LD      B,0             ;Zero out B
5222 09         00260           ADD     HL,BC           ;Point to end of string
5223 3620       00270           LD      (HL),20H        ;Blank out 00H
5225 218E52     00280           LD      HL,BUFF         ;Point to beginning of BUFF
5228 011010     00290           LD      BC,1010H        ;16 chars, Zero checksum
522B 7E         00300 LOOP      LD      A,(HL)          ;Get a character
522C ED44       00310           NEG                     ;NEGate it
522E 81         00320           ADD     A,C             ;Add to checksum total
522F 4F         00330           LD      C,A             ;Update total
5230 23         00340           INC     HL              ;Bump pointer
5231 10F8       00350           DJNZ    LOOP            ;Finish checksum
5233 FE40       00360           CP      40H             ;Less than 40H?
5235 3003       00370           JR      NC,NA           ;No, No Adjustment
5237 C640       00380           ADD     A,40H           ;Bump by 40H
5239 4F         00390           LD      C,A             ;Adjust total
523A 217C52     00400 NA        LD      HL,RESULT       ;Point to RESULT string
523D CD6744     00410           CALL    4467H           ;Display it
5240 79         00420           LD      A,C             ;Get checksum
5241 E6F0       00430           AND     0F0H            ;Mask out low bits
5243 CB3F       00440           SRL     A               ;Move high bits to low
5245 CB3F       00450           SRL     A
5247 CB3F       00460           SRL     A
5249 CB3F       00470           SRL     A
524B CD5D52     00480           CALL    ASCII           ;Convert 0-F to "0"-"F"
524E CD3300     00490           CALL    0033H           ;Display digit
5251 79         00500           LD      A,C             ;Get checksum
5252 E60F       00510           AND     0FH             ;Mask out high bits
5254 CD5D52     00520           CALL    ASCII           ;Convert 0-F to "0"-"F"
5257 CD3300     00530           CALL    0033H           ;Display digit
525A C32D40     00540           JP      402DH           ;Back to DOS
525D C690       00550 ASCII     ADD     A,90H           ;Quick hex-ASCII convert
525F 27         00560           DAA
5260 CE40       00570           ADC     A,40H
5262 27         00580           DAA
5263 C9         00590           RET
5264 57         00600 PROMPT    DEFM    'What is your serial #? '
     68 61 74 20 69 73 20 79 6F 75 72 20 73 65 72 69
     61 6C 20 23 3F 20
527B 03         00610           DEFB    03H
527C 54         00620 RESULT    DEFM    'The checksum is: '
     68 65 20 63 68 65 63 6B 73 75 6D 20 69 73 3A 20
5280 03         00630           DEFB    03H
0010            00640 BUFF      DEFS    16
5200            00650           END     START
00000 TOTAL ERRORS

ASCII   525D    BUFF    528E    LOOP    522B    NA      523A    PROMPT  5264
RESULT  527C    START   5200
```

"There! Now all of those Model III/4 owners who read Northern Bytes and wondered why my SU+ /CMD file technique didn't work can try again!"

Also in Volume 5, Number 4 we ran a short article entitled "CREATE A SELF-BOOTING DISKETTE USING NEWDOS/80" by Joachim Kelterbaum. Gary Bryce of Sydney, Australia tried to use the method shown, but found it wouldn't work for him (it kept coming up with "SYS ERROR" after boot). Gary goes on to state:

"Not to be deterred I investigated further and worked out how to do it. The procedure works for single density disks on a Model I, and double density disks on a Model III.

"Format a data disk (no system on it) - note that the TSR (Track Step Rate) for boot must be set correctly by PDRIVE before formatting the data disk. Now copy the /CMD file to be made self booting to the formatted data disk. Use the DIR d command (where d is the number of the drive containing the da. disk) to ensure that the file was saved in one contiguous block (usually it will have only one extent, as listed in the EXTS column of the directory display, the exception being those files over 32 grans or 41K long).

"Now use the DFS option of SUPERZAP to read relative sector 0 of the file, note the DRS (Drive Relative Sector) of this sector, return to the SUPERZAP menu (X), select the DTS option and enter the drive number and DRS, the resulting display will show the starting Track (TRK) and Sector (TRS) numbers of the file (for Mod I double density only, add 1 to the TRK). Using SUPERZAP, patch the following bytes of sector 0 in BOOT/SYS of the data disk in the following manner:

MODEL I -
Byte 12 - Start Sector # of the /CMD file.
Byte 13 - Start Track # of the /CMD file.
Byte 4B - change from C8 (RET Z) to C9 (RET).
MODEL III -
Byte 04 - Start Sector # of the /CMD file.
Byte 05 - Start Track # of the /CMD file.
Byte 3E - change from C8 (RET Z) to C9 (RET).

(Please note that I have not verified the Model III mod, as I have a Model I.) Now a few words about what types of files can be made self-booting. Generally speaking, any file which has no calls into DOS would be suitable (e.g. Cassette based games, utilities)."

Gary Bryce is the editor of SYDTRUG NEWS, the newsletter of the Sydney TRS-80 Users Group, and the above comments by Gary were reprinted from that publication. Their mailing address is P.O. Box 43, Erskineville 2043, AUSTRALIA.

Finally, on page 19 of Volume 5, Number 4 I reviewed Scripture Software's BIBLE SEARCH program. The current version is 1.4 and now includes the Old Testament as well as the New. In addition, a new method of text compression is used wh makes the disk files occupy less room on the disks - the enti. Bible (Old and New Testaments) now occupies only 16 single sided single-density disks (for those of you not familiar with "text compression", note that this does NOT mean that any part of the scriptures have been omitted or "condensed". It simply means that they have been saved on disk in such a way as to occupy fewer bytes on the diskette). For further information on pricing and disk formats available, contact Scripture Software, P.O. Box 6131-C, Orlando, Florida 32853 or telephone (305) 896-4264.

By the way, many of you felt I was a bit too hard on the King James Version of the Bible. Well, maybe, but I still can't understand King James era English very well, so I'll stick to a more modern translation. The author of Bible Search feels that the New International Version is more of a paraphrase than an actual translation, and also that some of the so-called "more reliable manuscripts" used in the newer versions really aren't (more reliable, that is). For folks like me who simply can't/won't read the original King James, he prefers the New King James Version, which is simply the original King James updated to modern English (yes, the "Thees" and "Thous" have been removed). In any case, I didn't mean to step on anyone's toes - except maybe those few narrow-minded individuals that insist that the King James Version is the ONLY Bible worth reading.

LETTERS DEPARTMENT - We've received quite a bit of mail this time around, so let's get right to it:

Date: Fri Jun 22, 1984 10:27 pm EDT
From: Paul Snively / MCI ID: 176-6817

TO: * Jack Decker / MCI ID: 102-7413
Subject: Mod 4 routines

June 22, 19(

Dear Jack,

I had a rather enlightening chat with Ted Carter of Micro-Labs, Inc. today. These are the people who brought us the Graphyx Solution hi-res boards for the Models III and IV, and the character generator board for the Model I. Anyway, he was kind enough to give us some address equivalencies from the III to the IV over the phone. The Model III address is on the left, and the equivalent address on the IV running under TRSDOS 6.1.2 is on the right. NOTE THAT THE MODEL IV ADDRESSES ARE ONLY VALID UNDER TRSDOS 6.1.2!!!

MODEL III	MODEL IV
260DH	39FBH
4121H	72ECH
40AFH	7142H
0A7FH	29F4H
0BF2H	2BBEH
0716H	2622H
1541H	38EAH
1547H	38F0H
19A2H	5886H
0847H	278BH
0A0CH	2982H
08A2H	27EDH
2337H	7345H

Well, that's about it. Also, the "normal" Model III RST 8, 10, and 18 routines are not valid on the IV. Ted said that he had to write those routines himself. So, here is the info. Feel free to publish in Northern Bytes as long as credit is given to Ted Carter of Micro-Labs, Inc. 902 Pinehurst, Richardson, Texas 75080. Talk to you later...

Regards,
Paul Snively

Date: Sat Jun 23, 1984 9:53 am EDT **RECEIPT
From: David R. McGlumphy / MCI ID: 181-7759

TO: * Jack Decker / MCI ID: 102-7413

Here's a letter that offers a challenge to someone who's looking for a programming challenge. I offer it to you strictly (!) verbatim.

"Dear Al: In the Feb 1982 (Pg 23) TRS-80 Microcomp News you have almost given me the answer to a question that the "Shack" experts say cannot be done.

I have a Model 4 (tape only) 64K upgrade, DMP 100 printer & Shack Recorder. I would like to conceal my programs from LIST, LLIST, CSAVE yet OK on RUN & CLOAD. I used your POKE PEEK(16548) + 256 * PEEK(16549), PEEK(16548) : POKE PEEK(16548) + 1 + 256 * PEEK (16549), PEEK(16549) and after the first RUN it will not list llist but the first line - any attempt to list a portion will lock it up for a re-Cload. It can be CSAVED & listed before the first RUN & CSAVED after the first RUN.

I am attempting a Password type operation with tape. Any simple solutions or can it be done?

Leonard M. Brown
R1 Box 277C
Denison, Texas 75020"

It's been a long time since we fooled around with tape, hasn't it, boys. I don't have any handy code already available to help Mr. Brown, nor do I have the time to help him more than this now, but this sounds like a fun project. I think I see a way to do it using the SYSTEM command to load a tape, but it also seems like I remember an article about CLOADing machine language. Is my memory failing me? Have at it, old-time hackers.

[Editor's note: Dave reports that he sometimes goes by the name Al Reudisuelli(?!). This explains the introduction to the above letter. If you knew Dave McGlumphy, you'd realize why he might have to use a phoney name sometimes. He once sent me a letter in a very official-looking envelope. The return address was "Federal Health Department, 1416 Government Street S.W., Washington, D.C. 83615" and in big letters on the outside of the envelope, it said, "HERE IS THE INFORMATION YOU REQUESTED ON HOW TO TREAT THE HERPES VIRUS AT HOME." Since then, I've noticed that my mailman puts on a pair of gloves before he opens my mailbox, and he refuses to bring postage-due mail to my door. Thanks, Dave. Your day will come (by the way, Dave's address is 4429 Paula Lane, Red Bank, Tennessee 37415 in case any of you practical jokers would like to trade notes).]

Date: Sat Jun 30, 1984 3:29 am EDT
From: David Dalton / MCI ID: 183-6752

TO: * Jack Decker / MCI ID: 102-7413
Subject: Northern Bytes...

Jack: I'm a regular customer of the Alternate Source, and I want to tell you how much I appreciate Northern Bytes.

A question: Do you know of any problems in using Superscripsit with NEWDOS80 (2.0 and 2.5, Model III) that are not corrected by Apparat's zaps? I have had the most peculiar thing happen repeatedly. When I use a programmable user key, the computer seems to drop to ROM BASIC and says Memory size? This continues until I reprogram any key, then the problem goes away for a while. I am now using Superscripsit 1.02.3, which, as far as I know, is the most recent. The same thing happened with earlier versions. As far as I know, my memory is perfect. The computer has been in for regular maintenance twice in the last 6 months. I have learned not to let it clobber my files by <W>riting to disk before I use a programmable key. But I must admit I *like* Superscripsit, and I am committed to using it to edit an 85,000 word novel. Too late to turn back now. If you know the answer I'll be delighted, and if answer by MCI I'll gladly send a donation to Northern Bytes, for I imagine your MCI bill must be a sight. I am using a Model III, NEWDOS 2.5, and a Compukit 10-meg hard disk. Thanks a bunch, and, again, Long Live Northern Bytes.

- David Dalton, 3558 Bowens Road, Tobaccoville, North Carolina 27050

[I am sysop of SF Writers Network, (919) 922-3308, 24 hours, 300/1200, TBBS].

[Editor's note: If anyone can help David, please contact him directly as I can imagine he's pretty desperate for an answer if he hasn't found one by now. Also please send a copy to us here at NORTHERN BYTES and we'll share the answer with everyone.]

Dear Jack,

Would you please help me by mentioning a problem I have with the Model 4P in Northern Bytes?

At the moment I have both a Model 4 and a 4P, but economics dictate that the 4 must go in favour of the portable. The trouble is that my program library is on 80 track double head (external) drives. The Model 4P does not support external drives though it does support an external hard disk via the 50 pin I/O. I have taken the 4P apart and attached a new drive cable (no pins omitted) which places the disk controller in the middle, the internal drives at the start and the external drives at the end of the chain. Despairingly, the 4P will only acknowledge drive 0 or 1, in any variation or multiples of drive 0 and 1. It would appear that the drive select lines are not implemented for the external drives (?). Can anyone help?

The Model 4P technical manual is not available in' Australia, is it available yet in the U.S.? I would also appreciate you mentioning that I would like to communicate with other TRS-80 users in the U.S. or Canada.

I look forward to each issue of Northern Bytes, keep up the good work, it is appreciated.

- Tony Domigan, P.O. Box 150, Thomastown, Victoria 3074, AUSTRALIA

[Editor's note: Once again, I suggest you write directly to Tony if you can help him, with a copy to us here at NORTHERN BYTES so we can pass on the solution to everyone. However, if you prefer you can send a reply to us and we'll forward it to Tony. My first thought was to wonder if, in the native Model 4 mode, a command such as SYSTEM (DRIVE=2,ENABLE) would have to be executed before an additional drive could be used, but since Tony also has a regular Model 4 I'm sure his copies of TRSDOS 6 are already set up to enable the additional drive. As for the 4P technical manual, I haven't heard anything on that, either. Anyone have any information they'd care to pass along?]

Dear Jack,

Thank you for Volume 5 Numbers 3 and 4. I have a number of comments and suggestions concerning these issues.

You mention your book on TRS-80 ROM Routines several times, without mention of price or availability. How much and where?

I think you missed the point in your review of Volume 6 of Dr. Dobbs. As the author of the only TRS-80 specific program in the volume I do not consider the tape routine useful any longer (Model I, tape only), but there was also a BASIC program which allowed embedded, relocatable assembler routines in hexadecimal form. I still use a similar routine, refined slightly since then. I am not a BASIC fan, but sometimes BASIC is quicker but the unrelocatable decimal values really bug me! So far I have never seen any published BASIC programs that use the method shown. If you are interested I could send an updated routine.

Concerning multiple double-sided disk drives on the Model I, there is a very simple method for expanding your system to more than 3 double sided or 4 single sided drives. The proviso is that only 3 or 4 of them can be used at any one time. I have 4 single sided drives, 2 80-track, one 40-track, and one 8-inch. I can boot from either 40 or 80 track disks with the flick of a switch. The method can be used to expand the number of drives connected. The switch is used to open or close the drive select line to one or more of the drives, or to swap the allocation of a drive. The leads can easily be mounted on the posts usually used for the drive select jumpers. Further details if desired.

Concerning the number of TRS-80 users "still out there". There may be several hundred thousand of them, but they do not seem to actively encourage new software development by buying software for the computers. After spending probably about 2000 hours developing a sophisticated program for the Model I, III, and 4 I have been disappointed by the user response. I'm not disappointed by the responses from users, but by the number of users. As an almost final resort I have thought of giving it away and asking for user contributions to cover costs.

Concerning the method for repositioning an output file to write new information at the beginning. I have recently used the method, but have not been able to test it on TRSDOS 1.3. Do you have any alternative for this operating system? I have tried it on Model I systems, and on TRSDOS 6 on the Model 4 without problems.

Concerning the backup of Super Utility + to a /CMD file, I have a method I have used on several versions of the program. It was developed for a friend who uses SU+. Personally I prefer Superzap and a homemade zap program. This method makes use of the fact that SU+ can modify memory, including SU+ itself. Simply enter a short routine which will save all registers, move the low portion to high memory, and then modify the program to jump to this routine on some event such as the SHIFT-BREAK key. The NEWDOS/80 DUMP command can then be used to save the result on disk. A routine then has to be added to restore the program and the registers, and move it back to low memory. Be careful to save the interrupt register, since SU+ checks this register for "illegal" tampering. Further details are available if desired.

Concerning Nate Salsbury's "Patch of a Patch". He insists on spelling my name incorrectly! He should stick to Arne, and leave the surname out of it.

Concerning the green or amber monitors for TRS-80 models. A friend bought the amber screen from Langley St. Clair for the Model III, plugged it in directly, and had no problems at all with it. About $100 if I remember rightly. I still have my original Model I monitor, which was modified for 220 Volts with an internal transformer. It was placed right next to the tube, causing it to have constant shakes. Now the transformer is removed, and I am using an external unit delivering 110 Volts, plus the voltages required by the expansion interface and the keyboard unit.

Concerning the reading of IBM diskettes on the TRS-80. Over a year ago I wrote a simple zap program to do this, being both able to read and write. I also modified Superzap at the time by moving the buffer. Instead of going to Debug, I found it easier to use the DM command in Superzap itself to display the sector. The zap program used to copy files used the NEWDOS/80 routines to read and write, and functioned by setting an appropriate PDRIVE value. By the way, the sectors are numbered from 1, not 0. The copy program was written without any information on the disk format, so it would only read contiguous files, and I have not bothered to update it since then. Also it would not read double-sided disks, since I had no means for testing this. I have not given up hope of extending the program some day, but would like to be able to read several other formats as well, such as the Model 4 CP/M formats, and perhaps also the Color Computer if I could get hold of a disk and a description of the format.

Concerning new commands in NEWDOS/80 or other operating systems. There is a very simple method for adding short commands, containing no more than 248 bytes of code. The command is simply assembled to reside in the DOS disk buffer area (4300H or 4400H, or perhaps other values). The single sector can then be used as a command without disturbing the remainder of memory. I have used it for setting up the printer, which I usually forget before entering programs. Many of my programs have been modified to allow calling NEWDOS/80 commands directly by preceding them with a /. For example my EDAS, word processor, VisiCalc, etc.

By the way, talking of EDAS 3.5, there is a bug in the cross-reference program. If you define enough names or references to fill up memory, then the cross-reference listing program will bomb out without warning. Mine has now been disassembled and modified to allow several passes through the file. It will run as long as all names using a single starting letter will fit in memory.

Also concerning EDAS 3.5, Roy Soltoff seems pleased that it will not run under NEWDOS/80. The only function which will not operate is the ability to specify some parameters on the program call line. The call to the routine which is not defined in NEWDOS can easily be bypassed, and Superzap can be used to modify the parameters if necessary. I am currently using a highly modified version of EDAS 3.5 under NEWDOS/80 without problems.

I think that's about all for those two issues. Keep up the good work now that there are no serious magazines left for the TRS-80 user.

- Arne Rohde, Pilevej 31, 7600 Struer, DENMARK

[Editor's note: Arne brings up several interesting points in his letter. Here are some comments in response: My book sells for $19.95 plus shipping through The Alternate Source (shipping is $3.00 in the U.S. and Canada). The method of repositioning an output file seems to work on everything EXCEPT TRSDOS 1.3 I have no idea why, but then TRSDOS 1.3 is my second l favorite DOS (in case you're wondering what my LEAST favorite DOS is, I'll just say that I like DOSPLUS, MULTIDOS, and NEWDOS/80 better than TRSDOS 1.3). Maybe some of our readers could help on this one. I think that breaking the protection on Super Utility has become something of a mania among TRS-80 users, everybody's got a method of doing it. I apologize for the misspelling of Arne's surname, on behalf of Nate Salsbury and NORTHERN BYTES. It certainly does seem to make more sense to use Superzap's DM function instead of exiting to DEBUG to read the expanded buffer. I don't know anything about the format of the disk directory on a Color Computer (but would like to, since I might have to try and read a CoCo disk someday), can anyone help with this? Finally, I hope Arne will document some of the patches and zaps he's mentioned and share them with the rest of us!]

Dear Jack:

Thanks for sending me a copy of Northern Bytes. That particular issue had little to interest a cassette-based amateur like me, but I was impressed with the expertise of your correspondents in patching machine language commercial software.

Although I'm cassette-oriented by choice, there is one area where it is slowing me down tremendously. My TRS-80 Tape Editor/Assembler, Version 1.1, records source and object code at 500 baud. As I gain ability to write longer assembly-language programs, the excessive recording time is beginning to drive me up a wall, but I don't know enough, yet, to be able to change the coding so that it will run at 1500 baud.

I was hoping that someone within your sphere of acquaintance has addressed this problem, and could provide me with some numbers, and instructions as to where to put them in the original program, to get me out of this bind.

Can you help? I'd appreciate it, immensely.

-Robert B. Koehler, R.D. #4, Box 174, Lake Walton Ro
Hopewell Junction, New York 12

[Editor's note: Can someone help Bob with his problem? It seems I vaguely recall seeing an article outlining such a conversion

sometime back in one of the '80 magazines (80-Micro or 80-U.S.), but don't remember when or where.]

Dear Jack;

Just read the latest issue of NORTHERN BYTES, and I have one question. Where do you find the time to write, key in, dig up, layout and do the million and one other things that have to be done for a newsletter?!?

I have but one complaint, though. The people doing most of the submissions are using NEWDOS!

Here are three one-liners for the Radio Shack Pocket Computer. I am using the PC-3 but I would think all the others have the functions to do these.

The pocket computer has much more memory available to the user than the calculator, but it's not as readily available. You don't have memory keys to store the display in a memory. These three lines give you that ability.

By hitting two keys, <DEF> and <X, V or Z> the display or calculation result is stored in X, V (and Y) or Z. Using <DEF>, <V> stores the number in Y and the number rounded to two decimal places in V.

```
500 "X" : AREAD X : WAIT : USING : PRINT "X=";X : END
501 "Z" : AREAD Z : WAIT : USING : PRINT "Z=";Z : END
502 "V" : AREAD Y : WAIT : USING : V=(INT((Y*100)+.5))/100 :
PRINT "V=";V;" Y=";Y : END
```

The first two I saw in a newsletter, the third was born out of necessity.

Keep up the good work.

- Dave Bower, 572 Longfellow Avenue, Virginia Beach, Virginia 23462

[Editor's note: Thanks, Dave, for the one-liners. As for the complaint about everyone using NEWDOS, I'd be happy to receive articles from users of other DOSes (even ones I don't personally happen to like), so as to provide more balanced content in NORTHERN BYTES!].

FORTH-WISE
by Paul Snively

You may recall from the last Northern Bytes that I am going to be writing a regular column in support of TASForth, the Forth language package from The Alternate Source, our benevolent benefactors. So as not to make a liar out of the editor, here is the column!

You may also recall that I have given some phone numbers which I said were BBS numbers where Forth was discussed. I'm going to touch on a couple of these in just a little more depth in this issue.

The phone numbers that I'd like to talk about are (415) 538-3580 and (206) 759-0615. Both of these BBSes are unique in that not only do they SUPPORT Forth, they are written IN Forth! The (415) number is 300 baud, whereas the (206) one is 300/1200 baud. Both use 8 data bits, 1 stop bit and no parity.

Both of these BBSes are "Forth Trees." What this means is that they are BBSes written in Forth and sold by the CommuniTree Group. The reason that they are called "trees" is that their message structure is - you may have guessed by now - a binary tree. To put it simply, there is a root node (message) on the tree called "CONFERENCES." Under this root are some branches. The root is always "CONFERENCES," but the branches depend upon the SYSOP (sometimes referred to as the Fairwitness) and upon the users. Some high level branches might be "CHAT," "INQUIRIES," "LANGUAGES," "LIBRARY," and "TO-OPERATOR."

When you first call up, the tree asks you how wide your screen is. The default is 80 characters because the Forth Tree was originally written on an Apple. Respond with 64 (or 80 for Model 4 users, etc.) The Tree will then ask you if you have lowercase capabilities. If the answer is "Yes," just press <ENTER>, otherwise type "NO" and press enter. The system will give you some hints as to how to read the help section, maybe give some current news (usually only two or three lines worth) and give you the COMMAND ? prompt.

OK. Now that you're on, how do you use the system? Well, you could go ahead and type READ CONFERENCES. That, however, would only get you what was in the root, i.e., CONFERENCES. You'd probably be bored. What you really SHOULD do is open your buffer for input (if you have that capability) and type "i conferences" and press <ENTER>. "i" is short for "index" (all of the Tree commands have abbreviations) and "conferences" is the node that you want an index of. The implication of this is that you can index any node, not just "conferences," and this is indeed true.

Once you have the index, you might want to log off. It's easy: just hang up. The Tree will reset automatically. Now dump your buffer out to disk and load it into your word processor or whatever and take a peek. You probably got some prompts and things, but you should also have the titles to all of the messages on the tree. Notice that many of the titles are indented. This means that the indented message is a branch to the one above it. If a message has a lot of "children," as they are called, then the children's titles may be indented quite a bit by the time you get down to the bottom of that branch.

Now, find the titles that sound interesting to you. Mark these so that you know what to ask for the next time you log on. For example, I have a list of titles that says:

```
INTERPRETIVE-DO
NEW-IDEAS
SQUISH-FORTH
SOUNDS-GOOD,BUT...
ONE-BETTER
SO-WHAT
INFIX-FORTH
INFIX-FORTH-2
ALTERING-COMPILED
OF-COURSE
```

These titles are ones that I intend to examine the next time that I log on to the system (hopefully later tonight.) Incidentally, these examples are from the Fig-Tree in California, and their number is the (415) one listed above.

Now that you know what you want to read, log back on and when you get the COMMAND ? prompt, type R <title>, where <title> is the title of the message that you are interested in. Don't type the braces; they are a convention I use to indicate that the phrase is a parameter.

The message will be printed. Pressing "S" will Stop the display until another key is pressed and pressing "C" will take you back to the COMMAND ? prompt. In fact, "C" will take you to the command prompt from ANY command that is executing. Remember this! It comes in handy for aborting long indices that waste valuable phone time.

Congratulations. You now know how to read messages on the Forth Tree. Another neat feature that you may want to take advantage of after your first experience with the Tree is the S option on reads. S is short for "starting," and it allows you to specify a date to start reading from. So, if you were on the Tree on 27-JUN-84 and logged on again on 5-JUL-84, you could type "R CONFERENCES S 28-JUN-84" and press <ENTER> and the Tree would list every message that had been posted since the last time you were logged on. Very handy! Of course, it puts the burden of keeping track of the date of your last logon on you, the user, but it's really not that bad.

One thing that you may have noticed through all of this: The system has no provision for finding out who you are. Neither of the Trees listed above has any password access things or inquiries as to who you are. These are open systems. Please respect the Fairwitness' forthrightness and refrain from using the system to curse, slander, and just generally make a mess of things. Just because they don't have passwords now doesn't mean that they can't in the future.

One last thing: the (206) Tree is running on a Model I TRS-80, and the Fairwitness of that Tree is also the author of that Tree, as well as the version of Forth that it runs under! Whew! This guy is prolific, and if you have any questions about TRS-80 Forth, he is the one who can answer them! He'll also be glad to tell you who to buy his QFORTH from. QFORTH is unlike TASForth in many respects, and I'm strongly considering buying QFORTH to add to my collection of languages.

Here are some actual messages from one of the Forth-Trees:

```
*** SQUISH-FORTH 2-MAY-84
 PARENT=NEW-IDEAS USAGE= 98
```

How would you like a Forth that doesn't need a metacompiler to develop optimized products? Instead, you develop and debug in the ordinary way. Then when you're done, the system squeezes down (in-place in memory if you wish). Not only are headers and

compiling/interpreting words eliminated if not needed, but all words never used by your product are automatically squished out - even throughout the nucleus - giving the most compact code possible.

How to do it?

Assume a microprocessor with position-independent code (otherwise, it's a little harder and less optimum). Implementation is easier if the Forth control structures use relative branching. For simplicity in this presentation, assume no advanced or tricky code, and assume that all code fields are located two bytes before their parameter fields. Address constants must not refer to locations inside the program.

Consider such a program which has been developed and now runs correctly. Let's call its final word 'GO'. Now say 'SQUISH GO'. The word 'SQUISH' itself will probably be deleted; but before then, here's what happens:

Without making any changes to the Forth system or to your program, 'SQUISH' builds a table in some spare memory. The table has one entry (of four words) for each dictionary word. The table in initialized to zeros.

The first table entry is the address of the code field of each word, in order, filled in by a single pass down the dictionary. The second entry is the total length of the object code (including CFA), of only those words which could possibly be executed by 'GO'. This entry is obtained by a recursive 'shadow execution' of 'GO', which traverses every control path of every colon definition (to reach every control path, simply don't branch on 'BRANCH' or '0BRANCH'), touching every word which could ever be executed, and marking the corresponding table-length entry. At the end of this shadow-execution, all words which still have zero in the length-entry are squeezed out of the table. Those are the words which could never have been executed by 'GO'.

Clearly a simple code routine could use this table (of addresses and lengths, of relevant words only) to 'CMOVE' each definition and compact the program. (The actual compaction must be done in code, because Forth will be destroyed if you choose to squish in-place.) But first we must plan to translate the old compiled addresses into new ones. Because this translation must be done in code, 'SQUISH' first makes it easy - by filling in the remaining two fields of the table.

The third table entry is the new address of the code field of each word (i.e. the address after that word's definition has been moved during the compaction). These addresses are obtained by a "shadow SQUISH", which does not actually compact any code but goes through the motions in order to compute the new, post-compaction addresses which the words will later have.

The fourth table entry gives addresses of code for shadow execution of special compiled words like 'LIT', compile-time 'IF', 'DO', semicolon, etc.

Now the little compaction routine in code does the job for real. It CMOVEs each relevant definition. Then if that word is a colon definition, it does a simple, non-recursive shadow execution of it (with the help of the fourth table entry), for the sole purpose of knowing for sure which words in the object code are indeed compiled addresses; these are translated by the table (look up in first entry, replace by third entry).

Any problems?

JSJ

*** SOUNDS-GOOD,BUT... 9-JUN-84
PARENT=SQUISH-FORTH USAGE= 36

Sounds like a good, innovative idea. Having written a decompiler or two though has given me a different outlook on the task of "shadow executing" a Forth program. The problem is with words like WORD, ." and so on that actually parse arguments out of the input stream. You wind up with a lot of special-purpose code to properly emulate these pesky instructions.

My other concern is with the limitation of position-independent code. I've written a couple of Forths and the requirement of complete position independence would have killed me both times. Some processors just don't have what it takes to run this kind of code efficiently.

Maybe the answer is to put together a cross-compiler that is less "user hostile" than ones currently available. That's my approach, anyway. I'll let you know how it turns out.

Joe Barnhart

*** ONE-BETTER 19-JUN-84
PARENT=SQUISH-FORTH USAGE= 19

How about a Forth that allows you to create the absolute minimal Forth Application (based on what your source needs), still allows full use of the editor/assembler/single-step-debugger etc. and allows execution of this minimal system (without ever outer-interpreter!) and without resident heads ..... hehehehe. (Yes ... I have it)

*** INFIX-FORTH 2-MAY-84
PARENT=NEW-IDEAS USAGE= 82

If you thought SQUISH-FORTH was weird...

This message and its followup (PREFIX-FORTH) suggest some of the most radical changes to Forth yet proposed. These ideas began half-seriously but keep looking better.

We start by adding infix to Forth. Why? Some people want it. Try writing the quadratic equation in Forth and Basic and see which version the public finds easier to read.

Then why hasn't infix been used before? It's easy to translate infix to postfix. But it's hard for an infix language to be fully extensible. What about operator precedence? And the bizarre restriction that functions take no more than two arguments and return no more than one result? And running out of special characters for the function names? The APL way isn't for everyone.

But note! what we do with infix, what we use it for, is not what needs to be extensible. It's hard to extend arithmetic; that's system-programmer work, not the everyday use of an extensible language.

So let's integrate infix and postfix in the same line. (Not as strange as it sounds; many languages integrate infix and PREfix, which is used for function calls.) And then the beauty is that infix becomes more general than usual. For infix expressions can now have missing elements, and the Forth stack will supply them.

We propose an ordinary, garden-variety Forth; it can even be Forth-83 standard. Its users never need know that anything is different. But if the text interpreter cannot find a word in the dictionary, it tries to make an infix "expression" out of it (an ordinary number qualifies as an infix expression, of course). Then the Forth compiler translates the infix to postfix, and compiles as ordinary Forth. (Note the run-time speed advantage over most other interactive infix language systems.)

Consider some examples of infix expressions. These may be used anywhere within ordinary Forth colon definitions.

Normal infix expressions are like '(A+B)*(C+D)'. That expression can be used verbatim in a normal line of infix Forth (without the single quotes). This particular expression will take nothing from the stack, and return one value to it (because it produces one result).

We also allow normally-illegal expressions like '(+B)*(C+D)'. This "expression" takes one argument from the stack, adds it to B, etc., and returns one result to the stack.

We also allow complete "assignment statements", e.g., 'X=(A+B)*(C+D)'. Here there is no effect on the stack; the computed value is not left there because it is stored in 'X'. The equal sign when used in an infix expression is essentially the FORTH "TO" concept, popular in systems in Europe.

Infix expressions must have no spaces within them (except as noted below), thereby distinguishing what is infix from what is ordinary Forth. 'WORD' will take each infix expression as one token, and when this token is not found in the dictionary, it will be parsed and translated by the (extended) Forth compiler.

Now let's mix infix and postfix. E.g. '5 A=' sets 'A' (a constant) to 5. '10+' and '10-' add ten to or subtract ten from the number on top of the stack. (Careful of '+10' or '-10', as '+' and '-' become unary if there is a space on the left and none on the right - a rule which gives us what we would expect anyway, the number 10 or -10 pushed onto the stack.)

Consider the infix "expression" '++'. It adds the top three stack numbers (because the three arguments which would normally be present in the infix expression are missing) and produces one result, which it returns to the stack. Consider '+' alone. It similarly adds the top two stack numbers and returns one result. Just as we would expect from ordinary Forth; infix and postfix have become one.

The set of infix operators ('+', '-', '*', '/', and '^') is not extensible; these five are all of them. But you can use infix expressions to build your own operations, obviously, since these

expressions can be used anywhere in colon definitions. And you can also use postfix, all of Forth and your own words, to build infix formulas. Not as obviously.

Consider '(A+B)*(HERE 10 +)*(C+D)'. Within parentheses may be an infix expression (without spaces), or ordinary Forth (with spaces). The ordinary Forth may itself include infix, nested to any depth.

Note some details:

(1) The expression above is taken and initially parsed as a single infix unit, despite the spaces, because the spaces are within parentheses used by an infix expression. Obviously a change to 'WORD' is necessary. Forth comments, which also use parentheses, are not affected, as we shall see.

(2) The spaces inside parentheses and adjacent to them need not actually be written, because, for infix purposes only, a delimiter (infix space-equivalent) is always assumed inside a left or right paren (never outside of a paren). So '(HERE 10 +)' is just as good as '(HERE 10 +)'. Even better - because the former cannot be confused with a comment, which requires a left paren alone.

(3) Note that under the rules we have given, the extreme cases '(HERE)', '(10)', and '(+)', etc., could be considered either infix or postfix. But that's no problem, because either way is equivalent.

To change the subject for a moment, consider type checking. Some people want it. The above infix system makes it feasible. For within infix formulas, type checking (plus automatic fixed-floating conversion, etc.) can clearly be done at compile time. And outside of the infix formulas, optional run-time type checking (e.g. using an extra type byte attached to every stack item) becomes significantly less expensive since the low-level arithmetic work, checked at compile time, can be compiled to run without this overhead. (And other low-level, speed-critical Forth routines can be defined and debugged, then recompiled with the run-time checking turned off.)

Tune in next time for another exciting installment, tentatively called 'PREFIX-FORTH'. It looks like we can turn Forth around to integrate the ordinary colon (it's already prefix), plus Basic-type functions like SIN or SQR (also prefix already), plus infix (already works well with prefix functions, e.g. in Basic), plus Forth extensibility, plus string stack and also input-stream arguments without the bother of quotes, plus handy command languages, plus variable numbers of arguments with conditional compilation - without losing anything we already have in Forth.

At least that's what it looks like now.

JSJ

*** ALTERING-COMPILED 12-MAY-84
PARENT=NEW-IDEAS USAGE= 60

Logo gives us the ability to redefine procedures (words) that are already the components of compiled words. In Forth, that is not so. It would be convenient if it were so. Is there a way to do it in Forth? Example: say you have a word "circumference" defined as follows: : CIRCUMFERENCE DIAMETER PI-TIMES ; where DIAMETER somehow fetches the diameter and PI-TIMES is defined : PI-TIMES 22 7 */ ;

Later you find the definition of pi as 355 113 */ and want to use it in the already defined CIRCUMFERENCE. Obviously you could define a new PI-TIMES and a new CIRCUMFERENCE - with or without a FORGET PI-TIMES first - but is there a way to redefine PI-TIMES without having to redefine CIRCUMFERENCE?

*** OF-COURSE 12-MAY-84
PARENT=ALTERING-COMPILED USAGE= 58

Yes, my child, FORTH can do all. Including changing the definition of previously defined words. Three methods are: (1) Patching the code field and the first two bytes of the parameter field to redirect to the later definition. (2) Finding all uses and hot patching (risky at best) and best of all (3) Using <<DEFER>> in the manner given by Henry Laxon in Issue 6, Volume 5 of FORTH DIMENSIONS. There is a wealth of knowledge in FORTH DIMENSIONS. Read and be amazed.

Thanx, THE GOOD DOCTOR

*** INTERPRETIVE-DO 23-MAY-84
PARENT=SUBMISSIONS USAGE= 95

The following is a DO word which can be used interpretively (usage follows):

```
: /DO 1 WORD TIB @ 80 ERASE HERE COUNT TIB @ SWAP CMOVE
    1+ SWAP DO 0 IN ! I INTERPRET LOOP 0 TIB @ ! 0 IN ! ;
```

Usage: 10 20 /DO LIST

The list of words after /DO until C/R are executed while the loop index is varied from 10 to 20. The current value of the loop index is left on the stack for each execution of the list of words.

Well, that should get you started. If you have any questions about TASForth, the Forth-Tree BBSes, or Forth in general, contact me, Paul Snively, c/o The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906, or directly at 2820 Jordan Drive, Columbus, Indiana 47203 telephone (812) 372-8789, or via MCI Mail, user name PSNIVELY. See you next issue!

A NEW MODEL 4 ROM? Unfortunately, it appears so. As many of you know, a NEW Model 4 has appeared with green screen (even in the U.S.A. - FINALLY!), relocated arrow keys (heaven help you if you have to use a <SHIFT><DOWN-ARROW> plus another key combination extensively within a program - this is a definite boo, hiss after all this time) and worst of all, revised ROMs that modify the original Model III ROM code BELOW 3000H!!! Those of you that have TRS-80 ROM ROUTINES DOCUMENTED can line out the phrase that states that the Model III and 4 ROMs are the same below 3000H (will I revise my book? Probably not right away. For one thing, I don't have access to one of the new Model 4's yet and for another, I'm getting rather tired of playing the "Catch-Up" game with Tandy).

There are times when I think Tandy should leave well enough alone, and (with the exception of the green screen, which was LONG overdue) I think this was one of them. If anyone has access to one of the NEW Model 4's and would like to document the ROM differences for the rest of us, we'd appreciate it (in fact, if the article were REALLY well done this might be one of those "once in a blue moon" articles that we'd actually PAY for!).

Thanks to Nate Salsbury and S. C. Williams for bringing this to my attention.

HOW TO GET FREE SOFTWARE is the title of a new book by Alfred Glossbrenner, published by St. Martin's Press. This book can save the average personal computer user hundreds of dollars. If you're a personal computer user and you're not independently wealthy, you need this book. This is especially true if you're into CP/M, since the author must document just about every source of free CP/M software there is. But, he doesn't stop there. He goes on to document the other sources of free software for just about every popular brand of personal computer now in existence.

Is the free software any good? Yes!! Is it truly free? Depends on where you get it. If you download it from a Bulletin Board System in your local calling area (and you're not on measured time service), then you could probably say it is truly free (assuming you don't put any value on the time it takes you to download the software). Otherwise, you may have to pay a few dollars to get a disk of "free" software sent to you. You might get it from a users group which may or may not charge you a few bucks. In any case, the software is truly a bargain, if not "free" in the strictest sense of the word.

As a NORTHERN BYTES reader, you already know about one source of free software for the TRS-80 - the TAS Public Domain Software Library. Yes, it's mentioned in the book, along with a couple of other projects your editor has been involved with (this "mention" takes about two pages - thank you, Alfred!). The Fairfield County TRS-80 User Group (c/o Alan Abrahamson, 10 Richlee Road, Norwalk, Connecticut 06851) is also mentioned as a group that has a large library of TRS-80 public domain programs (however, I'm not sure that they are quite prepared to deal with the mail inquiries that the mention in the book will generate, since their library has pretty much been for the benefit of their local members in the past - in fact, they rarely mention their library in their newsletter. Maybe that will change when the inquiries start pouring in!).

Glossbrenner's writing is very readable - not the dry, technical style of writing that is so common among computer-oriented literature! Novice users will appreciate this book both for the "readability" and for the fact that the author has brought together a lot of information that the novice user especially could benefit from, but that he might not otherwise be able to obtain without a lot of time, effort, and persistence in asking questions of more experienced users. Get this one for yourself or for your computer club library, you'll never regret it!

DIRSLOT/CMD - A file utility that allows the user to direct any file on a disk into a selected directory slot, (C) Joachim Kelterbaum, Frankenstr. 305, 4300 Essen 1, West Germany.

This program is of interest mainly to those programmers who write /SYS modules on their own. Those modules require a definite directory slot so they can be accessed by the DOS overlay loader. Usually it is quite a boring task to establish a file entry at the desired directory slot. This program makes it easy.

The command syntax is as follows:

DIRSLOT filename</ext:dnr>,dd

filename may be any valid filename including extension and/or drive number if that's convenient. The filename must be followed directly by a comma and a two digit decimal number. This number gives the slot number the file is to be positioned at. Example:

DIRSLOT TESTFILE/ABC:2,06

would place the file TESTFILE/ABC into directory slot number 6 (thats where SYS4/SYS normally resides on a system disk).

CAUTION !!! As this program allows any slot number from 00 to 99 (if the directory holds that many entries), you can easily destroy your system disk, if you don't exactly know what you're doing. You can even replace the BOOT/SYS or the DIR/SYS entry by your own file!!!

The program has error checking to some degree for your security. First, you are warned to read this documentation. Any invalid input command (parameter errors, slot number too large for directory in that particular case etc.) will abort the program. If you did not give a drive number the given file is searched and the first drive is selected that contains this file.

The program then checks the directory of that drive. If the file entry at the slot number you specified is an FXDE (extent entry - a case that happens every 1000 years) there will be no exchange done!! (it would be too annoying to rebuild most of the directory). The job is aborted then. If that slot contains no entry or an inactive entry, the exchange is done without further notice. If, however, there is an active file at the selected slot, the user will be given the name of that file. The program then asks, if the exchange shall be performed or not. Only a reply of 'y' will activate the exchange. Any other reply will abort the program. The program is smart enough to detect the length in granules of the directory and its location (specified by PDRIVE).

```
                00100 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00110 ;x            DIRSLOT/CMD              x
                00120 ;x                                     x
                00130 ;x   (C) Joachim Kelterbaum            x
                00140 ;x   Frankenstr. 305                   x
                00150 ;x   4300 Essen 1                      x
                00160 ;x   W. Germany                        x
                00170 ;x                                     x
                00180 ;x please , read document file         x
                00190 ;x DIRSLOT/DOC before you make use     x
                00200 ;x of this program                     x
                00210 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
8000            00220          ORG     8000H
8000 44         00230 FILDIR   DEFM    'DIR/SYS:'   ;Name of Dir-file to be modified
     49 52 2F 53 59 53 3A
8008 00         00240 NBOV     DEFB    0            ;space for drive number
8009 0D         00250          DEFB    0DH
                00260 ;
1E00            00270 DIRBUF   DEFS    7680         ;buffer to hold directory
                00280 ;
5200            00290          ORG     5200H
                00300                               ;constants
5200 00         00310 SNB      DEFB    0            ;will hold target slot #
5201 00         00320 ENR      DEFB    0            ;low dec. digit of target slot #
5202 00         00330 ZNR      DEFB    0            ;high        "
5203 00         00340 DNR      DEFB    0            ;target drive #
5204 00         00350 GRNNB    DEFB    0            ;# grans of directory in target disk
5205 00         00360 SLTMX    DEFB    0            ;max. allowable slot #
000F            00370 FILNAM   DEFS    15           ;holds name of file to be moved
0002            00380 SLTTX    DEFS    2            ;target slot # (string)
000B            00390 CONFIL   DEFS    11           ;converted filename (directory format)
5222 0A         00400          DEFB    0AH
5223 0D         00410          DEFB    0DH
5224 0000       00420 COUNT    DEFW    0            ;rel. pos # of target file in DIRBUF
5226 00         00430 XPOS     DEFB    0            ;x-pos       "
5227 00         00440 YPOS     DEFB    0            ;y-pos       "
5228 00         00450 SORGNB   DEFB    0            ;slot # of orig. file
5229 00         00460          DEFB    0
522A 00         00470 CNTZIL   DEFB    0            ;rel pos # of orig. file in DIRBUF
522B 00         00480          DEFB    0
522C 0000       00490 QUAD     DEFW    0            ;adress of target file in DIRBUF
522E 0000       00500 ZILAD    DEFW    0            ;  "      orig.      "
5230 0000       00510 HEORG    DEFW    0            ;  "      HIT entry for target
5232 0000       00520 HEZIL    DEFW    0            ;  "           "       orig.
5234 0A         00530 ZILMS    DEFB    0AH
5235 44         00540          DEFM    'Directory Slot # '
     69 72 65 63 74 6F 72 79 20 53 6C 6F 74 20 23 20
0002            00550 MSINS    DEFS    2
5248 20         00560          DEFM    ' currently holds the following entry:'
     63 75 72 72 65 6E 74 6C 79 20 68 6F 6C 64 73 20
     74 68 65 20 66 6F 6C 6C 6F 77 69 6E 67 20 65 6E
     74 72 79 3A
5260 0A         00570          DEFB    0AH
526E 0D         00580          DEFB    0DH
                00590 ;
0020            00600 FCB      DEFS    32           ;file control block
528F 00         00610 FCODE    DEFB    0            ;will hole error code
                00620 ;
0100            00630 BUFFER   DEFS    256          ;file buffer
                00640 ;
5390 2A         00650 SLTMS1   DEFM    'xx File xx '
     2A 20 46 69 6C 65 20 2A 2A 20
539B 03         00660          DEFB    3
539C 6E         00670 SLTMS2   DEFM    'now resides at Slot #'
     6F 77 20 72 65 73 69 64 65 73 20 61 74 20 53 6C
     6F 74 20 23
53B1 03         00680          DEFB    3
                00690 ;
53B2 0A         00700 FXDMS    DEFB    0AH
53B3 72         00710          DEFM    'required Slot is occupied by an FXDE'
     65 71 75 69 72 65 64 20 53 6C 6F 74 20 69 73 20
     6F 63 63 75 70 69 65 64 20 62 79 20 61 6E 20 46
     58 44 45
53D7 0D         00720          DEFB    0DH
53D8 0A         00730 ABOMS    DEFB    0AH
53D9 4A         00740          DEFM    'Job has been aborted'
     6F 62 20 68 61 73 20 62 65 65 6E 20 61 62 6F 72
     74 65 64
53ED 0A         00750          DEFB    0AH
53EE 0D         00760          DEFB    0DH
53EF 65         00770 QUST     DEFM    'exchange Slots (Y/N) ?'
     78 63 68 61 6E 67 65 20 53 6C 6F 74 73 20 28 59
     2F 4E 29 20 3F
5405 0D         00780          DEFB    0DH
5406 0A         00790 XCHMS    DEFB    0AH
5407 2A         00800          DEFM    'xx EXCHANGE SELECTED SLOTS xx'
     2A 20 45 58 43 48 41 4E 47 45 20 53 45 4C 45 43
     54 45 44 20 53 4C 4F 54 53 20 2A 2A
5424 0D         00810          DEFB    0DH
                00820 ;
                00830                               ;Start of program xxxxxxxxx
5800            00840          ORG     5800H
5800 7E         00850 START    LD      A,(HL)       ;get 1st char from input buffer
5801 FE0D       00860          CP      0DH          ;is it enter?
5803 2808       00870          JR      Z,ERR1       ;Y -> error
5805 116F52     00880          LD      DE,FCB       ;point to FCB
5808 CD1C44     00890          CALL    441CH        ;extract filespec.
580B 1821       00900          JR      OK
580D CDC901     00910 ERR1     CALL    01C9H        ;print error mssg.
5810 211E58     00920          LD      HL,MES
5813 11003C     00930          LD      DE,3C00H
5816 010F00     00940          LD      BC,15
5819 EDB0       00950          LDIR
581B C32D40     00960          JP      402DH        ;-> DOS
                00970 ;
581E 50         00980 MES      DEFM    'Parameter Error'
     61 72 61 6D 65 74 65 72 20 45 72 72 6F 72
582D 00         00990          DEFB    0
                01000 ;
582E 7E         01010 OK       LD      A,(HL)       ;get following 2 dec. digits
582F D630       01020          SUB     30H                  as target slot #
5831 38DA       01030          JR      C,ERR1
5833 FE0A       01040          CP      0AH
5835 30D6       01050          JR      NC,ERR1
5837 320252     01060          LD      (ZNR),A
583A 23         01070          INC     HL
583B 7E         01080          LD      A,(HL)
583C D630       01090          SUB     30H
583E 38CD       01100          JR      C,ERR1
5840 FE0A       01110          CP      0AH
```

```
5842 30C9      01120        JR      NC,ERR1
5844 320152    01130        LD      (ENR),A
5847 3A0252    01140        LD      A,(ZNR)
584A CB27      01150        SLA     A
584C 47        01160        LD      B,A
584D CB27      01170        SLA     A
584F CB27      01180        SLA     A
5851 80        01190        ADD     A,B
5852 47        01200        LD      B,A
5853 3A0152    01210        LD      A,(ENR)
5856 80        01220        ADD     A,B
5857 320052    01230        LD      (SNB),A     ;put slot # ->SNB
585A CDA85B    01240        CALL    SIGNUP      ;print sign up mssg.
585D 3E00      01250        LD      A,0         ;clear FILNAM buffer
585F 210652    01260        LD      HL,FILNAM
5862 77        01270        LD      (HL),A
5863 110752    01280        LD      DE,FILNAM+1
5866 010E00    01290        LD      BC,14
5869 EDB0      01300        LDIR
586B 212043    01310        LD      HL,4320H    ;point to filename in input buffer
586E 110652    01320        LD      DE,FILNAM
5871 7E        01330 TRANS  LD      A,(HL)      ;move it to FILNAM
5872 FE2C      01340        CP      ','
5874 2805      01350        JR      Z,OUT11
5876 12        01360        LD      (DE),A
5877 23        01370        INC     HL
5878 13        01380        INC     DE
5879 18F6      01390        JR      TRANS
587B 3E0D      01400 OUT11  LD      A,0DH       ;append 0DH
587D 12        01410        LD      (DE),A
587E 23        01420        INC     HL
587F 111552    01430        LD      DE,SLTTX    ;move target slot # to SLTTX
5882 0602      01440        LD      B,2
5884 7E        01450 TR2    LD      A,(HL)
5885 12        01460        LD      (DE),A
5886 23        01470        INC     HL
5887 13        01480        INC     DE
5888 10FA      01490        DJNZ    TR2
588A 116F52    01500        LD      DE,FCB      ;open target file to see,
588D 219052    01510        LD      HL,BUFFER   ;if it exists and on which
5890 0600      01520        LD      B,0         ;drive it is located
5892 CD2444    01530        CALL    4424H       ;open file
5895 3A7552    01540        LD      A,(FCB+6)   ;extract drive # from FCB
5898 320352    01550        LD      (DNR),A     ; -> DNB
589B CB27      01560        SLA     A           ;calc. adress of DDGA byte in
589D 47        01570        LD      B,A                          PDRVE table
589E CB27      01580        SLA     A
58A0 CB27      01590        SLA     A
58A2 80        01600        ADD     A,B
58A3 C609      01610        ADD     A,9
58A5 4F        01620        LD      C,A
58A6 0600      01630        LD      B,0
58A8 217143    01640        LD      HL,4371H
58AB 09        01650        ADD     HL,BC
58AC 7E        01660        LD      A,(HL)
58AD 320452    01670        LD      (GRNNB),A   ;put it to GRNNB
58B0 47        01680        LD      B,A         ;calc. max allowable # of slots
58B1 CB27      01690        SLA     A
58B3 CB27      01700        SLA     A
58B5 80        01710        ADD     A,B
58B6 D602      01720        SUB     2
58B8 CB27      01730        SLA     A
58BA CB27      01740        SLA     A
58BC CB27      01750        SLA     A
58BE 320552    01760        LD      (SLTMX),A   ;put it to SLTMX
58C1 47        01770        LD      B,A         ;check for alloable slot # input
58C2 3A0052    01780        LD      A,(SNB)
58C5 90        01790        SUB     B
58C6 2802      01800        JR      Z,ERR2
58C8 3824      01810        JR      C,OK2
58CA CDC901    01820 ERR2   CALL    01C9H       ;error if target slot # not
58CD 21D858    01830        LD      HL,MS2                       allowable
58D0 11003C    01840        LD      DE,3C00H
58D3 011300    01850        LD      BC,19
58D6 EDB0      01860        LDIR
58D8 C32D40    01870        JP      402DH
58DB 69        01880 MS2    DEFM    'illegal slot number'
    6C 6C 65 67 61 6C 20 73 6C 6F 74 20 6E 75 6D 62
    65 72
58EE 116F52    01890 OK2    LD      DE,FCB
58F1 CD2844    01900        CALL    4428H       ;close file again
58F4 3A0352    01910        LD      A,(DNR)     ;plug drive # into FILDIR buffer
58F7 C630      01920        ADD     A,30H
58F9 320880    01930        LD      (NBDV),A
58FC 210080    01940        LD      HL,FILDIR   ;prepare to open DIR/SYS on that drive
58FF CD1C44    01950        CALL    441CH       ;extract filespec.
5902 219052    01960        LD      HL,BUFFER   ;HL -> file buffer
5905 0600      01970        LD      B,0         ;256 byte records
5907 CD2044    01980        CALL    4420H       ;open file
590A CD3F44    01990        CALL    443FH       ;pos. to start of file
590D 3A0452    02000        LD      A,(GRNNB)   ;calc. # of sectors to be moved
5910 47        02010        LD      B,A
5911 CB27      02020        SLA     A
5913 CB27      02030        SLA     A
5915 80        02040        ADD     A,B
5916 47        02050        LD      B,A         ;put result to B
5917 11BA80    02060        LD      DE,DIRBUF   ;read (B) sectors and move to DIRBUF
591A C5        02070 TRFDIR PUSH    BC
591B D5        02080        PUSH    DE
591C 116F52    02090        LD      DE,FCB
591F CD3644    02100        CALL    4436H   ;READ SECTOR
5922 D1        02110        POP     DE
5923 219052    02120        LD      HL,BUFFER
5926 010001    02130        LD      BC,256
5929 EDB0      02140        LDIR
592B C1        02150        POP     BC
592C 10EC      02160        DJNZ    TRFDIR
592E 116F52    02170        LD      DE,FCB
5931 CD2844    02180        CALL    4428H       ;close file again
5934 CDE35A    02190        CALL    NAMCON      ;convrt. name of target file to DIR format
5937 211752    02200        LD      HL,CONFIL   ;search for target file entry in DIRBUF
593A 11BF82    02210        LD      DE,DIRBUF+205H
593D 060B      02220        LD      B,11
593F CD1D5B    02230        CALL    SEARCH
5942 ED4B2452  02240        LD      BC,(COUNT)  ;(BC) = rel pos # of file entry
5946 DA635B    02250        JP      C,NFDERR    ;error if not found
5949 CDC901    02260        CALL    01C9H       ;CLS
594C 79        02270        LD      A,C         ;calc. slot # of target file
594D 47        02280        LD      B,A
594E CB3F      02290        SRL     A
5950 CB3F      02300        SRL     A
5952 CB3F      02310        SRL     A
5954 322652    02320        LD      (XPOS),A
5957 CB27      02330        SLA     A
5959 CB27      02340        SLA     A
595B CB27      02350        SLA     A
595D 4F        02360        LD      C,A
595E 78        02370        LD      A,B
595F 91        02380        SUB     C
5960 322752    02390        LD      (YPOS),A
5963 3A2652    02400        LD      A,(XPOS)
5966 CB3F      02410        SRL     A
5968 CB3F      02420        SRL     A
596A CB3F      02430        SRL     A
596C CB27      02440        SLA     A
596E CB27      02450        SLA     A
5970 CB27      02460        SLA     A
5972 4F        02470        LD      C,A
5973 3A2652    02480        LD      A,(XPOS)
5976 91        02490        SUB     C
5977 47        02500        LD      B,A
5978 79        02510        LD      A,C
5979 CB27      02520        SLA     A
597B CB27      02530        SLA     A
597D CB27      02540        SLA     A
597F 4F        02550        LD      C,A
5980 3A2752    02560        LD      A,(YPOS)
5983 CB27      02570        SLA     A
5985 CB27      02580        SLA     A
5987 CB27      02590        SLA     A
5989 81        02600        ADD     A,C
598A 80        02610        ADD     A,B
598B 322852    02620        LD      (SORGNB),A  ;put result -> SORNB
               02630 ;
598E 3E02      02640        LD      A,2         ;set NTF to integer
5990 32AF40    02650        LD      (40AFH),A
5993 212852    02660        LD      HL,SORGNB
5996 CDF709    02670        CALL    09F7H       ;copy SORGNB to ACCUM
5999 CD8E0F    02680        CALL    0F8EH       ;convt to string
599C E5        02690        PUSH    HL          ;save adress
```

```
599D 3E0D    02700        LD    A,0DH       ;append 0DH
599F 213641  02710        LD    HL,4136H
59A2 77      02720        LD    (HL),A
59A3 219053  02730        LD    HL,SLTMS1   ;messg: target file found at
59A6 CD6744  02740        CALL  4467H
59A9 210652  02750        LD    HL,FILNAM
59AC CD6744  02760        CALL  4467H
59AF 219C53  02770        LD    HL,SLTMS2
59B2 CD6744  02780        CALL  4467H
59B5 E1      02790        POP   HL          ;slot # SORGNB
59B6 CD6744  02800        CALL  4467H
59B9 3A0052  02810        LD    A,(SNB)     ;calc. rel. pos of orig. file
59BC 0606    02820        LD    B,6         ;entry in DIRBUF
59BE CB3F    02830 DV64   SRL   A
59C0 10FC    02840        DJNZ  DV64
59C2 0606    02850        LD    B,6
59C4 CB27    02860 MU64   SLA   A
59C6 10FC    02870        DJNZ  MU64
59C8 57      02880        LD    D,A
59C9 3A0052  02890        LD    A,(SNB)
59CC 92      02900        SUB   D
59CD 47      02910        LD    B,A
59CE CB3F    02920        SRL   A
59D0 CB3F    02930        SRL   A
59D2 CB3F    02940        SRL   A
59D4 4F      02950        LD    C,A
59D5 78      02960        LD    A,B
59D6 E607    02970        AND   07H
59D8 CB27    02980        SLA   A
59DA CB27    02990        SLA   A
59DC CB27    03000        SLA   A
59DE 81      03010        ADD   A,C
59DF 82      03020        ADD   A,D
59E0 322A52  03030        LD    (CNTZIL),A  ;put result -> CNTZIL
59E3 211552  03040        LD    HL,SLTTX    ;inset target slot # (as string
59E6 114652  03050        LD    DE,MSINS                    into MSINS)
59E9 010200  03060        LD    BC,2
59EC ED80    03070        LDIR
59EE 210A82  03080        LD    HL,DIRBUF+200H;search for original file entry
59F1 3A2A52  03090        LD    A,(CNTZIL)                  at target slot #
59F4 47      03100        LD    B,A
59F5 112000  03110        LD    DE,32
59F8 19      03120 FDENTR ADD   HL,DE
59F9 10FD    03130        DJNZ  FDENTR
59FB 7E      03140        LD    A,(HL)      ;get 1st byte of that file entry
59FC FE90    03150        CP    90H         ;is it an FXDE ?
59FE 200F    03160        JR    NZ,NOFX     ;no ->NOFX
5A00 21B253  03170        LD    HL,FXDMS    ;yes: error + abort
5A03 CD6744  03180        CALL  4467H
5A06 210853  03190 ABORT  LD    HL,ABOMS
5A09 CD6744  03200        CALL  4467H
5A0C C32D40  03210        JP    402DH
5A0F CB67    03220 NOFX   BIT   4,A         ;is file active ?
5A11 2828    03230        JR    Z,XCHFLS    ;no : exchange file entries
5A13 0605    03240        LD    B,5         ;yes : display file name
5A15 23      03250 ADV    INC   HL
5A16 10FD    03260        DJNZ  ADV
5A18 111752  03270        LD    DE,CONFIL
5A1B 010B00  03280        LD    BC,11
5A1E ED80    03290        LDIR
5A20 213452  03300        LD    HL,ZILMS    ;ask if exchange is to be done
5A23 CD6744  03310        CALL  4467H
5A26 211752  03320        LD    HL,CONFIL
5A29 CD6744  03330        CALL  4467H
5A2C 21EF53  03340        LD    HL,QUST
5A2F CD6744  03350        CALL  4467H
5A32 CD4900  03360        CALL  0049H       ;get keybd. entry
5A35 E65F    03370        AND   5FH         ;convt to upper case
5A37 FE59    03380        CP    'Y'         ;is it Y ?
5A39 20CB    03390        JR    NZ,ABORT    ;no : abort
5A3B 210654  03400 XCHFLS LD    HL,XCHMS    ;yes : exchange entries
5A3E CD6744  03410        CALL  4467H
5A41 2A2452  03420        LD    HL,(COUNT)  ;get rel pos of target file entry
5A44 0605    03430        LD    B,5         ;and calculate adress in DIRBUF
5A46 29      03440 M32    ADD   HL,HL
5A47 10FD    03450        DJNZ  M32
5A49 110A82  03460        LD    DE,DIRBUF+200H
5A4C 19      03470        ADD   HL,DE
5A4D 222C52  03480        LD    (QUAD),HL   ;result to QUAD
5A50 116F52  03490        LD    DE,FCB      ;transfer file entry to FCB
5A53 012000  03500        LD    BC,32
5A56 ED80    03510        LDIR
5A58 2A2A52  03520        LD    HL,(CNTZIL) ;calc. adress of original file
5A5B 0605    03530        LD    B,5         ;in DIRBUF
5A5D 29      03540 MU32   ADD   HL,HL
5A5E 10FD    03550        DJNZ  MU32
5A60 110A82  03560        LD    DE,DIRBUF+200H
5A63 19      03570        ADD   HL,DE
5A64 222E52  03580        LD    (ZILAD),HL  ;result to ZILAD
5A67 ED5B2C52 03590       LD    DE,(QUAD)   ;move that file entry to slot of target file
5A6B 012000  03600        LD    BC,32
5A6E ED80    03610        LDIR
5A70 216F52  03620        LD    HL,FCB      ;move target file entry to slot of orig. file
5A73 ED5B2E52 03630       LD    DE,(ZILAD)
5A77 012000  03640        LD    BC,32
5A7A ED80    03650        LDIR
5A7C 3A2452  03660        LD    A,(COUNT)   ;exchange the HIT entries in an
5A7F CD8D5B  03670        CALL  CALC        ;analogous manner
5A82 223052  03680        LD    (HEORG),HL
5A85 3A2A52  03690        LD    A,(CNTZIL)
5A88 CD8D5B  03700        CALL  CALC
5A8B 223252  03710        LD    (HEZIL),HL
5A8E 2A3052  03720        LD    HL,(HEORG)
5A91 E5      03730        PUSH  HL
5A92 7E      03740        LD    A,(HL)
5A93 47      03750        LD    B,A
5A94 2A3252  03760        LD    HL,(HEZIL)
5A97 7E      03770        LD    A,(HL)
5A98 4F      03780        LD    C,A
5A99 78      03790        LD    A,B
5A9A 77      03800        LD    (HL),A
5A9B E1      03810        POP   HL
5A9C 79      03820        LD    A,C
5A9D 77      03830        LD    (HL),A
5A9E 116F52  03840        LD    DE,FCB
5AA1 210080  03850        LD    HL,FILDIR
5AA4 CD1C44  03860        CALL  441CH       ;extract filespec DIR/SYS:X
5AA7 219052  03870        LD    HL,BUFFER
5AAA 116F52  03880        LD    DE,FCB
5AAD 0600    03890        LD    B,0
5AAF CD2044  03900        CALL  4420H       ;open file
5AB2 1A      03910        LD    A,(DE)
5AB3 CBC7    03920        SET   0,A         ;set write protect state
5AB5 12      03930        LD    (DE),A
5AB6 CD3F44  03940        CALL  443FH       ;pos to start of file
5AB9 3A0452  03950        LD    A,(GRNNB)   ;now move DIRBUF back to file
5ABC 47      03960        LD    B,A
5ABD CB27    03970        SLA   A
5ABF CB27    03980        SLA   A
5AC1 80      03990        ADD   A,B
5AC2 47      04000        LD    B,A
5AC3 210A80  04010        LD    HL,DIRBUF
5AC6 C5      04020 TRF    PUSH  BC
5AC7 119052  04030        LD    DE,BUFFER
5ACA 010001  04040        LD    BC,256
5ACD ED80    04050        LDIR
5ACF E5      04060        PUSH  HL
5AD0 116F52  04070        LD    DE,FCB
5AD3 CD3C44  04080        CALL  443CH
5AD6 E1      04090        POP   HL
5AD7 C1      04100        POP   BC
5AD8 10EC    04110        DJNZ  TRF
5ADA 116F52  04120        LD    DE,FCB
5ADD CD2844  04130        CALL  4428H
5AE0 C32D40  04140        JP    402DH
             04150 ;
             04160 ;
5AE3 211752  04170 NAMCON LD    HL,CONFIL   ;convert filename in FILNAM to directory
5AE6 3E20    04180        LD    A,20H       ;format. i.e. 8 chrs. filename padded with blanks
5AE8 77      04190        LD    (HL),A      ;plus 3 chars extension (padded)
5AE9 111852  04200        LD    DE,CONFIL+1 ;put result to CONFIL
5AEC 010A00  04210        LD    BC,10
5AEF ED80    04220        LDIR
5AF1 210652  04230        LD    HL,FILNAM
5AF4 111752  04240        LD    DE,CONFIL
5AF7 0609    04250        LD    B,9
5AF9 7E      04260 TRFN   LD    A,(HL)
5AFA FE2F    04270        CP    '/'
5AFC 280C    04280        JR    Z,OUT1
5AFE FE3A    04290        CP    ':'
```

```
5800 C8       04300          RET     Z
5801 FE0D     04310          CP      0DH
5803 C8       04320          RET     Z
5804 12       04330          LD      (DE),A
5805 23       04340          INC     HL
5806 13       04350          INC     DE
5807 10F0     04360          DJNZ    TRFN
5809 C9       04370          RET
580A 23       04380 OUT1     INC     HL
580B 111F52   04390          LD      DE,CONFIL+8
580E 0603     04400          LD      B,3
5810 7E       04410 TRXT     LD      A,(HL)
5811 FE3A     04420          CP      ':'
5813 C8       04430          RET     Z
5814 FE0D     04440          CP      0DH
5816 C8       04450          RET     Z
5817 12       04460          LD      (DE),A
5818 23       04470          INC     HL
5819 13       04480          INC     DE
581A 10F4     04490          DJNZ    TRXT
581C C9       04500          RET
              04510 ;
              04520 ;
              04530 ;
581D C5       04540 SEARCH   PUSH    BC      ;search for string pointed to by (HL)
581E D5       04550          PUSH    DE      ;start search at (DE)
581F E5       04560          PUSH    HL      ;match string has (B) bytes
5820 1A       04570 CPIT     LD      A,(DE)  ;if search fails, increment (DE) by 32
5821 BE       04580          CP      (HL)    ;increment COUNT by 1
5822 2024     04590          JR      NZ,FAIL ;if file entry is not active, go on
5824 23       04600          INC     HL      ;C set if no match else reset
5825 13       04610          INC     DE
5826 7A       04620          LD      A,D
5827 B3       04630          OR      E
5828 2833     04640          JR      Z,BUFND
582A 10F4     04650          DJNZ    CPIT
582C E1       04660 FOUND    POP     HL
582D D1       04670          POP     DE
582E C1       04680          POP     BC
582F E5       04690          PUSH    HL
5830 D5       04700          PUSH    DE
5831 EB       04710          EX      DE,HL
5832 11FBFF   04720          LD      DE,-5
5835 19       04730          ADD     HL,DE
5836 7E       04740          LD      A,(HL)
5837 CB67     04750          BIT     4,A
5839 2808     04760          JR      Z,KILLED
583B D1       04770          POP     DE
583C E1       04780          POP     HL
583D ED4B2452 04790          LD      BC,(COUNT)
5841 AF       04800          XOR     A
5842 C9       04810          RET
              04820 ;
5843 D1       04830 KILLED   POP     DE
5844 E1       04840          POP     HL
5845 C5       04850          PUSH    BC
5846 D5       04860          PUSH    DE
5847 E5       04870          PUSH    HL
              04880 ;
5848 E1       04890 FAIL     POP     HL
5849 D1       04900          POP     DE
584A ED4B2452 04910          LD      BC,(COUNT)
584E 03       04920          INC     BC
584F ED432452 04930          LD      (COUNT),BC
5853 C1       04940          POP     BC
5854 E5       04950          PUSH    HL
5855 212000   04960          LD      HL,32
5858 19       04970          ADD     HL,DE
5859 EB       04980          EX      DE,HL
585A E1       04990          POP     HL
585B 18C0     05000          JR      SEARCH
              05010 ;
585D E1       05020 BUFND    POP     HL
585E D1       05030          POP     DE
585F C1       05040          POP     BC
5860 AF       05050          XOR     A
5861 3F       05060          CCF
5862 C9       05070          RET
              05080 ;
5863 CDC901   05090 NFDERR   CALL    01C9H
5866 216F5B   05100          LD      HL,NFMES
5869 CD6744   05110          CALL    4467H
586C C32D40   05120          JP      402DH
586F 66       05130 NFMES    DEFM    'file entry could not be found'
    69 6C 65 20 65 6E 74 72 79 20 63 6F 75 6C 64 20
    6E 6F 74 20 62 65 20 66 6F 75 6E 64
588C 0D       05140          DEFB    0DH
              05150 ;
588D 47       05160 CALC     LD      B,A     ;calculate adress of HIT entry in DIRBUF
588E CB3F     05170          SRL     A       ; (B) = rel pos. of entry in DIRBUF
5890 CB3F     05180          SRL     A       ; on exit (HL) = desired adress
5892 CB3F     05190          SRL     A
5894 4F       05200          LD      C,A
5895 78       05210          LD      A,B
5896 E607     05220          AND     07H
5898 6F       05230          LD      L,A
5899 2600     05240          LD      H,0
589B 0605     05250          LD      B,5
589D 29       05260 X32      ADD     HL,HL
589E 10FD     05270          DJNZ    X32
58A0 0600     05280          LD      B,0
58A2 09       05290          ADD     HL,BC
58A3 010A81   05300          LD      BC,DIRBUF+100H
58A6 09       05310          ADD     HL,BC
58A7 C9       05320          RET
              05330 ;
58A8 CDC901   05340 SIGNUP   CALL    01C9H   ;present sign-up message
58AB 21B558   05350          LD      HL,SUPMS
58AE CD6744   05360          CALL    4467H
58B1 CD4980   05370          CALL    8049H
58B4 C9       05380          RET
              05390 ;
58B5 20       05400 SUPMS    DEFM    '                    DIRSLOT'
    20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
    20 20 20 44 49 52 53 4C 4F 54
58D0 0A       05410          DEFB    0AH
58D1 0A       05420          DEFB    0AH
58D2 20       05430          DEFM    '        (C) Joachim Kelterbaum'
    20 20 20 20 20 20 20 20 43 29 20 4A 6F 61 63 68
    69 6D 20 4B 65 6C 74 65 72 62 61 75 6D
58F0 0A       05440          DEFB    0AH
58F1 0A       05450          DEFB    0AH
58F2 74       05460          DEFM    'this program may blow your diskettes if used
    68 69 73 20 70 72 6F 67 72 61 6D 20 6D 61 79 20                    unproperly'
    62 6C 6F 77 20 79 6F 75 72 20 64 69 73 6B 65 74
    74 65 73 20 69 66 20 75 73 65 64 20 75 6E 70 72
    6F 70 65 72 6C 79
5C29 0A       05470          DEFB    0AH
5C2A 79       05480          DEFM    'you are URGED to read the instructions first'
    6F 75 20 61 72 65 20 55 52 47 45 44 20 74 6F 20
    72 65 61 64 20 74 68 65 20 69 6E 73 74 72 75 63
    74 69 6F 6E 73 20 66 69 72 73 74
5C56 0A       05490          DEFB    0AH
5C57 0A       05500          DEFB    0AH
5C58 0A       05510          DEFB    0AH
5C59 70       05520          DEFM    'press any key to continue'
    72 65 73 73 20 61 6E 79 20 6B 65 79 20 74 6F 20
    63 6F 6E 74 69 6E 75 65
5C72 0D       05530          DEFB    0DH
              05540 ;
5800          05550          END     START
00000 TOTAL ERRORS

ABOMS   5308   ABORT   5A66   ADV     5A15   BUFFER  5290   BUFND   585D
CALC    588D   CNTZIL  522A   CONFIL  5217   COUNT   5224   CPIT    5820
DIRBUF  800A   DNR     5203   DV64    598E   ENR     5201   ERR1    580D
ERR2    58CA   FAIL    5848   FCB     526F   FCODE   528F   FDENTR  59F8
FILDIR  8000   FILNAM  5206   FOUND   582C   FXDMS   5382   GRNNB   5204
HEORG   5230   HEZIL   5232   KILLED  5843   M32     5A46   MES     581E
MS2     58DB   MSDMS   5246   MU32    5A5D   MU64    59C4   NAMCON  5AE3
NBDV    8008   NFDERR  5863   NFMES   586F   NOFX    5A0F   OK      582E
OK2     58EE   OUT1    580A   OUT11   587B   QUAD    522C   QUST    53EF
SEARCH  581D   SIGNUP  58A8   SLTMS1  5390   SLTMS2  539C   SLTNX   5205
SLTTX   5215   SNB     5200   SORGNB  5228   START   5800   SUPMS   58B5
TR2     5884   TRANS   5871   TRF     5AC6   TRFDIR  591A   TRFN    5AF9
TRXT    5810   X32     589D   XCHFLS  5A3B   XCHMS   5406   XPOS    5226
YPOS    5227   ZILAD   522E   ZILMS   5234   ZNR     5202
```

MERGELINE

by Thomas G. Hanlin III and David B. Lewis

Computing can be frustrating. Have you ever typed in a great program, only to be greeted by BASIC's "OUT OF MEMORY" error, or to find that the program is so badly set up that it runs too slowly? Fixing it usually means doing a lot of re-typing.

MERGELINE is a short (97-byte) utility that takes care of this problem. It allows you to merge lines in your BASIC programs quickly and easily, making editing very simple. MERGELINE runs on your TRS-80 Model I, III, or 4 or TRS-80-compatible computer, disk or non-disk, with any amount of memory.

The general syntax is MERGE (optional spaces) LINE (optional spaces) #####, where ##### is an existing line in your program. MERGELINE splices together this line and the line following it, reducing your program by one line and saving the bytes that BASIC uses to point to it. MERGELINE produces an error message if either line does not exist.

Here is an example of MERGELINE's use. Part of your program might be:

```
10 FOR G = 15360 TO 16383
20 POKE G,191
30 NEXT
```

After typing MERGELINE 10 <CR>, the program looks like:

```
10 FOR G = 15360 TO 16383:POKE G,191
30 NEXT
```

And typing another MERGELINE 10 puts the entire program on one line. MERGELINE is clearly a useful utility for editing your programs.

How to Load it

When you enter BASIC, answer the MEMORY SIZE? question with a number that leaves at least 97 bytes at the top of memory. The routine is totally relocatable, so you can put it anywhere in memory. For a 48K machine, you might use 65400. For 32K, 48900 is reasonable, and 32600 is fine for a 16K machine. Then load and run the program MERGELN/BAS in listing 1. Answer the LOADING ADDRESS question with the number you used for the MEMORY SIZE? query. Note that you can also give a negative number; because 32767 is the largest positive integer which can be expressed in two bytes, a larger positive integer is expressed as itself minus 65536. Thus, -136 is the same as 65400, and MERGELINE accepts both. You should see the word "DONE" printed on the screen. You can then type NEW and load your own program. The normal MERGE for disk-based machines will work if LINE does not follow.

How it Works

The BASIC program does little more than POKE the routine into memory. It also causes a jump to the MERGE routine to be first filtered through the MERGELINE routine. That way, the program can check for the LINE token.

The source code is in listing 2. It is just for demonstration; you should not run it directly from DOS. MERGELINE requires that BASIC be loaded.

The first part of the routine just checks for the LINE token. If it is not present, then the program jumps to the normal MERGE routine. Otherwise, the line number is stored in DE, and the lines are checked for in memory. If either is missing, MERGELINE returns to BASIC with an error message. Otherwise, it checks to make sure that the combined length of the two lines will not exceed about 245 bytes.

If there are no errors, the routine starts, at line 410. It replaces the end byte of the first line with a colon. Then MERGELINE squeezes out the four-byte index to the second line. ROM BASIC routines are used to reset all the pointers, and the new length of the program is stored. Then MERGELINE jumps back into BASIC, and you can continue with your program.

Notes

MERGELINE is a command, not a statement. It can be used in a program, but it will halt execution. The line number must also be numeric, and not an expression.

Also, be wary of producing a line too long to LIST or EDIT. Because BASIC replaces keywords with one-byte tokens, a line can be fewer than 255 bytes in memory, but be too long when re-expanded.

MERGELINE is a useful addition to your library. Enjoy it.

```
10 CLS:PRINTTAB(21)"MERGELINE VERSION 1.0":PRINT:PRIN
NPUT"LOADING ADDRESS";L:IFL<0THENL=L+65536
20 FOR X=LTOL+96:READA: N=X:GOSUB30: POKEN,A:B=B+A:NEX
T: IF B-11583THENPRINT"DATA ERROR":STOP ELSEN=L+5:GOSU
B30:POKEN,PEEK(16780):N=N+1:POKEN,PEEK(16781):POKE16780,
L-INT(L/256)*256:POKE16781,L/256:PRINT"DONE":END
30 IFN>32767THENN=N-65536
40 RETURN:DATA 43,215,254,156,194,0,0,35,205,90,30,225,205,44,
27,48,72,35,126,183,40,67,43,229,197,197,94,35,86,225,235
50 DATA 237,82,17,245,0,223,193,225,48,51,197,43,54,58,35,229,23
5,42,249,64,63,237,82,229,193,225,229,209,35,35,35,35,237
60 DATA 176,225,237,91,164,64,205,252,26,42,249,64,43,43,43,43,3
4,249,64,205,97,27,195,193,29,30,14,33,30,28,195,162,25
```

```
                00030 ;       MERGELINE
                00040 ; This is the source code for the routine.  It splices
                00050 ; BASIC lines together.
                00060 ;
                00070 ; Do not run this program directly.  Use the BASIC
                00080 ; program in this article.
                00090 ;
F000            00100         ORG     0F000H
                00110
F000 2B         00120         DEC     HL      ;POINT BACK TO MERGE TOKEN
F001 D7         00130         RST     10H     ;PROCESS OUT SPACES
F002 FE9C       00140         CP      156     ;CHECK FOR LINE TOKEN
F004 C20000     00150         JP      NZ,0000 ;IF NONE, USE USUAL MERGE COMMAND
                00160                 ;THE ADDRESS IS FILLED IN BY THE BASIC PROGRAM
F007 23         00170         INC     HL      ;POINT TO DIGIT
F008 CD5A1E     00180         CALL    1E5AH   ;GET LINE NUMBER. RESULT IN DE
F00B E1         00190         POP     HL      ;REMOVE RET ADDRESS
F00C CD2C1B     00200         CALL    1B2CH   ;SEARCH FOR LINE IN MEMORY
F00F 3048       00210         JR      NC,NOLINE       ;ERROR IF NO TARGET LINE
F011 23         00220         INC     HL      ;POINT TO MSB OF ADDRESS FOR NEXT LINE
F012 7E         00230         LD      A,(HL)
F013 B7         00240         OR      A       ;IF 0, THEN NO NEXT LINE TO SPLICE
F014 2843       00250         JR      Z,NOLINE        ;PRODUCE THE ERROR
F016 2B         00260         DEC     HL      ;START OF SECOND LINE
F017 E5         00270         PUSH    HL      ;SAVE IT
F018 C5         00280         PUSH    BC      ;SAVE START OF FIRST LINE?
F019 C5         00290         PUSH    BC
F01A 5E         00300         LD      E,(HL)  ;MOVE LINE ADDRESS
F01B 23         00310         INC     HL
F01C 56         00320         LD      D,(HL)  ; TO DE.
F01D E1         00330         POP     HL      ;START OF TARGET LINE
F01E EB         00340         EX      DE,HL   ;SWITCH POINTERS
F01F ED52       00350         SBC     HL,DE   ;HL=MERGED LINE'S LENGTH
F021 11F500     00360         LD      DE,245  ;DE=A REASONABLE LENGTH
F024 DF         00370         RST     18H     ;COMPARE HL AND DE
F025 C1         00380         POP     BC
F026 E1         00390         POP     HL      ;RESTORE STACK
F027 3033       00400         JR      NC,TOOBIG       ;IF LINES TOO LONG.
F029 C5         00410         PUSH    BC
F02A 2B         00420         DEC     HL      ;POINT TO END OF TARGET LINE
F02B 363A       00430         LD      (HL),':'        ;REMOVE ZERO END BYTE
F02D 23         00440         INC     HL
F02E E5         00450         PUSH    HL
F02F EB         00460         EX      DE,HL   ;DE IS EOL + 1.
F030 2AF940     00470         LD      HL,(16633)      ;HL IS END OF PROGRAM +1
F033 3F         00480         CCF
F034 ED52       00490         SBC     HL,DE   ;DISTANCE FROM THE EOL..
F036 E5         00500         PUSH    HL      ;..TO THE END OF THE PROGRAM..
F037 C1         00510         POP     BC      ;..INTO BC.
F038 E1         00520         POP     HL      ;HL=EOL +1..
F039 E5         00530         PUSH    HL
F03A D1         00540         POP     DE      ;.. AND SO IS DE
F03B 23         00550         INC     HL      ;SKIP OVER NEXT LINE'S POINTERS
F03C 23         00560         INC     HL
F03D 23         00570         INC     HL
F03E 23         00580         INC     HL
F03F EDB0       00590         LDIR            ;MOVE PROGRAM OVER UNNEEDED 4 BYTES
F041 E1         00600         POP     HL      ;BEGINNING OF LINE
F042 ED5BA440 00610           LD      DE,(40A4H)      ;BEGINNING OF PROGRAM
```

```
F046 CDFC1A   00620          CALL   1AFCH   ;FIX LINE POINTERS
F049 2AF940   00630          LD     HL,(16633)      ;END OF PROGRAM POINTER
F04C 2B       00640          DEC    HL      ;SHRINK THE PROGRAM BY FOUR BYTES
F04D 2B       00650          DEC    HL
F04E 2B       00660          DEC    HL
F04F 2B       00670          DEC    HL
F050 22F940   00680          LD     (16633),HL      ;SAVE NEW VALUE
F053 CD611B   00690          CALL   1B61H   ;CLEAR VARIABLES
F056 C3C11D   00700          JP     1DC1H   ;RETURN TO BASIC
F059 1E0E     00710 NOLINE   LD     E,14    ;CODE FOR UNDEFINED LINE ERROR
F05B 21       00720          DEFB   21H     ;HIDE NEXT INSTRUCTION
F05C 1E1C     00730 TOOBIG   LD     E,28    ;CODE FOR STRING TOO LONG ERROR
F05E C3A219   00740          JP     19A2H   ;SEND AN ERROR AND RETURN TO BASIC
F000          00800          END    0F000H
00000 TOTAL ERRORS

NOLINE  F059    TOOBIG  F05C
```

LOWER CASE DESCENDERS FOR LINE PRINTER VII AND DMP-100 by Robert B. Koehler [Editor's note: Bob is a cassette system user, and the programs below are intended for use with a cassette-based system. However, I believe the BASIC program (used only long enough to POKE the machine code into high memory) will work under Disk BASIC if BASIC is entered with zero files (use BASIC 0 or BASIC,0 under some DOSes)].

This program uses the graphics capabilities of the Line Printer VII and DMP-100 to substitute more conventional-looking characters for the lower case g,j,p,q, and y. As part of a word-processor or similar program, a line of up to 79 text characters will be processed and printed with each call. It is not relocatable. However, it can be re-assembled with any starting address in the ORG statement of the source program.

The accompanying demonstration program contains everything needed to POKE the machine code into the top of a 16K, 32K or 48K RAM. It can be streamlined, for use with any one of these memories, as follows:

For 16K RAM:

1. Add to the end of line 10- :POKE 16527,125
2. Delete lines 12 through 35 and 2000 through 3050

For 32K RAM:

1. In line 5, change 125 to 189
2. Add to the end of line 10- :POKE 16527,189
3. Delete lines 11, 13 through 35, 1000 through 1050 and 3000 through 3050

For 48K RAM:

1. In line 5, change 125 to 253
2. Add to the end of line 10- :POKE 16527,253
3. Delete lines 11, 12, 15 through 35, and 1000 through 2050

The subroutine is entered with the line to be printed in string AP. Two output strings are generated: T for the main bodies of the characters, and B for the descenders. Graphic line feeds are included at the end of each string.

Briefly, the subroutine works as follows:

The VARPTR addresses for T and B, and the length of AP plus 3, are stored on the stack. The length of AP is also put in register B, the address of the start of AP in DE, the address of the start of T in IX, and the address of the start of B in IY. If the length of AP is zero, graphics line feeds are put in T and B. Otherwise, the ASCII value for each character in AP is examined. If it is not one of the five to be changed, that value is put into T and the ASCII value for a blank goes into B. If the character is to be changed, the graphics codes for the main body of the character are added to T, and those for the descender to B, using the Block Move instruction, LDIR. Register C, which is accumulating a count of the number of bytes in T and B, is incremented for each unchanged character, and increased by nine for each changed character. Register B is decremented for each character of AP that is processed. When the processing is complete, the three graphics line feed characters are added to T, and C is incremented after each one. The value in C is then compared with the length of AP plus 3. If they are the same, there are no descenders in the line, so a length of 3 for B is stored in its VARPTR address, and the graphics line feed characters are put in B. If there are descenders in the line, any blanks at the end of B are removed, back to the last descender (this shortens the string, and saves printing time). The three graphics line feed characters are then added. The final length of B is stored in its VARPTR address, followed by the address of the start of B, then the same is done for T, and the subroutine ends.

Because of the small capacity off the buffer in the LP VII, the printing of a long line of text with multiple extenders could require as many as six passes of the print head. The DMP-100, with its larger buffer, will print such a line with two passes.

The BASIC program listing follows, and is in turn followed by the source code listing:

```
1 REM LOWER CASE DESCENDERS (LPDESC) FOR LP-VII AND DM
P-100
2 REM BY: ROBERT B. KOEHLER, R.D. #4, BOX 174
3 REM     HOPEWELL JUNCTION, NY  12533  (914) 226-7543
5 POKE 16561,11:POKE 16562,125
10 CLEAR 1000:DEFSTR A,B,T:DEFINT I-L:DIM K(2):L=0:POKE 16
526,12
11 FOR I=32012 TO 32312:READ J:POKE I,J:NEXT
12 FOR I=-17140 TO -16840:READ J:POKE I,J:NEXT
13 FOR I=-756 TO -456:READ J:POKE I,J:NEXT
15 INPUT "<1> FOR 16K
<2> FOR 32K
<3> FOR 48K";IM
20 ON IM GOTO 25,30,35
25 POKE 16527,125:GOTO 40
30 POKE 16527,189:GOTO 40
35 POKE 16527,253
40 CLS:PRINT "INPUT LINE(S) TO BE PRINTED - (79 CHARS/LIN
E MAX)"
50 AP="":INPUT AP:GOSUB 100:GOTO 50
100 T="":B="":K(0)=VARPTR(T):K(1)=VARPTR(B):K(2)=VARPTR(AP)
110 L=USR(VARPTR(K(0)))
120 LPRINT T;B;:RETURN
1000 DATA 205,127,10,14,3,94,35,86,35,13,40,3,213,24,246,26,71,19
8,3,245,235,35,94,35,86,14,0,221,33,57,126,253,33,28,127,120,254,0,
40,96,26,254,103,40,18,254,106,40,47,254
1010 DATA 112,40,49,254,113,40,51,254,121,40,53,24,57,217,33,223
,125,221,229,209,1,9,0,237,176,213,221,225,253,229,209,1,9,0,237,1
76,213,253,225,217,121,198,9,79,24,36,217,33,241,125
1020 DATA 24,221,217,33,3,126,24,215,217,33,21,126,24,209,217,33
,39,126,24,203,221,119,0,253,54,0,32,12,221,35,253,35,19,5,32,160,2
21,54,0,18,12,221,54,1,10,12,221,54,2,30
1030 DATA 12,241,225,185,32,20,54,3,253,33,28,127,253,54,0,18,25
3,54,1,10,253,54,2,30,24,18,65,253,43,253,126,0,254,32,32,3,5,24,24
4,253,35,112,24,224,35,17,28,127,115,35
1040 DATA 114,225,113,35,17,57,126,115,35,114,201,18,128,184,19
6,196,168,252,128,30,18,128,128,132,132,132,131,128,30,18,128,128
,128,128,128,253,128,30,18,128,128,132,132,132,131,128,30,18,128,
252
1050 DATA 168,196,196,184,128,30,18,128,135,128,128,128,128,128
,30,18,128,184,196,196,168,252,128,30,18,128,128,128,128,135,
128,30,18,128,188,192,192,160,252,128,30,18,128,128,132,132,132,1
31,128,30
2000 DATA 205,127,10,14,3,94,35,86,35,13,40,3,213,24,246,26,71,19
8,3,245,235,35,94,35,86,14,0,221,33,57,190,253,33,28,191,120,254,0,
40,96,26,254,103,40,18,254,106,40,47,254
2010 DATA 112,40,49,254,113,40,51,254,121,40,53,24,57,217,33,223
,189,221,229,209,1,9,0,237,176,213,221,225,253,229,209,1,9,0,237,1
76,213,253,225,217,121,198,9,79,24,36,217,33,241,189
2020 DATA 24,221,217,33,3,190,24,215,217,33,21,190,24,209,217,33
,39,190,24,203,221,119,0,253,54,0,32,12,221,35,253,35,19,5,32,160,2
21,54,0,18,12,221,54,1,10,12,221,54,2,30
2030 DATA 12,241,225,185,32,20,54,3,253,33,28,191,253,54,0,18,25
3,54,1,10,253,54,2,30,24,18,65,253,43,253,126,0,254,32,32,3,5,24,24
4,253,35,112,24,224,35,17,28,191,115,35
2040 DATA 114,225,113,35,17,57,190,115,35,114,201,18,128,184,19
6,196,168,252,128,30,18,128,128,132,132,132,131,128,30,18,128,128
,128,128,128,253,128,30,18,128,128,132,132,132,131,128,30,18,128,
252
2050 DATA 168,196,196,184,128,30,18,128,135,128,128,128,128,128
,30,18,128,184,196,196,168,252,128,30,18,128,128,128,128,135,
128,30,18,128,188,192,192,160,252,128,30,18,128,128,132,132,132,1
31,128,30
3000 DATA205,127,10,14,3,94,35,86,35,13,40,3,213,24,246,26,71,19
8,3,245,235,35,94,35,86,14,0,221,33,57,254,253,33,28,255,120,254,0,
40,96,26,254,103,40,18,254,106,40,47,254
3010 DATA112,40,49,254,113,40,51,254,121,40,53,24,57,217,33,223,
```

```
253,221,229,209,1,9,0,237,176,213,221,225,253,229,209,1,9,0,237,17
6,213,253,225,217,121,198,9,79,24,36,217,33,241,253
3020 DATA24,221,217,33,3,254,24,215,217,33,21,254,24,209,217,33,
39,254,24,203,221,119,0,253,54,0,32,12,221,35,253,35,19,5,32,160,2
21,54,0,18,12,221,54,1,10,12,221,54,2,30
3030 DATA12,241,225,185,32,20,54,3,253,33,28,255,253,54,0,18,253
,54,1,10,253,54,2,30,24,18,65,253,43,253,126,0,254,32,32,3,5,24,244
,253,35,112,24,224,35,17,28,255,115,35
3040 DATA114,225,113,35,17,57,254,115,35,114,201,18,128,184,196
,196,168,252,128,30,18,128,128,132,132,132,131,128,30,18,128,128,
128,128,128,253,128,30,18,128,128,132,132,132,131,128,30,18,128,2
52
3050 DATA168,196,196,184,128,30,18,128,135,128,128,128,128,128,
30,18,128,184,196,196,168,252,128,30,18,128,128,128,128,128,135,1
28,30,18,128,188,192,192,160,252,128,30,18,128,128,132,132,132,13
1,128,30
```

```
                00100 ; LPDESC - DESCENDERS FOR LP VII & DMP-100
B00C            00110          ORG     46396
B00C CD7F0A     00120          CALL    2687  ; VARPTR(K(0)) IN HL
B00F 0E03       00130          LD      C,3
B011 5E         00140 LPB      LD      E,(HL)  ; VARPTR(T & B) ON STACK,
B012 23         00150          INC     HL          ; VARPTR(AP) IN DE
B013 56         00160          LD      D,(HL)
B014 23         00170          INC     HL
B015 0D         00180          DEC     C
B016 2803       00190          JR      Z,LPA
B018 D5         00200          PUSH    DE
B019 18F6       00210          JR      LPB
B01B 1A         00220 LPA      LD      A,(DE)
B01C 47         00230          LD      B,A   ; LEN(AP) IN REGISTER B
B01D C603       00240          ADD     A,3
B01F F5         00250          PUSH    AF   ; LEN(AP)+3 ON STACK
B020 EB         00260          EX      DE,HL
B021 23         00270          INC     HL
B022 5E         00280          LD      E,(HL)  ; START. ADDR. OF AP IN DE
B023 23         00290          INC     HL
B024 56         00300          LD      D,(HL)
B025 0E00       00310          LD      C,0
B027 DD2139BE   00320          LD      IX,TST  ; START. ADDR. OF T IN IX
B02B FD211CBF   00330          LD      IY,BST  ; START. ADDR. OF B IN IY
B02F 78         00340          LD      A,B
B030 FE00       00350          CP      0       ; CHECK FOR EMPTY STRING
B032 2860       00360          JR      Z,LPS
B034 1A         00370 LPC      LD      A,(DE)   ; GET A CHARACTER
B035 FE67       00380          CP      103      ; IS IT A LWR.CASE G?
B037 2812       00390          JR      Z,LPD
B039 FE6A       00400          CP      106      ; IS IT A LWR. CASE J?
B03B 282F       00410          JR      Z,LPE
B03D FE70       00420          CP      112      ; IS IT A LWR. CASE P?
B03F 2831       00430          JR      Z,LPF
B041 FE71       00440          CP      113      ; IS IT A LWR. CASE Q?
B043 2833       00450          JR      Z,LPG
B045 FE79       00460          CP      121      ; IS IT A LWR. CASE Y?
B047 2835       00470          JR      Z,LPH
B049 1839       00480          JR      LPJ        ; NOT A SPEC. CHARACTER
B04B D9         00490 LPD      EXX
B04C 210FBD     00500          LD      HL,GST  ; ADDR. OF L.C. G CODE
B04F DDE5       00510 LPK      PUSH    IX   ; MOVE GRAPH. CODE FOR BODY
B051 D1         00520          POP     DE     ; OF CHAR. TO T
B052 010900     00530          LD      BC,9
B055 EDB0       00540          LDIR
B057 D5         00550          PUSH    DE
B058 DDE1       00560          POP     IX
B05A FDE5       00570          PUSH    IY   ; MOVE GRAPH. CODE FOR DESC.
B05C D1         00580          POP     DE    ; OF CHAR. TO B
B05D 010900     00590          LD      BC,9
B060 EDB0       00600          LDIR
B062 D5         00610          PUSH    DE
B063 FDE1       00620          POP     IY
B065 D9         00630          EXX
B066 79         00640          LD      A,C
B067 C609       00650          ADD     A,9
B069 4F         00660          LD      C,A
B06A 1824       00670          JR      LPL
B06C D9         00680 LPE      EXX
B06D 21F1BD     00690          LD      HL,JST   ; ADDR. OF L.C. J CODE
B070 18DD       00700          JR      LPK
B072 D9         00710 LPF      EXX
B073 2103BE     00720          LD      HL,PST   ; ADDR. OF L.C. P CODE
B076 1807       00730          JR      LPK
B078 D9         00740 LPG      EXX
B079 2115BE     00750          LD      HL,QST   ; ADDR. OF L.C. Q CODE
B07C 18D1       00760          JR      LPK
B07E D9         00770 LPH      EXX
B07F 2127BE     00780          LD      HL,YST   ; ADDR. OF L.C. Y CODE
B082 18CB       00790          JR      LPK
B084 DD7700     00800 LPJ      LD      (IX+0),A   ; PUT CHAR. IN T
B087 FD360020   00810          LD      (IY+0),32  ; PUT BLANK IN B
B08B 0C         00820          INC     C
B08C DD23       00830          INC     IX
B08E FD23       00840          INC     IY
B090 13         00850 LPL      INC     DE
B091 05         00860          DEC     B
B092 28A0       00870          JR      NZ,LPC
B094 DD360012   00880 LPS      LD      (IX+0),18   ; PUT GRAPH. LINE FEED
B098 0C         00890          INC     C             ; IN T
B099 DD36010A   00900          LD      (IX+1),10
B09D 0C         00910          INC     C
B09E DD36021E   00920          LD      (IX+2),30
B0A2 0C         00930          INC     C
B0A3 F1         00940          POP     AF          ; LEN(AP)+3
B0A4 E1         00950          POP     HL          ; VARPTR(B)
B0A5 B9         00960          CP      C           ; CHECK FOR NO DESC'S.
B0A6 2014       00970          JR      NZ,LPM
B0A8 3603       00980          LD      (HL),3      ; IF NO DESC'S. PUT ONLY
B0AA FD211CBF   00990          LD      IY,BST      ; GRAPH. LINE FEED IN B
B0AE FD360012   01000 LPR      LD      (IY+0),18
B0B2 FD36010A   01010          LD      (IY+1),10
B0B6 FD36021E   01020          LD      (IY+2),30
B0BA 1812       01030          JR      LPN
B0BC 41         01040 LPM      LD      B,C    ; SHORTEN B TO END AT LAST DESC.
B0BD FD2B       01050 LPP      DEC     IY
B0BF FD7E00     01060          LD      A,(IY+0)
B0C2 FE20       01070          CP      32
B0C4 2003       01080          JR      NZ,LPQ
B0C6 05         01090          DEC     B
B0C7 18F4       01100          JR      LPP
B0C9 FD23       01110 LPQ      INC     IY
B0CB 70         01120          LD      (HL),B
B0CC 18E0       01130          JR      LPR
B0CE 23         01140 LPN      INC     HL
B0CF 111CBF     01150          LD      DE,BST
B0D2 73         01160          LD      (HL),E  ; START. ADDR. OF B TO
B0D3 23         01170          INC     HL          ; VARPTR(B)+1&2
B0D4 72         01180          LD      (HL),D
B0D5 E1         01190          POP     HL
B0D6 71         01200          LD      (HL),C    ; LEN(T) AND START. ADDR. OF
B0D7 23         01210          INC     HL          ; T TO VARPTR(T)
B0D8 1139BE     01220          LD      DE,TST
B0DB 73         01230          LD      (HL),E
B0DC 23         01240          INC     HL
B0DD 72         01250          LD      (HL),D
B0DE C9         01260          RET             ; RETURN TO BASIC PROGRAM
B0DF 1280       01270 GST      DEFW    8012H     ;START OF L.C. G CODE
B0E1 88C4       01280          DEFW    0C488H
B0E3 C4A8       01290          DEFW    0A8C4H
B0E5 FC80       01300          DEFW    80FCH
B0E7 1E12       01310          DEFW    121EH
B0E9 8080       01320          DEFW    8080H
B0EB 8484       01330          DEFW    8484H
B0ED 8483       01340          DEFW    8384H
B0EF 801E       01350          DEFW    1E80H
B0F1 1280       01360 JST      DEFW    8012H     ;START OF L.C. J CODE
B0F3 8080       01370          DEFW    8080H
B0F5 8080       01380          DEFW    8080H
B0F7 FD80       01390          DEFW    80FDH
B0F9 1E12       01400          DEFW    121EH
B0FB 8080       01410          DEFW    8080H
B0FD 8484       01420          DEFW    8484H
B0FF 8483       01430          DEFW    8384H
B101 801E       01440          DEFW    1E80H
B103 1280       01450 PST      DEFW    8012H     ;START OF L.C. P CODE
B105 FCA8       01460          DEFW    0A8FCH
B107 C4C4       01470          DEFW    0C4C4H
B109 8880       01480          DEFW    8088H
B10B 1E12       01490          DEFW    121EH
B10D 8087       01500          DEFW    8780H
B10F 8080       01510          DEFW    8080H
B111 8080       01520          DEFW    8080H
```

```
BE13 801E    01530          DEFW   1E80H
BE15 1280    01540 QST      DEFW   8012H      ;START OF L.C. Q CODE
BE17 88C4    01550          DEFW   0C488H
BE19 C4A8    01560          DEFW   0A8C4H
BE1B FC80    01570          DEFW   80FCH
BE1D 1E12    01580          DEFW   121EH
BE1F 8080    01590          DEFW   8080H
BE21 8080    01600          DEFW   8080H
BE23 8087    01610          DEFW   8780H
BE25 801E    01620          DEFW   1E80H
BE27 1280    01630 YST      DEFW   8012H      ;START OF L.C. Y CODE
BE29 BCC0    01640          DEFW   0C0BCH
BE2B C0A0    01650          DEFW   0A0C0H
BE2D FC80    01660          DEFW   80FCH
BE2F 1E12    01670          DEFW   121EH
BE31 8080    01680          DEFW   8080H
BE33 8484    01690          DEFW   8484H
BE35 8483    01700          DEFW   8384H
BE37 801E    01710          DEFW   1E80H
00E3         01720 TST      DEFS   227        ;START OF T
00E3         01730 BST      DEFS   227        ;START OF B
0000         01740          END
00000 TOTAL ERRORS

BST   BF1C   GST   BD0F   JST   BDF1   LPA   BD1B   LPB   BD11
LPC   BD34   LPD   BD4B   LPE   BD6C   LPF   BD72   LPG   BD78
LPH   BD7E   LPJ   BD84   LPK   BD4F   LPL   BD90   LPM   BD8C
LPN   BDCE   LPP   BD8D   LPQ   BDC9   LPR   BDAE   LPS   BD94
PST   BE03   QST   BE15   TST   BE39   YST   BE27
```

NEWDOS/80 VERSION 2 MODEL I PATCHES contributed by Truman Krumholz:

1. To eliminate the double line feed when in the DOS READY mode: Zero byte F0H in sector 0 of SYS1/SYS. It is normally a 0DH.

2. To allow where you can either enter the date and time on power up or just press <ENTER> to bypass: Zero bytes 67H, 68H, 7AH and 7BH in sector 12 of SYS0/SYS. These bytes are normally 20H, EDH, 20H, and EDH.

OPTIONAL NEWDOS/80 VERSION 2.0 ZAPS are reprinted from the CINTUG (Cincinnati TRS-80 User Group) newsletter:

These ZAPs to NEWDOS/80 Version 2.0, Models I and III allow the changing of the PDRIVE parameter without writing to disk. The syntax for use is the same as before except "B" is used instead of "A" in the PDRIVE command. The zap allows the PDRIVE table in memory to be changed without changing the disk record.

******* ZAP 091 ******* 03/02/84 ******* V2M1 *******
SYS16/SYS,02,FB change FE 41 21 20 06 to C3 C0 51 00 00
SYS16/SYS,04,D4 change all zeroes to:
FE 41 12 20 03 C3 F1 4F FE 42 C2 F7 4F 3E 41 12 77 C5 E5 01 00 07 21 E2 4D 71 23 10 FC E1 C1 C3 F1 4F

******* ZAP 086 ******* 03/02/84 ******* V2M3 *******
SYS16/SYS,02,EB change 7E FE 41 12 20 to 7E C3 CB 51 20
SYS16/SYS,04,DF change all zeroes to:
FE 41 12 20 03 C3 E0 4F 3D FE 41 20 F8 C5 E5 01 00 07 21 D3 4D 71 23 10 FC E1 C1 18 E3 02

The following ZAP allows you to use the Series I Editor Assembler under NEWDOS/80 Version 2.0 on the Model III. It may also work on the Model I, but it has not been tried out. Remember that you must have applied ZAP 025 in order to transfer files from TRSDOS 2.3b over to NEWDOS/80 Version 2.0. The function of this zap is to reload the BREAK enable function of NEWDOS/80. If you are interested in patching other TRSDOS programs, the bytes to look for are 3A C9 32 78 44.

EDTASM/CMD,02,00 change 2A 0D 40 to C3 EA 5B
EDTASM/CMD,05,8D change 4C 69 63 65 6E 73 65 64 20 74 6E 20 to AE 3E C9 32 78 44 2A 0D 40 C3 6B 58

The next patches are for the FORTRAN package and also enable the BREAK key function. They are also for the Model III, and have not been checked out for the Model I. Note that FORTRAN/CMD is also called EDIT/CMD on some versions.

FORTRAN/CMD,02,DA change AF 32 1E 53 to C3 A8 7C 00
FORTRAN/CMD,43,54 change 00 00 00 00 00 00 00 00 00 00 00 00 to AF 32 1E 53 3E C9 32 78 44 C3 D2 54
F80/CMD,00,04 change C3 A1 57 00 00 00 00 00 to 3E C9 32 78 44 C3 A1 57
L80/CMD,00,07 change AF 32 15 43 to C3 E6 74 00
L80/CMD,35,72 change 30 00 00 00 00 00 00 00 00 00 00 00 to 3E C9 32 78 44 AF 32 15 43 C3 07 52

BACKING UP A COLOR COMPUTER DISK USING SUPER UTILITY PLUS - This one's short and sweet - use drive configuration settings T3D,35,17 and Super Utility will be able to backup your Color Computer disks on your Model I/III/4. Thanks to Bill Peek of the Indiana Software Group for this information.

TASMON UPGRADE AVAILABLE - By the time you receive this newsletter, the Model 4 version of TASMON (that runs in the native Model 4 mode) should be ready to ship. But don't go away just because you own a Model I or Model III - a TASMON upgrade is available for you as well. Not everyone will need the upgrade, but if you do you should contact The Alternate Source for pricing and availability.

The Model I version is upgraded to decode and display all of the known useful "undocumented" Z-80 opcodes. Now, if you use undocumented opcodes, TASMON won't choke on them. In addition, when tracing or single-stepping code, the last seven instructions executed are shown (in disassembled form) in the lower right portion of the display. This makes it a lot easier to figure out how you got to where you are now, in case a program takes a jump to someplace you didn't expect. Finally, a new command has been added - the "Q" command is now used to compare two blocks of memory. For example, a user might type (user input is underlined):

Q A000 B000 A154 B154

In the above example, the user told TASMON to compare two blocks of memory starting at A000H and B000H respectively. TASMON reported that all bytes were the same until it reached A154H and B154H, where the bytes stored at those locations were different. At this point the user could restart the compare by simply typing:

Q <ENTER>

and TASMON would begin comparing at A155H and B155H (in this example). This function could be useful in determining the differences (or patched instructions) in two similar code segments.

The Model III version has all the above enhancements, plus the keyboard scan routine has been revised so as to NOT use the Model III lookup tables (which seem to move around in the various versions of the Model 4 ROM!).

There is a cost to be paid for these enhancements, in terms of memory - the new versions occupy a little bit more than the 8K of memory used by the original version. But, if you have a need of the ability to disassemble undocumented opcodes, to see in disassembled form what you've just executed, to compare blocks of memory, or to run TASMON on a Model 4 in the Model III mode, you should contact The Alternate Source for price and availability of these upgrades.

[NOTICE - The above should not be taken as an unbiased review, but rather should be considered more of a news release. Your editor was responsible for much of the added code for the enhancements (but NOT for the native Model 4 version).]

CHRISTIAN COMPUTING MAGAZINE is a bimonthly publication for, you guessed it, Christian computer users. One benefit of this magazine is that they review and cross-reference software that might appeal to churches, church leadership, or Christians in general (but possibly not to the average computer user). Another feature is "Telecommunications Cross-Talk", which provides news of Christian-oriented Bulletin Board Systems (the issue I received mentioned two, at (817) 483-1264 and (713) 721-0888) and other telecommunications interests. This magazine is not a TRS-80 specific publication, but it does include coverage of TRS-80 software. The September-October 1984 issues had 28 pages in full size, glossy format (and this is apparently their third issue). Subscription rates at present are $13 U.S., $18 Foreign, or $3.00 per single issue. Address inquiries to Christian Computing Magazine, 72 Valley Hill Road, Stockbridge, Georgia 30281 or telephone (404) 474-0007.

INTERRPT/CMD

A MODULE BASED ON NEWDOS80/V2 [MODEL I VERSION] TO INTERRUPT A RUNNING PROGRAM AND SAVE A COMPLETE CORE-IMAGE TO DISK. THE INTERRUPTED PROGRAM MAY BE RESTARTED SOME LATER TIME

by Joachim Kelterbaum
(Frankenstr. 305, 4300 Essen 1, West Germany)

If you are the type of user who has programs that take hours to run, read on. Probably you have experienced a sudden reboot after 2 hours or so of running your favorite application. The whole time spent was just wasted, because you have to start over again - hoping not to get another one of those nasty reboots.

Well, how does the following sound?

Start your application and save the actual contents of RAM including registers, stack etc. to disk (about every hour). Then just resume execution. If you have an unwanted reboot or silent death or whatever, start from where you last saved RAM to disk. This can save some time and aggravation in the long run.

To activate the module just enter INTERRPT. If your application polls the keyboard from time to time via the device control block (BASIC does this all the time to detect SHIFT-@) you may just press SHIFT-CLEAR. From this very moment the whole machine status is saved to disk onto a file called MEMORY/CIM:0 (it has to be created before the first application and must allow for 39 granules of space). After the status has been saved, a reboot is done. Now you can either resume execution by entering INTERRPT,* or run anything else before you do so. That's all there is to it.

You will find a commented source listing of the module below. As it is (at least I hope so) well commented, I'll only deal with the major aspects of operation here:

If you enter the program via INTERRPT (not followed by "*"), first the original keyboard driver address is plugged into the instruction SCALL (line 1750). Then the module SCALL...END is relocated directly below HIMEM and its new entry address is stored to RELAD and to the DCB address. HIMEM is updated accordingly. Then the first sector of the file INTERRPT/CMD is written back to disk. This permanently updates RELAD for a future restart.

The instructions RLCT (line 830) to DONE (line 1090) update a few absolute addresses in the relocated module using RELTBL as a relocation table. This has to be done because not all of the Z80 code is 'truly relocatable'.

After this a sign on message is displayed ((FLAG) = 0) and a jump to DOS is done. Now the new keyboard intercept routine is active (lines 1750 to 1820). If and only if SHIFT-CLEAR is pressed, a jump to SHFCLR is done. Then all registers are saved to stack and the stackpointer is stored to STACK = 4023H (2 unused bytes in the video DCB). After this RAM from 3C00H to FFFFH except for 4D00H-51FFH (DOS overlay-area) is saved to the file MEMORY/CIM:0.

Note that there is no explicit file buffer specified in the open file command. Instead, IX is loaded with the FCB address, and IX+3, IX+4 (that is the address of the associated file buffer) is changed. By this means you save additional 256 bytes for a buffer. The buffer itself sort of 'floats' through RAM. When all RAM has been saved (ENDIT, line 2220) the file is closed, a message (MS) is displayed, a wait for a few seconds is done and a reboot is forced by the HALT instruction.

If you enter the routine by INTERRPT,* a jump to STAR (line 1250) is done. First the stackpointer is moved into 3FF0H (video RAM) so it won't be in the way while reloading the original RAM contents. This trick only works if you have all bits available in the video RAM (i.e. lowercase must be installed) -as far as I have been told, it does not work with the PMC80 for other hardware reasons-.

After this, the module (SCALL-END) is relocated to the address (RELAD) which has been written to disk before. The keyboard driver address is plugged again and a jump to RLCT is done, where some addresses in the relocated module are fixed up again. As (FLAG) = 1 now, the program continues at STARNT (line 1520). Then the entry address of RESTOR (line 2400) in the relocated module is calculated and a jump to that address is done. Now the RAM is read back from the file MEMORY/CIM again. It is stored beginning at 4000H (rel. record # 4 ff.), because the video RAM still acts as temporary stack. The DOS overlay-area is skipped again. When FFFFH is reached, the FCB is repositioned to record # 0, the old stack is restored, and the original screen contents are read back (4 records = 1024 bytes).

Then the FCB is positioned to EOF and the file is closed (this is necessary because the original contents of the FCB (line 3120) are needed for subsequent "shift-clear's".

Finally, all registers are restored. Now, everything is fixed up except for the overlay-area. We now simply plug 0 into addre 4317H to make DOS believe, it had loaded overlay # 0 (whi would be BOOT/SYS). When the next overlay request will follow DOS will take care of that by itself, so we won't have to bother with it.

Finally, we jump to SCALL which was the address where we interrupted our running program (this fixes the program counter).

Please, do read the comments at the beginning of the listing before using this program!

```
5200          00100         ORG     5200H
              00110 ;
              00120 ;INTERRPT/CMD
              00130 ;
              00140 ;(C) JOACHIM KELTERBAUM
              00150 ;
              00160 ;
              00170 ;This module intercepts the keyboard driver if activated.
              00180 ;By pressing shift-clear any running program can be
              00190 ;interrupted as long as it polls the keyboard via DCB.
              00200 ;When shift-clear is recognized, the whole RAM 3C00H -
              00210 ;FFFFH except DOS overlay area including all registers
              00220 ;and PC is saved to disk.
              00230 ;Then a reboot is executed.
              00240 ;When needed -some later time- enter INTERRPT,* from
              00250 ;DOS. Your interrupted program will be reloaded from
              00260 ;disk and will continue execution where it left off.
              00270 ;The following conditions must be met !!!
              00280 ;The program to be interrupted must poll the keyboard
              00290 ;via DCB from time to time (BASIC does this anyway).
              00300 ;The program must honour HIMEM.
              00310 ;This module relocates itself below HIMEM.
              00320 ;
              00330 ;It is primarily intended to be used when a program
              00340 ;runs over a long period of time.
              00350 ;Be careful, if your program uses disk-files, because
              00360 ;those files must stay in the same locations before
              00370 ;and after interruption unless they have been closed
              00380 ;before interruption.
              00390 ;This is due to the fact that the information of an
              00400 ;(eventually) open FCB is saved in the core-image and
              00410 ;will be reused after restart of the interrupted
              00420 ;program.
              00430 ;If you do not take this precaution, you might destroy
              00440 ;your disk !!!!!!!!!!!!!!!!!!
              00450 ;ATTENTION !!!!
              00460 ;
              00470 ;This program assumes an existing file MEMORY/CIM:0
              00480 ;Create it, if necessary. It will need 39 GRN's of
              00490 ;diskette space.
              00500 ;
              00510 ;
              00520 ;
5200 0000     00530 RELAD   DEFW    $-$     ;put addr. of relocated routine
              00540                         ;here
5202 7E       00550 START   LD      A,(HL)  ;get first parameter (x)
5203 FE2A     00560         CP      '*'     ;is it * ?
5205 CAEE52   00570         JP      Z,STAR  ;yes ==> STAR
5208 2A1640   00580         LD      HL,(4016H)      ;get KBD driver address
520B 225454   00590         LD      (SCALL+1),HL    ;plug into SCALL+1
520E 219455   00600         LD      HL,END          ;last byte of mod. to be
              00610                         ;relocated
5211 ED5B4940 00620         LD      DE,(4049H)      ;get HIMEM value
5215 014201   00630         LD      BC,END-SCALL+1  ; # bytes to move
5218 ED88     00640         LDDR                    ; move module
521A ED534940 00650         LD      (4049H),DE      ;update HIMEM
521E 13       00660         INC     DE              ;first addr. of relocated
              00670                         ;module
521F ED530052 00680         LD      (RELAD),DE      ;plug into RELAD
5223 ED531640 00690         LD      (4016H),DE      ;and KBD driver addr.
5227 0600     00700         LD      B,0     ;record length = 265
5229 113353   00710         LD      DE,FCB1 ;point to FCB of INTERRPT/CMD
```

```
522C 215353  00720        LD    HL,BUFF2; buffer for that file
522F CD2444  00730        CALL  4424H  ;open file
5232 CD3644  00740        CALL  4436H  ;read first sector
5235 210052  00750        LD    HL,RELAD        ;plug (RELAD)
5238 7E      00760        LD    A,(HL)          ;into rel. words
5239 325753  00770        LD    (BUFF2+4),A     ;4 and 5 of
523C 23      00780        INC   HL              ;buffer
523D 7E      00790        LD    A,(HL)          ;and
523E 325853  00800        LD    (BUFF2+5),A     ;
5241 CD3F44  00810        CALL  443FH           ;position to start of file
5244 CD3C44  00820        CALL  443CH           ;write back first sector
5247 ED5B0052 00830 RLCT  LD    DE,(RELAD)      ;relocate code from
524B 212453  00840        LD    HL,RELTBL       ;SCALL to END using
524E 7E      00850 NXTLOC LD    A,(HL)          ;RELTBL as relocation
524F 47      00860        LD    B,A             ;table
5250 23      00870        INC   HL              ;this method of relocation
5251 B6      00880        OR    (HL)            ;is described in
5252 2814    00890        JR    Z,DONE          ;TRS-80 ROM ROUTINES DOCUMENTEI
5254 E5      00900        PUSH  HL              ;by Jack Decker
5255 66      00910        LD    H,(HL)
5256 68      00920        LD    L,B
5257 19      00930        ADD   HL,DE
5258 4E      00940        LD    C,(HL)
5259 23      00950        INC   HL
525A 46      00960        LD    B,(HL)
525B 2B      00970        DEC   HL
525C D5      00980        PUSH  DE
525D EB      00990        EX    DE,HL
525E 09      01000        ADD   HL,BC
525F EB      01010        EX    DE,HL
5260 73      01020        LD    (HL),E
5261 23      01030        INC   HL
5262 72      01040        LD    (HL),D
5263 D1      01050        POP   DE
5264 E1      01060        POP   HL
5265 23      01070        INC   HL
5266 18E6    01080        JR    NXTLOC
5268 3A3253  01090 DONE   LD    A,(FLAG)        ;test if FLAG = 0
526B FE00    01100        CP    0       ;(means no X-option chosen)
526D C21753  01110        JP    NZ,STARNT; if X-option was chosen
5270 217952  01120        LD    HL,SGNMS        ;display sign-up
5273 CD6744  01130        CALL  4467H           ;message on screen
5276 C32D40  01140        JP    402DH           ;==> DOS
             01150 ;
5279 52      01160 SGNMS  DEFM  'Run-time interrupt routine activated'
     75 6E 2D 74 69 6D 65 20 69 6E 74 65 72 72 75 70
     74 20 72 6F 75 74 69 6E 65 20 61 63 74 69 76 61
     74 65 64
5290 0A      01170        DEFB  0AH
529E 73      01180        DEFM  'start by pressing shift-clear'
     74 61 72 74 20 62 79 20 70 72 65 73 73 69 6E 67
     20 73 68 69 66 74 2D 63 6C 65 61 72
52B8 0A      01190        DEFB  0AH
52BC 72      01200        DEFM  'recall interrupted program by entering'
     65 63 61 6C 6C 20 69 6E 74 65 72 72 75 70 74 65
     64 20 70 72 6F 67 72 61 6D 20 62 79 20 65 6E 74
     65 72 69 6E 67
52E2 0A      01210        DEFB  0AH
52E3 49      01220        DEFM  'INTERRPT,X'
     4E 54 45 52 52 50 54 2C 2A
52ED 0D      01230        DEFB  0DH
             01240 ;
52EE 21F03F  01250 STAR   LD    HL,3FF0H        ;X-option ...
             01260                 ;move Stack out of the way
             01270                 ;so it won't be bombed by being
             01280                 ;overwritten while restoring
             01290                 ;the original core-image
             01300                 ;Stack temporarily resides
             01310                 ;within Video-RAM
             01320                 ;This can only work, if lower-
             01330                 ;case is installed, so Video-RAM
             01340                 ;has all 8 bits accessable
52F1 F9      01350        LD    SP,HL
52F2 215354  01360        LD    HL,SCALL        ;move module SCALL-END
52F5 ED5B0052 01370       LD    DE,(RELAD)      ;to (RELAD)
52F9 014201  01380        LD    BC,END-SCALL+1  ;# bytes to be moved
52FC ED80    01390        LDIR  ; do it
52FE 3E01    01400        LD    A,1     ;set FLAG to 1
5300 323253  01410        LD    (FLAG),A

5303 2A1640  01420        LD    HL,(4016H)      ;get KBD driver addr.
5306 EB      01430        EX    DE,HL           ; --> DE
5307 2A0052  01440        LD    HL,(RELAD)      ;get start of new driver
530A 23      01450        INC   HL              ;plug old driver address
530B 73      01460        LD    (HL),E          ;into (RELAD+1)
530C 23      01470        INC   HL
530D 72      01480        LD    (HL),D
530E 2A0052  01490        LD    HL,(RELAD)      ;get new driver addr.
5311 221640  01500        LD    (4016H),HL      ;plug into DCB
5314 C34752  01510        JP    RLCT            ;==> RLCT
5317 2A0052  01520 STARNT LD    HL,(RELAD)      ;continue after moving
             01530                 ;routine SCALL-END
             01540                 ;while X-option is selected
531A 019F00  01550        LD    BC,RESTOR-SCALL ;calculate addr. of RESTOR
531D 09      01560        ADD   HL,BC   ;in relocated module
531E 222253  01570        LD    (JMP+1),HL      ;and
5321 C30000  01580 JMP    JP    $-$     ;jump to it
             01590 ;
5324 2B00    01600 RELTBL DEFW  FCBA+1-SCALL    ;relocation table
5326 6000    01610        DEFW  MSA+1-SCALL
5328 A600    01620        DEFW  FCBB+1-SCALL
532A 1F01    01630        DEFW  SCA+1-SCALL
532C 2B00    01640        DEFW  FCBX+2-SCALL
532E A300    01650        DEFW  FCBY+2-SCALL
5330 0000    01660        DEFW  0       ;end of table
5332 00      01670 FLAG   DEFB  0
5333 49      01680 FCB1   DEFM  'INTERRPT/CMD'
     4E 54 45 52 52 50 54 2F 43 4D 44
533F 0D      01690        DEFB  0DH
0013         01700        DEFS  19      ;fill up to 32 bytes
0100         01710 BUFF2  DEFS  100H    ;buffer
             01720 ;
4023         01730 STACK  EQU   4023H   ;unused 2 bytes in RAM used as
             01740 ;                    stack-pointer pointer
5453 CD0000  01750 SCALL  CALL  $-$     ;call original driver
5456 FE1F    01760        CP    31      ;was clear pressed ?
5458 C0      01770        RET   NZ      ;ret if not
5459 3A8038  01780        LD    A,(3880H)       ;is it shift-clear
545C CB47    01790        BIT   0,A
545E 2003    01800        JR    NZ,SHFCLR       ;yes -->SHFCLR
5460 3E1F    01810        LD    A,31    ;no, so
5462 C9      01820        RET           ;return
             01830 ;
5463 F5      01840 SHFCLR PUSH  AF      ;save all registers onto stack
5464 C5      01850        PUSH  BC
5465 D5      01860        PUSH  DE
5466 E5      01870        PUSH  HL
5467 DDE5    01880        PUSH  IX
5469 FDE5    01890        PUSH  IY
546B 08      01900        EX    AF,AF'  ;switch register sets
546C D9      01910        EXX
546D F5      01920        PUSH  AF      ;and save those, too
546E C5      01930        PUSH  BC
546F D5      01940        PUSH  DE
5470 E5      01950        PUSH  HL
5471 ED732340 01960       LD    (STACK),SP      ;store actual stack ptr.
5475 D9      01970        EXX           ;back to old register set
5476 08      01980        EX    AF,AF'
5477 0600    01990        LD    B,0     ;256-byte records
5479 DD212101 02000 FCBX  LD    IX,FCB-SCALL    ;points to FCB
547D 112101  02010 FCBA   LD    DE,FCB-SCALL    ;     same
5480 21003C  02020        LD    HL,3C00H        ;first buffer = 3C00
5483 CD2044  02030        CALL  4420H           ;open file MEMORY/CIM:0
5486 CD3C44  02040 CONT   CALL  443CH           ;write sector
5489 010001  02050        LD    BC,100H         ;update pointer
548C 09      02060        ADD   HL,BC           ;to next buffer
548D E5      02070        PUSH  HL      ;save HL
548E 01004D  02080        LD    BC,4D00H        ;is buffer at overlay-area
5491 B7      02090        OR    A
5492 ED42    02100        SBC   HL,BC
5494 E1      02110        POP   HL
5495 200B    02120        JR    NZ,OK   ;no -> OK
5497 210052  02130        LD    HL,5200H        ;yes, skip overlay-area
549A DD7503  02140        LD    (IX+3),L        ;update buffer adress
549D DD7404  02150        LD    (IX+4),H        ;in FCB
54A0 18E4    02160        JR    CONT
54A2 7D      02170 OK     LD    A,L     ;is end of RAM reached ?
54A3 B4      02180        OR    H
54A4 DD7503  02190        LD    (IX+3),L        ;update buffer addr.
```

```
54A7 DD7404   02200          LD     (IX+4),H        ;in FCB
54AA 200A     02210          JR     NZ,CONT         ;if not end of RAM go on
54AC CD2844   02220 ENDIT    CALL   4428H           ;close file
54AF CDC901   02230          CALL   01C9H           ;clear screen
54B2 217200   02240 MSA      LD     HL,MS-SCALL     ;display message MS
54B5 CD6744   02250          CALL   4467H
54B8 0605     02260          LD     B,5     ;hold it fo a while
54BA C5       02270 LPP      PUSH   BC
54BB 01FFFF   02280          LD     BC,0FFFFH
54BE CD6000   02290          CALL   60H
54C1 C1       02300          POP    BC
54C2 10F6     02310          DJNZ   LPP
54C4 76       02320          HALT           ;force REBOOT
              02330 ;
54C5 43       02340 MS       DEFM   'Core-Image has been saved'
    6F 72 65 20 49 60 61 67 65 20 68 61 73 20 62 65
    65 6E 20 73 61 76 65 64
54DE 0A       02350          DEFB   0AH
54DF 52       02360          DEFM   'Reboot follows !!!'
    65 62 6F 6F 74 20 66 6F 6C 6C 6F 77 73 20 21 21
    21
54F1 0D       02370          DEFB   0DH
              02380 ;
              02390 ;
54F2 0600     02400 RESTOR   LD     B,0     ;256-byte records
54F4 DD212101 02410 FCBY     LD     IX,FCB-SCALL    ;points to FCB
54F8 112101   02420 FCBB     LD     DE,FCB-SCALL    ;same
54FB 210040   02430          LD     HL,4000H        ;first buffer address
              02440                         ;you have to start reloading
              02450                         ;the old core image at 4000h
              02460                         ;because the screen is used
              02470                         ;as a temporary stack
              02480                         ;it is updated last !!
54FE CD2044   02490          CALL   4420H   ;open file
5501 010400   02500          LD     BC,4    ;record #4 (first after screen)
5504 CD4244   02510          CALL   4442H   ;position FCB there
5507 CD3644   02520 CONT2    CALL   4436H   ;read sector
550A 010001   02530          LD     BC,100H ;update buffer adress
550D 09       02540          ADD    HL,BC
550E E5       02550          PUSH   HL
550F 01004D   02560          LD     BC,4D00H        ;is it overlay-area
5512 B7       02570          OR     A
5513 ED42     02580          SBC    HL,BC
5515 E1       02590          POP    HL
5516 200B     02600          JR     NZ,OK2  ;if not -> OK2
5518 210052   02610          LD     HL,5200H        ;skip overlay-area
551B DD7503   02620          LD     (IX+3),L        ;update buffer address
551E DD7404   02630          LD     (IX+4),H        ;in FCB
5521 18E4     02640          JR     CONT2
5523 7D       02650 OK2      LD     A,L     ;is end of RAM reached ?
5524 B4       02660          OR     H
5525 DD7503   02670          LD     (IX+3),L        ;update buffer address
5528 DD7404   02680          LD     (IX+4),H        ;in FCB
552B 200A     02690          JR     NZ,CONT2        ;if not, continue reading
552D 010000   02700 END2     LD     BC,0    ;record 00 (first screen sector)
5530 21003C   02710          LD     HL,3C00H; buffer addr.
5533 DD7503   02720          LD     (IX+3),L        ;update in FCB
5536 DD7404   02730          LD     (IX+4),H
5539 CD4244   02740          CALL   4442H   ;position file there
553C 2A2340   02750          LD     HL,(STACK)      ;get stack-pointer of
553F F9       02760          LD     SP,HL   ;previously saved core image
5540 21003C   02770          LD     HL,3C00H
5543 0604     02780          LD     B,4     ;restore 4 sectors of old screen
5545 C5       02790 SCRLP    PUSH   BC
5546 CD3644   02800          CALL   4436H
5549 010001   02810          LD     BC,100H
554C 09       02820          ADD    HL,BC
554D C1       02830          POP    BC
554E DD7503   02840          LD     (IX+3),L
5551 DD7404   02850          LD     (IX+4),H
5554 10EF     02860          DJNZ   SCRLP
5556 CD4844   02870          CALL   4448H   ;position to EOF
5559 CD2844   02880          CALL   4428H   ;close file
555C 08       02890          EX     AF,AF'  ;restore all registers
555D D9       02900          EXX
555E E1       02910          POP    HL
555F D1       02920          POP    DE
5560 C1       02930          POP    BC
5561 F1       02940          POP    AF
5562 D9       02950          EXX
5563 08       02960          EX     AF,AF'
5564 FDE1     02970          POP    IY
5566 DDE1     02980          POP    IX
5568 E1       02990          POP    HL
5569 D1       03000          POP    DE
556A C1       03010          POP    BC
556B F1       03020          POP    AF
556C 3E00     03030          LD     A,0     ;plug 0 into # of cuccently loaded
556E 321743   03040          LD     (4317H),A       ;DOS overlay
              03050          ;to force DOS to reload the needed
              03060          ;overlay at the next RST 28H
              03070          ;This is neccesscary, because the overlay-
              03080          ;area is used during disk I/O by a
              03090          ;different module than was resident before
              03100          ;saving core-image
5571 C30000   03110 SCA      JP     SCALL-SCALL     ;back to KBD driver
5574 4D       03120 FCB      DEFM   'MEMORY/CIM:0'
    45 4D 4F 52 59 2F 43 49 4D 3A 30
5580 0D       03130          DEFB   0DH
0013          03140          DEFS   19
5594          03150 END      EQU    $
5202          03160          END    START
00000 TOTAL ERRORS

BUFF2  5353   CONT   5486   CONT2  5507   DONE   5268   END    5594
END2   552D   ENDIT  54AC   FCB    5574   FCB1   5333   FCBA   547D
FCBB   54F8   FCBX   5479   FCBY   54F4   FLAG   5332   JMP    5321
LPP    54BA   MS     54C5   MSA    54B2   NXTLOC 524E   OK     54A2
OK2    5523   RELAD  5200   RELTBL 5324   RESTOR 54F2   RLCT   5247
SCA    5571   SCALL  5453   SCRLP  5545   SGNMS  5279   SHFCLR 5463
STACK  4023   STAR   52EE   STARNT 5317   START  5202
```

MODEM80 PATCH - Modem80 is a terminal program sold by The Alternate Source, that has one big advantage over the "freebie" programs you might find on your local Bulletin Board System. That is that it is capable of uploading or downloading ANY length file (when downloading the only limit is the amount of free space on your diskette). Most other terminal programs use a buffer to store received data, and when that buffer is full, you're out of luck.

However, I have had one problem when using Modem8 have a Novation Auto-Cat modem, which thinks that it s...d answer any incoming phone calls if the DTR (Data Terminal Ready) line from the RS-232 interface is up. Modem80 sets the DTR and enables the MODEM during initialization, but doesn't reset the DTR and disable the MODEM when you use the X option to exit to DOS. Suppose, as in my case, you don't really want the MODEM to answer the phone. Well, if Modem80 has been run at least once since power-up, the DTR line will be set and the MODEM will greet callers with a loud squeal after the first ring, whereupon the caller will generally hang up very quickly! This happened to me once too often so I contacted Leslie Mikesell, the author of Modem80, for a patch to make Modem-80 disable the MODEM at exit.

Here is the patch to make Modem80 release an auto-answer MODEM by dropping DTR and RTS before exit to DOS, as provided by Les. It is in LDOS PATCH format, and the Dnn,nn numbers refer to the file sector and byte offset within the file sector (in hexadecimal), if you prefer to use a file zapper program to install.

```
.Patch to Modem80 to turn off DTR & RTS at exit
.
D05,09=CD 31 91
D04,4D=CD E1 95 21 21 91 CB C6 CB CE 18 E5
.
.End of patch
```

The corresponding load addresses for the patch would be at 91EDH and 9131H.

ARE ALL HARD DISK CONTROLLERS EQUAL? Apparently not. In a recent phone conversation with Jimmy Nord of Sacremento, California, he explained that the OMTI controller that he uses gives 20% more storage per drive than the normal controller. It has onboard memory and reads and writes full tracks rather than individual sectors. The drawback is that it i... bit slower than the normal controller (but at hard disk acc... speeds, who's going to notice anyway?). I have never heard of the OMTI controller before or since, so if anyone has any further information about it why not pass it along to us?

NEWDOS/80 MODEL I DISK ALARM - This program was sent to us by Paul Fransen, secretary of a TRS-80 users group in Holland (the country, not Holland, Michigan). It was written by one of the members of his group. What it does is that it patches your Model I NEWDOS/80 system disk so that when some unsuspecting person comes along and tries to boot your disk, they will be greeted by the sound of an alarm! Of course, you have to have an amplifier hooked up to the cassette port for this to work.

When YOU boot up your disk, you'll be rewarded with silence because you'll know which key to hold down during the boot process to disable the alarm. The default is the <SHIFT> key, but you can change that to any key you like.

Now you can trap that sneaky roommate, co-worker or employee that tries to use your system disk without your knowledge. You might be surprised to find out who you catch red-handed (and red-faced) with this program!

```
10 CLEAR1000:CLS
20 PRINTTAB(23)"-=: ALARM-DISK :=-
"STRING$(63,140):PRINT@832,STRING$(63,140)
30 PRINT@256,"This program let you make an alarm-disk!!
This is done by zapping SYS0/SYS of NEWDOS 2.0.
Except for the alarm, the DOS will work normally.
When you boot up an alarm will go off. To prevent this you have
to push down a ";
40 PRINT"certain key.
So, if an unqualified person tries to start up your computer
you will be alarmed by an awful sound."
50 PRINT@896,"One moment, please.....";
60 CMD"BREAK,N"
70 M=-32768
80 FORX=0TO274:READY:POKEM+X,Y:NEXT
90 OPEN"D",1,"SYS0/SYS"
100 FIELD1,196ASA$,59ASB$,1AS C$
110 GET1,13
120 NW$(0)=B$+STRING$(5,32)
130 FIELD1,1ASA$,4ASB$,57AS C$,1ASD$,55ASE$,138ASF$
140 GET1,14:CLOSE
150 NW$(1)=A$+C$+STRING$(6,32)
160 NW$(2)=E$+STRING$(9,32)
170 FIELD1,165ASA$,1ASQ$,90ASB$
180 OPEN"D",1,"BOOT/SYS"
190 GET1,3:CLOSE
200 GOSUB560:GOSUB590:PRINT@384,"Connect an amplifier to yo
ur system.
This is a demonstration.
Don't worry, your disk will not be zapped!
When you have seen and heard enough, push the <SHIFT>-key.
210 GOSUB560:CLS
220 DEFUSR=M:PRINT@576,CHR$(31)"Push the <SHIFT>-key, when
 you have seen enough...."
230 POKE14305,1:FORT=0TO200:NEXT:PRINT@0,;:FORN=0TO2:PR
INTNW$(N);:NEXT
240 Z=USR(0):IFPEEK(14464)=0THEN240
250 PRINT@576,CHR$(31)"If you like this alarm-zap push <ENTER
>.
If you don't, push <BREAK>."
260 IFPEEK(14400)=4THENPRINT@576,CHR$(31)"Thanks for your i
nterest!":CMD"BREAK":PRINT:PRINT:END
270 IFPEEK(14400)=1THEN290
280 GOTO260
290 PRINT@576,CHR$(31)"You're sure you want to zap SYS0/SYS ?

<ENTER> = yes
<BREAK> = no"
300 IFPEEK(14591)<>0THEN300
310 IFPEEK(14400)=4THEN260
320 IFPEEK(14400)=1THENPRINT@576,CHR$(31)"One moment, plea
se.....":GOTO340
330 GOTO310
340 OPEN"R",1,"SYS0/SYS"
350 FIELD1,249ASA$,3ASB$,4AS C$
360 GET1,12
370 READA,B,C:Z$=CHR$(A)+CHR$(B)+CHR$(C):LSETB$=Z$:PUT1,1
2
380 FIELD1,33ASA$,109ASB$,114AS C$
390 GET1,15
400 Z$="":FORX=1TO109:READA:Z$=Z$+CHR$(A):NEXT:LSETB$=Z$
410 PUT1,15:CLOSE
420 PRINT@320,"That's that. Your SYS0/SYS has been zapped.
When you boot up, your banner will flash and the alarm signal
will sound. Use <SHIFT> to stop this.
Everyone who sees this program now knows it is the <SHIFT>-key
."
430 PRINT"So maybe you would like to use another key (only one!).
Hold down the key you prefer until I have found it and....

REMEMBER THIS KEY!!!"
440 FORX=0TO7:M=14336+2[X:IFPEEK(M)=0THENNEXT:GOTO440
450 R$=INKEY$:P=PEEK(M):IFM<>14400THEN470
460 IFP=1THENR$="ENTER"ELSEIFP=2THENR$="CLEAR"ELSEIFP
=4THENR$="BREAK"ELSEIFP=16THENR$=CHR$(92)ELSEIFP=32TH
ENR$="BACKS"ELSEIFP=64THENR$="TAB"ELSEIFP=128THENR$=
"SPACE"
470 PRINT:PRINT"Found it!! Just wait one moment....."
480 IFM=14464THENR$="SHIFT":GOTO550
490 MM=INT(M/256):ML=M-256*MM
500 OPEN"R",1,"SYS0/SYS"
510 FIELD1,46ASA$,4ASB$,206AS C$
520 GET1,15
530 Z$=CHR$(ML)+CHR$(MM)+CHR$(254)+CHR$(P):LSETB$=Z$
540 PUT1,15:CLOSE
550 PRINT@320,CHR$(31)"Done!!

Don't forget to push "CHR$(34)R$CHR$(34)" after booting up!":FO
RT=0TO999:NEXT:CMD"S=BOOT"
560 PRINT@896,CHR$(30);"Push a key.....";
570 IFPEEK(14591)<>0THEN570
580 IFPEEK(14591)=0THEN580ELSERETURN
590 PRINT@256,;:FORI=1TO3:PRINTSTRING$(192,32);:NEXT:RETU
RN
600 DATA217,33,0,60,17,83,128,1,192,0,237,176,33,0,60,6,192,54,32
,35,5,32,250,30,150,14,1,121,183,40
610 DATA17,67,62,1,211,255,16,254,67,62,2,211,255,16,254,13,32,2
39,29,32,230,213,33,83,128,17,0,60,1,192
620 DATA0,237,176,1,20,0,5,32,253,13,32,250,6,192,54,32,35,5,32,2
50,209,217,201,128,128,128,140,140,188,140
630 DATA132,128,128,128,128,128,128,128,128,128,188,140,172,144
,128,128,128,184,172,144,128,128,128,188,172,144,128,184,172,148
640 DATA128,128,128,128,128,128,128,128,128,128,128,128,128,128
,128,128,128,128,32,32,32,32,32,32,32,128,128,128
650 DATA128,128,191,128,128,128,128,128,176,128,160,144,128,128
,191,128,128,149,128,160,190,177,176,187,180,128,128,191,128,139
660 DATA142,129,170,149,128,128,83,111,102,116,119,97,114,101,1
28,128,128,128,128,128,128,128,128,32,32,32,32,32,32,32,32
670 DATA32,128,128,131,141,142,129,128,128,140,128,128,130,141,
135,128,128,128,143,140,142,129,136,135,128,128,128,128,130,141
680 DATA128,143,128,128,128,128,138,133,128,128,82,111,116,116,
101,114,100,97,109,128,128,128,128,128,128,128,32,32,32,32
690 DATA32,32,32,32,32
700 DATA195,0,81
710 DATA217,33,0,60,17,106,81,1,192,0,237,176,58,128,56
720 DATA254,1,40,71,33,0,60,6,192,54,32,35,5,32,250,30
730 DATA150,14,1,121,183,40,17,67,62,1,211,255,16,254,67,62
740 DATA2,211,255,16,254,13,32,239,29,32,230,213,33,106,81,17
750 DATA0,60,1,192,0,237,176,1,20,0,5,32,253,13,32,250
760 DATA6,192,54,32,35,5,32,250,209,24,178,33,228,78,54,33
770 DATA35,54,137,35,54,80,217,195,228,78,2,2,0,81
780 '
```

a program of :

Joop van Dam
Hammarskjoldplaats 322
3069 RJ Rotterdam
HOLLAND
Telephone 31+10+208984

TRSDOS 1.3 PATCH PROGRAM provided by Don Brate - This BASIC program will automatically install various patches to a TRSDOS 1.3 system disk. Each patch is explained, and you are then asked whether you want the patch to be installed. Don says he does not know who "J.F.A. Electronics" (mentioned in line 3) is, but apparently they got the patches from Tandy and Bob Snapp.

```
1 CLEAR1000:DEFINTA-Z:DIMN%,P1%,P2%,Q$,PA$,P$(46):CLS:PRI
NTTAB(13)".....":REM UPDATED 05/01/82
2 PRINTTAB(13)".      TRSDOS 1.3 PATCHES      .
3 PRINTTAB(13)".   Courtesy J.F.A. ELECTRONICS   .
```

```
4 PRINTTAB(13)".....
5 PRINT:PRINT"The current version of TRSDOS is 1.3, dated Jul
y 1, 1981.
6 PRINT"This program will allow you to apply a series of correcti
ve
7 PRINT"patches which should bring your TRSDOS up to version.
 It
8 PRINT"will also list some patches, courtesy of BOB SNAPP of S
NAPP-
9 PRINT"WARE, which allow various optional modifications to TR
SDOS
10 PRINT"1.3. ==> Select only the patches you wish to apply. Thi
s
11 PRINT"program will generate a /BLD file containing your sele
cted
12 PRINT"patches and then you will be given further instructions.
13 GOTO81
14 PRINT:INPUT"Press <ENTER> to continue...";Q$
15 CLS:PRINTSTRING$(63,"*"):PRINTTAB(26)"CAUTION":PRINTS
TRING$(63,"*")
16 PRINT:PRINT"BEFORE RUNNING this program, you must get it
 on a TRSDOS 1.3":PRINT"diskette -- preferably the one you are g
oing to patch.
17 PRINT"If you have downloaded it to a 1.1 or 1.2 disk, put that
disk":PRINT"in Drive 1 and the TRSDOS 1.3 disk in Drive 0. After
 'RESET'
18 PRINT"go to BASIC and load it from Drive 1. You may get a cle
an ":PRINT"load and be able to use it immediately. (You probably
 should
19 PRINT"'SAVE' it to the TRSDOS 1.3 disk in Drive 0.) If you do
n't":PRINT"get a clean load, you will need to use cassette or 'XFE
RSYS'
20 PRINT"to get it to the TRSDOS 1.3 diskette. In any case, be s
ure":PRINT"you are running under TRSDOS 1.3 when you execute t
his patch":PRINT"program!!!";
21 PRINT" Press <ENTER> to continue or <BREAK> to exit...";:LI
NEINPUTQ$
22 CLS:PRINT:PRINT"Be sure you have a backup TRSDOS 1.3 disk
in drive 0 and then
23 PRINT"enter a filename (suggestion 'PATCHES') but NO /exten
sion:
24 LINEINPUTF$:IFLEN(F$)>8ORLEN(F$)<3ORINSTR(F$,"/")>0TH
EN22 ELSEF$=F$+"/BLD:0
25 PRINT:PRINT"Your DO FILE will be called: ";F$
26 PRINT:INPUT"Press <ENTER> to continue...";Q$
27 OPEN"O",1,F$
28 PA$="":CLS:PRINT:PRINT"Patches 1 thru 12 correct a problem
with 'XFERSYS', a problem
29 PRINT"with BASIC, a formatting error message, an error in fil
e load-
30 PRINT"ing, and change the release date to Jul 1. These patche
s are
31 PRINT"mandatory and should be excluded ONLY if you have pre
viously
32 PRINT"applied them.":PRINT
33 PRINT"APPLY PATCHES 1-12 <Y> OR <N>? ";
34 LINEINPUTQ$:IFQ$="N"THEN35 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO33 ELSEP1=1:P2=12:PA$=PA$+"1-12, ":G
OSUB127
35 CLS:PRINT:PRINT"Patch 13 eliminates the graphics during boo
t-up.":PRINT"APPLY PATCH 13 <Y> OR <N>? ";
36 LINEINPUTQ$:IFQ$="N"THEN37 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO35 ELSEP1=13:P2=13:PA$=PA$+"13, ":G
OSUB127
37 PRINT:PRINT"Patch 14 eliminates the boot up message from th
e TRSDOS":PRINT"version through the serial number.
38 PRINT"APPLY PATCH 14 <Y> OR <N>? ";
39 LINEINPUTQ$:IFQ$="N"THEN40 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO38 ELSEP1=14:P2=14:PA$=PA$+"14, ":G
OSUB127
40 PRINT:PRINT"Patch 15 eliminates the Tandy Copyright messag
e during boot.
41 PRINT"APPLY PATCH 15 <Y> OR <N>? ";
42 LINEINPUTQ$:IFQ$="N"THEN43 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO41 ELSEP1=15:P2=15:PA$=PA$+"15, ":G
OSUB127
43 PRINT:PRINT"Patch 16 bypasses the TIME? question during bo
ot-up.
44 PRINT"APPLY PATCH 16 <Y> or <N>? ";
45 LINEINPUTQ$:IFQ$="N"THEN46 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO44 ELSEP1=16:P2=16:PA$=PA$+"16, ":G
OSUB127
46 PRINT:PRINT"Patch 17 bypasses both DATE? and TIME? quest
ion during boot-up.
47 PRINT"APPLY PATCH 17 <Y> OR <N>? ";
48 LINEINPUTQ$:IFQ$="N"THEN49 ELSEIFQ$<>"Y"THENPRINT
HR$(27);CHR$(30);:GOTO47 ELSEP1=17:P2=17:PA$=PA$+"17, ":G
OSUB127
49 CLS:PRINT:PRINT"Patch 18 changes the periods at 'TRSDOS R
EADY' to spaces.
50 PRINT"APPLY PATCH 18 <Y> OR <N>? ";
51 LINEINPUTQ$:IFQ$="N"THEN52 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO50 ELSEP1=18:P2=18:PA$=PA$+"18, ":G
OSUB127
52 PRINT:PRINT"Patch 19/20 corrects an error in FORMAT wherei
n disk I/O error":PRINT"retry counter is left incorrectly at 2 (rec
omended patch).
53 PRINT"APPLY PATCHES 19/20 <Y> OR <N>? ";
54 LINEINPUTQ$:IFQ$="N"THEN55 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO53 ELSEP1=19:P2=20:PA$=PA$+"19, 20, "
:GOSUB127
55 PRINT:PRINT"Patch 21 allows you to support 65535 logical rec
ords by":PRINT"expressing those above 32767 as negative number
s.":PRINT"Eg. record # + (65536 * (record #)32767))
56 PRINT"APPLY PATCH 21 <Y> OR <N>? ";
57 LINEINPUTQ$:IFQ$="N"THEN58 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO56 ELSEP1=21:P2=21:PA$=PA$+"21, ":G
OSUB127
58 PRINT:PRINT"Patch 22/23 corrects 2 potential errors in the DI
RECTORY.
59 PRINT"APPLY PATCHES 22/23 <Y> OR <N>? ";
60 LINEINPUTQ$:IFQ$="N"THEN61 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO59 ELSEP1=22:P2=23:PA$=PA$+"22, 23, "
:GOSUB127
61 PRINT:PRINT"Patch 24/25 are mandatory patches to correct an
":PRINT"I/O error in a directory USR call.
62 PRINT"APPLY PATCHES 24/25 <Y> OR <N>? ";
63 LINEINPUTQ$:IFQ$="N"THEN64 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO62 ELSEP1=24:P2=25:PA$=PA$+"24, 25, "
:GOSUB127
64 PRINT:PRINT"Patch 26/27/28 allows DEBUG into low memory
RAM and ROM.
65 PRINT"APPLY PATCHES 26/27/28 <Y> OR <N>? ";
66 LINEINPUTQ$:IFQ$="N"THEN67 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO65 ELSEP1=26:P2=28:PA$=PA$+"26, 27,
28, ":GOSUB127
67 PRINT:PRINT"Patch 29 provides detailed ERROR MESSAGES fr
om TRSDOS rather":PRINT"than 'ERROR 05' type of messages.
68 PRINT"APPLY PATCH 29 <Y> OR <N>? ";
69 LINEINPUTQ$:IFQ$="N"THEN70 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO68 ELSEP1=29:P2=29:PA$=PA$+"29 ":GO
SUB127
70 PRINT:PRINT"This is a Patch to BASIC/CMD. It will allow you
 to read in a":PRINT"BASIC program with up to a 50% increase in
speed.":PRINT"If the program leaves less than 400 bytes in":PRI
NT"memory you will recieve an out of Memory error"
71 PRINT"APPLY PATHES 30-45 <Y> OR <N> ";
72 LINEINPUTQ$:IFQ$="N"THEN73 ELSEIFQ$<>"Y"THENPRINTC
HR$(27);CHR$(30);:GOTO71 ELSEP1=30:P2=45:PA$=PA$+"30-45 ";
GOSUB127
73 CLOSE#1:CLS:PRINT:PRINT"YOU HAVE SELECTED THE FOLL
OWING PATCHES:
74 PRINT:PRINTTAB(5)PA$
75 PRINT:PRINT"After reading these instructions, simply press '
RESET' and":PRINT"when you get TRSDOS READY type 'CLEAR'.
76 PRINT:PRINT"When the display settles down and you get TRSD
OS READY again,":PRINT"type DO ";F$;"
77 PRINT:PRINT"The patches will be applied to your TRSDOS 1.3
disk (which":PRINT"should still be in Drive 0) one by one. If you
see the
78 PRINT"error message 'String Not Found', ignore it! This mean
s only":PRINT"that the particular patch has already been applied
to your
79 PRINT"disk. Now press 'RESET'.
80 N=N+1:GOTO80
81 P$(1)="PATCH XFERSYS/CMD (ADD=548E,FIND=3500FD21,CHG
=FD360001)
82 P$(2)="PATCH BASIC/CMD (ADD=58F8,FIND=F1,CHG=00)
```

```
 83 F$(3)="PATCH *0 (ADD=503B,FIND=467269,CHG=536174)
 84 P$(4)="PATCH *0 (ADD=5044,FIND=31,CHG=32)
 85 P$(5)="PATCH *6 (ADD=5850,FIND=3A62,CHG=BF5F)
 86 P$(6)="PATCH *6 (ADD=5FBE,FIND=20697320616374,CHG=0D11
6544C31C44)
 87 P$(7)="PATCH *0 (ADD=5044,FIND=32,CHG=31)
 88 P$(8)="PATCH *0 (ADD=503A,FIND=20536174204D,CHG=576564
20204A)
 89 P$(9)="PATCH *0 (ADD=5040,FIND=6179,CHG=756C)
 90 P$(10)="PATCH *7 (ADD=579C,FIND=0955,CHG=3851)
 91 P$(11)="PATCH *7 (ADD=5135,FIND=207468652064,CHG=3F200
33A7D4E)
 92 P$(12)="PATCH *7 (ADD=513B,FIND=69736B657474,CHG=FE81C
A0D55C9)
 93 P$(13)="PATCH *0 (ADD=4E89,FIND=1B,CHG=27)
 94 P$(14)="PATCH *0 (ADD=4E95,FIND=1B,CHG=27)
 95 P$(15)="PATCH *0 (ADD=4E9B,FIND=1B,CHG=27)
 96 P$(16)="PATCH *0 (ADD=4EFE,FIND=215451,CHG=C32E4F)
 97 P$(17)="PATCH *0 (ADD=4EB8,FIND=213B51,CHG=C3394F)
 98 P$(18)="PATCH *1 (ADD=4E78,FIND=2E,CHG=20)
 99 P$(19)="PATCH *7 (ADD=4E55,FIND=12,CHG=0F)
100 P$(20)="PATCH *7 (ADD=4E5B,FIND=0C,CHG=09)
101 P$(21)="PATCH BASIC/CMD (ADD=5EE8,FIND=451E,CHG=012
B)
102 P$(22)="PATCH *10 (ADD=4E2A,FIND=3ADA4E,CHG=784F00)
103 P$(23)="PATCH *10 (ADD=4E47,FIND=02,CHG=03)
104 P$(24)="PATCH *10 (ADD=4E2E,FIND=CD3E4B,CHG=CD8A50)
105 P$(25)="PATCH *10 (ADD=508A,FIND=4469736B,CHG=4FC33E
4B)
106 P$(26)="PATCH *5 (ADD=4EDF,FIND=38E6,CHG=0000)
107 P$(27)="PATCH *5 (ADD=4F04,FIND=D0,CHG=C9)
108 P$(28)="PATCH *5 (ADD=506E,FIND=38E3,CHG=0000)
109 P$(29)="PATCH *4 (ADD=4E28,FIND=20,CHG=18)
110 P$(30)="PATCH BASIC/CMD (ADD=5BFE,FIND=2AA440,CHG=C
D8754)"
111 P$(31)="PATCH BASIC/CMD (ADD=5C07,FIND=FF,CHG=FE)"
112 P$(32)="PATCH BASIC/CMD (ADD=5C0D,FIND=CD535F7723,C
HG=CD9B540000)"
113 P$(33)="PATCH BASIC/CMD (ADD=53CC,FIND=8754,CHG=4A1
E)"
114 P$(34)="PATCH BASIC/CMD (ADD=5487,FIND=E17EFE26,CHG
=2323E5DD)"
115 P$(35)="PATCH BASIC/CMD (ADD=548B,FIND=C24A1ED7,CHG
=E1ED5BA4)"
116 P$(36)="PATCH BASIC/CMD (ADD=548F,FIND=C24A1EE5,CHG
=40013300"
117 P$(37)="PATCH BASIC/CMD (ADD=5493,FIND=219B54CD,CHG
=0901FF00)"
118 P$(38)="PATCH BASIC/CMD (ADD=5497,FIND=3F56E1C9,CHG
=EDB0EBC9)"
119 P$(39)="PATCH BASIC/CMD (ADD=549B,FIND=35013D3C,CHG
=DD7503DD)"
120 P$(40)="PATCH BASIC/CMD (ADD=549F,FIND=26751734,CHG=
7404DD36)"
121 P$(41)="PATCH BASIC/CMD (ADD=54A3,FIND=263C3675,CHG
=0500DDCB)"
122 P$(42)="PATCH BASIC/CMD (ADD=54A7,FIND=3C267516,CHG
=01EEDD34)"
123 P$(43)="PATCH BASIC/CMD (ADD=54AB,FIND=1A050C07,CHG
=0A2003DD)"
124 P$(44)="PATCH BASIC/CMD (ADD=54AF,FIND=1C121D01,CHG
=340B24C3)"
125 P$(45)="PATCH BASIC/CMD (ADD=54B3,FIND=1011,CHG=535F
)"
126 GOTO14
127 FORN=P1TOP2
128 PRINT#1,P$(N)
129 NEXTN
130 RETURN
```

PRINT YOUR OWN CHRISTMAS CARDS on your Epson (with Graftrax-Plus) or Gemini-10X or -15X printer. By using color ribbons and one or more software programs, you can create up to four different styles of Christmas cards using five colors. I've seen a sample and it looks really nice. Each program (which will do one card style) costs $14.00 on cassette or $15.00 on disk, and versions are available for the Models I, III (and 4 in III mode), and Color Computer with Extended BASIC. For information on the card programs, write to Francis S. Kalinowski, 16 N. Alder Drive, Orlando, Florida 32807.

By the way, if all you want are the color ribbons for the printer, they are available from P.F. Skeberdis, P.O. Box 27, Fremont, Michigan 49412 (phone (616) 924-3175). Apparently Mr. Kalinowski does not sell the ribbons, so in order to print your own cards, you have to get the program from Mr. Kalinowski and the ribbons from Mr. Skeberdis. The results are worth the effort, however. The only drawback is that while the cards are quite beautiful in and of themselves, they do not, in my opinion, properly capture the true religious significance of the holiday. But that is a matter of preference and conviction, and if your convictions aren't as intense as mine, I think you may find that these cards are just the thing to answer that time-honored question, "But what's a computer GOOD for?..."

IT'S AMAZING WHAT YOU FIND WHILE YOU'RE CLEANING DEPT. - While in the process of moving, I found what surely must be a collector's item: A "Tandy Computers 1978 Catalog"! Those of you that are relative newcomers to the computer game might be surprised to learn that Tandy Computers (located at 1500 One Tandy Center in Fort Worth) once issued a catalog which offered computers from many manufacturers, including the IMSAI 8080, VECTOR 1, XITAN alpha-2, Equinox 100, PolyMorphic System 8813 and 88-2, Southwest Technical 6800, ICOM 6800, Intecolor 8031, Processor Technology Sol-20, and others. Also featured were printers (a Centronics 779 was $1145), terminals and monitors, disk drives (a Shugart SA400 35 track MiniFloppy Diskette Drive sold for $355), expansion boards and CPU boards (including the Tandy CPU-1 which used an 8080A and was S-100 compatible), Software (you could get "Tandy Disk Basic" for $149.95 or "Microsoft BASIC" for $350), books, and miscellaneous parts and hardware (a 2708 1K EPROM sold for $24.95).

What, no TRS-80? Yes - on the last two pages of the catalog (back cover and inside back cover), the Model I was featured. The "Computer with Power Supply and Built-In Keyboard" (and Level I BASIC and only 4K of ROM) was $399.95. The "12 inch Video Display" (actually an RCA portable television set with tuner and sound section removed, and a bit of interface circuitry added) sold for $199.95, and the "Realistic CTR-41 Data Recorder" (which had been just a plain old Radio Shack cassette recorder until someone at Tandy decided it would make do for a "data recorder") sold for $49.95. That came to $649 for the package, but you could save $49.90 by buying the complete package for $599.95

Or you could go whole hog and buy the "Complete Radio Shack Microcomputer System" for $2995.00. This gave you all the items mentioned above (except that the keyboard contained 16K of RAM and Level II BASIC), a TRS-80 Line Printer (a 5x7 dot matrix printer that sold separately for $1299), a "Floppy Disk System" (actually one disk drive with cable, which sold for $499 separately), an Expansion Interface (the original buggy version, with NO expansion memory installed), and a (here's just what you've always wanted for three grand, folks) game cassette (probably the original Blackjack/Backgammon game cassette)!

You want some real software, you say? Well, you had four additional programs to choose from, all on cassette: A payroll program (which would handle twelve employees, $19.95), Math I (addition, subtraction, multiplication, division skills, $19.95), Kitchen (menus, conversion tables, directory, message center, $4.95) and Personal Finance (no other description, but probably a "use the computer to balance your checkbook" type program, $14.95).

Now I know why computer users don't tend to sit around and reminisce about the "good old days"!

NEWS RELEASE: "ONLINE TELECOMPUTING, INC. is proud to announce 'Independence Day For The Personal Computer User.' Starting July 4th ONLINE, the world's first 'FREE' electronic information service, will be accessible by an 800 number to all of the continental U.S. ACCESS NUMBER: (800) 438-2438. Operational Hours: Monday - Thursday 6 PM - 1 AM, Friday at 6 PM until 6 AM Monday (all times are Eastern Standard Time). For information on listing products or services contact our office in Georgia at (404) 998-7776."

I have managed to get on this service once or twice. They do not have a message service like your local BBS, but they are still pretty popular - it's nearly impossible to get through without getting a busy signal! Set up your auto-dialer program and give them a try...

MCI MAIL NOTES - I think that MCI Mail is really trying hard to make their system more "user friendly" (at risk of using a phrase that has been banished by the Unicorn Hunters!). I have noticed now that when you CREATE an MCI Mail letter, you can type in the recipient's FULL name and MCI Mail will still "find" them on the system. For example, to reach me, you could CREATE a message to JDECKER (but you would then have to pick me out from among a list of all the folks with a last name of Decker and a first initial of J that are on the system), OR your could CREATE a message to 102-7413 (my MCI Mail address, which you might not remember), OR you could CREATE a message to JACK DECKER, which would find me and me alone (at least until another Jack Decker signs onto MCI Mail!). Similarly, when you are sending an order to The Alternate Source, you can CREATE a message to either TAS or to THE ALTERNATE SOURCE.

Also, I mentioned last issue that MCI Mail did not have a local access number in Seattle (which would have been handy for folks in western Canada that might want to use the system). Well, they do now! The number is (206) 282-3077. There's also a new access number for Hawaii (it's a local call in the Honolulu area), it's (808) 545-3050. Of course, users in the mainland U.S. can still use the MCI Mail national toll-free access number (800) 323-0905. If you aren't a registered MCI Mail user, hit ENTER once or twice when you get modem tone, then type the word REGISTER for both User I.D. and password. You can then register electronically, or you can call (800) 424-6677 to speak with an MCI Mail representative. Remember, there are NO charges unless and until you actually SEND mail, and then it only costs $1.00 to send up to 7500 characters to another MCI Mail user.

THE COLOUR OF VDT'S AND THE EYE by James P. Johnson, O.D., F.A.A.O. is reprinted from USR(80):

I'll first quote from an erudite paper given by a pair of eminent professors at the University of Waterloo (that's in Ontario folks), Drs. J. Sivak and G. Woo.

"The longitudinal chromatic aberration of the human eye is substantial and therefore the colour of the phosphor chosen for a visual display terminal (VDT) will affect refractive state and accommodative demand. For most working distances green stimuli (= max. 520 nm) are optimal."

In translation this says essentially something that we've all known for some time: that it is easy seeing green! Now you know why. Or do you?

Light breaks up in the eye to its coloured components, much like a prism or rainbow. The brain re-integrates it so that we don't notice the aberration. However, we are all slightly nearsighted (myopic) to blue light and the opposite is true of red light - we strain a bit. The eye is most sensitive to light with a wavelength of 520 nanometers. So what's a nanometer? It is awfully darn small! (1E-9 meters) 520 of them make a yellowish green colour.

The effect of VDT'S on human health and comfort has been the subject of much discussion in such learned tomes as Readers Digest and Micro 80, and has even attracted the attention of that great institution of health care, the British Columbia Department of Labour. All agree there are problems, how many we don't know yet.

We do know, for instance, that the emission of X-Rays is less from the worst black & white phosphor tube over a period of 1 year than it is from one chest X-Ray every 2 years. You may put away the lead apron now.

The effect of VDT'S on the eye can affect physical comfort. If you normally need spectacles occasionally, it is probable that you will need them more while working in front of a computer terminal. Qualities other that colour have unpleasant effects on us. The natural stimulus to accommodation (better known as "strain to see close up") is a blur - so set your contrast to the best you can get. If your tube is blurred, your eyes could constantly be changing focus, and get tired out.

Another annoying habit of some phosphors is their very fast decay time, causing a barely perceptible flashing or flicker on the screen. This can be very tiring to the human system (this must be true - it was in Micro-80). Since I am due for more schooling on this subject next month I will leave this topic for a later article.

MOVING THE MODEL 4 ROM INTO RAM - If you have a Model 4 or 4P, you can move the ROM code into RAM, where it can then be patched, changed, or otherwise diddled with. I had previously published a method to do this, but as usual, someone else has a better way. The "someone else" in this case is Lyman Epp of Omaha, Nebraska, and this is the code he uses to accomplish the task:

```
                01000 ;Name: MOVE/SRC, assembled as MOVE/CMD
                01010 ;By  : Lyman Epp
                01020 ;Date: 04/12/84
                01030
                01040 ;This program moves ROM into RAM on a Model 4 or 4P
                01050 ;computer.  When this program is done executing, it is
                01060 ;in the 'RAM' mode.  This program is six bytes shorter
                01070 ;than the shortest method that I found (using LDIR and
                01080 ;moving the entire ROM image into high memory, etc...)
                01090 ;This method takes .229 seconds (at 4 MHz) to execute,
                01100 ;and 917,522 T states to execute everything but the final
                01110 ;RET statement.  The other method (LDIR) took .151
                01120 ;seconds and 602,232 T states to execute.
                01130
5200            01140          ORG     5200H
5200 F3         01150 START    DI
5201 21FF37     01160          LD      HL,37FFH        ;START LOCATION
5204 AF         01170 LOOP     XOR     A
5205 D384       01180          OUT     (84H),A         ;PUT IN ROM MAP
5207 46         01190          LD      B,(HL)          ;GET BYTE
5208 3C         01200          INC     A
5209 D384       01210          OUT     (84H),A         ;PUT IN RAM MAP
520B 70         01220          LD      (HL),B          ;SAVE BYTE
520C 2B         01230          DEC     HL              ;AND DO IT AGAIN
520D B4         01240          OR      H               ; UNTIL YOU GET TO
520E F20452     01250          JP      P,LOOP          ;  LOCATION 0000H
5211 FB         01260          EI
5212 C9         01270          RET
5200            01280          END     START
00000 TOTAL ERRORS

LOOP    5204    START   5200
```

MODEL 4P MEMORY EXPANSION by Ian Webb, Saratog. California is reprinted from the Cabrillo Computer Society Model I,III,4 SIG Technical Notes:

I had occasion to try to expand a Model 4P to 128K. It appears as though there is NO information on the 4P available. All my magazines seem to ignore this machine. Radio Shack is tight-lipped about it and there doesn't seem to be a technical manual available.

To expand the machine to 128K you need about an hour of time and eight 4164 dynamic RAM ICs. I bought mine from DoKay, 2100 De La Cruz Boulevard, Santa Clara, California 95050. They cost $5.45 a piece for 200 nanosecond devices which is what I recommend you buy.

Open your machine, insert the memory chips in the eight available sockets AND move the AMP shorting block from E12-E13 to E11-E12.

Test your memory by using MEMDISK from TRSDOS 6.1.2 [SYSTEM (DRIVE=2,DRIVER="MEMDISK")].

If everything checks, you have it. Congratulations. If you care, 128K RAM buttons are available from the Shack for about $1.59 each.

CP/M PUBLIC DOMAIN DISKS - John Cramer has most CP/M Public Domain Library disks and will supply them for a nominal cost to computer users. He will copy them to the user's formatted disk, or will supply the disk. Average cost is $5 per single-sided disk, $6 per double-sided disk (price includes disks). John supports 8" single density and over 40 5" CP/M and CP/M86 formats, including Radio Shack Models I/III/4, Montezuma Micro, Omikron, CPM+. Libraries maintained include Osborne, SIG/M, CPMUG, TRS-80 Mod 1, IBM-PC Blue, & Piconet. Send a 37 cent self-addressed stamped envelope for theme lists of all libraries to John Cramer, Box 28606, Columbus, Ohio 43228-0606. If yo don't send the SASE, you won't get a reply! John says he can als transfer files between various CP/M formats.