NORTHERN BYTES

Double Issue: Volume 7 Number 3

NORTHERN BYTES PRICE INCREASE EFFECTIVE THIS ISSUE

Due to the lack of growth in our readership, we feel that it is necessary to raise the price for NORTHERN BYTES. This and all future issues will be $[illegible] per issue. However, we are not just doubling the price, we are also doubling the page count to 48 pages. So, in effect you are getting two of our former size issues for the price of two (which is another way of saying that you're still paying the same price per page). Doubling the page count enables us to save money on both printing and mailing costs, which might mean we'll be around just a little longer.

If you are on our credit card or "to be invoiced" mailing list and do not wish to pay the higher price, please let us know as soon as possible, and we will remove you from our mailing list. If you are truly upset about being charged the additional $2.00 for this issue, just drop us a line and we'll credit your Visa or Mastercard account for $2.00 and drop you from the mailing list. I do apologize for not giving advance notice about this increase, but it was a last-minute decision to keep us from drowning in red ink.

For those of you who have "free" issues coming, this and each future issue will count as two of our former size issues. So, you will see your "issues left" count on your mailing label reduced to half of what it would have been had we stayed with the old size. In other words - suppose your last mailing label said that you had four issues left. The issue you now have in your hand counts for two of those issues, and the next issue will count for the final two of those issues. Thus, the label on this issue will say that you have one issue left. You didn't really lose two issues - we just had to adjust the count. For those that had an odd number of issues remaining, the adjustment will be made in your favor. Those contributing articles to Northern Bytes will receive three of the new size issues free, as opposed to the six free issues we used to give at the former size.

Of course, you'll be getting NORTHERN BYTES less often now - possibly much less often. Is this the beginning of the end? Yes, I'm afraid it is. It is only a matter of time until the last NORTHERN BYTES rolls off the press. Since NORTHERN BYTES is an irregular publication, we have the luxury of letting it "fade into the sunset" by gradually publishing fewer and fewer issues. So, we probably won't ever have a formal "this is the last issue" type announcement - you'll just notice someday that you haven't seen an issue for six or eight months, and you'll know it's gone.

But why? Most of our current readers seem to love NORTHERN BYTES. I get letters almost daily that say how much our readers appreciate us, and pleading with us not to discontinue publication. To quote Barrett Strong, "Your love gives me such a thrill, but your love won't pay my bills..." [in case you're too young to remember, Barrett Strong was a singer in the early Motown era. He had one big hit, called "Money (That's What I Want)". Evidently he made as much money as he wanted from that one song, because he was never heard from on the charts again].

Well, I've never done NORTHERN BYTES for money - and it's a good thing, because it's never really covered its own expenses. You may have noticed that we carry very little advertising (not by choice), but what you may not realize is that our readership levels have never been so large that our NORTHERN BYTES mailing would not fit in the back seat of a very small car.

It isn't as though we haven't tried. We did a "mass mailing" on CompuServe that resulted in our sending out several hundred free sample copies, but we got very few regular readers from that. We even bought a classified ad in 80-Micro. Still not much luck. But what is really disappointing is that we expected that "word of mouth" advertising by our readers would inspire others to sign up. That hasn't happened in any significant numbers.

The point is, we can't continue to operate in the red (well, the pink, anyway) forever. The editor would have liked to seen the day when he could have received a real salary for editing this rag, but apparently that is not to be. We've been on shaky ground just trying to cover expenses.

The biggest reason, however, for cutting back/discontinuing NORTHERN BYTES is that the editor has put in hundreds of hours on this publication, and now feels that this time could be spent much more effectively in other pursuits. Call it an early midlife crisis if you like, but at age 34 I am hardly successful by any measure. That's partly because I'm somewhat of an unconventional person (I don't mind dressing nicely, but I draw the line at wearing a necktie - I just won't do it!), but mainly because my best efforts seem to have been put into somewhat unproductive pursuits. If, by this time, NORTHERN BYTES had grown and reached the point where some employees could be hired and the work load distributed among more people, then I would be more than happy to keep it going. This has not happened, however, and I am not a glutton for punishment.

I do have one request for our readers. Please believe me when I say that we appreciate the articles and programs that have been contributed to NORTHERN BYTES in the past. However, in all fairness I must now ask that you not send in any more articles for NORTHERN BYTES, unless they are with no strings whatsoever (in other words, you don't expect a reply or your disk back, and you won't get upset if the article is never used). I don't want to be responsible for any unsolicited mail, and there is a chance that future contributions may never be used. I'd hate to see anyone go through all the trouble of writing an article, only to have it just lie here. Charley at The Alternate Source says that he is willing to look at anything you might want to send, and will consider it both for NORTHERN BYTES and/or the TAS Public Domain Library, and for any commercial possibilities you may have overlooked, so you may be better off in the long run to send your articles and programs directly to TAS.

Also, I intend to cut back on answering mail, which (as you know if you are a regular reader) has always been somewhat of a problem for me. I get lots of mail, and rather than try to pick and choose which letters deserve an answer, I am just going to quit answering any of them (with a very few exceptions). From now on, the only sure way to get a reply from me is by telephone, at (906) 632-3248. The reason for this is that I am a very slow typist (especially when composing letters) and since I am trying to free up some time for other activities, it wouldn't make much sense for me to quit doing NORTHERN BYTES, only to spend all of my time answering mail. So, call, don't write (and don't call every night, either!). Overseas readers may consider themselves somewhat exempt from this requirement (I know how expensive transoceanic telephone calls are), but I am not guaranteeing that I will answer ANY mail anymore. Again, if you have a question or comment, you might be better off addressing it to Charley at The Alternate Source.

I do apologize for these actions. If you really want to see me reverse my decision (to gradually disengage myself from NORTHERN BYTES), then sign up a few of your friends as regular readers. If it becomes profitable to publish issues (I'd say "again" but it never really was), we might do a few more and do them more frequently. And, I do appreciate all the nice comments and compliments I have received about this publication over the past couple of years. I feel that NORTHERN BYTES readers are some of the nicest people around, and in one way I feel terrible even thinking about discontinuing this publication. But, I have other obligations that I feel take precedence over NORTHERN BYTES. Please allow me some breathing room, and we'll see whether I can crank out another issue or two before the year ends.

- Jack Decker, Editor

THE EXTERMINATOR

This is the case of "the bugs that leave tracks". In the TRSDOS 1.3 patches that were given in Volume 6, Number 8, patch number 13 was supposed to change the disk track count. Then, last issue, Andy Levinson wrote in to say that a couple of the patch lines were omitted, and gave two more patch lines. Those are in error; the "*7" and "*14" are reversed.

I finally decided to dig out my old copy of TRSDOS™ COMMENTED by Soft Sector Marketing (no longer in print, unfortunately) and see what the actual patches are. So, without further ado, here are the patches to change the track count on a copy of TRSDOS 1.3:

```
PATCH *Ø (ADD=4926,FIND=NN,CHG=MM)
PATCH *Ø (ADD=499B,FIND=NN,CHG=MM)
PATCH *6 (ADD=5CØ6,FIND=NN,CHG=MM)
PATCH *6 (ADD=5D53,FIND=NN,CHG=MM)
PATCH *7 (ADD=53FF,FIND=NN,CHG=MM)
PATCH *7 (ADD=54EF,FIND=NN,CHG=MM)
PATCH *7 (ADD=58Ø2,FIND=NN,CHG=MM)
PATCH *7 (ADD=5CD4,FIND=NN,CHG=MM)
PATCH *14 (ADD=4EB8,FIND=NN,CHG=MM)
```

Also, change the last byte in the bootstrap sector to 'MM'

In the above patches, replace NN with the original number of tracks in hexadecimal (this will be 28 for a standard 40 track disk, since 28 hexadecimal = 40 decimal). MM should be replaced with the new track count desired, again in hexadecimal (use 50 if you want 80 tracks). For more information, see the letter from Andy Levinson in the letters column of this issue (which arrived just as this issue was going to layout). I hope this ends the confusion!

Also, bugstalkers, be sure to see the letter from Art Rasmussen in this issue's letters column, which describes a bug that appeared in his letter in LAST issue's letters column...

LETTERS DEPARTMENT

Reminder: Persons sending letters intended for publication should send them on magnetic media or via Compuserve [72167,161], Delphi [TASIO], or MCI Mail [109-7407] (especially if longer than a couple of paragraphs). If you are NOT using Allwrite (or Newscript) and your word processor offers the option to save your file in ASCII format, please do so (especially if using SuperScripsit!). Your cooperation in this matter will help us to bring you a better newsletter!

Dear Jack,

..... Re: NORTHERN BYTES Volume 7, Number 2, page 22, Model 4 SuperScripsit zaps [to remove the triangles that appear in place of two spaces] - I could not make the two patches work. Try:

```
PATCH SCRIPSIT/CTL (D14,28=18:F14,28=2Ø)
PATCH SCR35/CTL (DØ1,3Ø=18:FØ1,3Ø=2Ø)
```

John Southcombe

[Oops - either we goofed or you have a different version of SuperScripsit; I'm not sure which. Thanks for the correction!]

Dear Jack,

I have a Model I and so was very upset when Infocom discontinued releasing Model I versions of their games. After several discouraging phone calls to them I gave up on ever seeing a new Infocom game on my good old Mod I. So obviously my eyes lit up when I saw the note about converting Mod III Infocom to Mod I. I immediately sat down to try the suggested process and failed to make it work at all. After talking with Bob Seaborn I found out that he was using NEWDOS/80 version 2, while I was using LDOS (and also that the location of the input buffer in the Mod I is 4318 not 4218 so zap 21 25 42 to 21 18 43, not to 21 18 42). I switched to NEWDOS and still had no success. Remembering something else in the discussion with Bob I realized there was a password problem. Bob's solution was to change the SYSTEM parameter AA=N disabling passwords. However, there is another way. The Mod III Infocom command file looks for a data file with password SMC so using the ATTRIB command to set ACC and UPD passwords = SMC solved that problem. One could also zap the password from within the command file so that it looks for FILESPEC/DAT not FILESPEC/DAT.SMC.

I then transferred the files to LDOS and again failed to have it work. I then discovered that the Infocom command file also looks at HIGH$ which in Mod III is at 4411 and in Mod I is at 4049 (so find 11 44 and change to 49 40. I found this in sector 1).

At this point all worked fine and continued to work when transferred to hard disk. What a pleasure since at that point I used Hypercross to transfer Infocom data files from CP/M format as well as from MS-DOS format and they all worked fine. Only one command file from Mod III is necessary. Just rename it to whatever the data file is. I am now happily playing Hitchhiker, Sampler, Zork I, Enchanter, and Cutthroats as CMD (all renamed from the Hitchhiker command file for Mod III) and DAT files on hard disk on my Mod I. I imagine data files from CoCo will work too, but haven't tried that yet (I am trying to locate the Sampler for CoCo as listed in the latest Radio Shack Express Order Catalog but there doesn't seem to be a part number for it. Strange! It's cheaper to experiment on the $7.95 size)

That note in Northern Bytes has made me quite happy and I am very glad I have been subscribing to it. Keep up the good work. I, for one, really appreciate your efforts for Mod I users.

Mike Riskin

[Glad to be of help, Mike, and sorry about the bug. And, thanks for the additional information!]

Dear Jack,

I am an avid consumer of NORTHERN BYTES which means I read and re-read every single issue several times. I only wish I had discovered you earlier than the Volume 5 issues. Thanks for your insights, wisdom, and sticking with the task of communicating to us ever shrinking body of "believers" in TRS-80 computers.

I appreciate your comments, patches, and helpful hints in the use of ALLWRITE! program. I wish PROSOFT would put out a newsletter similiar to LESCRIPT, but I guess cost is prohibitive. I have used ALLWRITE! exclusively on a daily basis for more than a year now, having stomped to death several copies of SUPERSCRIPSIT due to its interminable habit of losing disk sectors and then refusing to load what it could salvage. I have yet to lose my first sector with ALLWRITE! and am very thankful to Chuck Tesler for his expertise.

Secondly, thanks for SD/CMD program on PD#14. My mouth watered when I bought the Model III version and now it is satisfied with the Model 4 version. I am looking forward to being able to imbed the date in my letters without having to worry about what it is! Thanks for your knowledge and for sharing it.

Thirdly, since you seem to use ALLWRITE! also, I want to share with you a procedure I use which speeds up the movement considerably between editing and formatting modules by using MEMDISK as the primary storage disk. If you think it worthwhile, feel free to share this procedure in NORTHERN BYTES. It is not particularly ingenious, but it is very useful.

NOTE THAT SOME LINES THAT FOLLOW WILL NEED TO HAVE THE WORD "(printer)" REPLACED WITH THE DISK FILE NAME OF WHATEVER PRINTER IS BEING USED. If a second printer is in use, do NOT rename those files as there is not room in MEMDISK for another set of printer files.

1. Execute the following JCL (or type each line individually from DOS).

```
Rename al/cmd to al2/cmd
Rename al/def to al2/def
Rename alf/cmd to alf2/cmd
Rename (printer)/def to (printer)2/def
Rename (printer)/tab to (printer)2/def
```

2. Using ALLWRITE! write the following JCL and save it under "START/JCL"

```
ad  ;cm include ONLY if you have the SD/CMD
    program from PD disk #14.  If you don't have it,
    order it from TAS now.  You need it!
system (sysres=1)
system (sysres=2)
system (sysres=3)
system (sysres=4)
system (sysres=1Ø)
system (sysres=12)
system (drive=2,driver="memdisk")
d
d
y
copy al2/cmd:Ø to al/cmd:2
copy al2/def:Ø to al/def:2
copy alf2/cmd:Ø to alf/cmd:2
copy (printer)2/def:Ø to (printer)/def:2
copy (printer)2/tab:Ø to (printer)/tab:2
```

al
//stop

3. Save this JCL file under file name START/JCL
4. Go to DOS and type "AUTO DO START" <ENTER>
5. If you have the SD/CMD program from PD Disk #14, type from DOS, SYSTEM (DATE=NO) [also be sure to use the SETDATE/FIX file from PD #14 - type DO SETDATE/FIX to install the patches that keep TRSDOS from erasing the date from memory during a reboot -Editor]
6. Reboot the disk.

Now each time you boot the ALLWRITE! working disk, after the date is entered you can go for your first cup of coffee for the morning while the JCL file is working. You will find the speed up between Editor and Formatter significant. Unfortunately, there is not room in MEMDISK to copy SYS0/SYS which would allow one to make MEMDISK drive 0 and make the system truly efficient.

Jack, if you think this is a worthwhile procedure to share, please use it on a Public Domain disk as well.....

..... One little addition to the forum on disassembly of SUPERSCRIPSIT begun in NORTHERN BYTES Volume 7, Number 2: According to the manual, the S/CTL module is the serial driver for printers hooked to the RS-232C port rather than the Centronics parallel port. It can be eliminated from the disk to save space if not needed.

LASTLY, have you considered putting NORTHERN BYTES on a disk format so we could each load it into our word processor? The size of type is hard on poor eyesight! Just an off-the-wall thought.

Thanks for your time. Please don't give up on NORTHERN BYTES. It's about all that is left out here for us would be hackers!

Steve Torkko - Faith Lutheran Church,
2741 Sherman Avenue, North Bend, Oregon 97459

[Thanks for passing along the information, Steve. We do put some text from NORTHERN BYTES on the Public Domain Library disks, but mostly it is program documentation. Most of my text files are in ALLWRITE! format and have lots of formatting codes included. Not only that, but I may print out an article and then wind up actually using it six months or a year later - and just try and find the original text file then!

TAS is seriously thinking about starting a BBS (using the SYSLINK program) and if they do and there is any demand for it, I may consider uploading the text of some articles, but I may just leave the ALLWRITE formatting codes in and let others remove them if they don't use ALLWRITE (or change them as needed if they don't use a DMP-2100P printer). However, this would take a lot of extra time and effort, which is something I don't have much more of to put into NORTHERN BYTES.]

Dear Jack,

I would like to echo the expressions of others and compliment you on the new look of Northern Bytes. It is much easier to read and this is a very important factor for the over 40 crowd.

Next, you have published several articles on self-booting system disks for the Model 4P for NEWDOS/80 and for DOSPLUS but I haven't seen any for MULTIDOS. I was talking to my good friend Bob White in Honolulu, known to some as the 'GURU OF THE WEST', and he mentioned he had discovered where to zap MULTIDOS 1.6 so it will boot on the 4P.

On Track 17, Sector 0, change Relative Byte CD from 28 to 42. Then make sure you have at least 10 Grans free on your system disk, and copy MODELA/III program to it from a TRSDOS disk and you will have a Self-Booting 4P system disk.

This zap will only work on the 1.6 version of MULTIDOS and not on version 1.7. Bob said he will work on the zaps for the 1.7 version and pass them along.

Bill Baker

[MULTIDOS is a great DOS, but does some things in a non-standard way that can give you a bit of grief if you're not careful. In particular, it may use non-standard filenames for BOOT/SYS and/or DIR/SYS, or place the directory entries for those files in non-standard places. I discovered the hard way that a single-density disk produced by MULTIDOS cannot be WRDIRPed by NEWDOS/80 because of this. I don't know if your zap corrects a similar non-standard condition, but thanks for sharing it.]

Dear Jack,

... The "big" computer news here [in Australia] is that the Model 4 is "on special" until sold out ($A1499 against $A1999). As well, the "big names" in Radio Shack software (Profile 4+, SuperScripsit, etc.) are also "on special" at 1/4 to 1/3 of their normal price. No 4P's are left. We believe that there are no plans to introduce the 4D to Australia (Tandy Australia has not yet replied to two of our letters regarding the future of the Model 4 in Australia). We have heard that production of the 4D has ceased in the U.S.A. Is this true? We are not worried about hardware support - Tandy has a good reputation in this area - software support is something else! As far as I know TRSDOS 6.x is incompatible with any other computer currently sold in Australia.

There is strong evidence of continuing demand for a quality 8 bit computer that is marketed with an extensive range of good software (here in Australia). As a group, I would say that sales of the 4/4P have been taking customers away from the 1000/1000HD where there is more profit. Oh well, I suppose that's life (business!).

Bert Smith

[I could speculate on the fate of the Model 4D but it would just be repeating rumors, something I don't want to do. If all else fails, consider the used computer market as a source for acquiring additional Model 4's and 4P's. As those people who always have to have the latest thing (or those who are terribly frightened of being stuck with an "obsolete" computer) sell their systems to "move up" (?) to MS-DOS, Model 4's and 4P's will undoubtedly begin to appear on the "pre-owned" market.

As for software support, this may be "the best of times and the worst of times" for TRS-80 Model I/III/4 owners. True, some software vendors have pulled their TRS-80 product line completely, but others have responded by lowering prices and in some cases, placing programs that were formerly sold commercially into the public domain. I have given up trying to predict what will happen in the future - I'm usually wrong anyway - but with hundreds of thousands of Model I/III/4 TRS-80's still around and in daily use (including my old Model I), I'm sure that there will be companies that will continue to offer software support for some time yet. Unfortunately, Tandy may not be one of them, from all appearances.

On the other hand, if software producers are getting as much response from Model I/III/4 owners and users as we have here at NORTHERN BYTES, then I can understand why they would want to move on to greener pastures. You can't produce software (or a newsletter!) for a market of only a few hundred people!]

Dear Jack,

... This old fashioned non-electronic mail is temporary. I had to terminate MCI... I need a service which will make (paper) deliveries, and answer my letters. I used MCI mail to send orders and letters to the U.S.A. for friends and club members. When I allowed MCI to use my credit card, they stopped sending monthly accounts; my letters asking fora accounts, probably a dozen, were ignored. I have three cases of MCI (paper) not reaching the addressee, two in Europe, one in the U.S.A. (LSI with an order for LDOS which you kindly looked into for me).

I could not recover my costs without an accounting. So, I kept my use of MCI to a minimum, but received a charge of $50 and more each month. This was greater than the cost of Midas. Tony Domigan says this is not possible. If there were increased rates, I was never notified...

I received, yesterday from MCI, a brief invoice summary for October 1985 (only) for $16, which amount had been charged in November, and again December, the October account was $76.48! Worse, it was addressed to

Clifford Richards
Boronia Road #3
Australia

But it found me!

... The continuing arguments about the merits of NEWDOS versus LDOS/TRSDOS make good reading in Northern Bytes. NEWDOS was abandoned by its maker and Powersoft did not support double sided NEWDOS drives in Super Utility. I was forced to TRSDOS 6 and LDOS. After living through the unbelievable early

TRSDOS, its very name was a dirty word and I vowed it would never again disgrace my VDU.

I learned to respect TRSDOS 6 and LDOS 5.1.4. and to enjoy using them. There is no rule that you must be loyal only to one brand of DOS. My advice is, use whatever software which turns you on.

Don Singer (letter in Volume 7, Number 2) is correct about Profile 4 + not running with KSMPLUS on TRSDOS 6.2. "Inquire, Update, Add," & number 1 on the runtime menu uses top memory. Profile 4 + works splendidly with good old "plain vanilla" KSM. Profile III + (HD) on LDOS 5.1.4 is not so affected.

Better, a brilliant program, SUPERLOG 4 makes and saves programmed keys. 19K is available for use and can be loaded from DOS without using any high memory. Stock paragraphs for letters can be inserted into Lazy Writer, addresses and notes can be kept. Useful, too, patches can be applied from SL4 (resident in Memory), printing commands and Help Files. The bad news is that Lazy Writer seems to be the only WP program to accept data.

I bought a program called MICROZAP from SOTA in Vancouver, Canada, which cost me over $100 Australian. It is still advertised in 80 Micro, they recently gave it (rightly) a poor review. Sota claimed it would Zap NEWDOS Double sided disks. It would not. I returned the disk in November last year so that they could send a copy of the program which would perform as advertised. They have not responded, nor to a further MCI letter. They must still be in business as they advertised it April 80-Micro.

Do you know anything about them? That is the final last program I will buy without using it first, do you blame me?

Samuel Wells asked about the efficiency of the 1 meg memory board for the Model III. Happy to tell him all I have found. I have the Alpha Technology (A1A) board in an old type Model 4, table top. The Model 4 runs faster than the III, so he would benefit more than I, from the increased speed of execution.

Software to access the added memory is by different authors. I have TRSDOS 6 2, from Security and Software of Houston, Texas. Versions for NEWDOS, LDOS and CP/M are available. TRSDOS and LDOS give 5 drives. With NEWDOS/80, you lose a drive. I use the Ramdrive as Drive 0, under

```
Drive :0  Ram Disk  03/29/86   Free Space =  750.00K/  896.00K  Files = 221/256
Drive :1  SUPERLOG  11/14/85   Free Space =   45.00K/  360.00K  Files = 205/256
Drive :2  PRONTO    12/04/85   Free Space =   72.00K/  360.00K  Files = 186/256
Drive :3  DATADISK  03/19/86   Free Space =  555.00K/  720.00K  Files = 206/256
Drive :4  DATADISK  03/29/86   Free Space =  273.00K/  720.00K  Files = 213/256
```

Ramdrive moves the drives up one. In the above, Superlog disk was the booting disk. I have the two banks of memory + 896K and can run both Superlog 4 and PRONTO or Double Duty in the old memory banks. Software has been written so that DOS will access all banks, the programs to use it this way will follow. Lescript already has a version which allows 1 Meg of copy (which would take Sam past his 120 page limitation).

Loading takes less than a minute (the initial load takes more time because it writes a file to disk). Configurations can be made to different drives. If I used a drive other than :0, I would not need the system files, but there is no point in this. If I boot another DOS or Super Utility, Supermem finds Ramdisk was active and restores the files, that is if you did not turn off the power.

I have printed here, a "DIR 0 (S,I)". This shows that, with systems installed, there remains more than a double sided 80 track drive. The missing systems are System 13, System 0 (used mainly in booting) and the Debug systems. One of the tricks, take an example from Webster, is to leave DICT3/EW (which updates) on line in floppy or hard disk. Files updated of course, show a "+".

```
Drive :0  Ram Disk   56 Cyl, DDEN, Free =  754.00K /  896.00K,  Date 29-Mar-86
Filespec    MOD  Attr  Prot   LRL  #Recs  EOF  File Size  Ext    Mod Date
-------------------------------------------------------------------------------
BACKUP/CMD       IP    EXEC   256    26  107     8.00K    1    01-May-85
BOOT/SYS         SIP   EXEC   256     4  255     2.00K    1
DECKER       +         FULL   256     4   76     2.00K    1    18-Apr-86
DIR/CMD                FULL   256    11  244     4.00K    1    05-Sep-85
DIR/SYS          SIP   READ   256    64  255    16.00K    1
EDIT/CMD                FULL   256    28  205     8.00K    1    24-Oct-85
FORMAT/CMD       IP    EXEC   256    19   14     6.00K    1    01-May-85
FT                     FULL   256    26  116     8.00K    1    07-Oct-85
HEADING                FULL   256     2  128     2.00K    1    17-Apr-86
HEADING/CO             FULL   256     2  168     2.00K    1    29-Mar-86
IFC/CMD                FULL   256    32  164     8.00K    1    02-May-84
IFCLIST/CMD            FULL   256     5   86     2.00K    1    18-Apr-84
L/CMD                  FULL   256    29   93     8.00K    1    18-Mar-86
LPNT/CMD               FULL   256    27  117     8.00K    1    12-Sep-85
P1/CMD                 FULL   256     2   16     2.00K    1    18-Dec-85
SYS1/SYS         SIP   NO     256     6   65     2.00K    1    14-Sep-85
SYS10/SYS        SIP   NO     256     2    0     2.00K    1    15-Mar-84
SYS11/SYS        SIP   NO     256     4  253     2.00K    1    15-Mar-84
SYS12/SYS        SIP   NO     256     4  235     2.00K    1    15-Mar-84
SYS2/SYS         SIP   NO     256     6   40     2.00K    1    14-Sep-85
SYS3/SYS         SIP   NO     256     4   44     2.00K    1    15-Mar-84
SYS4/SYS         SIP   NO     256     6   26     2.00K    1    15-Mar-84
SYS6/SYS         SIP   NO     256    50  242    14.00K    1    14-Sep-85
SYS7/SYS         SIP   NO     256    26  178     8.00K    1    15-Mar-84
SYS8/SYS         SIP   NO     256    37   14    10.00K    1    15-Mar-84
U/CLU                  FULL   256    15  162     4.00K    1    03-Sep-85
X1                     FULL   256     1  255     2.00K    1    27-Nov-85
X5                     FULL   256     1  255     2.00K    1    06-May-85
X9                     FULL   256     1  255     2.00K    1    28-Aug-85
===============================================================================
          29 Files out of  29 selected, Space =   142.00K
```

A problem with NEWDOS on Ramdrive is that it seems impossible to increase directory space from 10 sectors. With all the NEWDOS fans out there, will it be long before someone writes the zaps which will enable the fifth drive?

Superscripsit would run faster and smoother... Programs I use, Multiplan and TK!Solver, used to run at snail's pace and I am not patient by my nature. Every Model 4 program improves to my liking, including Profile 4 +, it seems to give them new life.

Cliff Richards

[Cliff, sorry about hacking up your letter - you made several points and I tried to keep as many of them in as possible.

Regarding MCI, I have noticed that their efforts toward keeping the small volume user have become almost nil. I suspect that they are trying to position themselves as an extension of the electronic mail systems now used by many large companies. AT&T is starting a new electronic mail service (called AT&T Mail, naturally) and what little I have read about their pricing structure seems to indicate that they are going to be almost a clone of MCI Mail (at perhaps a slightly lower cost). This could make a big impact on the bottom line of MCI Mail. It would be interesting to see these two communications giants get into an electronic mail "price war"!

I don't know much of anything about SOTA Computing Systems, Ltd. except that I have heard that their "Fast 80 BBS" leaves something to be desired. If anyone has had any experience with this firm or their products, please let us know about it.

I don't think many software manufacturers would mind if you "try before you buy", so long as you actually DO buy if you like and intend to use the program (if any software manufacturer writes in to disagree with this opinion, I warn you now that you'll probably wish you hadn't. With all the junk that's been marketed, I think it's high time for a little up front disclosure of what a prospective purchaser is going to get. Note that I'm NOT condoning "piracy", but I'm also not condoning the software vendors that produce shoddy products and then try to hide behind the copyright laws when a customer asks for a refund).

Anyway, thanks for all the information you've passed along!]

Baa Hum-BUG BUG BUG

That's exactly what is in my article on creating a double-sided self-booting NEWDOS/80 disk for the 4P (See Volume 7 Number 2 page 5). Step 9 in the procedure says: "Move to sector 171.", it should say, "Move to sector 172." Sorry about the inconvenience.

Recently there have been several questions regarding SuperScripsit's file structure. About a year ago I ran across the following article written by Tom Price. I believe that Tom was the author, or at least one of the authors, of Model 4 Scripsit.

SuperSCRIPSIT Document Files

This article will explain in detail the structure of SuperSCRIPSIT document files, which may be of assistance in repairing damaged files with the aid of a 'zap' program or file editor.

Each SuperSCRIPSIT file has four distinct areas:
Record 0 - Document header and other vital information.
Records 1 thru 4 - Disk block index.
Record 5 - List of new page markers.
Records 6 thru to the end - Disk blocks containing the text.

Before getting into the details, it is necessary to define the meaning of a "disk block". It is 1K in size, consisting of four 256-byte records. Each block contains a 7-byte header (explained later), up to 985 bytes of text, paragraph and control information, and 32 bytes of overflow space to accommodate minor changes without starting a new block. Each of the four possible header/footer pages will occupy its own block, if present. A document may not contain more than 174 blocks. Blocks are numbered from 0 to 173, with Block 0 starting at document record 6.

Document Record 0 - Header

Byte	Description
00	ID - Always E0, identifies a SS document
01 - 18	24 bytes for the document name
19	Maximum lines per page in half-line increments
1A	Pitch - PS=00
1B	Line spacing in half-line increments
1C - 23	8 bytes for printer driver filename
24 - 25	Page number to start footers
26 - 27	Page number to start headers
28	Odd footer length in half-lines
29	Odd header length in half-lines
2A	Even footer length in half-lines
2B	Even header length in half-lines
2C	Horizontal cursor position on video display at document close
2D	Vertical cursor position on video display at document close
2E	Column position of cursor at document close
2F - 30	Document line number of cursor at document close
31 - 45	Tab line 0 (21 bytes)
46 - 5A	Tab line 1 (21 bytes) [NOTE] - The 21 byte tab line contain 168 bits, each bit representing a column position. If a bit is set, there is a tab stop at that position.
5B - 71	Bit map of disk block allocation (23 bytes) [NOTE] - The first 174 bits of this map represent disk blocks 0-173. If a bit is set, that block is allocated. If the bit is reset, the block is available.
72	Disk block number of odd footer (if any)
73	Disk block number of odd header (if any)
74	Disk block number of even footer (if any)
75	Disk block number of even header (if any)
76	Disk block number of tab line table (FF=none) [NOTE] - This block has space for 48, 21 byte tab lines, which added to the two available in this record, make up the maximum of 50 tab lines per document.
77	Number of tab lines currently assigned to the document.
78 - 97	Name of Author (32 bytes)
98 - B7	Name of Operator (32 bytes)
B8 - D7	Comments (32 bytes)
D8 - FF	Not currently used.

Disk Block Index - Records 1 thru 4

Byte 00 of the index contains the number of active text blocks in the entire document, not including any blocks assigned to headers, footers or tab lines. Starting with Byte 01 of Record 1, there is room for 174, five-byte groups, each group representing a disk block containing text. The groups are arranged in the actual order of the document's text as printed. The meaning of each byte in a group is as follows.

Byte	Description
00	Disk block number (range 00 to AD). If the value is FF, it denotes the end of the index and all following bytes have no meaning. The actual record number in the file can be found by multiplying the block number by 4 and adding 6.
01 - 02	Actual length of valid text in the block
03 - 04	Number of lines of text in the block. The upper nibble of byte 04 is used to contain block control information as follows: Bit 7 - set if the first line of the block is a whole line, not part of a line from a previous block. Bit 6 - set if the block contains an open marker '[' Bit 5 - set if the block contains a close marker ']' Bit 4 - set if the block has been changed (edited)

List of New Page Markers - Document Record 5

This record contains the location of any hard page breaks (^) in the document. Byte 00 is the number of markers in the document, followed by 127 pairs of bytes (byte FF is not used). Each non-zero byte pair contains the line number, relative to the start of the document (line 0), where a page break is located.

Disk Block Structure

Each disk block starts with a 7 byte header. Bytes 00-01 contain the number of text bytes following the header. This value must be identical to bytes 01-02 of the index group for this block. Bytes 02-06 of the header comprise the default paragraph format for the block as follows.

Byte	Description
02	Column containing the left margin
03	Column containing the right margin
04	Column containing the indent tab
05	Number of the tab line in use.
06	Control byte - following bits are used: Bit 4 - Set if the paragraph is frozen Bit 3 - Set if the paragraph is centered Bits 2 thru 0 - used to indicate line spacing in half-lines.

After the header comes the actual test. If the block starts with a complete paragraph (line), there will be a 5 byte paragraph group identical in format to the default group in the block header followed by an EF control byte indicating the end of the paragraph info. Then comes the actual text of the paragraph, terminated by an FD, denoting a hard carriage return. This will be followed by a 5-byte paragraph info group for the next paragraph, followed by an EF, and so forth until the number of bytes shown in Bytes 00-01 of the block has been reached. Everything after this point is MEANINGLESS; the text continues on the next block shown in the disk block index. In the text, the following control codes may be encountered:

Code	Meaning
E5	Null - deleted text
EC	Soft page marker
ED	Hard page marker
EF	End of paragraph control info
F0	Start block marker (shows as '[' on screen)
F1	End block marker (shows as ']' on screen)
F2	Normal tab
F3	Align tab
F5	'Code' for printer control (underline, bold, etc.)
F6	Filler bytes for insert mode
F7	Space compression for two succeeding spaces (delta)
F8	Soft carriage return replacing a space
F9	Soft carriage return replacing a double space
FA	Hard hyphen
FB	Hard space (for hyphenation)
FC	Hard carriage return during inserting
FD	Hard carriage return
FF	End of file/text

Armed with the above information, a user with a clobbered file may be able to zap it to the point where it will load properly and allow final repair with the normal SuperSCRIPSIT editing functions. The information in the disk block index MUST agree with what is actually contained in the disk blocks themselves. For example, when the FF byte is encountered in the index, it is a sign that the previous block should contain the FF end of text code somewhere among the valid bytes of that block.

Tom Price
02/17/84

[Letter resumes here]

I know this information has helped me on several occasions. What I would really like is to find out why SuperScripsit crashes. I have loaded SuperScripsit files only to find a section with garbage (control) characters which defy deletion. Attempting to delete

usually results in locking up the program or forcing a reboot, or I find my cursor out in never never land with a tab line that is impossible to correct. I think the problem is that the byte count in the block index does not match the byte count at the beginning of the block, but how did it get that way???

Another problem that I have had is that SuperScripsit has once or twice written a file right on top of the boot track, and of course you know what that does to a disk. I have checked all the SuperScripsit files against the original masters (I'm using version 1.3 on TRSDOS 1.3) and checked the DOS files against an original master and everything seems fine, anyone out there with any ideas???

I guess that's why I use Scripsit, with PowerSoft's PSCRIPT patch, instead of SuperScripsit. It has never crashed.

Art Rasmussen
612 West Hillcrest, Keene, Texas 76059

[This information should be a SuperScripsit hacker's delight. Thanks for sending it, Art!]

Dear Jack:

With mud in my face, I am reporting a dual error in patches to TRSDOS 1.3 that I sent you by my letter of 2/9/86. The patches were publilshed in Northern Bytes Volume 7, Number 2. I received the issue just a few days ago but the error was caught by reader William Baker of Independence, Missouri.

I am referring to my two patches necessary to augment the patches in Volume 6, Number 8, for changing the disk track count for TRSDOS 1.3. I had reversed the "asterisk" values in the two patches and also had forgotten that one of the two patches overlaid a corrective patch that I had previously written for TRSDOS 1.3

Corrected, the two patches should be:

```
PATCH *7 (ADD=54EF,FIND=27,CHG=xx)
PATCH *14 (ADD=4EB8,FIND=28,CHG=xx)
```

where "xx" should be replaced by "28", "2A", or "50" for 40, 42, and 80 tracks respectively.

For the curious, the patch to overlay 7 is in the middle of the TRSDOS backup utility. Before doing a backup, TRSDOS looks to ensure that all tracks that are allocated on the source disk exist on the destination disk. Without the corrective patch above, TRSDOS 1.3 only compares the first 39 tracks. That is no problem unless a higher track is allocated on the source disk but is locked out on the destination disk.

I had stated that the *7 patch had a "FIND=28". That is the value that should be there. As distributed, TRSDOS 1.3 instead uses "27" which is wrong. To correct this, I suggest that all TRSDOS 1.3 users apply the *7 patch above. If one has standard 40 track drives, the patch is:

```
PATCH *7 (ADD=54EF,FIND=27,CHG=28)
```

The second patch, to overlay 14, is part of the rarely used but documented $RAMDIR system call. The above patch becomes necessary when the disk track count is changed so that $RAMDIR returns the correct number for free space information (note that this is not the routine used by the TRSDOS 1.3 FREE command).

The $RAMDIR call is quite useful and indeed was carried over to TRSDOS 6. The call dumps key information about active files to a RAM buffer. I suspect the call is often ignored due to a major bug: $RAMDIR miscounts file name lengths when some files have extensions and some do not. The following patch corrects this bug:

```
PATCH *14 (ADD=4F00,FIND=0E,CHG=0F)
PATCH *14 (ADD=4F11,FIND=010D000901,CHG=0E0D090E03)
PATCH *14 (ADD=4F16,FIND=0300BE2810,CHG=BE2812F53E)
PATCH *14 (ADD=4F1B,FIND=F53E2F1213,CHG=2F1213DD2B)
```

This is a bona fide bug and the patch should be considered mandatory.

The TRSDOS manual (mine at least) does not mention that $RAMDIR skips "system" files (I refer to files with a system attribute in a TRSDOS 1.3 directory listing--not the "true" TRSDOS 1.3 system overlays). The following is an optional patch to $RAMDIR that forces $RAMDIR to also return information about the system files:

```
PATCH *14 (ADD=4EEB,FIND=CB76,CHG=AF00)
```

Two last comments about Volume 7, Number 2: In response to Vern Hester (as if he did not know), the Model 4 Technical Reference Manual states that the speed for the disk drive is 300 RPM plus/minus 1.5%. I am not a hardware person (I was forced to buy the book to get the TRSDOS 6 software calls). As a layperson, I did not see anything that said that drives should be adjusted off 300 RPM--just that they could be.

Finally, for Art Rasmussen (Volume 7, Number 2, page 5), the "secret" about bit 5 of byte CD of the GAT sector was mentioned in the newsletter of SAGATUG, my local TRS-80 club, and mentioned with credit to us (and others) in Northern Bytes Volume 6, Number 3, at page 13.

Andy Levinson
11575 Sunshine Terrace, Studio City, California 91604-3835

[Andy, your letter arrived just in the nick of time to be published in this issue. I had already written my remarks in THE EXTERMINATOR about this problem, but decided to reprint your letter anyway because you go into the problem in a lot more detail than I did. Thanks for all of the information!]

REPLACE YOUR Z-80 WITH A SUPER CHIP

David G. Huffman (1345 Williams Street, Des Moines, Iowa 50317) recently sent me some information on a project that he has been working on. Apparently the Z-80 CPU used in the TRS-80 Models I/III/4 and in the LNW computer (which is what David uses) can be replaced with a Hitachi HD64180 CMOS Microprocessor. The HD64180 is NOT pin-compatible with the Z-80, but offers much more, thus hardware types may wish to tackle a conversion project.

Some of the features of the HD64180 include: Fast operating frequency (up to 10 MHz). On-chip Memory Management Unit supports 512K byte memory and 64K I/O address space. Two-channel Direct Memory Access Controller with memory-memory, memory-I/O, and memory-memory mapped I/O transfer capability. WAIT input and Wait State Generator for slow memory and I/O devices. Programmable dynamic RAM refresh addressing and timing. Two-channel, full duplex Asynchronous Serial Communication Interface with programmable Baud Rate Generator and MODEM control signals. Clocked serial I/O port with high speed operation (up to 300K bps at 6 MHz). Two-channel 16-bit Programmable Reload Timer for timing and output waveform generation. Programmable interrupt controller manages 12 interrupt sources (8 internal, 4 external) with three interrupt modes. "Dual Bus" interface compatible with all standard memory and peripheral LSI.

Enhanced standard 8-bit software architecture features include: Upward compatible with existing Z-80, 8085, and 8080 system and application software. Optimized for higher performance of standard 8-bit operating systems (e.g. CP/M80). Enhanced instruction set including high speed multiply. I/O address relocation for board level product compatibility. SLEEP, IOSTOP, and SYSTEM STOP low power modes. Many existing Z-80/8080 instructions require fewer clock cycles to execute, thus, even with no change in CPU clock speed, programs may run 10% to 20% faster.

David reports that there is only one problem in replacing a Z-80 with a HD64180, and that is that the HD64180 does not recognize the "undocumented" Z-80 instruction set (those machine language instructions that, although they do not appear in the official Z-80 instruction set, nevertheless are recognized by virtually every Z-80 ever made). David has found that a surprising amount of TRS-80 software makes use of the undocumented instructions. In his LNW conversion, he has figured out a way to divert those instructions to a section of code that translates the "undocumented" instruction to standard Z-80 code, executes it and returns to the original program.

It is my understanding that David is working on a Model 4 conversion using the HD64180. If you are a hardware hacker, you may wish to obtain data and specification sheets on this chip from your nearest Hitachi representative (or a Hitachi regional office in Burlington, Massachusetts; Dallas, Texas; Itasca, Illinois, San Jose, California, or Woodland Hills, California). In Canada contact an office of Longman Sales, Inc. (Mississauga, Ontario; Ottawa, Ontario; or Kirkland, Quebec).

Will this chip spawn the next generation of modifications for the Models I/III/4? It certainly seems to offer a lot of potential!

PROFILE 4+ ENHANCEMENT PROGRAMS
by Don Singer

At work I use Radio Shack's PROFILE™ database manager on a Model II. Over the years we have added every enhancement offered by Radio Shack and the Small Computer Company, resulting in a very sophisticated and powerful, but expensive (about $800) package. PROFILE III PLUS offers many of the same features for my 4P, but again requires several expensive enhancements. Also, it runs only under TRSDOS 1.3, which I do not want to use. PROFILE 4+ offers most of the features I want at a more reasonable price, and runs under TRSDOS 6 which I like, but some capabilities are still lacking. Partly at the urging of Dwain Sutton, who uses PROFILE 4+ extensively in his ranching business, I wrote three BASIC utilities to perform some of these functions. Dwain wrote a menu program to run my programs.

Each PROFILE database (or "file") consists of up to four ASCII files called "segments", having the same filename with different extensions (/KEY, /DAT, /DA2, /DA3). The records in each segment can be 1 to 256 characters long, and segments from one file can be on different drives. Data can be sorted on up to five fields and selected using Boolean operators on up to 12 fields from any segment. Sorts can be stored in up to six Index files (ending in /IX1 or /IY1-5), and indexes can be built from other indexes. PROFILE has many other powerful features including sophisticated entry screens, limited math formulas, multiple label and report formats (also somewhat limited), the ability to re-structure files after data is entered, and the ability to build a small file and expand it as data is added.

To use these programs, you should understand PROFILE's file structure, and know how to build PROFILE Indexes to achieve the desired results. You should also be able to manipulate and RENAME files in TRSDOS and understand how to enter BASIC with the proper number of files and RUN a program. You should also be able to deal with an aborted BASIC program. I have made liberal use of error trapping, but have not trapped every possible error. (The programs have detailed on-screen instructions concerning the number of files and other information needed to use them. The instructions can be bypassed by the user. Also, programs may be run individually or called from the menu program, RM/BAS.)

One of PROFILE's problems is that "deleted" records are filled with blanks and kept in the file. If new data is not added, the blank records accumulate, taking up valuable disk space, and slowing sorting and selecting. SUBFILE (SUBFIL/BAS) removes these blank records and writes all records with data to a destination file specified by the user. The destination file segments can then be RENAMED to the original, source filename and used in PROFILE. SUBFIL can also write a destination file containing non-blank records selected with a PROFILE Index, (e.g. to archive inactive records before deleting them from the active database).

REPLACE/BAS, lets you fill fields of your choice with a "literal" (e.g. "0.00" for Starting Balance, "1986" for Current Year or a series of spaces to "blank" a field). The literal can be left- or right-justified within the field (padded with spaces). Again, this can be done for all records in a file, or for specific records chosen with a PROFILE Index.

TRANSFER/BAS, is a "poor man's relational database". It transfers data in specified fields between two PROFILE data files. The field and segment numbers in the source and data fields may be different. (For example, data from field #4 in segment #1 of the source file may be moved to field #21, in destination file segment #3.) Data may be moved from EACH record in the Source File to the "same-numbered" record in the Destination File, or SOURCE and DESTINATION records may be selected using a PROFILE Index from each file.

These programs have the potential to scramble data if you misunderstand or don't follow the directions, use the wrong Index(es), or build Index(es) incorrectly for the desired result, so it's important to have good backups of the data. I suggest checking the data carefully after using these programs to verify that it is OK. Be sure to check the results before KILLing your old files, deleting archived records, or using your new data files for large printouts or other applications. This is not meant to discourage use of these programs, but to encourage CAREFUL use. So far the programs have worked well, and no problems have been found.

RENAME/BAS allows you to create a new database by modifying an existing one without having to RENAME each file individually in TRSDOS.

RM/BAS is a menu program which runs the above programs from a menu.

If these programs are converted to Model III BASIC, they should work with PROFILE III data files also, but as far as I know this has not been tried.

SUBFIL/BAS

```
10 REM          * SUBFIL/BAS - Placed in the public domain by
DAS, 09/15/85  *
20 REM          * Archives records from PROFILE 4 data file to a
destination *
30 REM          * file, while REMOVING records "deleted" under
PROFILE 4(tm) *
40 REM
50 CLS: PRINT@ (6,20), CHR$(16);" S U B F I L E  F O R  P R O F I
L E 4 "
60 PRINT: PRINT TAB(29) "     By Don Singer
70 PRINT TAB(31), "3726 Skyline Drive"
80 PRINT TAB(29), "Scottsbluff, Ne 69361"
90 FOR I=1 TO 3000:NEXT
100 REM
110 REM          * INITIALIZE AND INSTRUCTIONS *
120 REM
130 CLS: CLEAR 5000:DEFINT A-Z: DIV$=STRING$(80,"="): CN$="To
continue, press <ENTER>..."
140 EX$(1)="/KEY": EX$(2)="/DAT": EX$(3)="/DA2": EX$(4)="/DA3"
150 PRINT@ (2,10), "Do you want instructions (Y/N)?": GOSUB 5030:
IF YN$="N" GOTO 430
160 CLS: PRINT "This program can be used in two ways:"
170 PRINT: PRINT TAB(5) "---> (A) Remove the blank records
'Deleted' by PROFILE, and write"
180 PRINT TAB(14) "all records with data to a destination file"
190 PRINT: PRINT TAB(5) "---> (B) Archive selected records to a
destination file using"
200 PRINT TAB(14) "a PROFILE Index"
210 PRINT: PRINT " Before starting either option, you must be
ready to supply the program with the following information:"
220 PRINT: PRINT TAB(5) "1. The name of your PROFILE (Source)
file."
230 PRINT TAB(5) "2. The number of segments in your Source file."
240 PRINT TAB(5) "3. The name of your Destination file."
250 PRINT TAB(5) "4. The disk drive # for EACH segment of the
Destination file."
260 PRINT: PRINT "To continue press <ENTER>...": GOSUB 6000: CLS
270 PRINT "To Archive from an index, you must also know:"
280 PRINT: PRINT TAB(5) "5. Which PROFILE Index you want to use
PRINT Index (by #) or INQUIRY."
290 PRINT TAB(5) "6. The logical record length (LRL) of the Index
file."
300 PRINT: PRINT "The LRL is listed in the directory (DIR) on your
data disk."
310 PRINT "The INQUIRY index ends in '/IX1'. THE PRINT Indexes
end in '/IY1', '/IY2' etc."
320 PRINT: PRINT "(You will have a chance to see a DIRectory
later!)"
330 PRINT: PRINT "Finally, enter BASIC with FILES=2*(No. of
segments) + 1."
340 PRINT: PRINT: PRINT: PRINT DIV$: PRINT "Do you need to
exit the program now to get some of this information"
350 PRINT "or to re-enter BASIC with the proper number of FILES
(Y/N)?"
360 PRINT: PRINT DIV$: GOSUB 5030: IF YN$="Y" THEN RUN "RM/BAS"
400 REM
410 REM          * INPUT USER DATA *
420 REM
430 CLS: INPUT "Name of Source file";SF$:IF LEN(SF$)>8 THEN PRINT
"8 CHARACTERS OR LESS, PLEASE": GOTO 430
440 SF$=SF$+STRING$(8-LEN(SF$),48)
450 INPUT "How many segments (1-4)";NS: IF NS<1 OR NS>4 GOTO 450
460 IF NS=1 THEN GOTO 490
470 PRINT: PRINT "Did you remember to enter BASIC with ";2*NS+1;"
FILES?"
480 PRINT "If not, do you want to exit now to do it (Y/N)?": GOSUB
5030: IF YN$="Y" THEN RUN "RM/BAS"
490 PRINT: INPUT "Name of Destination file";DF$:IF LEN(DF$)>8 THEN
PRINT "8 CHARACTERS OR LESS, PLEASE": GOTO 490
500 DF$=DF$+STRING$(8-LEN(DF$),48)
```

```
510 FOR I=1 TO NS
520 PRINT "DESTINATION Drive # (0-3) for segment # ";I;:INPUT D$(I)
530 IF LEN(D$(I))>1 THEN PRINT "Enter a number from 0 to 3
WITHOUT a colon (:)":GOTO 520
540 NEXT I
550 PRINT: PRINT "Do you want to use an Index?":GOSUB
5030:IX$=YN$: IF IX$="N" GOTO 640
560 PRINT: INPUT "Index to use <I> OR <1-5>";I$
570 IF I$="I" OR I$="i" THEN I$="/IX1" ELSE I$="/IY"+I$
580 PRINT: PRINT "Do you want to see a DIRectory of your Index
files to read LRL's ?"
590 GOSUB 5030: IF YN$="Y" THEN CLS: SYSTEM "DIR /I": PRINT CN$:
GOSUB 6000
600 INPUT "LRL for Index";LRL
610 CLS: PRINT "*** WARNING ***": PRINT: PRINT "If something halts
the program during data transfer (eg. a Disk I/O error),"
620 PRINT "type the word 'CLOSE' and press <ENTER> at the 'Ready'
prompt to avoid damage"
630 PRINT "to your files!"
640 PRINT: PRINT DIV$: PRINT "Insert Source and Destination data
disks and press <ENTER> when ready.":GOSUB 6000
700 REM
710 REM          * GET SEGMENT LENGTHS *
720 REM
730 OPEN "R",1,SF$+"/MAP"
740 IF LOF(1)=0 THEN GOTO 10090
750 IF LOF(1)<>NS THEN GOTO 10030
760 FIELD 1, 1 AS X$,2 AS L$,253 AS Y$
770 FOR SEG=1 TO LOF(1)
780 GET 1, SEG
790 LS(SEG)=CVI(L$)
800 IF SEG=1 THEN EX$(1)="/KEY" ELSE IF SEG=2 THEN EX$(2)="/DAT"
810 IF SEG=3 THEN EX$(3)="/DA2" ELSE IF SEG=4 THEN EX$(4)="/DA3"
820 NEXT SEG
830 CLOSE
1000 REM
1010 REM         * OPEN DATA FILES FOR I/O *
1020 REM
1030 FOR SEG=1 TO NS
1040 OPEN "R", SEG, SF$+EX$(SEG), LS(SEG)
1050 OPEN "R", SEG+NS, DF$+EX$(SEG)+":"+D$(SEG), LS(SEG)
1060 FIELD SEG, LS(SEG) AS SK$(SEG)
1070 FIELD SEG+NS, LS(SEG) AS DK$(SEG)
1080 NEXT SEG
1090 IF IX$="N" THEN GOTO 1230 ELSE IB=2*NS+1
1100 OPEN "R", IB, SF$+I$,LRL
1110 FIELD IB, LRL-3 AS X1$, 2 AS DR$, 1 AS Y1$
1200 REM
1210 REM         * SETUP TO ARCHIVE FILES USING INDEX *
1220 REM
1230 CLS: N=0
1240 IF IX$="N" THEN GOSUB 2030: GOTO 1310
1250 FOR LR=4 TO LOF(IB)
1260 PRINT@85, "Processing record # ";LR-3
1270 GET IB, LR
1280 DR=CVI(DR$)
1290 GOSUB 3030
1300 NEXT LR
1310 CLOSE: PRINT: PRINT N;"  Records Transferred.": PRINT: PRINT
CN$
1320 GOSUB 6000: RUN "RM/BAS"
2000 REM
2010 REM         * SETUP TO READ FILES WITHOUT INDEX *
2020 REM
2030 FOR DR=1 TO LOF(1)
2040 PRINT@85, "Processing record # ";DR
2050 GOSUB 3030
2060 NEXT DR
2070 RETURN
3000 REM
3010 REM         * READ FILES, DISCARD EMPTY RECORDS, WRITE NEW
FILES *
3020 REM
3030 FOR I=1 TO NS
3040 GET I,DR
3050 NEXT I
3060 IF LEFT$(SK$(1),1)=CHR$(0) THEN GOTO 3130
3070 N=N+1
3080 FOR I=1 TO NS
3090 LSET DK$(I)=SK$(I)
3100 PUT I+NS, N
3110 NEXT I
3120 PRINT@120, "Last record transferred  ";N
3130 RETURN
5000 REM
5010 REM         * INPUT ROUTINES *
5020 REM
5030 YN$=INKEY$: IF YN$="" GOTO 5030
5040 IF YN$="y" THEN YN$="Y"
5050 IF YN$="n" THEN YN$="N"
5060 IF YN$<>"Y" AND YN$<>"N" THEN GOTO 5030
5070 RETURN
6000 EN$=INKEY$: IF EN$="" GOTO 6000
6010 IF EN$<> CHR$(13) THEN GOTO 6000 ELSE RETURN
10000 REM
10010 REM        * ERROR ROUTINES *
10020 REM
10030 CLOSE: CLS: PRINT
10040 PRINT "* ERROR *": PRINT
10050 PRINT "The # of segments you input does not match the # of
segments in"
10060 PRINT "file "+SF$+"/MAP. Please check your data and start
over."
10070 PRINT: PRINT "Press 'Y' to re-enter file data, 'N' to quit":
GOSUB 5030
10080 IF YN$="Y" THEN GOTO 430: ELSE PRINT: PRINT "Files closed":
PRINT: PRINT CN$: RUN "RM/BAS"
10090 CLOSE: KILL SF$+"/MAP": CLS: PRINT
10100 PRINT "* ERROR *": PRINT: PRINT "File ";SF$;"/MAP";" Not
Found."
10110 GOTO 10070
65000 REM        * PROFILE is a trademark of Tandy Corporation *
```

REPLACE/BAS

```
10 REM          * REPLACE/BAS - Placed in the public domain by
DAS, 01/10/86 *
20 REM          * Replaces user-specified field(s) in a PROFILE
4(tm) data    *
30 REM          * file with specified Literal(s) for all records
or for        *
40 REM          * records selected using a PROFILE index
*
50 REM
60 CLS: PRINT@ (6,20), CHR$(16);" R E P L A C E   F O R  P R O F I
L E 4 "
70 PRINT: PRINT TAB(29) "     By Don Singer     "
80 PRINT TAB(31), "3726 Skyline Drive"
90 PRINT TAB(29), "Scottsbluff, Ne 69361"
100 FOR I=1 TO 3000:NEXT
200 REM
210 REM          * INITIALIZE AND INSTRUCTIONS *
220 REM
230 CLS: CLEAR 5000:DEFINT A-Z: DIV$=STRING$(79,"="): ER$="***
ENTRY ERROR - TRY AGAIN ***": CN$="To continue, press <ENTER>..."
240 EX$(1)="/KEY": EX$(2)="/DAT": EX$(3)="/DA2": EX$(4)="/DA3"
250 DIM
DT$(100),DTR$(100),H$(100),LD(100),NC$(37),NC(37),JU$(5,37),C(5,37)
260 PRINT@ (2,10), "Do you want instructions (Y/N)?": GOSUB 5030:
IF YN$="N" GOTO 630
270 PRINT: PRINT TAB(33) "***************"
280 PRINT TAB(33) "*** WARNING ***"
290 PRINT TAB(33) "***************"
300 PRINT: PRINT: PRINT "This program manipulates your valuable
PROFILE data files, and has the power"
310 PRINT "to RUIN them if you make a mistake!  So for your own
protection, PLEASE have"
320 PRINT "sufficient Backups, and test your changes BEFORE
putting them in use.": PRINT
330 PRINT TAB(35) "Thank You!": PRINT: PRINT: PRINT CN$: GOSUB
6000: CLS
340 CLS: PRINT "This program can be used in two ways:"
350 PRINT: PRINT TAB(5) "---> (A) Replace fields with constant
values specified from the "
360 PRINT TAB(14) "keyboard for ALL records"
370 PRINT: PRINT TAB(5) "---> (B) Replace fields in records
selected using"
380 PRINT TAB(14) "a PROFILE Index"
390 PRINT: PRINT " Before starting either option, you must be
ready to supply the program with the following information:"
400 PRINT: PRINT TAB(5) "1. The name of your PROFILE (Source)
```

```
file."
410 PRINT TAB(5) "2. The number of segments in your Source file."
420 PRINT TAB(5) "3. The new data to 'PLUG INTO' the contents of
each field"
430 PRINT TAB(8) "you want to replace."
440 PRINT TAB(5) "4. Whether you want the new data RIGHT- or
LEFT-JUSTIFIED in each field."
450 PRINT: PRINT TAB(8) "NOTE: If replacement data is longer than
the field length, it will"
460 PRINT TAB(8) "be LEFT-JUSTIFIED, with the extra characters
cut off on the right!"
470 PRINT: PRINT "To continue press <ENTER>...": GOSUB 6000: CLS
480 PRINT "To Archive from an index, you must also know:"
490 PRINT: PRINT TAB(5) "5. Which PROFILE Index you want to use -
PRINT Index (by #) or INQUIRY."
500 PRINT TAB(5) "6. The logical record length (LRL) of the Index
file."
510 PRINT: PRINT "The LRL is listed in the directory (DIR) on your
data disk."
520 PRINT "The INQUIRY index ends in '/IX1'. THE PRINT Indexes
end in '/IY1', '/IY2' etc."
530 PRINT: PRINT "(You will have a chance to see a DIRectory
later!)"
540 PRINT: PRINT "Finally, you must enter BASIC with FILES = No.
of segments + 1."
550 PRINT: PRINT: PRINT: PRINT: PRINT DIV$: PRINT "Do you need to
exit the program now to get some of this information"
560 PRINT "or to re-enter BASIC with the proper number of FILES
(Y/N)?"
570 PRINT: PRINT DIV$: GOSUB 5030: IF YN$="Y" THEN CLOSE: RUN
"RM/BAS"
600 REM
610 REM          * INPUT USER DATA *
620 REM
630 CLS: INPUT "Name of Source file";SF$:IF LEN(SF$)>8 THEN PRINT
"8 CHARACTERS OR LESS, PLEASE": GOTO 630
640 SF$=SF$+STRING$(8-LEN(SF$),48)
650 INPUT "How many segments (1-4)";NS: IF NS<1 OR NS>4 GOTO 650
660 IF NS<=2 THEN GOTO 690
670 PRINT: PRINT "Did you remember to enter BASIC with ";NS+1;"
FILES?"
680 PRINT "If not, do you want to exit now to do it (Y/N)?": GOSUB
5030: IF YN$="Y" THEN CLOSE: RUN "RM/BAS"
690 PRINT: PRINT "Do you want to use an Index?":GOSUB
5030:IX$=YN$: IF IX$="N" GOTO 780
700 PRINT: INPUT "Index to use <I> OR <1-5>";I$
710 IF I$="I" OR I$="i" THEN I$="/IX1" ELSE I$="/IY"+I$
720 PRINT: PRINT "Do you want to see a DIRectory of your Index
files to read LRL's ?"
730 GOSUB 5030: IF YN$="Y" THEN CLS: SYSTEM "DIR /I": PRINT CN$:
GOSUB 6000
740 INPUT "LRL for Index";LRL
750 CLS: PRINT "*** WARNING ***": PRINT: PRINT "If something halts
the program during data transfer (eg. a Disk I/O error),"
760 PRINT "type the word 'CLOSE' and press <ENTER> at the 'Ready'
prompt to avoid damage"
770 PRINT "to your files!"
780 PRINT: PRINT DIV$: PRINT "Insert Source data disks and press
<ENTER> when ready.":GOSUB 6000
800 REM
810 REM          * READ /MAP FILE, DISPLAY FIELDS *
820 REM          * AND SELECT FIELDS TO BE REPLACED *
830 REM
840 NF=0: J=0: OPEN "R",1,SF$+"/MAP"
850 IF LOF(1)=0 THEN GOTO 10090
860 IF LOF(1)<> NS THEN GOTO 10030
870 FIELD 1, 1 AS X$,2 AS L$,13 AS Y$,239 AS Z$
880 FOR SEG=1 TO LOF(1): GET 1, SEG: LS(SEG)=CVI(L$): NF(SEG)=0
890 IF ASC(Z$)=0 THEN GOTO 920
900 NF=NF+1: NF(SEG)=NF(SEG)+1: LH=ASC(Z$): H$(NF)=MID$(Z$,2,LH)
910 LD(NF)=ASC(MID$(Z$,LH+2,1)): LSET Z$=MID$(Z$,LH+3): GOTO 890
920 CLS: PRINT "SEGMENT #  ";SEG
930 PRINT " #/FIELD HEADING";STRING$(16,32);"LENGTH";"     #/FIELD
HEADING";STRING$(16,32);"LENGTH"
940 PRINT DIV$: IF NF(SEG)<18 THEN K=NF(SEG) ELSE K=18
950 FOR I=1 TO K: J=J+1: PRINT USING "##";J;:PRINT
"-";H$(J);STRING$(32-LEN(H$(J)),32);:PRINT USING "###";LD(J): NEXT I
960 IF NF(SEG)<=18 THEN GOTO 980 ELSE P=0
970 FOR I=K+1 TO NF(SEG): J=J+1: P=P+1: PRINT @ (P+2,41), USING
"##";J;:PRINT " ";H$(J);STRING$(32-LEN(H$(J)),32);:PRINT USING
"###";LD(J);:NEXT I
980 PRINT @ (21,0),"How many fields are to be replaced in segment
#";SEG;:INPUT NC$(SEG): IF NC$(SEG)="" THEN PRINT ER$: GOTO 980
990 IF ASC(NC$(SEG))<48 OR ASC(NC$(SEG))>57 THEN PRINT ER$: GOTO
980
1000 NC(SEG)=VAL(NC$(SEG)): IF NC(SEG)<1 THEN GOTO 1160
1010 FOR I=1 TO NC(SEG): IF I=1 THEN PRMT$="FIRST" ELSE
PRMT$="NEXT"
1020 PRINT @ (22,0),"Enter 'FIELD #,JUSTIFICATION (R/L)' of
";PRMT$;" field to replace -eg. <3,R>";: INPUT C(SEG,I),JU$(SEG,I)
1030 IF JU$(SEG,I)="r" THEN JU$(SEG,I)="R" ELSE IF JU$(SEG,I)="l"
THEN JU$(SEG,I)="L"
1040 IF JU$(SEG,I)<>"R" AND JU$(SEG,I)<>"L" THEN PRINT@ (23,0),
"Input JUSTIFICATION (R/L) for field # ";C(SEG,I);" Again  ";: INPUT
JU$(SEG,I): GOTO 1030
1050 FOR T=1 TO 600: NEXT T: PRINT @ (22,0), SPACE$(78): NEXT I
1060 CLS: PRINT "In Segment # ";SEG;" Data in these fields will be
replaced:"
1070 PRINT: PRINT "FIELD HEADING";:PRINT TAB(25) "FIELD #
JUSTIFICATION": PRINT DIV$
1080 FOR I=1 TO NC(SEG): PRINT H$(C(SEG,I));TAB(30);:PRINT USING
"##";C(SEG,I);: PRINT TAB(41) JU$(SEG,I):  NEXT I
1090 PRINT: PRINT "(Press <CTL><:> for Hardcopy) - Are these
correct (Y/N)?": GOSUB 5030
1100 IF YN$="N" THEN J=J-NF(SEG): GOTO 920
1110 CLS: FOR I=1 TO NC(SEG)
1120 PRINT "Enter new data to replace contents of Field #
";C(SEG,I);" (";H$(C(SEG,I));")"
1130 PRINT "?";:LINE INPUT DTR$(C(SEG,I))
1140 PRINT "Is this correct?": GOSUB 5030: IF YN$="N" THEN GOTO
1120
1150 NEXT I
1160 NEXT SEG
1170 CLOSE
1200 REM
1210 REM          * OPEN DATA FILES FOR I/O *
1220 REM
1230 NF=0: FOR SEG=1 TO NS
1240 OPEN "R", SEG, SF$+EX$(SEG), LS(SEG): DU=0
1250 FOR I=1 TO NF(SEG)
1260 NF=NF+1: FIELD SEG, (DU) AS DUM$, LD(NF) AS DT$(NF):
DU=DU+LD(NF)
1270 NEXT I
1280 NEXT SEG
1290 IF IX$="N" THEN GOTO 1430 ELSE IB=NS+1
1300 OPEN "R", IB, SF$+I$,LRL
1310 FIELD IB, LRL-3 AS X1$, 2 AS DR$, 1 AS Y1$
1400 REM
1410 REM          * SETUP TO READ FILES USING INDEX *
1420 REM
1430 CLS: N=0
1440 IF IX$="N" THEN GOSUB 1630: GOTO 1510
1450 FOR LR=4 TO LOF(IB)
1460 PRINT@85, "Processing record # ";LR-3
1470 GET IB, LR
1480 DR=CVI(DR$)
1490 GOSUB 1730
1500 NEXT LR
1510 CLOSE: PRINT "DONE!": PRINT
1520 PRINT CN$: GOSUB 6000: RUN "RM/BAS"
1600 REM
1610 REM          * SETUP TO READ FILES WITHOUT INDEX *
1620 REM
1630 FOR DR=1 TO LOF(1)
1640 PRINT@85, "Processing record # ";DR
1650 GOSUB 1730
1660 NEXT DR
1670 RETURN
1700 REM
1710 REM          * READ FILES, REPLACE DATA, WRITE FILES *
1720 REM
1730 FOR SEG=1 TO NS
1740 GET SEG, DR
1750 FOR I=1 TO NC(SEG)
1760 IF JU$(SEG,I)="L" THEN LSET DT$(C(SEG,I))=DTR$(C(SEG,I))
1770 IF JU$(SEG,I)="R" THEN RSET DT$(C(SEG,I))=DTR$(C(SEG,I))
1780 NEXT I
1790 PUT SEG, DR
1800 NEXT SEG
1810 RETURN
```

```
5000 REM
5010 REM        * INPUT ROUTINES *
5020 REM
5030 YN$=INKEY$: IF YN$="" GOTO 5030
5040 IF YN$="y" THEN YN$="Y"
5050 IF YN$="n" THEN YN$="N"
5060 IF YN$<>"Y" AND YN$<>"N" THEN GOTO 5030
5070 RETURN
6000 EN$=INKEY$: IF EN$="" GOTO 6000
6010 IF EN$<> CHR$(13) THEN GOTO 6000 ELSE RETURN
10000 REM
10010 REM       * ERROR ROUTINES *
10020 REM
10030 CLOSE: CLS: PRINT
10040 PRINT "* ERROR *": PRINT
10050 PRINT "The # of segments you input does not match the # of
segments in"
10060 PRINT "file "+SF$+"/MAP. Please check your data and start
over."
10070 PRINT: PRINT "Press 'Y' to re-enter file data, 'N' to
quit":GOSUB 5030
10080 IF YN$="Y" THEN GOTO 630 ELSE PRINT "Files closed.": PRINT:
PRINT CN$: GOSUB 6000: RUN "RM/BAS"
10090 CLOSE: KILL SF$+"/MAP": CLS: PRINT
10100 PRINT "* ERROR *": PRINT: PRINT "File ";SF$;"/MAP";" Not
Found"
10110 GOTO 10070
65000 REM       * PROFILE is a trademark of Tandy Corporation *
```

TRANSFER/BAS

```
10 REM          * TRANSFER/BAS - Placed in the public domain by
DAS on       *
20 REM          * 01/21/86. Transfers data from & to selected
fields in    *
30 REM          * PROFILE(tm) 4 Source and Destination files.
*
40 REM
50 CLS: PRINT@ (6,20), CHR$(16);" T R A N S F E R   F O R  P R O F
I L E 4 "
60 PRINT: PRINT TAB(29) "    By Don Singer    "
70 PRINT TAB(31), "3726 Skyline Drive"
80 PRINT TAB(29), "Scottsbluff, Ne 69361"
90 FOR I=1 TO 3000:NEXT
100 REM
110 REM         * INITIALIZE AND INSTRUCTIONS *
120 REM
130 CLS: CLEAR 5000: DEFINT A-Z: DIV$=STRING$(79,"=")
140 EX$(1)="/KEY": EX$(2)="/DAT": EX$(3)="/DA2": EX$(4)="/DA3"
150 ER$="*** ENTRY ERROR - TRY AGAIN ***": CN$="To continue,
press <ENTER>..."
160 LH$=" #/FIELD HEADING"+STRING$(16,32)+"LENGTH": RH$="   "+LH$
170 DIM JU$(99),DT$(99,2),H$(99,2),LD(99,2),SF(99),DF(99)
180 PRINT@ (2,10), "Do you want instructions (Y/N)?": GOSUB 5030:
IF YN$="N" GOTO 600
190 PRINT: PRINT TAB(33) "****************"
200 PRINT TAB(33) "*** WARNING ***"
210 PRINT TAB(33) "****************"
220 PRINT: PRINT: PRINT "This program manipulates your valuable
PROFILE data files, and has the power"
230 PRINT "to RUIN them if you make a mistake!  So for your own
protection, PLEASE have"
240 PRINT "sufficient Backups, and test your changes BEFORE
putting them in use.": PRINT
250 PRINT TAB(35) "Thank You!": PRINT: PRINT: PRINT CN$: GOSUB
6000: CLS
260 CLS: PRINT "This program transfers data in specified fields
between two PROFILE data"
270 PRINT "files. The records to be used are chosen from the
files in two ways:"
280 PRINT: PRINT TAB(5) "---> (A) Data is moved from EACH record
in the Source File to the"
290 PRINT TAB(14) "same-numbered record in the Destination File."
300 PRINT: PRINT TAB(14) "NOTE: This option should be used only if
SOURCE and DESTINATION"
310 PRINT TAB(14) "files have the SAME number and order of
records (or order is not"
320 PRINT TAB(14) "important - eg. a new Destination file with no
data)."
330 PRINT: PRINT: PRINT TAB(5) "---> (B) SOURCE and DESTINATION
records are selected using a PROFILE"
340 PRINT TAB(14) "Index from each file."
350 PRINT: PRINT TAB(14) "NOTE: Both Source and Destination file
Indexes should be"
360 PRINT TAB(14) "'freshly-built' using IDENTICAL Sort and
Selection criteria"
370 PRINT TAB(14) "to prevent scrambling of the transferred data.
In particular,"
380 PRINT TAB(14) "they MUST have the SAME NUMBER of records
sorted in the SAME order!"
390 PRINT: PRINT: PRINT: PRINT CN$: GOSUB 6000: CLS
400 PRINT "Before starting either option, you must be ready to
supply the program with the following information:"
410 PRINT: PRINT TAB(5) "1. The name of your SOURCE file."
420 PRINT TAB(5) "2. The number of segments in your SOURCE file."
430 PRINT TAB(5) "3. The name of your DESTINATION file."
440 PRINT TAB(5) "4. The number of segments in your DESTINATION
file."
450 PRINT: PRINT CN$: GOSUB 6000: CLS
460 PRINT "To Transfer using Indexes, you must also know:"
470 PRINT: PRINT TAB(5) "5. Which PROFILE Indexes you want to use
- PRINT Indexes (by #) or INQUIRY."
480 PRINT TAB(5) "6. The logical record length (LRL) of the Index
file."
490 PRINT: PRINT "The LRL is listed in the directory (DIR) on your
data disk."
500 PRINT "The INQUIRY index ends in '/IX1'. THE PRINT Indexes
end in '/IY1', '/IY2' etc."
510 PRINT: PRINT "(You will have a chance to see a DIRectory
later!)"
520 PRINT: PRINT "Finally, enter BASIC with:"
530 PRINT: PRINT TAB(5) "FILES = (No. of Source segments) + (No.
of DESTINATION segments) + 2."
540 PRINT: PRINT: PRINT: PRINT DIV$: PRINT "Do you need to EXIT
the program now to get some of this information"
550 PRINT "or to re-enter BASIC with the proper number of FILES
(Y/N)?"
560 PRINT: PRINT DIV$: GOSUB 5030: IF YN$="Y" THEN CLOSE: RUN
"RM/BAS"
570 REM
580 REM         * INPUT USER DATA *
590 REM
600 CLS: INPUT "Name of Source file";F$(1):IF LEN(F$(1))>8 THEN
PRINT "8 CHARACTERS OR LESS, PLEASE": GOTO 600
610 F$(1)=F$(1)+STRING$(8-LEN(F$(1)),48)
620 INPUT "How many segments (1-4)";NS(1): IF NS(1)<1 OR NS(1)>4
GOTO 620
630 PRINT: INPUT "Name of Destination file";F$(2):IF LEN(F$(2))>8
THEN PRINT "8 CHARACTERS OR LESS, PLEASE": GOTO 630
640 F$(2)=F$(2)+STRING$(8-LEN(F$(2)),48)
650 INPUT "How many segments (1-4)";NS(2): IF NS(2)<1 OR NS(2)>4
GOTO 650
660 PRINT: PRINT "Do you want to use Indexes?":GOSUB
5030:IX$=YN$
670 IF IX$="N" OR IX$="n" THEN FI=NS(1)+NS(2): GOTO 760 ELSE
FI=NS(1)+NS(2)+2
680 PRINT: INPUT "SOURCE Index to use <I> OR <1-5>";I$(1)
690 IF I$(1)="I" OR I$(1)="i" THEN I$(1)="/IX1" ELSE I$(1)="/IY"+I$(1)
700 PRINT: PRINT "Do you want to see a DIRectory of your Index
files to read LRL's ?"
710 GOSUB 5030: IF YN$="Y" THEN CLS: SYSTEM "DIR /I": PRINT
CN$:GOSUB 6000
720 INPUT "LRL for SOURCE Index";LRL(1)
730 PRINT: INPUT "DESTINATION Index to use <I> OR <1-5>";I$(2)
740 IF I$(2)="I" OR I$(2)="i" THEN I$(2)="/IX1" ELSE I$(2)="/IY"+I$(2)
750 INPUT "LRL for DESTINATION Index";LRL(2)
760 PRINT: PRINT DIV$: PRINT "Did you enter BASIC with ";FI;"
Files?"
770 PRINT "If not, do you want to EXIT now to do it (Y/N)?": PRINT
780 PRINT DIV$: GOSUB 5030: IF YN$="Y" THEN CLOSE: RUN "RM/BAS"
790 CLS: PRINT "*** WARNING ***": PRINT: PRINT "If something halts
the program during data transfer (eg. a Disk I/O error),"
800 PRINT "type the word 'CLOSE' and press <ENTER> at the 'Ready'
prompt to avoid damage"
810 PRINT "to your files!"
820 PRINT: PRINT DIV$: PRINT "*** Insert Source and Destination
data disks and press <ENTER> when ready. ***": GOSUB 6000
900 REM
910 REM         * READ SOURCE & DESTINATION FILES *
920 REM
930 NX=0: FOR I=1 TO 2
```

```
940 IF I=1 THEN SD$="SOURCE" ELSE SD$="DESTINATION"
950 GOSUB 2030
960 NEXT I
1000 REM
1010 REM        * OPEN DATA FILES FOR I/O *
1020 REM
1030 CLS: PRINT: PRINT TAB(30) "...OPENING FILES...": NB=0: FOR I=1
TO 2
1040 NF=0: FOR SEG=1 TO NS(I): NB=NB+1
1050 OPEN "R", NB, F$(I)+EX$(SEG), LS(SEG,I): DU=0
1060 FOR J=1 TO N(SEG,I)
1070 NF=NF+1: FIELD NB, (DU) AS DUM$, LD(NF,I) AS DT$(NF,I):
DU=DU+LD(NF,I)
1080 NEXT J: NEXT SEG: NEXT I
1090 IF IX$="N" THEN GOTO 1230
1100 FOR I=1 TO 2: NB=NB+1
1110 OPEN "R", NB, F$(I)+I$(I),LRL(I)
1120 FIELD NB, LRL(I)-3 AS X1$, 2 AS DR$(I), 1 AS Y1$
1130 IB(I)=NB: NEXT I
1200 REM
1210 REM        * SETUP TO READ FILES USING INDEX *
1220 REM
1230 CLS: N=0
1240 IF IX$="N" THEN GOSUB 1430: GOTO 1310
1250 FOR LR=4 TO LOF(IB(1))
1260 PRINT@85, "Processing record # ";LR-3
1270 GET IB(1), LR: GET IB(2), LR
1280 DR1=CVI(DR$(1)): DR2=CVI(DR$(2))
1290 GOSUB 1530
1300 NEXT LR
1310 CLOSE: PRINT "DONE! - Files Closed": PRINT
1320 PRINT CN$: GOSUB 6000: RUN "RM/BAS"
1400 REM
1410 REM        * SETUP TO READ FILES WITHOUT INDEX *
1420 REM
1430 FOR DR1=1 TO LOF(1): DR2=DR1
1440 PRINT@85, "Processing record # ";DR1
1450 GOSUB 1530
1460 NEXT DR1
1470 RETURN
1500 REM
1510 REM        * READ FILES, TRANSFER DATA, WRITE FILES *
1520 REM
1530 NB=0: FOR I=1 TO 2
1540 IF I=1 THEN DR=DR1 ELSE DR=DR2
1550 FOR SEG=1 TO NS(I): NB=NB+1
1560 GET NB, DR
1570 NEXT SEG: NEXT I
1580 FOR M=1 TO NX
1590 IF JU$(M)="L" THEN LSET DT$(DF(M),2)=DT$(SF(M),1) ELSE RSET
DT$(DF(M),2)=DT$(SF(M),1)
1600 NEXT M
1610 NB=0: FOR I=1 TO 2
1620 IF I=1 THEN DR=DR1 ELSE DR=DR2
1630 FOR SEG=1 TO NS(I): NB=NB+1
1640 PUT NB, DR
1650 NEXT SEG: NEXT I
1660 RETURN
2000 REM
2010 REM         * READ /MAP FILES, DISPLAY FIELDS *
2020 REM
2030 NF(I)=0: J=0: OPEN "R",1,F$(I)+"/MAP"
2040 IF LOF(1)=0 THEN GOTO 10090
2050 IF LOF(1)<> NS(I) THEN GOTO 10030
2060 FIELD 1, 1 AS X$,2 AS L$,13 AS Y$,239 AS Z$
2070 FOR SEG=1 TO LOF(1): GET 1, SEG: LS(SEG,I)=CVI(L$): N(SEG,I)=0
2080 IF ASC(Z$)=0 THEN GOTO 2120
2090 NF(I)=NF(I)+1: N(SEG,I)=N(SEG,I)+1: LH=ASC(Z$):
H$(NF(I),I)=MID$(Z$,2,LH)
2100 LD(NF(I),I)=ASC(MID$(Z$,LH+2,1)): LSET Z$=MID$(Z$,LH+3): GOTO
2080
2110 RPT=0
2120 CLS: PRINT "SEGMENT #  ";SEG, "FILE  ";F$(I)
2130 PRINT LH$;RH$
2140 PRINT DIV$: IF N(SEG,I)<18 THEN K=N(SEG,I) ELSE K=18
2150 FOR M=1 TO K: J=J+1: PRINT USING "##";J;:PRINT
"-";H$(J,I);STRING$(32-LEN(H$(J,I)),32);:PRINT USING "###";LD(J,I): NEXT
M
2160 IF N(SEG,I)<=18 THEN GOTO 2180 ELSE P=0
2170 FOR M=K+1 TO N(SEG,I): J=J+1: P=P+1: PRINT @ (P+2,41), USING
"##";J;:PRINT "-";H$(J,I);STRING$(32-LEN(H$(J,I)),32);:PRINT USING
"###";LD(J,I);:NEXT M
2180 IF I=1 THEN GOSUB 3030
2190 IF I=2 THEN GOSUB 4030
2200 IF RPT=1 THEN GOTO 2110
2210 NEXT SEG
2220 IF RPT=2 THEN GOTO 930
2230 CLOSE
2240 RETURN
3000 REM
3010 REM        * PICK SOURCE FIELDS *
3020 REM
3030 PRINT @ (21,0),"How many ";SD$;" fields in segment #";SEG;" (0
to skip)";:INPUT NC$(SEG): IF NC$(SEG)="" THEN PRINT ER$: GOTO 3030
3040 IF ASC(NC$(SEG))<48 OR ASC(NC$(SEG))>57 THEN PRINT ER$: GOTO
3030
3050 NC(SEG)=VAL(NC$(SEG)): K=NX: IF NC(SEG)<1 THEN GOTO 3140
3060 FOR M=1 TO NC(SEG): NX=NX+1: IF M=1 THEN PRMT$="First" ELSE
PRMT$="Next"
3070 PRINT @ (22,0),"Enter Field #  of ";PRMT$;" ";SD$;" field";:
INPUT SF(NX)
3080 FOR T=1 TO 600: NEXT T: PRINT @ (22,0), SPACE$(78): NEXT M
3090 CLS: PRINT "In Segment # ";SEG;" These are the ";SD$;" fields:"
3100 PRINT: PRINT LEFT$(LH$,32): PRINT DIV$
3110 FOR M=K+1 TO NX: PRINT USING "##";SF(M);:PRINT "-";H$(SF(M),I):
NEXT M
3120 PRINT: PRINT "(Press <CTL><:> for Hardcopy) - Are these
correct (Y/N)?": GOSUB 5030
3130 IF YN$="N" THEN J=J-N(SEG,I): NX=NX-NC(SEG): RPT=1
3140 RETURN
4000 REM
4010 REM        * PICK DESTINATION FIELDS *
4020 REM
4030 FOR M=1 TO NX: IF DF(M)>0 THEN GOTO 4080
4040 PRINT @ (21,0), "For SOURCE field #";SF(M);"- ";H$(SF(M),1);" (If
Destination field not"
4050 PRINT @ (22,0), "in this segment, enter <0> to skip)";": Enter
DESTINATION field # ";:INPUT DF(M)
4060 FOR T=1 TO 600: NEXT T: PRINT @ (21,0), SPACE$(79)
4070 PRINT SPACE$(79)
4080 NEXT M
4090 IF SEG<LOF(1) THEN GOTO 4260
4100 FOR M=1 TO NX: JU$(M)="L"
4110 IF LD(SF(M),1)<LD(DF(M),2) THEN GOSUB 4300
4120 IF LD(SF(M),1)>LD(DF(M),2) THEN GOSUB 4400
4130 NEXT M
4140 IF EX=1 THEN GOSUB 4490
4150 IF NX<18 THEN K=NX ELSE K=18
4160 M=0
4170 CLS: PRINT TAB(20) "SOURCE"; TAB(57) "DESTINATION"
4180 PRINT LEFT$(LH$,32);SPACE$(6);LEFT$(RH$,28);"JUSTIFICATION":
PRINT DIV$
4190 FOR J=1 TO K: M=M+1
4200 PRINT USING "##";SF(M);: PRINT
"-";H$(SF(M),1);STRING$(38-LEN(H$(SF(M),1)),32);: PRINT USING
"##";DF(M);: PRINT "-"
H$(DF(M),2);STRING$(32-LEN(H$(DF(M),2)),32);JU$(M)
4210 IF M<NX THEN NEXT J
4220 PRINT CN$: GOSUB 6000
4230 IF M<NX THEN GOTO 4170
4240 PRINT @ (21,0), "(Press <CTL><:> for Hardcopy) - Are these
correct (Y/N)?": GOSUB 5030
4250 IF YN$="N" THEN CLOSE: CLS: PRINT "PLEASE RE-ENTER FIELD
DATA": RPT=2
4260 RETURN
4300 CLS: PRINT "Source field ";SF(M);"-";H$(SF(M),1);" is"
4310 PRINT "shorter than Destination field ";DF(M);"-";H$(DF(M),2)
4320 PRINT: INPUT "Enter JUSTIFICATION for data (R/L) ";JU$(M)
4330 IF JU$(M)="r" THEN JU$(M)="R" ELSE IF JU$(M)="l" THEN
JU$(M)="L"
4340 IF JU$(M)<>"R" AND JU$(M)<>"L" THEN PRINT: INPUT "Input
JUSTIFICATION (R/L) for this field again  ";JU$(M)
4350 RETURN
4400 CLS: PRINT "* WARNING *": PRINT: PRINT "Source field
";SF(M);"-";H$(SF(M),1);" is"
4410 PRINT "LONGER than Destination field ";DF(M);"-";H$(DF(M),2)
4420 PRINT: PRINT "Data will be LEFT-Justified, and the last
";LD(SF(M),1)-LD(DF(M),2);"characters"
4430 PRINT "will be LOST."
4440 PRINT: PRINT "To prevent this, you must lengthen the
```

```
Destination Field with"
4450 PRINT "PROFILE's 'Define Files' option. "
4460 PRINT: PRINT "If you want to do this, make a note of this
destination field Number and"
4470 PRINT "EXIT at the next opportunity!"
4480 PRINT: PRINT CN$: GOSUB 6000: EX=1: RETURN
4490 CLS: PRINT "Do you want to EXIT to lengthen Destination
fields (Y/N)?"
4500 GOSUB 5030: IF YN$="Y" THEN CLOSE: PRINT "Files closed.":
RUN "RM/BAS"
4510 RETURN
5000 REM
5010 REM        * INPUT ROUTINES *
5020 REM
5030 YN$=INKEY$: IF YN$="" GOTO 5030
5040 IF YN$="y" THEN YN$="Y"
5050 IF YN$="n" THEN YN$="N"
5060 IF YN$<>"Y" AND YN$<>"N" THEN GOTO 5030
5070 RETURN
6000 EN$=INKEY$: IF EN$="" GOTO 6000
6010 IF EN$<> CHR$(13) THEN GOTO 6000 ELSE RETURN
10000 REM
10010 REM       * ERROR ROUTINES *
10020 REM
10030 CLOSE: CLS: PRINT
10040 PRINT "* ERROR *": PRINT
10050 PRINT "The # of segments you input does not match the # of
segments in"
10060 PRINT "file "+F$(I)+"/MAP. Please check your data and start
over."
10070 PRINT: PRINT "Press 'Y' to re-enter file data, 'N' to
quit":GOSUB 5030
10080 CLOSE: IF YN$="Y" THEN GOTO 600 ELSE PRINT: PRINT "Files
Closed.": PRINT: PRINT CN$: GOSUB 6000: RUN "RM/BAS"
10090 CLOSE: KILL F$(I)+"/MAP": CLS: PRINT
10100 PRINT "* ERROR *": PRINT: PRINT "File ";F$(I);"/MAP";" Not
Found"
10110 GOTO 10070
65000 REM       * PROFILE is a trademark of Tandy Corporation *
```

RENAME/BAS

[Editor's note: This is an adaptation of a Model II program that originally appeared in the April, 1984 issue of the TRS-80 Microcomputer News. Also, in the interest of conservation of newsletter space, blank remark lines (which contained only a line number and an apostrophe) have been DELETED at the following line numbers: 190, 9000, 10500, 10600, 11700, 12500, 13000, 13500, 13600, 14300, 14400, 15800, 15900, 17300, 18400, 18500, 19700, 19800, 20800, 20900, 21900, 22000, 22500, 22600, 23400, 23500, 24100, 24200.]

```
100 'RENAME/BAS
110 'TRS-80 MICROCOMPUTER NEWS, APRIL 1984
120 'BY CALVIN ROBERTS
130 'PO BOX 22413
140 'SAN DIEGO, CA 92122
150 'ENTERED AND CONVERTED TO MODEL IV PROFILE(tm) 4+
160 'BY DWAIN SUTTON
170 'BUGGYPOLE RANCH
180 'ASHBY, NE  69333
200 '
8200 CLEAR 4000
8300 ON ERROR GOTO 23000
8400 DIM D$(100)
8500 DIM E$(100)
8600 DIM C$(100)
8700 DIM F$(100)
8800 DIM G$(100)
8900 R$=CHR$(16):N$=CHR$(17)
9100 '
9200 'MENU
9300 '
9400 CLS:A=0:B=0:PRINT TAB(25);R$" PROFILE FILE MANIPULATION MENU
"N$
9410 PRINT @(1,0),STRING$(80,"_")
9500 A$=" ":B$=A$:FOR J=0 TO 50:C$(J)=A$:D$(J)=A$:E$(J)=A$
9600 F$(J)=A$:G$(J)=A$:NEXT:J=0
9700 PRINT:PRINT
9800 PRINT TAB(30);"1 - Single File Rename":PRINT
9900 PRINT TAB(30);"2 - TOTAL FILE RENAME":PRINT
10000 PRINT TAB(30);"3 - Return to Menu":PRINT
10100 PRINT TAB(30);"4 - Exit to TRSDOS":PRINT
10110 PRINT @(19,0),STRING$(80,"_")
10200 PRINT TAB(32);R$" Enter Selection "N$
10300 INPUT Y:ON Y GOTO 11000,12100,12900,13400
10400 GOTO 9400
10700 '
10800 'Single File Rename
10900 '
11000 GOSUB 14000
11100 PRINT "Extent :";
11200 LINE INPUT C$:PRINT
11300 GOSUB 14800
11400 D$(0)=A$+"/"+C$:E$(0)=B$+"/"+C$
11500 L=0:GOSUB 20200
11600 PRINT:GOSUB 22400:GOTO 9400
11800 '
11900 ' All file Rename
12000 '
12100 GOSUB 14000:GOSUB 14800
12200 GOSUB 17700:GOSUB 18900
12300 GOSUB 20200:GOSUB 21400
12400 GOSUB 22400:GOTO 9400
12600 '
12700 ' Exit to PROFILE IV+ Menu
12800 '
12900 RUN"RM/BAS"
13100 '
13200 ' Exit to TRSDOS
13300 '
13400 CLS:SYSTEM
13700 '
13800 ' Subroutine to Input Old PROFILE File Name
13900 '
14000 CLS:PRINT "Old PROFILE File Name :";
14100 LINE INPUT A$
14200 PRINT:Z$=A$:GOSUB 23900:A$=Z$:RETURN
14500 '
14600 ' Subroutine to Input New PROFILE File Name
14700 '
14800 PRINT "New PROFILE File Name :";
14900 LINE INPUT B$
15000 PRINT:Z$=B$:GOSUB 23900:B$=Z$
15100 '
15200 ' Check to see if Old and New File Names are the Same
15300 '
15400 IF A$<>B$ THEN GOTO 15700
15500 PRINT "Files Cannot Have the same File Names !!"
15600 FOR V=1 TO 1000:NEXT:GOTO 9400
15700 PRINT:RETURN
16000 '
16100 ' Data Statements
16200 '
16300 DATA MAP,KEY,DAT,DA2,DA3:'                          File Segments
16400 DATA PM0,PM1,PM2,PM3,PM4,PM5,PM6,PM7,PM8,PM9:' Screen formats
16500 DATA PR0,PR1,PR2,PR3,PR4,PR5,PR6,PR7,PR8,PR9:' Report formats
16600 DATA LB0,LB1,LB2,LB3,LB4,LB5:',LB6,LB7,LB8,LB9:' Label formats
16700 DATA SL0,SL1,SL2,SL3,SL4,SL5:',SL6,SL7,SL8,SL9:' SuperScripsit
format
16800 DATA VC0,VC1,VC2,VC3,VC4,VC5:',VC6,VC7,VC8,VC9:' VisiCalc
format
16900 DATA MTH:'                                          Math file
17000 DATA IX1,IY1,IY2,IY3,IY4,IY5
17100 DATA 999
17400 '
17500 ' Subroutine to Add Extents to Old File Names
17600 '
17700 READ C$(J)
17800 IF C$(J)="999" THEN 18300
17900 D$(J)=A$+"/"+C$(J)
18000 J=J+1
18100 GOTO 17700
18200 L=J-1
18300 RESTORE:RETURN
18600 '
18700 ' Subroutine to Add Extents to New File Names
18800 '
18900 J=0
19000 READ C$(J)
19100 IF C$(J)="999" THEN 19500
```

```
19200 E$(J)=B$+"/"+C$(J)
19300 J=J+1
19400 GOTO 19000
19500 L=J-1
19600 RESTORE:RETURN
19900 '
20000 ' Subroutine to RENAME Files
20100 '
20200 FOR K=0 TO L
20300 NAME D$(K) AS E$(K):IF C=1 THEN 20600
20400 PRINT D$(K);"------> ";E$(K)," Completed"
20500 A=A+1:F$(A)=D$(K)
20600 C=0
20700 NEXT:RETURN
21000 '
21100 ' Subroutine to Inform User of Number and Files RENAMED
and files
21200 ' NOT FOUND
21300 '
21400 CLS:PRINT "FILES RENAMED: "A," ","FILES NOT FOUND: "B
21500 PRINT
21600 FOR L=1 TO J STEP 2
21700 K=L+1:PRINT F$(L),F$(K)," ",G$(L),G$(K)
21800 NEXT:RETURN
22100 '
22200 '  Pause Subroutine
22300 '
22400 PRINT "Hit "R$" <ENTER> "N$" to Continue :";:INPUT V:RETURN
22700 '
22800 ' Error Subroutine for FILE NOT FOUND
22900 '
23000 IF ERR <>53 THEN 24600
23100 PRINT "File ";D$(K);" NOT on Disk"
23200 B=B+1:G$(B)=D$(K):C=1
23300 RESUME NEXT
23600 '
23700 ' Subroutine to Pad PROFILE File Name to Eight Characters
23800 '
23900 X=LEN(Z$):X1=8-X:X1$=STRING$(X1,"0")
24000 X$=Z$+X1$:Z$=X$:X=0:X1=0:RETURN
24300 '
24400 ' Error Subroutine for
24500 ' ALL Other Errors
24600 PRINT "ERROR Number ";ERR;" Occurred at Line Number
";ERL:PRINT
24700 PRINT "Do you wish to resume ":INPUT Y$
24800 IF Y$<>"N" THEN 23300
24900 END
65000 REM       * PROFILE is a trademark of Tandy Corporation *
```

RM/BAS

```
5 '"RM/BAS" BY Dwain Sutton
50 R$=CHR$(16):N$=CHR$(17)
100 CLS:PRINT @20," P R O F I L E (tm)  IV   U T I L I T I E S"
110
PRINT"_____"
_____"
115 PRINT @(4,30),R$" BASIC PROGRAMS "N$
120 PRINT @(6,20),"1 - Archive Records or Delete Empty Records"
130 PRINT @(7,20),"2 - Replace Field Data (with literals)"
140 PRINT @(8,20),"3 - Transfer Records ('Lookup' between two
files)
150 PRINT @(9,20),"4 - Rename Files"
190 PRINT @(14,20),"X - Exit to DOS"
195 PRINT @(21,0),STRING$(80,"_")
200 PRINT @(22,28),R$" < Enter Selection > "N$
300 K$=INKEY$ :IF K$="" THEN GOTO 300
310 IF K$="1" THEN RUN"SUBFIL/BAS"
320 IF K$="2" THEN RUN"REPL/BAS"
330 IF K$="3" THEN RUN"XFER/BAS"
340 IF K$="4" THEN RUN"RENAME/BAS"
500 IF K$="X" OR K$="x" THEN CLS: SYSTEM
510 GOTO 300
65000 REM       * PROFILE is a trademark of Tandy Corporation *
```

OPTIMIZE YOUR 80-TRACK DOUBLE SIDED DOUBLE DENSITY TRSDOS 6 SYSTEM DISKS

by S. K. Pramanik
Sarpsborgvej 56, 7600 Struer, Denmark

OPTTDOS6/JCL puts an optimized copy of TRSDOS 6 to a double-sided 80 track disk. This puts the directory on track 40, with the DOS and some other often used files around it. However, LDOS 5.x must be used (use a backup copy), which is patched as required to put files on the required tracks. The JCL file carries its own documentation. A warning - check the size of the files in the particular version of TRSDOS against the documentation, and if necessary, make some adjustments.

```
.OPTTDOS6/JCL for use with LDOS 5.4
.   by SKP - August 18, 1985.
.   to optimize LDOS 6.1.2 system diskette
.   for 80 track double sided, double density diskettes
.
PATCH SYS8/SYS.SYSTEM:0 (D00,FE=2E 23)
.   to start file allocation at track 35
.
backup /jcl:2 :1
.   on tracks 35.00-35.35 (reserve 6 grans)
.
PATCH SYS8/SYS.SYSTEM:0 (D00,FE=2E 24)
.   to start file allocation at track 36
.
copy backup/cmd.utility:2 :1
.   on track 36.00-36.29 (5 grans)
copy format/cmd.utility:2 :1
.   on track 36.30-37.11 (3 grans)
copy help/cmd.utility:2 :1
.   on track 37.12-37.23 (2 grans)
copy basic/ov1.basic:2 :1
.   on track 37.24-37.35 (2 grans)
.
backup SYS1/SYS:2 :1 (s)
.   if SYS1 is requested, we also get SYS10, 11, 12 & 13
.   sys  1 on track 38.00-38.05 (1 gran)
.   sys 10 on track 38.06-38.11 (1 gran)
.   sys 11 on track 38.12-38.17 (1 gran)
.   sys 12 on track 38.18-38.23 (1 gran)
.   sys 13 on track 38.24-38.29 (1 gran)
backup SYS2/SYS:2 :1 (s)
.   on track 38.30-38.35 (1 gran)
.
backup SYS0/SYS:2 :1 (s)
.   on track 39.00-39.23 (4 grans)
backup SYS3/SYS:2 :1 (s)
.   on track 39.24-39.29 (1 gran)
backup SYS4/SYS:2 :1 (s)
.   on track 39.30-39.35 (1 gran)
.
backup SYS6/SYS:2 :1 (s)
.   on track 41.00-42.17 (9 grans)
backup SYS7/SYS:2 :1 (s)
.   on track 42.18-43.11 (5 grans)
backup SYS8/SYS:2 :1 (s)
.   on track 43.12-44.17 (7 grans)
.
copy dos/hlp:2 :1
copy basic/cmd.basic:2 :1
.
PATCH SYS8/SYS.SYSTEM:0 (D00,FE=2E 00)
.   to restore file allocation from track 0
.
copy boot/sys.lsidos:2 :1
.   on track 00.00-00.17 (3 grans)
backup SYS5/SYS:2 :1 (s)
.   on track 00.18-00.23 (1 gran)
backup SYS9/SYS:2 :1 (s)
.   on track 00.24-00.29 (1 gran)
.
.   the rest
copy patch/cmd.utility:2 :1
copy conv/cmd.utility:2 :1
copy forms/flt.filter:2 :1
copy ksm/flt.filter:2 :1
copy log/cmd.utility:2 :1
copy repair/cmd.utility:2 :1
. click/flt
. com/dvr
. comm/cmd
. floppy/dct
. tape100/cmd
```

GUIDED TOUR THROUGH THE MODEL III TRSDOS DIRECTORY

[Reprinted from Data Important to Members Everywhere (DIME), the newsletter of the Northern Illinois Computer Owners League (NICOL).]

This article will describe and explain the important features of the Model III TRSDOS directory. You'll see that it is quite similar in most respects to the Model I directory except for the way in which the DOS SYSTEM files are handled.

As in the Model I, the directory is located on Track #17 (11H), and occupies the entire track. Because of the Model III's double density format, this amounts to 18,256 byte sectors. The first sector (#0) contains the Granule Allocation Table (GAT), the diskette Master Password encode, the diskette name and any AUTO command which may be active. Sector #1 contains the Hash Index Table (HIT) and the information by which TRSDOS is able to locate the SYSTEM files. The remaining sixteen sectors of the directory track contain the directory entries for the diskette's user files, five entries per sector.

Let's take a look at Track #17 in detail:

1. Sector #0 (the GAT sector) - The first 40 bytes (00H to 27H) make up the GAT and represent Tracks 0 to 39 on the diskette. Each byte is a "bit map" of the allocated granules in the corresponding track. Bits 0 through 5 represent the 6 granules of the track and bits 6 and 7 are not used (always 0). Thus, a fully allocated track would appear as a '3F' in the GAT. In binary this would be 00111111. If a byte is '25', a binary 00100101, it means that the first, third and sixth granules are allocated.

Bytes CEH and CFH of Sector #0 are the encoded Master Password for the diskette. This is normally D38F for "PASSWORD". Bytes D0H - DFH are the diskette's name in ASCII and the creation date in MM/DD/YY format. Bytes E0H - FFH are used to store any AUTO command which may have been set up for the diskette. If byte E0H is a '0D' (carriage return), no AUTO command is in effect.

2. Sector #1 (the HIT sector) - The first 80 bytes (00H to 4FH) make up the Hash Index Table (HIT). Each non-zero bytes is the hash code for an active file in the directory, and its position in the HIT indicates in which sector and where in the sector the directory entry is located. For example, on a standard Radio Shack TRSDOS diskette, the first two bytes are 'F0' and 'F4', which are the hash codes for BASIC/CMD and CONVERT/CMD. A glance at Sector #2 will confirm that these files are the first two entries in that sector. The Model III uses the same algorithm as the Model I for computing hash codes.

Bytes E0H - FDH are used to store the size and location of the TRSDOS SYSTEM files. There are 15 pairs for bytes and each pair represent one of the SYS files. The first byte of the pair contains the granule offset and the size of the file in granules. The second byte of the pair is the number of the track in HEX where the file starts. To see how this works, let's take a look at bytes E4H - E5H which are '4210'. The '42' is 01000010 in binary notation. Because the byte has two pieces of information embedded in it, we'll write it as 010 00010. The '010' is the granule offset from the start of the track which is also TWO. Remember that a granule is three sectors in the Model III. The second byte of the pair is '10' which means that the file starts on Track #16 (10H). Putting it all together, we see that the file starts on Track #16 (10H), beginning on Sector #6 (offset of TWO granules) and occupies TWO granules or six sectors. This particular file, by the way, is very similar to SYS2/SYS on the Model I. The fifteen pairs of bytes in this area represent the TRSDOS resident module and fourteen overlay modules making up the entire DOS. The sixteenth pair is presently 'FFFF' and is not in use, no doubt being reserved for a future overlay, like a spooler (just a guess). Since the user never calls the DOS modules directly, they need not have any names and the information in these byte pairs is all that TRSDOS needs in order to locate them when program execution calls for them.

3. Sectors #2 through #17 (Named directory entries) - There is room for five, three line (48 byte) directory entries on each of these sixteen sectors. This gives a total capacity of 80 named files. The sixteenth line on each sector is not used, so Radio Shack placed a '(c) 1980 Tandy' on it as a filler. Let's look at the first entry in Sector #2 to see how it is constructed. This is the entry for BASIC/CMD and it looks like this:

```
5E08 5000 0042 4153 4943 2020 2043 4D44
0000 EF5C 1200 0E06 FFFF FFFF FFFF FFFF
FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
```

The first byte, '5E' contains the file type, its visibility and protection level. In binary this is 01011110 and we'll write it as 0 1 0 1 1 110 to make it easier to break it down. Bits 0-2 are the protection level. In this case, it is 6 (EXECute only). Bit 3 is the visibility bit. A '1' means that the file is invisible, as in this case. If bit 4 is a '1', the file is an active file and has a corresponding HIT entry. Bit 5 is usually '0', but if it should be a '1', it means that the file has a backup limitation. Radio Shack has said in a newsletter that certain programs like Scripsit and VisiCalc will be limited to two backup copies. Whether this can be circumvented by setting bit 5 to '0', I have no idea. Might be worth trying. If bit 6 is a '1', the file is a SYSTEM file. If it is a '0', it is a USER file. If the Model III TRSDOS has been set up using the same methods as the Model I, bit 7 will be '0', for a Primary directory entry and will be a '1' for an EXTENDED directory entry. I don't know if this is true for the Model III. If not, a file will be limited to 12 extents. In any case, bit 7 is '0' for all my files.

The second and third bytes are simply the file creation month and year in HEX, in this case '08 50', meaning 08/80. The fourth byte shows the location of the EOF byte in the file's final sector. The fifth byte is the Logical Record Length of the file (00 means 256 bytes). The sixth through the sixteenth bytes are the file name in ASCII, in this example, BASIC CMD.

Bytes 17-18 and 19-20 are the UPDATE and ACCESS password encodes, respectively. For this file, we see '0000 EF5C'. 'EF5C' is the code for NO password, so BASIC/CMD's has an UPDATE password assigned but no access password.

Bytes 21-22 indicate the total number of sectors in the file in LSB-MSB format. So BASIC/CMD has 18 (12H) sectors.

The next 26 bytes are arranged in pairs and are used for the file EXTENTS. 'FFFF' indicates no further extents. The first byte is the number in HEX where the extent begins. In our example, this is '0E' meaning Track #14. The second byte of the pair contains the file's granule count and offset, encoded in exactly the same way as for the TRSDOS system modules. For BASIC/CMD, this byte is '06' which is 000 00110 in binary notation. Thus BASIC/CMD occupies 6 granules and starts at the beginning of the track (no offset).

SUPER SCRIPSIT

by Jack Bognuda

[Reprinted from BITS & BYTES, a publication of the TRS-80 SYSTEM-80 Computer Group, 16 Laver Street, MacGregor, Queensland 4109, Australia.]

Apparat issued ZAP Number 83 for zapping the Model I SUPERSCRIPSIT for TRSDOS to enable it to run with Model I NEWDOS/80. To enable this TRSDOS version to run under NEWDOS it is necessary to make the following changes:-

```
SCRIPSIT/CMD Sector 09, 1F
        change:
                                                C93A
        D9AB 4FCD F04A C021 004D 0123 00CB 1E38
        0104 CB1E 3801 040D 2320 F204 0528 0105
        2105 00CD 6666 0604 CD46 667D 3222 7EAF
        C900 0000

        to:
                                                C906
        0411 333C 21DC AC1A 7713 D630 FE0A 3003
        0102 005B 2310 F036 8400 0000 0605 CD66
        6606 04CD 4666 7D24 2528 023E FF32 227E
        AFC9 0000

SCR17/CTL Sector 00,31
        change:
        69FE 3038 F9FE 3430 F5CD 9E75 0E00 3271
        42CD 1944 3E0F CD33 0021

        to:
        6932 728D 216E 8DCD 1944 2805 F6C0 CD09
        4418 0644 4952 2030 0D21

SCR17/CTL Sector 02,A2
        change:  57 08CB 8F52  to:  57 4450 8D52
```

These Zaps include provision to enable you to read the directory. Although not listed in the text of the Menu which appears initially on SUPERSCRIPSIT, if you press "D", the directory will come up.

A COMPREHENSIVE NON-REVIEW

by Gary Bullard

[This article is reprinted from ECHO, the newsletter of the Green Country Computer Association.]

This will be a sad story of a quest that ended in failure. Yet the ending is not necessarily an unhappy one because the intrepid adventurer who wandered into unknown territory gained knowledge and became wiser. Sadder, but wiser.

The intrepid adventurer was me. Sure, many people think I am a computer expert. Just goes to show how many people I've managed to fool. While I may know all there is to know about some aspects of computer life (such as: Apples are better than Tandys †) there are still some areas where I am a novice.

[† NORTHERN BYTES editor's note - he didn't fool ME - I could tell he wasn't a computer expert, right off the bat!!!]

Telecommunications is one such area. Actually, I have owned a modem for several years. But I have only used it for local bulletin board systems and occasionally to connect with the computer where I work. Simple, easy, straightforward. A matter of doing remotely what I would be doing at a terminal in the same building.

It became time to expand my horizons. Commercial database services such as CompuServe and the Source have been around for years; it was time I tried them out. Most service companies offered a "Starter Kit," designed to get you logged on and introduced to their command protocols and services. Often these kits would offer enough "free connect time" to offset the cost of the kit, thus allowing you to sample the database's services before committing yourself to a subscription.

Before I could decide which one I wanted to try first, I got a call from Western Union. They were inaugurating a new service called EasyLink and they wanted to give me a free three month trial subscription. All I would have to pay for would be any service not directly related to connect time, such as telegrams or mail sent through the post office.

Sounded good, so I accepted. Soon I received a hefty package containing a directory of users, the reference manual and guidebook, and a password.

The directions for logging onto EasyLink were complicated, even for somebody used to reading IBM manuals. After a few unsuccessful attempts at logging on I finally got through. Then the real fun began.

First, the prompts were very cryptic. Instead of "READY", or "OK", or simply a cursor, I would see "PTS" on the screen. The book admitted that this meant something like, "Prepare To Send". And that was one of the more reasonable examples. Some of the commands were equally cryptic, if not more so. For instance, "XXXX" might mean "send this file". Very mnemonic, yes?

I explored the system a couple of times but got little out of it except frustration. The book was complicated enough that it was a few weeks before I attempted to log on the first time. That cost me.

Why did it cost me? Because Western Union sent an electronic letter to my electronic mailbox welcoming me to the EasyLink service. When I did not access my mailbox within a few days, they printed the mailbox's contents and sent them to me through the post office. Then they charged me for the service.

When I realized what they were doing I called and asked them to cancel my subscription immediately and stop writing and sending mail to me.

Apparently they did not believe I received the first letter so they sent another. And charged me for that one, too. So I wrote them a letter and sent a couple of copies to their brass hats explaining what I thought of their service, and asking again to be removed from their subscription files. They got the message this time and quit.

Maybe I'm a slow learner, but I figured there HAD to be something better than EasyLink. Every computer store and bookstore was selling a box labeled, "CompuServe Starter Kit." The box contained a user's manual, password, telephone numbers, and five "free" hours of connect time. Furthermore, according to the box, you could access almost any service that CompuServe offered during that five hours. If you liked what you saw, you could become a regular subscriber.

Unfortunately, those five free hours cost nearly forty dollars. Thrifty person (cheapskate) that I am, I waited until I found a box for only twenty-six dollars before I bought one.

The CompuServe manual is much easier to understand than the EasyLink manual. The commands, if not always obvious, are at least reasonable. The login protocol did not require a degree in cryptography. And the services sounded useful and fun. I was ready and eager to try it out, even though the kit limited me to 300 baud and evening hours.

Another rude awakening. When I logged into CompuServe I was met with a message informing me that I had to give them my credit card number or bank account number. It seemed an unreasonable request, since I had already paid for five hours of connect time, but the computer insisted. If I had any problem with that requirement I should call their customer service number.

So I did. The lady who answered the phone was very polite, but confirmed that until I provided a credit card number or my bank account number I would not get access to CompuServe. Never mind that I had paid for a "CompuServe Starter Kit" that promised five hours of connect time; they changed the rules. Now a user must subscribe first, THEN he will be allowed to use the five hours.

This posed a dilemma: I don't use credit cards and I refuse to let anybody else write checks on my bank account. I would be happy to pay any bills I run up, but I don't want any electronic transfers depleting my meager account when I am not looking. CompuServe insisted that they send a statement before they draw the money out of my account, but they would not send me a bill and let me pay that way.

It sounded like they want the deck stacked in their favor. If a mistake is made and they overcharge me, they will have the money and I will have to fight to get it back. The other way, if they overcharged me, I could withhold payment until the problem is straightened out.

Just out of curiosity I went to my bank and asked how the electronic fund transfer worked. I was told that they get a tape with the transactions. Their computer reads the tape and treats it as gospel. Whatever that tape says to deduct from my (or anybody's) account gets deducted. I asked the bank what happened if somebody overcharged? I was told that the money was deducted anyway, and I could fight the system to get it back. And what if my account didn't have enough money? Then I would be charged a $15 overdraft fee.

In other words, giving somebody your bank account number is like giving them a blank check. It is worse than a credit card number because you can stop payment on a credit card purchase, and thefts are limited to fifty dollars in most cases.

That meant that I was stuck with a "Starter Kit" that was useless and worthless. The only person who could use it would be somebody willing to give out his credit card number or bank account. I was simply out $26 unless I could find somebody to take it off my hands. Fortunately, I found somebody at the following GCCA meeting.

Twice burnt, thrice shy, right? I told you I was a slow learner. Maybe some of the big database services are unreasonable, but surely not all were?

I found another service called Software Express Videotex. Their starter kit only cost $15, but they only promised two hours of "free" connect time. They had fewer services than CompuServe, but their primary function was software distribution through telephone lines. You would choose the program you wanted, download it, then they would bill you for the purchase.

Fifteen dollars seemed a small gamble, so I sent my money off. Back came a user's manual, password, and a catalog of Apple and IBM software at very reasonable prices.

Oh, happy day! Finally I was going to get to play on a big database service. I immediately logged in and was met by a request for my credit card number or bank account number. Deja vu!

Enough is enough, I sold that package in much the same way I got rid of the CompuServe package. No more, until one of these companies begin billing for services rendered. There are too many hands in my pockets now for me to add another one.

Not that I suspect any of these companies of being dishonest, or having any deliberate intentions of cheating me. But I am not in the habit of paying my bills with blank checks and I do not plan to start. Even an honest mistake could be a nightmare with that system.

This non-review tells you nothing about EasyLink, CompuServe, or Software Express. It is only a sad tale of one fellow's unsuccessful attempt to explore some on-line computer services.

The moral of the story (if there is one) would be that if you are a person who does not use credit cards and who refuses to divulge your bank account number, don't buy any starter kits!

they are a waste of time and money, unless you get lucky and can sell them to somebody who can use them.

Some time in the near future I would like to write a review of as many Tulsa bulletin board systems as possible. Along with that article would be a list of bulletin boards and their numbers.

All Sysops who would like to be included in this compilation please let me know about your systems. Please leave me a message on GCCA's Computercenter at (245-3456), Green Country Computer Association Associates: GTABBS at (663-7305), and The Phantom at (585-9437), or TCS at (628-0662) and I will get it [these are in area code 918].

Specifically, I would like to know: the name of your board, the telephone number, baud rates supported, hours of operation, the kind of computer running the board, what BBS program is being used (and did you write it?), special features your board provides, and any comments you would like to add about your BBS.

All non-sysops are asked to leave a message to me describing your favorite bulletin boards, which features you like (or hate), and what features you would like that do not presently exist.

Any and all comments from sysops and users will be appreciated. The more responses I get, the sooner this compilation will be complete.

[NORTHERN BYTES editor's note: I assume that Gary would not mind receiving responses from users outside of the Tulsa area as well.

As for the billing problems, I concur completely with Gary's comments. I, also, am one who does not use a credit card (and neither does Charley Butler at The Alternate Source, which is one reason that TAS offers invoice billing of NORTHERN BYTES). I rather do resent the fact that some businesses feel that you are a non-person if you do not have a credit card. Well, I may be the last of the non-persons, but I'm just stubborn enough that if I ever need a rental car and nobody will rent me one because I don't have one of the magic cards, I'll just walk to the nearest highway and stick my thumb out (I've travelled many miles that way in my lifetime). These credit card companies are modern day robber barons, as they increase their interest rates while all other lending institutions are decreasing theirs. I admit there are occasional advantages to having a credit card (such as if you do a lot of ordering through the mail), but not enough for me to pay $18 a year for one. And even if I had one, I certainly wouldn't give the number to an online computer service, where a computer error (such a failure to notice that I have logged off the system until a couple of days later) could result in a unexpectedly high charge to my account.

For the same reason, I'm not about to authorize anyone to dip into my bank account (not that it would matter much, there's never any money there anyway). So what is the answer? I think that these big online services should be willing to check your credit rating when you sign up (they certainly have the resources to do it!) and if you're not a real deadbeat, they should be willing to extend you a credit limit of, say, $25 or $50. If you wanted more you could prepay to your account. Likewise, for those with poor credit ratings, they should be willing to accept prepayments to the account (this still leaves them in control of a portion of your money, but at least you know they can't electronically abscond with more than what you've deposited - and they're protected because they can shut you off if your prepaid account runs dry). At present, the only online service that I know of that offers an option of prepayment to an account is American PeopleLink - and even they don't seem to encourage that method of payment.

CompuServe temporarily shut The Alternate Source off recently because they had just discovered (somehow) that TAS was operating under Chapter 11 protection (which they have been for over a year now). Since TAS signed up with CompuServe AFTER the Chapter 11 went into effect (and since CompuServe is using the Electronic Funds Transfer to make sure that they get theirs first, anyway), there is not much likelihood that they are in any danger of not being paid. Nevertheless, even with the deck stacked in their favor, they felt it prudent to simply disable the account, without so much as the courtesy of a phone call to find out what the situation was. Admittedly, once things were straightened around (TAS had to call THEM to find out why the account wasn't working), they did apologize and did provide the free use of a demo account for a couple of days to make amends. But, it sure seems like they are awfully paranoid about not getting paid. Perhaps, in their rush to sign up the Big Bu$ine$$ u$er, they've forgotten about the personal computer users that helped make them what they are today - and there's still a lot of us that don't have credit cards!

And then there's the whole can of worms about privacy of Electronic Mail (or lack of it) on systems such as CompuServe. But I'll save that for another issue...]

MISCELLANEOUS HINTS AND TIPS
Condensed from old issues of NORTHERN BYTES

1. Did you ever wonder why, when you use a BASIC LPRINT statement to print numeric variables, a carriage return is "forced" after 132 characters or less (if you don't send one in your program)? If you're not familiar with this effect, type FOR X=1 TO 1000: LPRINT X;: NEXT from BASIC READY. If you have an 80 column printer, you'll see it print alternate full and partial lines. The reason for this is a CP 84H (84H=132 decimal) instruction found at 20D9H in the ROM. A similar effect occurs if you use a comma as a field separator (use a comma instead of a semicolon in the above demonstration to see the effect). In this case you will find that partial lines will be printed even if you use string variables. The culprit here is a CP 70H instruction found at 211EH in the ROM. If you have an 80 column printer and a Model 4P, you may wish to zap the MODELA/III ROM image with values more appropriate for your printer.

2. If you have <u>one</u> key on your keyboard that seems particularly bouncy (returns more than one character when you hit it) or that you sometimes have to hit two or three times before it "takes", try opening up your computer and re-soldering the contacts on the underside of the keyboard printed circuit board. I have had this problem with several of the keys on my early Model I keyboard. Apparently not enough solder was used during the automated soldering process, and after repeated use the solder connection tends to crack and make intermittent contact. The solution is simple - apply a little more solder (be sure to use a low-wattage soldering iron and good electronic type solder). Note that this advice may not apply to the later keyboards (I'm not sure how they were constructed), but it certainly is applicable to the original Model I keyboards!

3. Want a power on indicator lamp on your Model III? If you have some experience in electronics, you can wedge an NE-2 neon lamp between the bottom of the RESET switch and the mounting stud below it - it's said to be a perfect fit. Use a little extra wire and a dropping resistor, insulate everything thoroughly (remember this is 120 volts you're working with), and connect the lamp to AC power at the main power switch (just below the reset lamp). This will cause the reset button to glow a soft orange when the power is on. If you can't figure out how to do this from these instructions, you probably shouldn't attempt this yourself anyway - seek help from an experienced hardware hacker! [This hint extracted from an old issue of the Marin County TRS-80 Users Group (MCTUG) Newsletter.]

4. If you have a Model I with an LNW double density adapter (Doubler), you should be aware that a Z-80 HALT instruction will <u>not</u> reset this Doubler to single density - which means that in some cases the computer will not reboot properly (as when the BOOT command is issued from some DOSes). You can force the Doubler to change modes by loading a value of FEH (for single density) or FFH (for double density) to memory location 37ECH (14316 decimal). [Extracted from the LNW User Group Newsletter].

5. If you are internally configuring disk drives, you should be aware that the MX (or MUX) switch or shunt is <u>not</u> necessarily set the same way in all drives on the system. The problem is that some drive manufacturers define an open MX switch in the same way that other manufacturers define a closed MX switch. In my system, I have both Tandon and TEAC drives. The MX switch on the Tandon is OPEN while the same switch on the TEAC is CLOSED - and the system won't operate properly with any other combination.

On a related subject, if you are ever troubleshooting a Model I and find that it works fine with only one drive connected but refuses to operate with a second drive hooked up, you probably have the drive cable turned upside down, so that the "pins pulled" side of the connectors are on the wrong side of the drive edge connectors. However, if you configure the drives internally (setting the appropriate drive select switches in each drive), then you <u>can</u> use the drive cable turned upside down (connectors turned over at the expansion interface and each drive). Why would you want to do that? Well, it can save you the expense of buying a new drive cable if you add double sided drives to your system (which MUST be internally configured to operate properly).

HOW TO WASTE MONEY ON COMPUSERVE

by Jack Decker

It all started with the best of intentions. I had written an article suggesting a new protocol that would permit message-base sharing between Bulletin Board Systems (so that two BBS's on opposite ends of a major metropolitan area could offer a common message area that would be shared by both BBS's, for example. The protocol might also be extremely useful to Amateur (Packet) Radio operators (this does not mean I have any greater appreciation of the A.R.R.L., it just means the protocol might be useful in that situation).

Anyway, I figured that the article could stand a bit of critique before I put it in NORTHERN BYTES. Since my deadline was fast approaching, I wanted to put it out where I could get some more-or-less immediate feedback, and besides, I wanted it to be available to those who might be able to make use of it as quickly as possible. What better place to place it than on CompuServe, right? The great electronic utility that lets everybody send their thoughts to the nation - surely this was the way to go.

(As you read farther, keep in mind that I've been heavily into computers for over seven years, and into telecommunications on and off for the last four or five. I'd hate to think what an inexperienced user would think about the "user-friendliness" of CompuServe.)

What I wanted to do was to get the file where it could be read by as many people as might be interested in the subject. There is a telecommunications Special Interest Group (SIG) on CompuServe (it's operated by Micro-Systems Software, the DOSPLUS people), but I thought it might not hurt to also get the file into the telecommunications area of at least a few of the other popular SIGs - perhaps some of the machine specific ones, which would of course include at least one of the TRS-80 related SIGs.

Unfortunately, Howard Hughes didn't leave me a penny in his will, so I didn't want to spend all night uploading the same file over and over to the Data Libraries of the various SIGs at 1200 baud. Wouldn't it be easier, I reasoned, to upload it to the system once, and then copy it to wherever I wanted to put it. It's logical thinking like that that gets you into trouble. Ask Murphy.....

Now, there is another fly in the ointment here. The nearest packet network node to me here in the Great White North is in Marquette, Michigan, and it happens to be a Tymnet node. Tymnet is fairly efficient on straight ASCII transfers, but reminds me of molasses in January (at the North Pole) when doing XMODEM transfers. First a 256 byte block is transmitted, at a nice, fast rate of 1200 baud. Then it is transmitted through the packet network. Then you wait (and wait, and wait...) for an ACK byte to come back from CompuServe so that the next block can be transmitted. You're paying for 1200 baud service, and getting an effective baud rate of about 35, because each block has to be acknowledged before the next block can be sent - and that sometimes takes several seconds!!

Of course, I didn't know any better, so I proceeded to begin uploading my lengthy document to my personal file area. About 25 minutes later, the upload was complete. Let's see, that's 25 minutes of 1200 baud (ha!) use of CompuServe and 25 minutes on my long distance carrier...

Well, at least now I could zap the file over to all those Data Libraries, so that everyone could read my wonderful ideas that would revolutionize telecommunications <*>. Right? Guess again, byte breath!

[For those of you not into discussion-oriented Bulletin Boards, I should explain that the <*> symbol indicates that the writer is speaking with "tongue in cheek", which is what the symbol is supposed to represent. It usually indicates that the statement just made should not be taken at face value.]

As it turns out, there is a SUBmit command in the Data Library command list (it does not appear on the menu, but will appear if you type HELP instead of a Data Library menu item), which allows you to transfer files to the Data Library from the public file area. Now, I should have realized that "public file area" and "personal file area" are not the same thing. But, I had never seen a reference to the "public" file area elsewhere on the system. So, for a while, I tried to use SUBmit to transfer my wonderful prose to the TELECOMM data library. And, of course, it kept telling me it couldn't find the file (and that I should try again in 24 hours!). I didn't quite understand that, since the file was perfectly visible in my personal file area. Hmm, maybe I'd better hang up and switch to 300 baud...

Back into CompuServe and on about the third reading of the help text for SUBmit, the light dawned. Well, thought I, what is this public file area that they talk about? A little more investigation (and some lucky guessing) turned up the fact that it is now called ACCESS.

Okay, so I go to ACCESS, and find that it also has a SUBmit command. And, lo and behold, the purpose of the SUBmit command in ACCESS is supposed to be to transfer files to the public ACCESS area from your personal file area. Okay, so this is going to be a two step process. I SUBmit my file to ACCESS from my personal file area, then once it is in ACCESS, I SUBmit it to the data libraries of my choice.

Right. If you believe that, you probably also believe in Santa Claus and the Easter Bunny.

Well, I proceeded to SUBmit my file to ACCESS, and after a couple of false starts, got it over in just the way I wanted it (coded with the proper keywords, etc.). And I got a message telling me that the file would be available on ACCESS within 24 hours. This is CompuServe's version of "the check is in the mail", but at the time it did seem to explain the message I had received back in the data library, instructing me to wait 24 hours.

Nothing to do now but sign off, and wonder just how much time I've already blown. Tymnet disconnects just as CompuServe is about to tell me how long I've been on the system. I mutter something about how maybe it was against nature for Tymnet and CompuServe to marry, and go to bed. Then it hits me - I could have probably uploaded the file five times in the time I took uploading the file and the trying to figure out how to move it. Oh, well, at least tomorrow night I can start sending the file all over.

Yup. You guessed it - the file was nowhere to be seen on ACCESS. It acted as though it had never heard of that file.

Well, I think to myself, I can outsmart this system. Rather than trying to put the file in a Data Library, I'll just send it via EasyPlex (CompuServe's optimistic title for their electronic mail service) to each of the SIG System Operators (SYSOPs). But first, I have to know their CompuServe I.D. number. I go to the TELECOMM sig and spend several minutes there, trying to find the SYSOP's I.D. number. I finally find it. I disconnect and compose a message offline, explaining what is coming in the next message (my plan is to upload a "cover letter" first, then send the text file itself in a separate message). So, I compose the cover message, call back and upload it and send it via EasyPlex. Now, the moment we've all been waiting for - I tell CompuServe that I want to USE my text file from my personal file area for the next message. It cheerfully complies for a moment, then starts spewing out an endless stream of system warning messages telling me that the file is too long and has apparently overflowed EasyPlex's buffer. A multi-million dollar computer system and it can't even handle a message that almost fits into the memory of a TRS-80 Model I (with ALLWRITE! co-resident in memory, no less).

Now I have to send another message explaining the situation and requesting that my "cover letter" be ignored. Well, I think, maybe the system goofed. I go back to ACCESS and once again SUBmit my file to the public access area. Then I go to FEEDBACK (your chance to talk back to CompuServe and ask them how the heck to do something that should be simple, but isn't). I leave a message asking how to transfer the file from my personal file area to a Data Library of a SIG. Then I sign off and wonder how many times I could have uploaded the file in the time I was on tonight.

This is becoming an obsession with me, so the next day I telephone CompuServe on their "Customer Service" line. I have learned to have something to read handy while calling anybody's "Customer Service" line, because chances are you are going to be treated to several minutes of "elevator music" interspersed with obnoxious announcements promoting the company's services (all the time that you're getting madder and madder that you're having to wait, they're making you listen to commercials. This takes a lot of nerve. The guy that invented the machine that injects commercials into music-on-hold is probably related to the character that invented the parking meter). CompuServe's announcements informed me to have my user I.D. ready, and that all "snap-pak" expiration dates have been extended indefinitely. They could have told me this once and I wouldn't have minded, but unfortunately, they repeated it several times. I finished two chapters in the book I was reading, and vowed never to buy a "snap-pak".

Finally a customer service representative came on the line, and I gave him my simple request. How, I asked, do you copy a file from my personal file area into a SIG's Data Library. He said he'd go check with someone who had recently done that. One

chapter later, he came back and said that it couldn't be done. There was just no way.

"Well", I said, "how about if I SUBmit it to ACCESS and then SUBmit it from there to the SIG's Data Library? He didn't know anything about the SUBmit command. "Just go to any Data Library and type HELP", I said, "and you'll see it listed." "Oh", he said (after some searching through the system), "you type SUBmit and then upload your file." "No, UPLoad does that. Read the HELP file on SUBmit". At this point I could tell that I was more familiar with SUBmit than he was - at least I'd HEARD of it! Well, he agreed that my proposed method of going through ACCESS should work. But, I noted that there was not a tone of confidence in his voice as he said it.

Since the notice that the file would be online in 24 hours obviously could not be believed, I decided to wait 48 hours and try again. When I logged on, I had an EasyPlex message waiting. It was from CompuServe, and was a reply to my FEEDBACK message. It said, "While it is possible to transfer a file laterally from the Personal File Area to the EasyPlex area, it is not possible to do so from the Personal File Area to one of the Special Interest Groups data libraries." You did notice that they failed to mention the size limitation on file transfers to the EasyPlex area? Doesn't anybody send long letters anymore?

Three paragraphs of obviously canned text followed, giving the complete idiot's guide on how to add files to database libraries. Since I felt like a complete idiot at the time, I read it, but it didn't tell me anything I didn't already know.

Hope springs eternal. I go to ACCESS, hoping that just maybe my file has been transferred there. Need I even bother to mention what I didn't find? I do a quick scan and note that there is nothing on ACCESS newer than three days old. So much for "within 24 hours."

I give up. I've been licked. I decide I am not going to keep calling back, day after day, waiting for "them" to decide to move my file. I go to the TELECOMM library and do a straight ASCII upload (no error checking/correction, but who cares? At this point I just want to get it uploaded). The upload takes about 10 minutes, including the time I spend manually typing in the keywords and description of the file. I am ready to cry. Or maybe to move in the middle of the night, and not give CompuServe my forwarding address...

A few days later I check in once more, and find that the file is not in ACCESS nor in the TELECOMM Data Library, and I have no idea why. At this point my feelings turn to disgust. It's no wonder that new users have such a hard time figuring out the system.

At least I can justify the time spent as not being a total waste, since it forms the basis for a newsletter article that will serve to 1) warn others that you can't move a file from your personal file area to a Data Library on CompuServe, and 2) perhaps provide a touch of humor in this otherwise dreary publication. Who am I kidding? Nobody else would have been dumb enough to try to do such an apparently logical thing, and I could dig out my exchange newsletter files and reprint the 992 variations on "Murphy's Laws" that have appeared in those publications if I wanted to inject some humor, for a lot less money.

It occurs to me that with my old TRS-80 Model I, I can copy files from any file area to any other (throw out that bum who said "but there's only one file area to start with!"). I can even copy from one disk to another! You know, I'll bet that even an IBM or an Apple or a Commodore can do the same thing! What is CompuServe running - an EINAC?

Ironically, my proposed protocol (you know - the one I've been trying to get into the Data Library throughout this article!), if widely implemented, would give some of the advantages of belonging to a service like CompuServe to Bulletin Board System users. Judging from my experiences in trying to copy one lousy text file, I'd say such a system is long overdue.

Say, you don't suppose that someone at CompuServe READ my file, and decided..... naw, couldn't be.....

WHAT GOES ON HERE?

[This article is reprinted from the Milwaukee Area TRS-80 User's Group (M.A.T.U.G.) newsletter.]

There doesn't appear to be anything special or particularly out of the ordinary about the BASIC program given in the listing following. When I run it in Model III BASIC, it produces an output of 17 triples of integers, supposedly all of the solutions there are for the selected range of the variable Z, disregarding obvious interchanges of the variables X and Y.

When I run it in every version of TRSDOS 6 and DOSPLUS IV BASIC that I have, however, three of the 17 solutions are not calculated or displayed.

I have tried to revise the listing so that it will run satisfactorily in the Model 4 using essentially the same approach as the original, but I have not succeeded. What is so unusual about this seemingly prosaic program?

By itself, this example is not of great significance. Yet I can't help thinking that there are many other subtle traps out there waiting to be sprung on us, when the results may not be so innocuous.

This is another instance of the necessity of subjecting a program to a severe shake-down cruise before accepting it as being free of bugs. Even then, can we be sure it is correct? We have not yet dispelled "The Illusion of Precise Computing".

```
100 REM: Solutions of X*X+Y*Y=Z*Z*Z in integers for Z in the
range 1-20
110 U$="(## ## ##)                    " : REM 22 blank spaces
for Model III, 30 blank spaces for Model 4
120 FOR Z = 1 TO 20
130 M = Z * Z * Z
140 FOR X = 0 TO M
150 N = Z * Z * Z - X * X
160 IF N < M/2 THEN 210
170 Y = INT(10000 * SQR(N)) / 10000
180 IF Y - INT(Y) > .00001 THEN 200
190 PRINT USING U$; X, Y, Z;
200 NEXT X
210 NEXT Z
220 END
```

CONTROL YOUR CRT CONTROLLER

by Mike Dinn

[This article is reprinted from the Canberra Micro-80 Users' Group Inc. newsletter, 113 Owen Dixon Drive, Evatt ACT 2617, Australia.]

In the good old days when Model I and System 80 were state of the art, we seemed to have a lot of "hackers" around. The word seemed to refer to those who enjoyed digging into their into their hardware and/or software almost for the sake of it. The way that "hacker" is now used by the media seems to have changed the definition somewhat, but irrespective of title I still get great satisfaction out of doing the same sort of things with my model 4P as I did with my System 80, LNW expansion interface, Omikron mapper and LNW doubler. The possibilities are not quite as broad, but if the Editors bully me enough, I'll try and write an occasional series on some of the things you can do with your Model 4 so that we can get more people interested and contributing. Here is one on the Model 4's CRT controller.

The Motorola 6845 is a 40 pin programmable CRT Controller. By loading its registers, you can control such things as the number of characters in a row, character lines on the screen, distance between lines, hardware scrolling, hardware cursor and react to the position of a light pen. The Model 4 uses what is called a 68045 which is a "hard-wired" version of the 6845, and only allows the hardware scroll option. Selection between 80 by 24 and 64 by 16 is done by toggling pin 3 which is normally used for light pen input. Other than this the 6845 and 68045 are pin compatible, and therefore the obvious thing for a hacker to do is try a 6845. In good hacking tradition, though, this is not quite straight forward, but I suggest that is part of the fun and challenge.

The first problem is that the BOOT ROM carefully clears all of the CRTC registers to zero, so that you end up with no video display at all. It should be possible to modify the ROM to load the registers but I'm not at that point yet. However, all is not lost. It turns out that the TRSDOS boot sequence DOES load the registers. The values controlling the lines per page can be readily zapped in Track 0, Sector 0 from 18 hex (24 decimal) to 19 hex (25 decimal) and lo and behold, you have 25 lines displayed. Scan lines per row can be changed from 8 to 11 so that the screen is much more readable.

I'll give details in the next article on how to do this, and how to cope with DOSPLUS (which does not load registers at Boot time), Model III NEWDOS/80 64 by 16 mode, and CP/M mode. Meanwhile, for those who wish to join me, buy yourself a 6845 (should be less than $10), and try to get hold of "The CRT Controller Handbook" by Gerry Kane, Osborne/McGraw-Hill, 1980.

PROPOSAL FOR A COMMON MESSAGE BASE PROTOCOL
by Jack Decker

[PLEASE NOTE: This is preliminary version 0.0 of this proposal. At this point I am welcoming suggestions and comments for a final version of this protocol.]

1. INTRODUCTION

It seems to me that there ought to be a way to share message bases among separate computer Bulletin Board Systems that share common interests. The primary use of message base sharing would be in large metropolitan areas, where suburban areas may be able to place local calls to the central city but not to other suburbs, and conversely, the local calling areas of suburban areas may reach out to include areas that are not part of the local calling area of the central city. As an example, consider the following diagram:

Suppose that, in this example, the local calling area of the central city includes both suburbs "B" and "C", but not the more distant suburbs "A" and "D". The local calling area of suburb "B" includes the central city and suburb "A", but not suburbs "D" and "E" on the far side of the city. Similarly, the local calling area of suburb "C" includes the central city and suburb "E", but not suburbs "A" and "B". It is obvious that in this situation, no one BBS will be able to offer service to the entire metropolitan area, at least not without installing expensive call-forwarding lines in suburbs "B" and "D" (and perhaps other suburbs in a north-south direction).

However, if several BBS's were capable of sharing a common message base, it would be possible to have full interaction between all users in the area, but each BBS user would place a call to a BBS located in his local calling area.

Besides all of the obvious reasons for offering this type of service, the offering of a common message base among several BBS's can actually help increase usage on all of the BBS's participating in the system. It seems that one reason for the high failure rate of BBS's is that users expect to be able to interact with a large number of other users. If a board can provide a large number of new messages each time a user calls in, it tends to hold the interest of the board's users.

Note that this "sharing" of message bases is most practical where there are no toll charges incurred for calls between the host BBS systems. This is due to the large amount of data that needs to be transmitted. However, this type of message base sharing can be practical in any of the following situations:

a) Between affiliated BBS's within a metropolitan area, as stated above.

b) Between BBS's that can interconnect through a packet network, such as GTE Telenet's "PC Pursuit" service.

c) Between BBS's connected by private communications circuits, such as BBS's within a company interconnected through the company's own leased circuits, or via low cost (i.e. WATS) telephone lines.

d) Between BBS's interconnected via amateur (packet) radio.

e) To a lesser extent, between BBS's interconnected via normal long-distance telephone lines, if a very high-speed modem is used.

f) To a lesser extent, between BBS's that can utilize an electronic mail service or information utility for delivery of the message data, at a lower price than that of a regular long distance telephone call between the two points.

There are probably other situations where message-base sharing can be useful - your comments are welcome.

2. WHAT THIS PROTOCOL IS DESIGNED TO DO

This protocol specifies standard batch header and message header formats for sending messages from one system to another. Any BBS that utilizes this protocol should be able to send its message base to any other BBS using the same protocol.

The protocol is designed to be easy to implement and to transfer a minimum amount of necessary information for message base sharing (while leaving room for future expansion). It is designed to take a batch of messages to be transferred and send them out in a standard format, via any available transmission method including electronic mail systems.

3. WHAT THIS PROTOCOL IS NOT

It is NOT an interactive protocol, but rather a one-way protocol that requires no response from the receiving BBS. This allows a batch of messages to be sent from one computer to several others at the same time, as via amateur radio or as a file left on an information utility. It is designed for one-way transmission. With proper programming, two computers could simultaneously transmit their message bases to each other via phone lines, though no error detection/correction would be possible in this case.

It is NOT a file transfer protocol. However, the file created for transmission using this protocol can be sent using XMODEM or some other error correcting file transfer protocol (and this is RECOMMENDED where possible).

It is NOT a text compression protocol, though it is certainly recommended that if transmission time is a factor, the file created for transmission be SQUeezed before sending and UNSQUeezed upon receipt at the destination system(s).

It does NOT specify formatting of individual messages. Some BBS systems allow "file mail", which permits one user to send a binary program file or similar type of file to another user, and this would not be possible if constraints are placed upon line length, allowable characters, etc. The protocol itself uses only ASCII characters in the ASCII range 32-127, but it is up to the source and destination systems to agree upon allowable characters within individual messages. When sending a batch of messages to an unknown system, it is suggested that a limitation of using only printable ASCII characters be employed.

4. SYSTEM IDENTIFICATION

Each BBS participating in a message base interchange must be uniquely identified, so that the origin and destination of a message can be properly determined. I propose the use of an eight character code to uniquely identify each BBS participating in the system. Although users of this protocol may in fact specify any code they like to uniquely identify a participating BBS, it is suggested that the following conventions be adhered to in choosing the unique code to identify each BBS:

a) For BBS's connected to dial-up telephone lines in the U.S.A. or Canada, the I.D. code will be:

1+AREA CODE+FIRST 3 DIGITS OF PHONE NUMBER+1 LETTER (A-Z)

Example: A BBS with the telephone number (705) 253-5366 would use the I.D. code 1705253A - note that the first (or only) BBS in a given area code and exchange would use the letter A as the final character of the I.D. Any subsequent BBS's that begin operation in the same telephone area code and exchange would use the suffix B, C, D, etc. Since this scheme permits up to 26 separate BBS's in a given telephone exchange to be identified, the chance of running out of suffix letters is extremely small.

b) For BBS's connected to dial-up telephone lines outside of the U.S.A. or Canada, the I.D. code will be:

COUNTRY CODE+CITY CODE (OMIT LEADING ZERO)+FIRST DIGITS OF PHONE NUMBER (to give total 7 digits)+1 LETTER (A-Z)

Example: A BBS in Sydney, Australia which gives out the number (02) 332-2494 (where 02 is the city code) would use the I.D. code 6123322A, where:

61 is the country code,
2 is the city code, minus the leading zero
3322 is the first digits of the phone number, to give 7 digits total,
A is the suffix as given under "a" above.

Note that U.S. and Canadian BBS's actually follow this scheme also, since "1" is the country code for these countries.

c) For BBS's connected via amateur (packet) radio, the I.D. code will be the operator's call sign, truncated to eight characters if necessary, or padded with spaces at the right if less than eight characters.

d) For BBS's that do not directly connect to telephone lines, but rather offer connections through an intermediary (such as a packet switching network), any unique, eight character code (preferably a mnemonic for the system name) may be used, provided it is not already in use. However, the code should not resemble the codes used by dial-up boards or amateur radio operators (preferably they should be all letters, no numbers). Since this would likely be the situation with only a very few systems, I do not anticipate problems in leaving this situation relatively undefined.

5. MESSAGE "PASS-THRU" CAPABILITY

It should be noted that one of the strengths of this protocol is that it is designed to facilitate "pass-thru" of messages from one system to another. This means that a message or batch of messages may be propagated throughout a network of BBS's by "passing through" other systems. As an example of how this might work, please refer back to the diagram in (1) above. Suppose that a user in Suburb "A" wishes to send a message to a user located in Suburb "D", and let's further suppose that there are BBS's in the central city and in suburbs "B" and "C". One can easily see where it might be desirable for the sender to upload his message to the suburb "B" BBS, which would in turn upload it to the central city BBS, which would in turn upload it to the suburb "C" BBS, where it could then be downloaded by the recipient in suburb "D".

It is my understanding that most BBS programs that have message transfer capability (e.g. FIDO and the first version of SYSLINK) do not have the capability to pass messages through an intermediate system, even when doing so would avoid a telephone toll charge as in the example above. With this protocol, individual messages or an entire batch of messages can be forwarded through as many systems as necessary.

6. BATCH HEADER

When a batch of messages are to be sent from one system to another, it shall be preceded by a 256 byte "batch header" which will give certain information that the receiving system may need, as follows:

INFORMATION	BYTE COUNT
Originating BBS I.D.	8
Destination System Password	16
Batch Originator I.D.	8
Batch Origination Year	2
Month	2
Day	2
Hour	2
Minute	2
Second	2
Private (letter "P") or Public (space character)	1
Reserved for future definition	51
Current destination ID (of BBS receiving this batch)	8
Additional routing ID's	120

Here is an explanation of each item, in turn:

a) Originating BBS I.D. - The eight character I.D. of the BBS that originated this call. If a batch is "passed thru" more than one system, this I.D. will change as each new BBS replaces the batch I.D. with its own.

b) Destination System Password - If desired, the receiving system may issue passwords of up to 16 characters to all systems from which it expects to receive data. The password would then be inserted here. Note that this will also change as a batch is "passed thru" various systems. Items a) and b) form a unique combination that identify the calling BBS to the receiving system, and verify that the calling BBS has a right to leave messages.

c) Batch Originator I.D. - This will be the same as a) except when a batch is "passed thru" more than one system. In that case, this field is NOT changed as it passes through each BBS, thus it retains the I.D. of the BBS that FIRST originated the batch.

d) Date and time fields - Two characters each for year, month, day, hour, minute, second. This will normally be the batch creation time, i.e, the time at which the unsent messages are collected into a batch in preparation to be sent. These do not change as a batch is "passed thru" systems. All times should be given in GMT (Greenwich Mean Time, also known as Coordinated Universal Time) which can then be converted to local time by the receiving system, if desired.

e) Private or Public - if this field is blank (contains an ASCII space character) then the batch is considered public and all public messages therein may be added to the message base of any boards that the batch "passes thru". If a letter "P" is found in this field, the batch is private and intended for the destination BBS only, not any intermediary BBS's. Although I cannot imagine why anyone would want to create a private batch, I have provided a field definition for the purpose.

e) Reserved for future definition. Suggestions are welcome!

f) Non-defined - This field is intentionally left open for individual systems to use as they please. This could contain, for example, a suggested filename under which to save this batch file, or the name of the message board that this batch of messages came from. Receiving systems are under no obligation to honor anything placed in this field, but may do so by mutual agreement between sending and receiving BBS's.

g) First destination I.D. - The I.D. of the system to which this batch is being sent, provided for verification purposes.

h) Second through Sixteenth destination I.D.'s - If this entire batch is to be "passed thru" several systems, the originating BBS will create a routing list for the batch, starting with the first BBS to which the batch is to be sent (which will be placed in (g) above). The subsequent BBS's that are to receive the batch are then listed in order. After the final entry, any leftover fields should contain all space characters. As each BBS receives the batch, it will "move up" the routing list so that the First Destination I.D. is dropped and the Second Destination I.D. becomes the new First Destination I.D., etc. (it will also change items (a) and (b) before passing the batch along, and if the batch is public, it may add this batch to its own message base before passing the messages along).

If the Sixteenth Destination I.D. contains eight asterisks ("********"), this means that an additional 256 byte sector of routing information follows. This will allow up to 32 more routing I.D.'s. If that second sector also ended with eight asterisks, a third routing sector would follow (and so on). I cannot imagine routing a batch of messages through even the 16 destinations permitted in the first sector, but have provided the capability to define as many sectors of routing information as needed.

It is the responsibility of each receiving BBS to "look down" the routing list and to send the batch along to ALL BBS's that are within the local calling area of the BBS, It must adjust the routing list accordingly when doing this. For example, suppose that it looks down the list and finds that the Second, Seventh, and Tenth Destination I.D.'s are within its local calling area. It should then send the batch to the Second Destination I.D. (with the Third through Sixth Destination I.D.'s left in the routing list and all others removed); to the Seventh Destination I.D. (with the Eighth and Ninth Destination I.D.'s left in the routing list), and to the Tenth Destination I.D. (with only any Destination I.D.'s after the Tenth remaining in the routing list).

Once again, please note that when a message is "passed thru" various systems, ONLY items (a), (b), (g), and (h) of the batch header are modified before sending the batch along.

It should be mentioned that there may be times when a batch cannot be routed properly. This would happen when a receiving BBS looks at the next destination I.D. and either does not recognize the I.D. as valid, or is unable to complete the connection to the destination BBS. Those writing BBS programs to utilize this protocol may wish to implement the automatic sending of an "unable to deliver batch" type message when this occurs, that would go back to both the batch originator and to the system from which the batch was received (items (a) and (c)). However, since this protocol is designed to be as simple to implement as possible, the actual handling of such an event will be left to the individual BBS operators.

Finally, note that in the batch routing procedure, there may be times that it will be desirable to specify that a batch is to be delivered to all BBS's on a list kept on file at a distant system (for example, a batch might be delivered to a central city BBS, with instructions to deliver the batch to all BBS's on a list that includes all of the suburban BBS's). This would be by agreement between systems only, but for the sake of uniformity, I suggest that such list names begin with a single asterisk (*) to distinguish them from a BBS I.D. code.

7. INDIVIDUAL MESSAGE HEADER

Each individual message will be preceded by a 256 byte "message header" which will give certain information that the receiving system may need, as follows:

INFORMATION	BYTE COUNT
Originating BBS I.D.	8
Originating BBS message number	8
Originating User I.D.	32
Message creation Year	2
Month	2
Day	2
Hour	2
Minute	2
Second	2
Private (letter "P") or Public (space character)	1

Message File Type (T=Text, B=Binary)	1
Message Length (number of bytes, in decimal)	8
Reserved for future definition	10
Reply to: message number	8
Addressee System I.D.	8
Addressee User I.D.	32
Message Subject Line	64
Non-defined (for individual BBS use)	64

Here is an explanation of each item, in turn:

a) Originating BBS I.D. - The eight character I.D. of the BBS that originated this message. This does not change, even if a batch is "passed thru" more than one system.

b) Originating BBS Message Number - The message number of the message on the BBS that originated the message. This is required for replies.

c) Originating User I.D. - This is the sender's name or I.D. number, as it is known on his home system. Required for replies.

d) Message Creation Date and time fields - Two characters each for year, month, day, hour, minute, second. All times should be given in GMT (Greenwich Mean Time, also known as Coordinated Universal Time) which can (and probably should) be converted to local time by the receiving system, if desired.

e) Private or Public - if this field is blank (contains an ASCII space character) then the message is considered public and may be added to the public message area of the destination BBS (and any BBS's that the message batch "passed thru", if the batch was public). If a letter "P" is found in this field, the message is private and should be displayed only for the designated recipient.

f) Message File Type - an actual message will show the letter "T" for text. If a binary file (a program, etc.) is sent as mail, then this field will contain a "B" and the recipient should probably be offered the option of downloading the "message" using an error-correcting protocol such as XMODEM. This is included for systems that support "File Mail" (e.g. SYSLINK). Other codes in this field may be supported at a later time.

g) Message length - an ASCII number giving the actual length in bytes of the message to follow. The receiving system should expect another message header to immediately follow the message, except where the end of file is reached.

h) Reserved for future definition. Suggestions are welcome!

i) Reply to: message number - If this message is a reply to a previous message, then the number of the previous message will be shown in this field (otherwise it should contain all spaces). Note that this should be the message number on the Addressee's home BBS, which will be found in the (b) field of the original message.

j) Addressee System I.D. - The I.D. of the system to which this message is to be sent. For replies to messages, this will come from the (a) field of the original message.

k) Addressee User I.D. - This is the recipient's name or I.D. number, as it is known on his home system.

l) Message Subject Line - The subject of the message, as entered by the sender. On replies to messages, the sender should be prompted as to whether he wants to keep the original subject or enter a new one (this is a major deficiency of many BBS programs, in that after a few back-and-forth exchanges of messages, the named subject often has nothing to do with the actual message contents!).

m) Non-defined - This field is intentionally left open for individual systems to use as they please. This could contain, for example, a suggested filename for binary files, or an extension of the message field. Receiving systems are under no obligation to honor anything placed in this field, but may do so by mutual agreement between sending and receiving BBS's.

8. INDIVIDUAL MESSAGE FORMAT

This is not specified due to the desire to accommodate "File Mail", or the transfer of binary files. The receiving BBS should be able to take an incoming text message and add any necessary carriage returns or linefeeds required for text formatting. This protocol will not enforce mandatory carriage returns, linefeeds, or text formatting of any kind. The receiving BBS should also be able to split an incoming message into multiple messages, if the originating BBS allows a longer message size than the receiving BBS.

Note that nothing prevents users of this protocol from setting their own standards for text formatting. However, if a system accepts incoming batches from unknown systems, it should be prepared to deal with unformatted text files and binary files.

9. END OF MESSAGES (END OF BATCH FILE)

This can be indicated either by the end of the batch file (as detected by the file transfer program used), or by a message header containing only 256 space (ASCII 32 decimal) characters. The dummy message header containing all spaces should always be used by the batch creator, since some file transfer utility programs may not give an accurate end-of-file mark.

10. BATCH COMBINING

When a system receives a batch that is to be "passed thru" to another system, nothing in this protocol prevents it from combining that batch with any other batches destined for that system that have exactly the same routing. This is why the originating BBS I.D. is shown in each message header. Combining batches may be advantageous when transmitting using file transfer methods that do not permit multiple files to be transmitted in one batch.

11. SUGGESTED FORMAT FOR AUTOMATED BATCH TRANSFER

As previously mentioned, this protocol is designed to be general enough that it can be used for batch transfer via many different means of communication, including those that are inherently one-way. However, for normal computer-to-computer batch transfer over dial-up phone lines, the following procedure is suggested:

a) Originating computer prepares batch for sending by selecting all "NEW" messages (all messages not previously sent to the destination BBS) and combining them into a file using this protocol. If possible (and the receiving system can handle it), the file should then be "SQUeezed" before sending. Presumably, the originating BBS would maintain a table of BBS's with which it exchanges message bases, which would include such data as the BBS phone number (which would include any carrier access codes necessary to reach the BBS via an alternate long distance carrier), the baud rate at which the BBS operates, whether the BBS accepts SQUeezed files, the time at which calls to this BBS should be placed (necessary if toll calls are involved!), and the message number of the last message sent to this BBS.

b) Originating computer configures RS-232 for proper baud rate, etc. and dials recipient computer. When connection is made, originator sends carriage return characters at rate of approximately one per second until an incoming data flow is sensed. Originator waits for data flow to stop (assumes user I.D. prompt) and transmits its System I.D. code (and a carriage return). It then waits for the data flow to again start and stop (assumes a password prompt) and transmits a password it has stored for the system.

c) At this point recipient computer should be able to tell that it is connected to another computer, so it sends an 05H (ENQ) byte to enquire as to what type of batch file is being sent. Originator sends one of two bytes: An 06H (ACK) to indicate that it is about to send a normal batch file, or a 1FH (US) to indicate that a SQUeezed file is being sent (which the recipient computer will need to UNSQUeeze after the call is completed). If recipient computer receives any other code, it will send another 05H byte and wait for a valid response.

d) Upon receiving the 06H or 1FH byte, the recipient computer transmits an 02H (STX) and goes to XMODEM (or upward-compatible version thereof) protocol to begin first batch file transfer. The first batch file is transmitted using XMODEM and saved to a filename selected by the recipient computer.

e) Once the first batch file has been transferred successfully, the recipient computer send another 05H (ENQ) byte to enquire as to whether another batch file is to be transmitted. If originator has another file, it sends an 06H (ACK) or 1FH (US) to acknowledge that another file (normal or squeezed, respectively) is to be sent. Recipient then transmits an 02H (STX) and switches back to XMODEM to receive the next file. Note that more than one batch file may be transmitted in a session, to allow batches that have been "passed thru" from other BBS's to all be transmitted in the same session with the batch from the originator BBS.

f) When originator has no more batch files to send, it transmits a 15H (NAK) in response to the 05H (ENQ) byte sent by the recipient. It then transmits either an 04H (EOT) byte to signal the end of transmission (and hangs up), or, more commonly, it will send an 02H (STX) byte to state that it will receive any batch files that the recipient has for it.

g) If the 02H byte is sent, the recipient will either respond with a 04H (EOT), which means it has nothing to send and wishes to end the transmission (and hangs up), or it responds with an 06H

(ACK) or 1FH (US) to acknowledge that it is about to send a file. The roles of originator and recipient are then reversed, the new recipient (former originator) sends another 02H byte when actually ready to begin XMODEM transfer of the file. At the end of the file transfer, everything proceeds as in (e) above (remember the roles of originator and recipient have been reversed at this point). Of course, when the (new) originator has no more batch files to send, it MUST end the transmission by sending the 04H (EOT) byte and hanging up.

NOTE: If a file is so badly garbled in transmission that XMODEM gives up and times out, it is probably best for the timed-out computer to simply drop carrier. If the lines are that bad, there's no guarantee that any control bytes sent would be received accurately. A retry of the transmission may then be made.

Once the batch file(s) have been received and the connection is broken, the BBS should automatically switch to a utility program that a) UNSQUeezes the batch file, if necessary, b) adds the messages from the batch file to the message base on the BBS (except where the batch file and/or individual messages are private and therefore, are not to be added), and c) determines whether the batch file is to be "passed thru" to any other systems, and takes appropriate action if so.

Note that messages imported from another system will probably not have a compatible message numbering scheme, thus those messages will need to carry two message numbers - the current number on the system, for "new message" reading purposes, and the original message number on the originating BBS (for reply purposes).

It is highly recommended that message headers displayed to the BBS users display the originating BBS I.D. number and the originating BBS message number on all "imported" messages. Display of the local BBS message number is also recommended. Really sophisticated BBS software will take the trouble to reconstruct all reply chains when a new batch of messages have been received. This will not be especially easy for the BBS programmer, but all of the necessary information to reconstruct message threads IS present in the message headers.

12. DATA FORMAT FOR HEADERS

Numeric data in batch and message headers should be right-justified, with leading spaces or zeroes. All other data should be left-justified and padded with spaces at the right, if necessary. If a given data item is not available or not used, it should be filled with space (32 decimal) characters.

13. CONCLUSION (AND AUTHOR'S SOAPBOX)

I hope that, at some point in time, message base sharing becomes a reality for BBS users. I also hope that computer users will wake up to the fact that phone companies are planning on charging for all local calls, which means there will be no such thing as a "free" BBS. The public deserves access to a portion of the radio spectrum that is specifically earmarked for computer-to-computer transmissions, and that is NOT controlled by either commercial interests (such as the telephone companies) or by the amateur radio fraternity (the powers that be in amateur radio seem to feel that THEY had to learn the antiquated Morse code, so the rest of us should be made to suffer in a similar fashion. No, thanks). Donald L. Stoner, W6TNS has filed a proposal with the FCC for the creation of a "Public Digital Radio Service" (the number given to this proposal by the FCC is RM-5241). I suggest that those interested in computer-to-computer communications take the time to learn about and support this proposal (Don Stoner's address is 6014 East Mercer Way, Mercer Island, Washington 98040). A report on this proposal was printed in NORTHERN BYTES Volume 7, Number 1.

PRINTER HEAD CLEANING

by Eddie Reddish
12 Pooley Street, Pakuranga, Auckland, New Zealand

My printer was starting to display the lack of attention that I was giving to it. I had always just left it to itself and only occasionally dusted it. I noticed that when I started it from 'cold' that the first character printed was never clear. Most times only some of the pins would 'fire'. This was getting to be rather frustrating as it was spoiling the printouts and usually meant that I would have to print another copy.

I thought that what was happening was that the ink from the ribbon was causing the pins to stick inside the printhead. In my printer, Star Gemini 10X, there are 9 pins arranged vertically. These pins are very close together. All other dot matrix printers have a similar arrangement. The problem was: How do you clean the print head?

All print heads that I have seen are held to the head carriage by two screws. The print head has a flexible lead connecting it to the printer's circuit board.

METHOD OF CLEANING:

1. Write a small program as follows:

```
10 LPRINTTAB(0)CHR$(73)CHR$(106)  'Exercise all pins - prints
capital "I" and small "j".
20 GOTO 10
```

2. With the printer power OFF, move the print head about one quarter of its length of travel from the left hand side. This will enable you ready access to the two screws holding the print head to its carriage. Undo and remove the two screws. Carefully lift the print head away from its carriage, but leave the connecting cable plugged into its position in the printer's circuit board. You will find as you look at the face of the print head that a considerable amount of 'gunge' will have attached itself to the print face. Using a lint free cloth soaked in ISOPROPYL ALCOHOL wipe the face of the print head.

3. Having done the easy bit the problem remains of how to clean the inner part of the print head. A method of working the alcohol into the print head and flushing out the 'gunge' is as follows:

(a) In the bottom of a shallow plastic lid place 3 or 4 folds of paper. Cover this paper with ISOPROPYL ALCOHOL to a depth of, say, 2 to 3 mm. [approximately 1/8"].

(b) Place the print head 'nose down' into the plastic lid, holding it just clear of the bottom (a small stand is recommended - the head should not be resting on the bottom because the pins have to have room to emerge from the head). The paper serves as a last resort safety buffer.

(c) Turn the printer ON.

(d) RUN the above program. The pins should fire continuously and in doing so will draw the alcohol up into the print head and wash out the jewel block.

(e) BREAK the program.

(f) Turn the printer power OFF. Dry the print head with a lint free cloth and refix the print head to its carriage with the two screws.

[Editor's note - If you find that some pins are still a bit sticky, you may wish to apply a very small amount of lubricant to the pins. WD-40 will work, but some do not recommend its use because they say it contains some ingredients that are really intended for use on heavy machinery, not small devices such as a print head. Silicone sprays are NOT recommended, they tend to gum up. Many electronic supply stores carry lubricants that are specifically designed for use on small electronic-mechanical parts (such as volume controls, switches, etc.) and these may be the safest, IF they do not contain components that harm plastics. Keep in mind that most printer ribbons contain lubricants (unless they are an off-brand, or were re-inked using stamp pad ink or some other ink that contains no lubrication), so unless you really need the extra lubrication, don't apply anything!]

IMPORTANT

1. To avoid damaging the connecting cable, it is important that the head carriage doesn't move very far from the extreme left hand side. That is the reason for the TAB(0) in LINE 10.

2. Other makes of printers may require the use of different CHR$ codes to exercise their pins. Check your printer manual for suitable codes and amend the program if necessary.

3. It is STRONGLY RECOMMENDED that you test RUN this program, to insure that the head carriage doesn't move further to the right than the TAB(1) position whilst the program is running, before unscrewing the print head screws.

DISCLAIMER

Neither the author nor this publication accepts liability for any damage resulting to any computer equipment or printer caused by persons following the instructions in this article, as it is presented for information purposes only. If you don't know what you're doing, we recommend that you leave the servicing of your printer to competent personal.

SHOVELS TO TEASPOONS

by Laurie Bisman

[This article is reprinted from CHRISTCHURCH-80, newsletter of the Christchurch-80 Users' Group, P.O. Box 4118, Christchurch, New Zealand.]

I have always been frustrated by the way NEWDOS/80 copies files. The method it uses could easily be modified to eliminate problems such as unwanted directory entries should a copy backfire. I often have a need to copy programs and data from disks which hold a lot of information down to disks which are a lot smaller capacity-wise. This causes another problem of course, that of knowing just how many files from disk "A" will actually fit onto disk "B". Although not a complete solution, this program goes part of the way to solving the problems.

I suggest that you eliminate the REMs when you type it in. It is quite small without them. I also suggest that ACCEL3 or ABASIC should be used to speed things up.

As files are created on disk zero, ensure that you have five or six grans free and that the disk isn't write protected.

[NORTHERN BYTES editor's note: Many of the lines in the program below contain linefeeds, which force the remainder of a BASIC line to begin on a new line in the printout. Thus, where you see a line that does not begin with a BASIC line number, you should assume it to be a continuation of the previous line. When entering this program into your computer, the linefeeds can be entered by using the down-arrow key.]

```
10 '   *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
       *   COPY FROM LARGE TO SMALL DISKS WITH    *
       *       /ILF FILES AND /JCL FILES          *
       *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

20 '           Laurie Bisman  -  1/1986.

30 '-------------------------------------------------------------
40 ' This program will allow easy copying from disks which have
     large capacity to disks which have a smaller capacity.
     For example: You may have some programs stored on a double
50 ' sided 80 track disk and you want to transfer all to single
     sided 40 track disks. Problem!! How many programs will fit
     onto the 40 track disk?
60 ' This program scans the source disks directory, compiling a
     series of /ILF files and finally producing a /JCL file for
     use in copying. Once this has been done, the /JCL file is
70 ' executed immediately. You will be prompted to change disks
     as required.
80 '-------------------------------------------------------------

                    Program commences....

90 '=============================================================
     CLEAR some string-space and dimension arrays to hold the
     program names and the gran sizes.  Set up various other
100 'variables as counters, pointers and constant values used
     in the directory-scanning section.
110 '=============================================================
120 CLEAR5000:DEFINTA-Z:DIMPR$(256),PR(256):C=1:A$="":P=32:P1=1:P2=2
55:TG=0:CLS
130 INPUT"SOURCE DISK NUMBER ";SD$
140 INPUT"TARGET DISK NUMBER ";TD$
150 INPUT"FORMAT TARGET DISK  (Y/N) ";FMT$
160 D=INSTR("YyNn",FMT$):IFD=0THEN150
170 IFD<3 THEN FMT$="FMT"ELSE FMT$="NFMT"
180 INPUT"How many grans on target disk "; GT
190 '
    -------------------------------------------------------------
200 'Build correct DIRECTORY name and prepare to look at file.
     An error will be generated immediately, but is trapped to
     allow program to continue.
210 '-------------------------------------------------------------
220 FI$="DIR/SYS:"+SD$
230 ONERRORGOTO950  '*  TRAP ERROR CAUSED BY LOOKING AT
"DIR"  *
240 OPEN"D",1,FI$  '   **  GO GET FILE  **
250 FIELD1,96ASGRANS$,112ASDUMMY$,16ASND$
       '  **  PREPARE TO GET NUMBER OF GRANS  **
260 FIELD1,32ASA$(1),32ASA$(2),32ASA$(3),32ASA$(4),32ASA$(5),32ASA$(
6),32ASA$(7),32ASA$(8)
       '  **  PREPARE TO GET EACH FILESPEC  **
270 CLS:PRINT@0,"READING DIRECTOR SECTOR"
280 FORL=3TOLOF(1):GET1,L  '
    **  LOOK AT DIR SECTORS AFTER "GAT" AND "HIT"  **
290   PRINT@25,LOF(1)-L
300 '
310 '-------------------------------------------------------------
320 '  Build file names and store them in the array PR$(n).
       Calculation of gran sizes is performed (subroutine),
       and gran sizes are stored in array PR(n).
330 '-------------------------------------------------------------
340   FORI=1TO8
350     IFASC(LEFT$(A$(I),1))<>16THEN440  '
          **  IGNORE KILLED, INVISIBLE OR SYSTEM FILES  **

360     T1$=MID$(A$(I),6,8):T2$=MID$(A$(I),14,3)
370     IF T2$="   "THEN390
380     IFRIGHT$(T2$,1)=" "THENT2$=LEFT$(T2$,LEN(T2$)-1):GOTO380
390     IFRIGHT$(T1$,1)=" "THENT1$=LEFT$(T1$,LEN(T1$)-1):GOTO390
400     IF T2$="   "THEN420
410     T1$=T1$+"/"+T2$  '
          **  PUT TOGETHER IN CONVENTIONAL FORM  **

420     M=0: GOSUB 830  '
          **  GET NUMBER OF GRANS FOR FILESPEC  **

430     PR$(C) =T1$:PR(C)=M:C=C+1
440   NEXTI
450 NEXTL
460 CLOSE
470 '
480 '-------------------------------------------------------------
490 ' Set pointer variable (PTR) to 65.  ASCII 65 is capital A.
      Create and open an appropriate /ILF file (TEMP + a letter
      of the alphabet + extension /ILF).
500 ' Write program names into new  /ILF file counting the gran
      sizes, when sufficient programs have been written to file
      that file is closed, another is created and the whole lot
510 ' is done again until all programs are accounted for.
520 '-------------------------------------------------------------
530 PTR=65
540 GOSUB 1010
550 FOR I=1TOC-1:PRINT #1,PR$(I):TG=TG+PR(I)
560   IFTG+PR(I+1)>GT THEN CLOSE:PTR=PTR+1:GOSUB1010:TG=0
570 NEXTI:CLOSE
580 '
590 '-------------------------------------------------------------
600 ' Now the /JCL file is created and built according to how
      many /ILF files were created. Each /ILF file is used in
      copying one 'SET' of programs to one disk.
610 ' Once a disk is full, a prompt is issued to change disks
      and the next copy is done.
620 ' After this process has finished,  it may be repeated by
      simply typing  'DO COPY'.  However, all temporary files
      may be deleted by typing 'DO TIDY'.
630 '-------------------------------------------------------------
640 OPEN"O",1,"COPY/JCL:0"
650 OPEN"O",2,"TIDY/JCL:0"
660 FORI=65 TO PTR
670   PRINT#1,"COPY "+SD$+" to "+TD$+",,"+FMT$+ " CBF NDMW ILF=T
EMP"+CHR$(I)+"/ILF"
680   PRINT#2,"KILL TEMP"+CHR$(I)+"/ILF"
690   IFI<PTR THEN PRINT#1,CHR$(129);"CHANGE DESTINATION DISK"
700 NEXT
710 PRINT#1,CHR$(131);"
    To tidy up (DELETE) /ILF & /JCL files,
    type 'DO TIDY' <ENTER>"

720 PRINT#2,"KILL COPY/JCL":PRINT#2,"KILL TIDY/JCL":CLOSE
730 '
740 '-------------------------------------------------------------
750 '  The completed job is now performed.
760 '-------------------------------------------------------------
770 CMD"S=DO COPY"
780 '
790 '-------------------------------------------------------------
800 ' This is the subroutine which calculates the number of
      grans each file uses on disk.
```

```
810 ' On EXIT the gran size for the current file is left in
      the variable 'M'
820 '-------------------------------------------------------
830 FORN=23 TO 32  '  **  LOAD "EXTENT" TABLE INTO ARRAY  **
840   TEMP(N-22)=ASC(MID$(A$(1),N,1))
850 NEXTN
860 FORN=2 TO 10 STEP 2  '  **  TALLY NUMBER OF GRANS  **
870   IF TEMP(N)=P2 AND TEMP(N-1)=P2 THEN 890
880   M=M+TEMP(N)-(FIX(TEMP(N)/P)*P)+P1
890 NEXTN
900 RETURN
910 '
920 '-------------------------------------------------------
930 ' This is the error trap line, any error barring a corrupt
      directory GAT sector will be ignored. An error is caused
      by reading a sector as these sectors are  READ PROTECTED
940 '-------------------------------------------------------
950 IFERR=106THENRESUME960ELSERESUMENEXT
960 PRINT@64,"This disc has a corrupted  ";CHR$(34);"HIT";CHR$(34);"
sector.
(2nd byte should be  C4)."
970 '
980 '-------------------------------------------------------
990 ' This subroutine creates and opens successive /ILF files.
1000 '------------------------------------------------------
1010 OPEN"O",1,"TEMP"+CHR$(PTR)+"/ILF:0"
1020 RETURN
```

THAT $"#%&" RADIO SHACK PRINTER ROUTINE

by Jay D. Mann

[This article is reprinted from CHRISTCHURCH-80, newsletter of the Christchurch-80 Users' Group, P.O. Box 4118, Christchurch, New Zealand.]

Not long ago I wrote about my infuriating experience with an Okidata printer, in which the real trouble turned out to be the arrogant assumption by the authors of the TRS-80/SYSTEM-80 ROM that when I wrote "LPRINT CHR$(11)" that I didn't know what I was talking about, and really wanted a string of line feeds!

Since then, two further annoying tricks have shown up. The first was a bug in an otherwise very elegant program "DOCLIST/BAS" in the book "Basic Faster and Better" by L. Rosenfelder. This program, which Laurie Bisman referred me to, turns a scrunched BASIC program into an elegant indented listing. But when I ran it, it produced a good first page, then four or five pages with one line per page, then a full page, and so forth. The trap was in line 3240. Rosenfelder PEEKs location 16425, where the TRS-80 ROM authors generously provided a line counter for the printer. If he finds more than 50 lines so far, then he LPRINTs CHR$(12), a form-feed. Of course the ROM can't leave that alone; it converts CHR$(12) into a bunch of line feeds. OK, so that works with an unintelligent printer. But then the stupid ROM does not reset the counter at 16425 to zero!!!! So the next program loop, another form-feed is issued. There is some value of 16425 where it will indeed be reset, but only after the profits of N. Z. Forest Products have been incremented considerably.

So, if you want to use the excellent DOCLIST program, change line 3240 to read as follows:

```
3240 IF INSTR(OP$,"P")=0 THEN LPRINT " ": IF PEEK(16425) >50 THEN
LPRINT CHR$(12);: POKE(16425,0): IF INSTR(OP$,"S") THEN GOSUB
3000: GOSUB 3100 ELSE GOSUB 3100
```

The only difference is that I've installed a POKE to reset the line counter... doing what Radio Shack should have done.

Another problem came up while trying to produce Dragon Curve plots on my printer. The method involved producing a giant bit-mapped image on a disk file, then LPRINTing these files on paper. Naturally a lot of bytes were empty, and I ran into the old Radio Shack problem of not outputting CHR$(0).

So I tried Alan Johnstone's printer driver, which he claims will put out all bytes. Sorry, Alan, not so! The first part of his driver checks register A for being zero, and jumps back to the TRS-80 ROM if so. And the blasted ROM will not LPRINT a zero.

I needed a way to print CHR$(0)! The first method was to Zap Alan Johnstone's DOS. Use Superzap, modifying SYS0/SYS, file relative Sector 23. Put a few zeroes in as follows:

```
[old:]     30 [5A]     B7 CA D1 05 18 25 ...
[new:]     30 [5A]     B7 00 00 00 18 25 ...
```

[NORTHERN BYTES editor's note: The "5A" in square brackets is there because that is the location to zap in the version of Alan Johnstone's DOS that appears on TAS Public Domain Library disk #ND-1. The "30" was the starting byte to zap given in the original article, and probably refers to an earlier version of Alan's DOS.] This wipes out the CHR$(0) test in Alan's DOS. Remember, it only happens the next time you Boot, and check it on a spare copy of the disk!

That won't help people who don't use that DOS. So here is a tiny, stupid, unassuming Assembly program that provides a printer driver that will print anything you want it to! Change the Origin to any convenient location. Invoke it from DOS level before you enter BASIC.

.....(the assembly program follows this article -Ed)

Why did Radio Shack write such a stupid printer handler? I think it was a mixture of an attempt to compensate for then current Radio Shack printers, plus a mixture of half-baked incompleted attempts to emulate bigger computers. The Radio Shack printers put in automatic line feeds after carriage returns, so the TRS-80 chops out line feeds! The printers didn't know how to handle a form-feed, so the TRS-80 did it. But why did the ROM writers dump out CHR$(0)? In the days of the mechanical telepriners like the ASR-33, the printer was out of action while the head was leisurely returning to the left margin after a CR. It was common to have a system variable called CRFILL that would produce a string of CHR$(0) after a carriage return, just to give the printer time to recover. So the Radio Shack authors decided to intercept the CHR$(0) at the ROM level instead of leaving it to the printer to ignore them or not. Not only that, they didn't follow through by installing anything like a CRFILL option, so there isn't any way within the ROM to put out a bunch of nulls.

We all know about the straightout bug in the Model I printer driver, where 67 lines instead of 66 lines are printed per page. What I cannot understand is how the geniuses in Texas continued to produce ROM revisions that retained all of the bugs of the first version.

```
                00100 ; SIMPLE-MINDED PRINTER DRIVER
                00110 ; BY JAY MANN     25/8/85
                00120 ;    KEEPS ITS STUPID FINGERS OUT OF THE PIE!
                00130 ;    AND TRANSMITS ALL REPEAT ALL CODES TO PRINTER
                00140 ;    WITHOUT CHANGING ANY OF THEM!
                00150
F000            00160           ORG     0F000H
F000 2115F0     00170 START     LD      HL,LPTEST-1     ; PROTECT MEMORY
F003 224940     00180           LD      (4049H),HL      ; HIMEM
F006 22B140     00190           LD      (40B1H),HL      ; BASIC HIMEM PTR
F009 11CEFF     00200           LD      DE,-50          ; CLEAR STRING AREA
F00C 19         00210           ADD     HL,DE
F00D 22A040     00220           LD      (40A0H),HL      ; CLEAR 50 TO BEGIN WITH
F010 2116F0     00222           LD      HL,LPTEST       ; CHANGE FOR SYSTEM-80
F013 222640     00224           LD      (4026H),HL      ; INSTALL NEW HANDLER
                00230
F016 3AE837     00240 LPTEST    LD      A,(37E8H)
F019 E6C0       00250           AND     0C0H            ; MASK ALL BUT BUSY&PAPER
F01B 2808       00260           JR      Z,GO
F01D 3A4038     00270           LD      A,(3840H)
F020 CB4F       00280           BIT     1,A             ; CHECK FOR <CLEAR>
F022 C0         00290           RET     NZ              ; EMERGENCY BAILOUT
F023 18DB       00300           JR      START
F025 79         00310 GO        LD      A,C
F026 32E837     00320           LD      (37E8H),A       ; OUTPUT TO PRINTER
F029 C9         00330           RET
                00340
F02A DBFD       00350 START1    IN      A,(0FDH)        ; FOR SYSTEM-80
F02C E6C0       00360           AND     0C0H
F02E 2809       00370           JR      Z,G01           ; OK, NOT BUSY
F030 3A4038     00380           LD      A,(3840H)       ; CHECK KEYBOARD
F033 CB4F       00390           BIT     1,A             ; EXIT IF <CLEAR>
F035 2005       00400           JR      NZ,EXIT
F037 18F1       00410           JR      START1
F039 79         00420 G01       LD      A,C             ; GET CHARACTER
F03A D3FD       00430           OUT     (0FDH),A        ; PRINT IT
F03C C9         00440 EXIT      RET
F000            00450           END     START
00000 TOTAL ERRORS

EXIT    F03C    GO      F025    G01     F039    LPTEST  F016    START   F000
START1  F02A
```

MULTI DISK FORMATTING UTILITY
for Models I, III, 4 (in Model III mode) and LDOS
by Ian H. Robertson

[Reprinted from the Adelaide Micro-User News, 36 Sturt Street, Adelaide, South AUSTRALIA 5000.]

The other day I decided it was time to clean up my disks, get rid of all the rubbish one inevitably collects, and utilise some of the wasted space. After very little effort I discovered I had at least 15 spare disks which I set about formatting there and then.

Before going any further with the story, I should add that I recently added two 80 track double sided drives to my system. One of the new drives is used as :0 and holds all my utilities, wordprocessors and so on. The other is used as :1 while one of my 40 track drives is configured as :2. I haven't yet decided what I'll do with the other 40 tracker.

Having formatted two disks I got to wondering why on earth there isn't a really simple way of repeating the formatting process (ala MS/PC-DOS), or of changing the parameters to cater for differing densities, tracks, sides, etc.

So I decided to fix the problem. The accompanying BASIC program is the result. Now, now, all you SU+ owners!! I know you can use SU to achieve the same result, but in doing so it is necessary to reboot the system afterwards. I wanted a utility that was specific for the job and allowed a normal return to DOS upon completion. This utility fits the bill nicely - for my purposes anyway.

Of course it is possible to configure KSM to carry out the same task but that means you suffer the additional overhead of more memory. I already have a significant number of drivers in high memory, as well as a comprehensive number of keys configured for KSM, so I elected to go another way.

This program will be available in the [Adelaide] Club's Public Domain set so if you don't want to type in the accompanying listing, arrange to obtain a copy from the Librarian or Secretary. [Now that it has appeared in NORTHERN BYTES, it will probably show up on a TAS Public Domain Library disk someday. -editor]

HOW IT WORKS

The Multiple Disk Formatting Utility is specifically designed to be used in a multi-drive, multi-parameter environment. It cannot be used in its current format by single drive owners.

It is also designed specifically for LDOS owners, although I see little reason for it not to be suitable for owners of other DOS's, provided DOS commands can be called from BASIC and that the correct parameters are passed to the FORMAT/CMD program.

I doubt that it can be used with TRSDOS 2.3.

The program is quite simple and I've tried to lay it out so it's easy to follow. I have not used any special tricks to achieve the end result, nor have I put special effort into the structure. If you feel that it can/should be improved, please feel free to do so.

The first line CLEARs some string space and sets up the error trapping routine. This routine is required because errors encountered during format, which cause FORMAT to abort, return a "System Command Aborted" message which causes BASIC to stop running at the current line number.

This error message may occur because the format process was unsuccessful (too many errors), or the disk was write protected, or you decided to abort the process with the <BREAK> key etc. etc.

The last line - RESUME #### forces BASIC to CONTinue.

The next line branches to the routine to set up the intro. screen and the default parameters. You may want to change these parameters to suit your own system, particularly if you are not using 80 track drives.

You are then given a menu which displays the current (default) settings. If they are OK the program branches to the formatting routine. If they are not OK the branch is to a CHANGE menu which gives the option of changing any, or all of the specified parameters.

The CHANGE menu is designed to "flip" between two settings for each option.

If you select option 1, Drive No., which is currently set at "1", it will flip to Drive No. "2".

If you select option 2, Density, which is currently set at "Dden", it will flip to "Sden".

If you select option 3, Sides, which is currently set at "2", it will flip to "1".

If you select option 4, Cylinders, which is currently set at "80", it will flip to "40".

When you have configured the parameters to suit, press <ENTER> and you'll be taken into the FORMAT routine.

The FORMAT routine commences by asking you to insert your disk into the currently selected drive to commence formatting, or select option "1" or "2" to return to the DEFAULT or CHANGE menus if you're not satisfied with the current settings and want to start again.

If you are happy, it goes away and formats your disk.

Upon return from FORMAT, you are then given the option of formatting another disk or returning to DOS.

If you wish to format another disk you have the option of returning to the DEFAULT menu or the CHANGE menu.

If you select the DEFAULT menu the program resets the variables DR$, DE$, SI$ and CY$ by branching to the necessary routine near the end of the program and then displaying the DEFAULT menu.

If you select the CHANGE menu, the program branches to a routine which strips out the previously added "extras" in the FORMAT routine and then displays the CHANGE menu with the previously selected parameters, exactly as you had specified.

NOTES

If you have a mix of drives with track numbers (say 40, 77 & 80) not catered for in the "flip" routine to change number of tracks, you may wish to replace this particular area with something like:-

```
#### REM ===== CHANGE NUMBER OF CYLINDERS (TRACKS) =====
#### REM
#### CLS
#### PRINT "Current number of Cylinders = "CY$
#### PRINT
#### LINEINPUT"Please key-in new number of Cylinders"CY$
#### IF CY$ < "35" OR CY$ > "80" PRINT:PRINT "Number of
Cylinders out of range (35 to 80)!!:GOTO ####
```

(The GOTO#### in the last line would branch to the "PRINT" statement following the line 'PRINT"Current number of Cylinders ="CY$')

```
#### GOTO #### (the start of the CHANGE menu routine.)
```

So there you have it. Have some fun!!!

```
100 CLEAR 1000:ON ERROR GOTO 1440
110 REM
120 REM ===== SET DEFAULTS AND GET TITLE PAGE =====
130 REM
140 GOSUB 1210
150 REM
160 REM ===== SET UP DEFAULT SCREEN =====
170 REM
180 CLS
190 PRINT@ 20, "MULTIPLE DISK FORMAT UTILITY"
200 PRINT@ 84,STRING$(28,131)
210 PRINT
220 PRINT"Default settings are:        Drive       = "DR$
230 PRINT"                             Density     = "DE$+"den"
240 PRINT"                             Sides       = "SI$
250 PRINT"                             Cylinders   = "CY$
260 PRINT
270 PRINT
280 PRINT"Is this Okay?                <Y>es or <N>o"
290 GOSUB 340
300 IF A$ = "Y" OR A$ = "y" THEN 370
310 IF A$ = "N" OR A$ = "n" THEN 790
320 PRINT:PRINT "Must be <Y> or <N> !!
330 GOTO 290
340 A$="":A$=INKEY$:IF A$ = "" THEN 340
350 RETURN
360 REM
370 REM ====== FORMAT ROUTINE ======
380 REM
390 DE$=DE$+"DEN"
400 SI$="SIDES="+SI$
410 CY$="CYL="+CY$
```

```
420 CLS
430 PRINT@192,"Insert your disk in Drive :";DR$;" and press
<ENTER> to continue"
440 PRINT@320,"                              - OR -"
450 PRINT@448,"To return to MAIN (Default) Menu - Key...<1>"
460 PRINT@576,"                              - OR -"
470 PRINT@704,"To return to CHANGE Menu  - Key.....<2>"
480 GOSUB 340
490 IF A$=CHR$(13) GOTO 530
500 IF A$="1" GOSUB 1370:GOTO 180
510 IF A$="2" GOSUB 740 :GOTO 790
520 GOTO 480
530 F$="FORMAT :"+DR$+" ("+DE$+","+SI$+","+CY$+",Q=N)"
540 CMD F$
550 PRINT
560 PRINT"Format another disk? <Y>es or <N>o"
570 GOSUB 340
580 IF A$ = "Y" OR A$ = "y" THEN GOSUB 720:GOTO 610
590 IF A$ = "N" OR A$ = "n":PRINT:PRINT"Returning to DOS":FOR X=0
TO 1000:NEXT:CLS:CMD "S
600 GOTO 570
610 REM
620 REM ===== RETURN TO DEFAULT OR CHANGE MENU  =====
630 REM
635 CLS
640 PRINT:PRINT"To return to MAIN (Default) menu - Key... <1>"
650 PRINT:PRINT"                             - OR -"
660 PRINT:PRINT"To return to CHANGE menu - Key..... <2>"
670 GOSUB 340
680 IF A$="1" GOSUB 1370:GOTO 180
690 IF A$="2" GOTO 790
700 GOTO 670
710 REM
720 REM ===== RESTORE VARIABLES =====
730 REM
740 DE$=LEFT$(DE$,1)
750 SI$=MID$(SI$,7,1)
760 CY$=MID$(CY$,5,2)
770 RETURN
780 REM
790 REM ===== CHANGE DEFAULT SETTINGS =====
800 REM
810 CLS
820 PRINTTAB(13)"Which settings do you wish to change?"
830 PRINT
840 PRINTTAB(13)"Parameter                Current Setting"
850 PRINTTAB(13) STRING$(37,131)
860 PRINT
870 PRINTTAB(13)"<1> Drive No.            "DR$
880 PRINTTAB(13)"<2> Density              "DE$+"den"
890 PRINTTAB(13)"<3> Sides                "SI$
900 PRINTTAB(13)"<4> Cylinders            "CY$
910 PRINT
920 PRINTTAB(13)"Key-in the number of your choice"
930 PRINT
940 PRINTTAB(13)"Press <ENTER> when finished"
950 GOSUB 340
960 IF A$=CHR$(13) THEN 370
970 A=VAL(A$)
980 IF A <1 OR A >4 THEN 950
990 ON A GOTO 1010,1060,1110,1160
1000 REM
1010 REM =====CHANGE SETTING OF DRIVE NUMBER =====
1020 REM
1030 IF DR$="1" THEN DR$="2":GOTO 790
1040 IF DR$="2" THEN DR$="1":GOTO 790
1050 REM
1060 REM ===== CHANGE DENSITY =====
1070 REM
1080 IF DE$="S" THEN DE$="D":GOTO 790
1090 IF DE$="D" THEN DE$="S":GOTO 790
1100 REM
1110 REM ===== CHANGE NUMBER OF SIDES =====
1120 REM
1130 IF SI$="1" THEN SI$="2":GOTO 790
1140 IF SI$="2" THEN SI$="1":GOTO 790
1150 REM
1160 REM ===== CHANGE NUMBER OF CYLINDERS (TRACKS) =====
1170 REM
1180 IF CY$="80" THEN CY$="40":GOTO 790
1190 IF CY$="40" THEN CY$="80":GOTO 790
1200 REM
1210 REM ===== SET UP TITLE SCREEN =====
1220 CLS
1230 PRINT STRING$(64,179)
1240 PRINT@274,"Multiple Disk Format Utility"
1250 PRINT@338,STRING$(28,131)
1260 PRINT@406,"(C) Ian H. Robertson"
(Can you copyright it and put it in the public domain, Ian? EdG)
1270 PRINT@473,"23 Hillary Cra"
1280 PRINT@535,"Modbury North 5092"
1290 PRINT@603,"Australia"
1300 PRINT@665,"Phone 2630653"
1310 PRINT@729,"10th June 1985"
1320 PRINT@896, STRING$(64,179);
1330 FOR X=15360 TO 16319 STEP 64: POKE X,191: NEXT
1340 FOR X=15423 TO 16319 STEP 64: POKE X,191: NEXT
1350 FOR X=0 TO 2000:NEXT
1360 REM
1370 REM ===== SET UP DEFAULTS =====
1380 REM
1390 DR$="1":REM SET DEFAULT DRIVE NUMBER
1400 DE$="D":REM SET DEFAULT DENSITY
1410 SI$="1":REM SET DEFAULT NUMBER OF SIDES
1420 CY$="40":REM SET DEFAULT NUMBER OF CYLINDERS (TRACKS)
1430 RETURN
1440 RESUME 550
```

TRSDOS 6.2 PASSWORD CHECKING

by Mike Zarowitz

Many schemes to by-pass TRSDOS 6.2 password checking have evolved since the operating system was released. However, all of these metheds require disk zapping directory records to either set the password to 8 blank spaces or to inform DOS that a password is not required. It would be nice if there was a NEWDOS-like command that simply turned password checking on or off. Quite by serendipity, I have found this switch in TRSDOS, and it can easily be set and reset from BASIC without disk zapping. To turn off password checking, issue:

POKE &H77,(PEEK(&H77) OR 128).

To turn it back on:

POKE &H77,(PEEK(&H77) AND 127).

The permanent patch is:

```
.patch to turn off password checking
.apply to SYSØ/SYS.LSIDOS
DØØ,83=8Ø
FØØ,83=ØØ
```

For the machine language programmer, these actions set and reset bit 7 of NFLAG$, the so-called "networking" flag. I don't have any TRSDOS backup-limited programs, but this raises an interesting possibility. The TRSDOS source code describes NFLAG$ bit 1-5 and our bit 7 as "reserved", yet bit 7 is actually used to control password checking. It is possible that some of bits 1-5 control the number of backups of certain disks. This should be easy to test as the standard value of NFLAG$ is 00 hex. If you have a backup-limited disk, check byte 00,83 of SYS0/SYS.LSIDOS to see if it is other than zero. If so, zap it to 00 hex and see if it now allows more backups.

For those who must guard against turning off of password checking with their system, either of the following patches to SYS2/SYS.LSIDOS will ignore the status of NFLAG$:

```
.either patch ignores status of NFLAG$
.apply to SYS2/SYS.LSIDOS
.
.patch #1: convert JR Z  to  JR
DØ2,27=18
FØ2,27=28
.end of patch #1
.
.patch #2: NOP out adjustment of filename password value
DØ2,2D=ØØ,ØØ
FØ2,2D=54,5D
.end of patch #2
```

MODEL 4P NEWDOS/80 VERSION 2 SELF-BOOTING SYSTEM DISKS FOR STANDARD 40-TRACK AND FOR 80-TRACK DOUBLE-SIDED DRIVES

PLUS: HOW TO INSTALL A DRIVE SELECT SWITCH ON YOUR 4P

By Peter Goed
22 Shields Street, Redcliffe, Queensland 4020, Australia
Telephone: 011 + 61 + 7 + 203 4882

(Adapted from an original idea by Tony Domigan published in NORTHERN BYTES, and help from Glen McDiarmid).

1. CREATE A 40-TRACK SINGLE SIDED BOOTING SYSTEM DISK

The methods previously published in Northern Bytes seem to assume quite a lot of knowledge on the reader's part, including that SuperUtility + is available to the reader (not necessarily so!). However, this method uses NEWDOS/80's own utility, SUPERZAP, which comes with NEWDOS/80 and is very adequate (indeed, more suitable) for the job and I propose to lead the reader step by step through the creation process -- on the assumption that he/she may not know very much about disk zapping. You must take great care to follow the steps EXACTLY as set out.

To create a 40-Track NEWDOS disk with corrected clock speed on the 4P using a Model III version of NEWDOS/80 and the MODELA/III file that comes on the TRSDOS 6.x system disk and the MODELA/III disk that comes with the 4P:-

1. Boot TRSDOS 6.x and FORMAT a 40-track SINGLE DENSITY disk in drive 1. Using the copy command, copy the MODELA/III file from the TRSDOS 6.x disk to the formatted disk in drive 1; then remove that disk from drive 1 and put aside.

2. Using the MODELA/III disk that comes with the 4P in drive Ø RESET the computer whilst holding down the <P> key until the disk swap prompt appears on the screen.

3. Remove the MODELA/III disk from drive Ø and insert the Model III NEWDOS/80 disk in its place and hit the <ENTER> key and NEWDOS/80 banner and READY prompt will appear.

4. Set the PDRIVE for drive 1:

PDRIVE,Ø,1,TI=A,TD=E,TC=42,SPT=18,TSR=Ø,GPL=2,DDSL=17,DDGA=2,A<ENTER>

Note. A normal 4P drive can comfortably format 42 tracks - check yours and if it cannot handle this, set TC to suit your drive's capability. You may even find that TC=43 is OK.

5. Insert a fresh disk in drive 1. To get all your System files onto the new disk

COPY,Ø,1,,FMT,CBF,/SYS <ENTER> etc.

6. You now have to create a read-protected sector at DRIVE 1, TRACK 17, SECTOR Ø, so that at bootup the machine firmware will think that it is looking at the GAT sector of the disk directory.

Using SUPERZAP's DD, go to SECTOR 3Ø6 of your new disk thus:-

SUPERZAP <ENTER> <ENTER> 1,3Ø6 <ENTER>

and do the following steps:-

MODØØ Ø1ØØ ZTFF <Enter> <Enter> <Y> <Enter> (This will cause FE Hex bytes to be loaded to address ØØØØH at bootup)

MODCD 43 <Enter> <Y> <Enter> (This byte tells the firmware as follows:)

BIT 7 = Ø (system disk)
BIT 6 = 1 (double density)
BIT 5 = Ø (single sided)
BIT 4 = Ø (not used)
BIT 3 = Ø (not used)
BIT 2 = Ø }
BIT 1 = 1 }} these three bits must equal
BIT Ø = 1 } one less than the GPT (grans per track) used in formatting the disk (with NEWDOS, 2 Granules Per Lump, 2 Lumps Per Track = 4 hence set 3)

SCOPY 1,3Ø6 <Enter> <Y> <Enter> <Enter> (this will read-protect the sector to allow the firmware to think that it is a directory sector of a TRSDOS 6.x disk.

7. Go to sector 3Ø7 by pressing <;>. This sector will become the dummy Hash Index Table (HIT) for the dummy directory we are creating. As the MODELA/III is found by the firmware without reference to the HIT sector, it is only necessary for the firmware to harmlessly load this sector to address ØØØØH where it will be overwritten by the MODELA/III file.

MODØØ Ø1FE ZTFF <Enter> <Enter> <Y> <Enter>

8. Go to sector 3Ø8 by pressing <;>. This sector will become the directory entry for the firmware to find the MODELA/III file.

MODØØ Ø1ØØ ZTFF <Enter> <Enter> <Y> <Enter>
MOD2Ø 1ØØØ ØØØØ ØØ4D 4F44 454C 4120 2Ø49 4949
9642 9642 ØØØØ 111A FFFF FFFF FFFF FFFF
<Enter> <Y> <Enter>
SCOPY 1,3Ø8 <Enter> <Y> <Enter> <Enter>

You have now created a directory entry for the MODELA/III file that tells the firmware to find the file starting at Track 17 Sector 1. The BOOTSTRAP loader already knows that it has to load 39H sectors and does not look for how many to load.

9. Set the PDRIVE on drive Ø to read single density on drive 1:-

PDRIVE,Ø,1,TI=A,TD=A,TC=4Ø,SPT=1Ø,TSR=3,GPL=2,DDSL=17,DDGA=2,A

(don't forget the ,A). Remove your new disk from drive 1 and place the single density copy of the MODELA/III file that you created earlier into drive 1. Enter SUPERZAP and with the DFS feature determine where the MODELA/III file resides on the disk (it should occupy DRS 1Ø to 66).

Place the new NEWDOS disk with the dummy directory in drive Ø (DO NOT REBOOT AT THIS STAGE) and with the CDS feature of SUPERZAP with SOURCE as 1,1Ø and DESTINATION as Ø,3Ø9 with a SECTOR COUNT of 57, you will copy over the MODELA/III file to sectors 3Ø9-365 inclusive on your new disk.

1Ø. Next we have to create a directory entry for NEWDOS to find the file, which, to save confusion with the dummy entry we will name MODEL3/ROM and make it an invisible file. With SUPERZAP go to DRS 172 (this is the first directory entry sector for NEWDOS under the PDRIVE specs shown above)

MOD8Ø 5F2Ø ØØØØ ØØ4D 4F44 454C 3320 2Ø52 4F4D
5678 1234 3DØØ 1E2C FFFF FFFF FFFF FFFF
<Enter> <Y> <Enter>

Now press <-> twice to backstep to sector 17Ø (the GAT - granule allocation table sector)

MOD1E FEFF
FFFF FFFF FF <Enter> <Y> <Enter>

(This will protect sectors 3Ø5 to 369 from being overwritten by NEWDOS).

Go to Sector 171 and

MOD8Ø 43 <Enter> <Y> <Enter> (43 is the HASH code for MODEL3/ROM we are here placing at the correct directory location spot in the hash index table).

11. At this stage we can test our creation out. Remove both disks from the drives and turn off the power, wait a generous 2Ø seconds and switch on, insert your NEW NEWDOS disk in drive Ø and hit Reset. If you have followed all the steps correctly, after about 2Ø seconds you should have the NEWDOS logo and NEWDOS/8Ø READY prompt appear on the screen.

12. Set the PDRIVE for drive 1 the same as drive Ø but with TC=4Ø. Set the SYSTEM options to your requirements and press Reset.

13. Insert the Model III NEWDOS/8Ø master disk in drive 1, and

COPY,1,Ø,,NFMT,CBF,CFWO

Select all files other than the system (/SYS) files (which are already on the disk) to be copied. Make sure to do the copy as above or you are likely to end up with some files being missed in the new master copy.

14. To correct the real time clock to be accurate at both 2 and 4 MHz it is necessary to alter two bytes in SYSØ/SYS (it would pay for you to check the real time clock against a stop watch to see if you have enough variation to need this mod). The first is at FRS 2,24H change 1EH to Ø4H; the second is at FRS 11,AØH change from 19H to 1DH. If you have a MODELA/III file that has the full word COPYRIGHT (not abbreviated to (c)) at the very end of the file (check with SUPERZAP DFS feature FRS59) then you will have to alter FRS 57,54H from 1EH to 33H (it should be preceded by 24 42 CD 8E). In the later versions of this file the byte has already been altered so Tandy must have realised a timing problem with the real time clock.

You now have a full BOOTING 4Ø+ track master copy of NEWDOS/8Ø for your 4P. Place a write protect on this copy and back it up to several working copies by:-

COPY,Ø,1,,FMT,BDU <ENTER> (BDU ensures a complete CLONE copy, no matter what you've done, so should be used if you have a non-standard disk to copy!).

2. CREATE AN 80-TRACK DOUBLE SIDED BOOTING SYSTEM DISK

If you have fitted an 80-track double-sided drive (drive #1) to your 4P and have set up a switch so you can turn it into drive 0, then you need this booting 80-track version of

NEWDOS/80. (The arrangement for switching your drives is fully covered in the next section of this article.)

To create this disk you will need to have previously made the 40-track self-booting NEWDOS disk described in the previous section of this article. To prepare a double-sided double-density disk under NEWDOS, you MUST use SUPERZAP -- SuperUtility + is incapable of handling this configuration!!

1. With your 40-track self-booting disk in drive 0 set the PDRIVE for your 80-track drive as follows.

TI=A,TD=G,TC=80,SPT=36,TSR=0(or 1,2,3),GPL=8,DDSL=17,DDGA=4

****** ensure that DDSL=17 ******

2. COPY,0,1,,FMT,CBF,/SYS (this will put all the SYS files onto the 80-track disk in the correct places). I assume here that the 80-track drive is drive #1 -- if otherwise, then use the correct drive number.

3. The next step is to create a file space for the MODELA/III file:

CREATE,MODELA/III:1,LRL=256,REC=57 <ENTER>

4. Use SUPERZAP,DFS,MODELA/III:1 to find where on your 80-track disk the file is located (it should be at DRS 20-76).

5. With SUPERZAP go to sector 612 on your 80-track disk.

a. MOD00 ZTFF move the cursor to byte CC and change 00 to 2D (excess of tracks over 35. 2D HEX=45 DEC). Alter byte CD from 00 to 62 (to indicate 80-track double-sided double-density). Hit <ENTER> <Y> <ENTER> and with <;> move to sector 613.

b. MOD00 ZTFF <ENTER> <Y> <ENTER> and with <;> move to sector 614.

c. MOD00 ZTFF <ENTER> <Y> <ENTER> SCOPY drive,614 <Y> <ENTER>

d. type in K and at the prompt answer 612 then type SCOPY drive,612 <Y> <ENTER>

e. type in K and at the prompt answer 614. We now have to write a false directory entry for the MODELA/III file pointer.

MOD00 1000 0000 004D 4F44 454C 4120 2049 4949
9642 9642 3900 0091 FFFF FFFF FFFF FFFF
<ENTER> <Y> <ENTER>

6. Hit X to return to the menu and with DFS go to DIR/SYS:dn and FRS0 and MOD0F from F8 to FC to protect the dummy directory. Return to the menu with X and with CDS <Y> 0,309 <ENTER> dn,20 <ENTER> 57 <ENTER> the MODELA/III file sectors on your system disk will be copied over to your 80-track disk from the 40-tracker (this puts the file where the BOOT ROM can find it).

7. Now COPY,0,dn,,NFMT,CBF,CFWO and copy over the remainder of the NEWDOS files to your 80-track disk **(MAKE SURE THAT YOU DO NOT COPY OVER THE MODEL3/ROM FILE!!!!).** If you need to do the alterations for the real time clock refer to the 40-track boot disk section of this article.

8. You must now rename the MODELA/III file:

RENAME MODELA/III:dn MODEL3/ROM <ENTER>

and make it invisible with:

ATTRIB,MODEL3/ROM:dn,INV

9. If you have two 80-track drives you can back up this disk with a simple COPY,dn1,dn2. Alas, if, like me, you have only one 80-track drive in the system, then you will have to make a backup copy by the COPY,0,0 and do a lot of disk swaps (you will still find it very worthwhile for the amount of storage space left on the disk after all your favourite files are on it).

3. HOW TO INSTALL A DRIVE SELECT SWITCH ON YOUR 4P

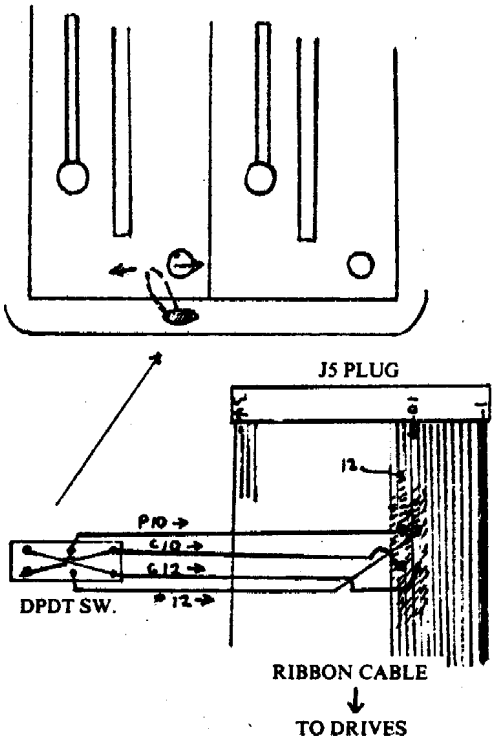
If you fit an 80-track drive to your Model 4P then this is for you. It will allow you to select an alternative drive as drive 0 by the flick of a switch (internal or external drives may be switched).

You will need to remove the case from your 4P (thus voiding the warranty, if still current) and install a switch in front of the drive 0 by drilling a hole on the right hand front side of the drive 0 recess in the horizontal plastic recess in the case. If you buy a small double-pole double-throw (DPDT) switch from Tandy, a ¼" hole will be about right. You will need to solder some wires to the switch and to the cable near J5 at the back of the drives. Counting from the dark edge of the cable, cut line 10 (drive 0) about 1" from the J5 header plug and do the same to line 12 (drive 1), strip a minimum amount of each of the cut wires on both sides of the cut and solder wires (four in all) long enough (about 14") to reach the switch.

The two wires that you soldered to the short ends nearest to J5 are soldered to the two centre terminals of the switch. The cable end of the cut wire at J5 10 will hook up to the end terminal of the switch on the side where J5 10 is terminated. You now run a wire from here to the diagonally opposite end terminal. Now connect the other wire (cable side J5 12) to one of the two terminals left over and run a wire from there to the opposite diagonal (spare) end terminal. Mount the switch so drive 0 is selected if the switch lever is pointing left from the operator's viewpoint. The four leads to the switch are best routed between the drive's mounting bracket and the top shield of the main printed circuit board.

Place the cover back on the computer and you can now swap drive 0 and 1 by the flick of a switch. If you were doing drive 2 or 3 instead of 1 use J5 14 (drive 2) or J5 6 (drive 3) as required.

The leads are marked on the diagram below thus: P10 = plug side of the cut on line 10, P12 = plug side of the cut on line 12, C10 = cable or drives side of the cut on line 10, C12 = cable side of the cut on line 12.



ZAP TO SYS18/SYS FOR IMPROVED BASIC LISTING

by Phil Ereaut

[Reprinted from BITS & BYTES, a publication of the TRS-80 SYSTEM-80 Computer Group, 16 Laver Street, MacGregor, Queensland 4109, Australia.]

When using the NEWDOS/80 BASIC extra list facilities, a problem can occur when using the Up Arrow to list previous lines. The previous line numbers are listed under the existing lines in reverse order, and can cause confusion in reading the screen, particularly when frequently changing backward and forward from Up Arrow to Down Arrow.

This ZAP gives an indication on the screen, when changing from forward listing to backward listing and vice versa. The indication is in the form of a single line gap, with an ARROW at the beginning of the blank line. This arrow will only be displayed on the changeover, any following operation of the same key will be as normal.

With SUPERZAP, make the following changes to Relative Sectors 1 and 3 :-

SYS18/SYS,01,C5 change 00 C3 33 2B 52 to 00 C3 60 55 52

SYS18/SYS,03,70 change All Zeros to:
E5 21 5F 55 3A 27 52 F5 BE 28 17 FE 5B 28 0B FE
0A 20 0F 7E FE 5B 3E 5C 28 08 CD 33 00 3E 0D CD
33 00 F1 77 E1 C3 33 2B

DISPLAY DIRECTORIES BY PASSWORD

[Reprinted from the Corpus Christi TRS-80 Users' Group newsletter:]

DIRPW/CMD is a program I wrote to display directories according to the password encode of the access password. The command syntax is:

DIRPW <drive> <password>

drive specifies the drive number to display. If omitted, the program defaults to drive 0. **password** is an optional parameter that specifies what password a file must have to be displayed. If omitted, all files will be displayed. The routine was originally written for NEWDOS/80 version 2, but I think it should work for other DOS's as well that use a standard TRSDOS 2.3 directory format.

[NORTHERN BYTES editor's note: Be sure to note the position of the closing single quotation mark in line 200. It is very important to note that the string literal in this line is exactly 32 characters long, which means that there are 23 spaces following the ":0" in this line.]

```
5200            00100         ORG     5200H
                00110
4424            00120 DSOPEN  EQU     4424H
4428            00130 DSCLOS  EQU     4428H
4436            00140 DSRD    EQU     4436H
4442            00150 DSPOS   EQU     4442H
0033            00160 VDCHAR  EQU     0033H
002B            00170 KBCHAR  EQU     002BH
                00180
0100            00190 UREC    DEFS    100H
5300 44         00200 DIRFCB  DEFM    'DIR/SYS:0
     49 52 2F 53 59 53 3A 30 20 20 20 20 20 20 20 20
     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
5320 20         00210 PWDVAL  DEFM    '       '
     20 20 20 20 20 20 20 20
5328 0000       00220 HSHVAL  DEFW    0000H
                00230
532A 3E0D       00240 START   LD      A,0DH           ;DISPLAY C/R
532C CD3300     00250         CALL    VDCHAR
532F 7E         00260         LD      A,(HL)          ;GET FIRST CHAR
5330 FE3A       00270         CP      '9'+1           ;IS IT NUMERIC?
5332 300E       00280         JR      NC,PWSET
5334 FE30       00290         CP      '0'
5336 3004       00300         JR      C,PWSET
5338 325853     00310         LD      (DIRFCB+8),A    ;YES, STORE AS DRIVE #
533B 23         00320         INC     HL
533C 7E         00330         LD      A,(HL)
533D FE0D       00340         CP      0DH
533F 2801       00350         JR      Z,PWSET
5341 23         00360         INC     HL
5342 0608       00370 PWSET   LD      B,8             ;GET PASSWORD TO CHECK
5344 115353     00380         LD      DE,PWDVAL       ;STORE IT IN PWDVAL
5347 7E         00390 PWSET1  LD      A,(HL)
5348 FE20       00400         CP      ' '
534A 2809       00410         JR      Z,HSRTN
534C FE0D       00420         CP      0DH
534E 2805       00430         JR      Z,HSRTN
5350 12         00440         LD      (DE),A
5351 23         00450         INC     HL
5352 13         00460         INC     DE
5353 10F2       00470         DJNZ    PWSET1
5355 215353     00480 HSRTN   LD      HL,PWDVAL+7     ;COMPUTER HASH CODE FOR PW
5358 11FFFF     00490         LD      DE,0FFFFH
535B 0608       00500         LD      B,8
535D C5         00510 HSRTN1  PUSH    BC
535E 7B         00520         LD      A,E
535F E607       00530         AND     07H
5361 4F         00540         LD      C,A
5362 7B         00550         LD      A,E
5363 07         00560         RLCA
5364 07         00570         RLCA
5365 07         00580         RLCA
5366 A9         00590         XOR     C
5367 07         00600         RLCA
5368 4F         00610         LD      C,A
5369 E6F8       00620         AND     0F8H
536B 47         00630         LD      B,A
536C 79         00640         LD      A,C
536D 07         00650         RLCA
536E E61F       00660         AND     1FH
5370 A8         00670         XOR     B
5371 AA         00680         XOR     D
5372 5F         00690         LD      E,A
5373 79         00700         LD      A,C
5374 E60F       00710         AND     0FH
5376 47         00720         LD      B,A
5377 79         00730         LD      A,C
5378 07         00740         RLCA
5379 07         00750         RLCA
537A 07         00760         RLCA
537B 07         00770         RLCA
537C A8         00780         XOR     B
537D C1         00790         POP     BC
537E AE         00800         XOR     (HL)
537F 57         00810         LD      D,A
5380 3620       00820         LD      (HL),20H
5382 2B         00830         DEC     HL
5383 10D8       00840         DJNZ    HSRTN1
5385 ED532853   00850         LD      (HSHVAL),DE     ;DE HAS HASH VALUE
5389 115300     00860         LD      DE,DIRFCB       ;OPEN DIR/SYS
538C 215200     00870         LD      HL,UREC
538F 0600       00880         LD      B,0
5391 CD2444     00890         CALL    DSOPEN
5394 C21A54     00900         JP      NZ,DIRR10
5397 010200     00910         LD      BC,2            ;POSITION TO REC #2
539A CD4244     00920         CALL    DSPOS
539D 2078       00930         JR      NZ,DIRR10
539F 115300     00940 DIRRD1  LD      DE,DIRFCB       ;GET ONE RECORD
53A2 215200     00950         LD      HL,UREC
53A5 CD3644     00960         CALL    DSRD
53A8 2804       00970         JR      Z,DIRRD2
53AA FE06       00980         CP      6
53AC 206C       00990         JR      NZ,DIRR10
53AE 0608       01000 DIRRD2  LD      B,8
53B0 C5         01010 DIRRD3  PUSH    BC              ;GET ONE FDE
5381 E5         01020         PUSH    HL
5382 7E         01030         LD      A,(HL)          ;TEST FOR VALID FDE
5383 E608       01040         AND     08H
5385 EE10       01050         XOR     10H
5387 2051       01060         JR      NZ,DIRRD9       ;INVALID, GET NEXT FDE
5389 111200     01070         LD      DE,18           ;VALID FDE,TEST PW
538C 19         01080         ADD     HL,DE
538D 5E         01090         LD      E,(HL)
538E 23         01100         INC     HL
538F 56         01110         LD      D,(HL)
53C0 2A2853     01120         LD      HL,(HSHVAL)
53C3 D5         01130         PUSH    DE
53C4 E5         01140         PUSH    HL
53C5 119642     01150         LD      DE,4296H
53C8 B7         01160         OR      A
53C9 ED52       01170         SBC     HL,DE
53CB E1         01180         POP     HL
53CC D1         01190         POP     DE
53CD 2805       01200         JR      Z,DIRR12        ;GO IF VALID
53CF B7         01210         OR      A
53D0 ED52       01220         SBC     HL,DE
53D2 2036       01230         JR      NZ,DIRRD9       ;NO MATCH; NEXT FDE
53D4 E1         01240 DIRR12  POP     HL
53D5 E5         01250         PUSH    HL
53D6 110500     01260         LD      DE,5
53D9 19         01270         ADD     HL,DE
53DA 011808     01280         LD      BC,0818H        ;DISPLAY FILENAME
53DD 7E         01290 DIRRD4  LD      A,(HL)
53DE FE20       01300         CP      ' '
53E0 2804       01310         JR      Z,DIRRD5
53E2 CD3300     01320         CALL    VDCHAR
53E5 0D         01330         DEC     C
53E6 23         01340 DIRRD5  INC     HL
53E7 10F4       01350         DJNZ    DIRRD4
53E9 7E         01360         LD      A,(HL)
53EA FE20       01370         CP      ' '             ;EXTENSION?
53EC 2806       01380         JR      Z,DIRRD6
53EE 3E2F       01390         LD      A,'/'           ;DISPLAY EXTENSION
53F0 CD3300     01400         CALL    VDCHAR
53F3 0D         01410         DEC     C
53F4 0603       01420 DIRRD6  LD      B,3
53F6 7E         01430 DIRRD7  LD      A,(HL)
```

```
53F7 FE20      01440          CP      ' '
53F9 2804      01450          JR      Z,DIRRD8
53FB CD3300    01460          CALL    VDCHAR
53FE 0D        01470          DEC     C
53FF 23        01480 DIRRD8   INC     HL
5400 10F4      01490          DJNZ    DIRRD7
5402 41        01500          LD      B,C             ;TAB TO NEXT 16TH POS
5403 3E20      01510          LD      A,' '
5405 CD3300    01520 DIRR11   CALL    VDCHAR
5408 10FB      01530          DJNZ    DIRR11
540A E1        01540 DIRRD9   POP     HL              ;GET NEXT FDE
540B C1        01550          POP     BC
540C CD2B00    01560          CALL    KBCHAR
540F B7        01570          OR      A
5410 2008      01580          JR      NZ,DIRR10
5412 112000    01590          LD      DE,20H
5415 19        01600          ADD     HL,DE
5416 10F8      01610          DJNZ    DIRRD3
5418 1805      01620          JR      DIRRD1
541A 110053    01630 DIRR10   LD      DE,DIRFCB       ;CLOSE DIR/SYS
541D CD2B44    01640          CALL    DSCLOS
5420 3E0D      01650          LD      A,0DH           ;DISPLAY C/R
5422 CD3300    01660          CALL    VDCHAR
5425 C9        01670          RET                     ;AND EXIT
532A           01680          END     START
00000 TOTAL ERRORS

DIRFCB  5300    DIRR10  541A    DIRR11  5405    DIRR12  53D4    DIRRD1  539F
DIRRD2  53AE    DIRRD3  53B0    DIRRD4  53DD    DIRRD5  53E6    DIRRD6  53F4
DIRRD7  53F6    DIRRD8  53FF    DIRRD9  540A    DSCLOS  4428    DSOPEN  4424
DSPOS   4442    DSRD    4436    HSHVAL  5328    HSRTN   5355    HSRTN1  5350
KBCHAR  002B    PWDVAL  5320    PWSET   5342    PWSET1  5347    START   532A
UREC    5200    VDCHAR  0033
```

SCRCONV/BAS

by Oswald Cooper

[This article is reprinted from Bits and Pieces, the newsletter of The Northeast Computer Club.]

The following program will take the output from a SuperScripsit selection file from Profile 4 Plus and create a "Name and Address" file and a text file for use with the form letters option of Allwrite. The program asks for the source file and a destination drive for the new files. The new file names are the same as the source file but with the extension of "NAA" and "TXT". If a source or destination drive number is not specified then all drives will be searched for the source and the destination will default to drive 1. If a source drive is specified but no destination, then the destination defaults to the source drive.

You are then asked to provide the code character that Allwrite will use. This defaults to "@". The program then requests the code letter to use and defaults to "A" if none is specified.

The program then prints the header information from the source file to the next file (as comment lines) and transfers the data to the new data file. On completion the data file may be loaded into Allwrite and the last two lines deleted. These are the end of file markers used by SuperScripsit but Allwrite will generate one more letter from the list, but with blank data items. I use this copy of the letter (along with a copy of the name and address file) as a file record of the information sent. The input file is not changed in any way, so the data in that file can still be used by SuperScripsit.

The text file (FILENAME/TXT) contains comment lines defining the variables to be used and the selection code used along with the file name and the control word to use the name and address file.

Line numbers underlined are referenced by "GOTO" or "GOSUB".

[NORTHERN BYTES editor's note: Don't type in the lines that begin with REM, they are for information purposes only.]

```
10      REM SCRCONV/BAS
20      CLEAR
30      DEFINT A-Z: CLS: GOTO 10000
REM PRINT A MESSAGE CENTERED ON A LINE:  ESTABLISH PRINT@ PR
IOR TO ENTRY
400     PRINT TAB(40-LEN(PROMPT$)/2);PROMPT$:RETURN
10000   PROMPT$="SCRCONV/BAS      VERSION 01.02.01      OSWALD CO
OPER"
10010   PRINT @0,;:GOSUB 400:PROMPT$=STRING$(LEN(PROMPT$),"-")
10020   PRINT @80,;:GOSUB 400
10030   PRINT "THIS PROGRAM WILL GENERATE THE FOLLOWING FILES
 FROM A"
10040   PRINT "PROFILE 4 PLUS SUPERSCRIPSIT SELECTION FILE:"
10050   PRINT "     NAME AND ADDRESS FILE:     FILENAME/NAA
10060   PRINT "     ALLWRITE TEXT FILE   :     FILENAME/TXT
10070   PRINT
10080   PRINT "TYPE IN THE SOURCE FILE NAME AND DESTINATION DR
IVE NUMBER"
10090   PRINT "     OF THE OUTPUT FILES, I.E.:     FILENAME/SR1:s :d
10100   PRINT "IF 's' ONLY IS SPECIFIED THEN 'd' BECOMES 's' ALSO"
10110   PRINT "DEFAULT DESTINATION DRIVE IS 1 IF NO DRIVES SPECI
FIED."
10120   PRINT
10130   PRINT "PRESS << ENTER >> TO QUIT OR"
10140   LINE INPUT "     TYPE THE INPUT FILE NAME:  ";FILESPEC1$
10150   IF FILESPEC1$="" THEN PRINT:PRINT "JOB ABORTED.": GOTO 65
000
10160   IF LEFT$(RIGHT$(FILESPEC1$,2),1)<>":" THEN DRIVE$=":1"
10170   I=INSTR(FILESPEC1$,":"):IF I<>0 THEN DRIVE$=RIGHT$(FILESPEC1
$,2)
10180   I=INSTR(FILESPEC1$,"/"):IF I<>0 THEN FILESPEC2$=LEFT$(FILES
PEC1$,I)+"NAA"+DRIVE$:FILESPEC3$=LEFT$(FILESPEC1$,I)+"TXT"+DRIVE$
10190   IF INSTR(FILESPEC1$,"/")=0 THEN FILESPEC2$=FILESPEC1$+"/NA
A"+DRIVE$:FILESPEC3$=FILESPEC1$+"/TXT"+DRIVE$
10200   PRINT "OUTPUT DATA FILE IS:     "FILESPEC2$
10210   PRINT "ALLWRITE TEXT FILE IS:  "FILESPEC3$
10220   PRINT
10230   LINE INPUT "ENTER CHARACTER FOR CODE (DEFAULT = :  ";CO
DE.CHAR$
10240   IF CODE.CHAR$="" THEN CODE.CHAR$="@"
10250   LINE INPUT "ENTER CODE TO USE (DEFAULT = A):  ";CODE$
10260   IF CODE$="" THEN CODE$="A"
10270   PRINT
10280   OPEN"I",1,FILESPEC1$
10290   OPEN"O",2,FILESPEC2$
10300   OPEN"O",3,FILESPEC3$
10310   PRINT "ALL FILES OPENED."
10320   PRINT #3,";CM  FILENAME:  "+FILESPEC3$
10330   PRINT #3,";RD "+CODE.CHAR$+","+FILESPEC2$
10340   PRINT #3,";CM  CODE USED:  "+CODE$
10350   PRINT #3,";CM  INFORMATION HEADER BLOCK
10360   PRINT "WRITING INFORMATION HEADER BLOCK TO "FILESPEC3$
" ...";
11000   IF EOF(1) THEN GOTO 12000
11010   LINE INPUT #1,A$
11020   COUNT=COUNT+1
11030   IF LEN(A$)=0 THEN PRINT #2,CODE.CHAR$+CODE$:GOTO 12000
11040   B$=MID$(A$,2,LEN(A$)-2)
11050   PRINT #3,";CM         "+CODE.CHAR$+CODE.CHAR$+MID$(STR$(CO
UNT),2)+" "+B$
11060   GOTO 11000
12000   COUNT=COUNT-1
12010   PRINT #3,";CM  ==========================================
==========
REM     ADDITIONAL DEFAULT CONTROL WORDS CAN BE ADDED TO TH
E TEXT FILE HERE
                I.E., TURN OFF FORMATTING :  PRINT #3,";FO OFF"
                OR TO ADD THE CURRENT DATE:  PRINT #3,"@@0"
REM     AN ADDITIONAL COMMENT CAN BE ADDED TO THE TEXT FILE
 BY INCLUDING THE
        FOLLOWING:  LINE INPUT "ENTER COMMENT:  ";COMMENT.TEXT
$
                    PRINT #3,";CM  "+COMMENT.TEXT$
12020   CLOSE 3
12030   PRINT " DONE."
12040   PRINT "WRITING DATA FILE "FILESPEC2$: PRINT
12500   IF EOF(1) THEN GOTO 14000
12510   LINE INPUT #1, A$
12520   IF LEN(A$)=0 THEN PRINT #2,CODE.CHAR$+CODE$:GOTO 13000
12530   B$=LEFT$(A$,LEN(A$)-1)
12540   B$=RIGHT$(B$,LEN(B$)-1)
12550   COUNTER=COUNTER+1
12560   IF COUNTER>COUNT THEN COUNTER=1
12570   C$=CODE.CHAR$+CODE.CHAR$+MID$(STR$(COUNTER),2)+" "
12580   B$=C$+B$
12590   PRINT #2,B$
13000   GOTO 12500
14000   CLOSE
14010   PRINT "JOB COMPLETE."
14020   PRINT "TYPE 'AL "FILESPEC3$"' TO EDIT THE FILE."
65000   END
```

MORE ON INPUT WITH INKEY

by Alf West

[Reprinted from the TRS-80 SYSTEM 80 Computer Group newsletter (16 Laver Street, MacGregor, Queensland 4109, Australia).]

The uses of INKEY$ for reading and acting on the keyboard input was mentioned in issue number 23 [of the TRS-80 SYSTEM 80 Computer Group newsletter] in the article "A Comprehensive Look (and Peeks) at the Keyboard". This article develops ideas for a more comprehensive use of the INKEY$ command.

We will start with the usual "press any key to continue" type of INKEY$ statement which is used to give an indefinite delay while you read instructions or a long list of information, viz :-

```
10 PRINT"List of Instructions or Information ..
20 PRINT"PRESS ANY KEY TO CONTINUE"
30 A$ = INKEY$: IF A$="" THEN 30
40 .. Program continues .....
```

This does not always work as you might expect because if some galah happens to lean on the keyboard and depress a key while the computer is printing the list of instructions or information on the screen, you find that there was no delay and the program progresses through to line 40 (and beyond) before you had a chance to read and digest the information printed on the screen by line number 10. To prevent that happening, it is desirable to fudge in the following line immediately before the INKEY$ statement.

```
25 POKE 16537,0
```

This zeros the address which holds the last character which had been entered from the keyboard and "cleans the slate" for the INKEY$, and the program will not proceed until a key is pressed after the computer gets to line number 25. The inclusion of this POKE is often desirable with the INKEY$ statement to avoid spurious results where the operator may "pussy foot" on the keyboard while a program is running.

Leaving this aspect aside, let's look at line number 30 which loops on itself until a key is pressed. The INKEY$ could not be added at the end of line number 20 as the loop would cause the "PRESS ANY KEY TO CONTINUE" to be printed over and over again until a key was pressed. The following line is a much better alternative to the above lines 20, 30 and 40.

```
PRINT"PRESS ANY KEY TO CONTINUE" : FORJ=0TO0: J=(INKEY$=""): NEXT:..continue
```

We will examine the statements :- FOR J = 0 TO 0 : J = (INKEY$="") : NEXT in that line in more detail as the little demonstration programs which will follow are developments of this concept.

In normal cases, a FOR NEXT loop from 0 to 0 would only execute once, but the logic statement (INKEY$="") is true when no key is pressed, and so returns a -1. Thus poor "J" just gets nowhere. He becomes -1, then gets incremented by the NEXT to 0, but goes back to -1, and so stays in that predicament until such time as a key is pressed. When that happens the logic statement (INKEY$="") is false, so returning a "0" and "J" then gets back on his feet and the loop is exited to the rest of the program. As you see, this is a much more elegant approach and the "INKEY$ loop" can be inserted wherever required in a multi-statement line.

Some programmers like to have a flashing cursor to indicate to the operator that he should do something. We can enlarge on the statements underlined above with some logic to turn the cursor on and off as follows :-

```
10 PRINT"PRESS ANY KEY TO CONTINUE "; : POKE 16418,140
20 FORJ=0 TO 0 : J=(INKEY$="") : JJ=(JJ+1) AND 7
30 PRINT CHR$(14 AND (JJ=1) OR 15 AND (JJ=5)); : NEXT : PRINT
40 .. Program continues .....
```

The address 16418 holds the value of the cursor character, so that is poked with the value of a graphic block to make it stand out this is optional depending upon what you feel like and if your ordinary cursor is O.K. then the POKE can be ignored [it won't work on an unmodified Model I computer, but should work as shown on a Model III or 4 (III mode). Note that if your DOS provides for a flashing cursor, this routine will be unnecessary -editor]. The INKEY$ routine starts and finishes as before but in the Logic statement at the end of line 20, JJ starts at zero and is incremented as the FOR NEXT loop goes around until it is 7. The "AND 7" then sets JJ back to zero. The first statement in line 30 initially prints "CHR$(0)" which does nothing until JJ equals 1 when it prints "CHR$(14)" which is the control code to turn on the cursor, or alternatively until JJ equals 5 when it prints "CHR$(15)" which is the control code to turn off the cursor. Hence the cursor flashes to warn the operator until he acts on the message and presses a key, when the program then falls out of the loop as it does in the general case as previously underlined.

Leaving the flashing cursor bit, now assume that we want to know what key was actually pressed. We allocate some string variable to INKEY$, let's say IN$ and the following demo program will tell us which key.

```
10 CLS : PRINT"THIS IS THE START - PRESS ANY KEY"
20 FOR J=0 TO 0 : IN$=INKEY$ : J=(IN$=""): NEXT
30 PRINT"THIS IS THE END - YOU PRESSED THE <"IN$"> KEY"
```

Line 20 in this case is exactly the same as the previous underlined statement except that we have pre-defined IN$ as being equal to INKEY$, and so by printing IN$ after the loop, we see what key was pressed.

This now leads to the obvious extension of these ideas so as to take alternative action according to what key the operator pressed. Just as the INKEY$ was used in an ordinary FOR NEXT loop (you might say, not so "ordinary"), the INKEY$ can be used in an INSTR test. (Note: Although the INSTR command is normally only applicable to Disk Basic, it can be used in Level II Basic if you use Warwick Sands' Tape Operating System, a copy of which is available from the Club Library)

The following demonstration program illustrates the INSTR/INKEY$ use in selecting an item in a menu :-

```
10 CLS:PRINT"SELECT MENU ITEM ( <A> <B> <C> <D> OR <E> ) ?" :
F$="XABCDE"
20 FOR J=2 TO 2:J=INSTR("XABCDE",INKEY$): NEXT: ON J-2 GOTO
100,200,300,400,500
100 PRINT"THE <A> MENU ROUTINE CONTINUES ....":END
200 PRINT"THE <B> MENU ROUTINE CONTINUES ....":END
300 PRINT"THE <C> MENU ROUTINE CONTINUES ....":END
400 PRINT"THE <D> MENU ROUTINE CONTINUES ....":END
500 PRINT"THE <E> MENU ROUTINE CONTINUES ....":END
```

The FOR NEXT loop in line number 20 here is not from 0 to 0 but from 2 to 2 due to a peculiarity of the INSTR function and this peculiarity causes the need for the "X" (it can be any unwanted character) at the beginning of the INSTR brackets. To explain - if you were seeking say the first item on the menu, and entered "A" in a normal INSTR test, e.g. F = INSTR("ABCDE","A") then F would be 1, but when you say F = INSTR("ABCDE",INKEY$) the computer comes up with an answer of "1" when no key is pressed, i.e. when INKEY$ = "". (You would expect the answer of "0" because there is no null string in the string "ABCDE" under test, but this is not so.) Because you get "1" in any case when no key is pressed, the FOR NEXT loop must go to "2". Likewise you must put in a "dummy" character (X in this case) in the INSTR tested string to make it "XABCDE" so that the "acceptable" characters will start from the second position onwards to give J a value sufficient to exit the loop. That is the expression :- "J = INSTR("XABCDE",INKEY$)" gives J a value of 1 if no key or the "X" is pressed, a value of 0 if any other unwanted key is pressed and only gets a value of 2 or more to allow exit of the loop if one of the wanted keys is pressed. The normal increment after a loop means that you use "J-2", (not J-1) in the "ON v GOTO" statement. (If all this "explanation" has confused you, don't worry because the program works - so try it out.) [Some DOSes have fixed the INSTR function of Disk Basic so that it DOES return a zero when no key is pressed. However, the above routine should work under ANY of the popular DOSes -editor].

An extension of the previous program to require the operator to input a given number of answers, no more and no less, to only allow particular "acceptable" answers, and to ensure that his answers are all different, i.e. that he does not repeat himself, is given in the following demo. program :-

```
10 CLS: LN=3: PRINT"SELECT"LN"ITEMS FROM  ( <A> <B> <C> <D> OR
<E> ) ?"
20 F$="XABCDE": OP$=STRING$(LN,32)
30 FOR I=1 TO LN
40 FOR J=2 TO 2 : IN$=INKEY$ : J=INSTR(F$,IN$) : NEXT
50 IF INSTR(OP$,IN$) THEN 40
60 PRINT IN$, : MID$(OP$,I,1)=IN$ : NEXT
70 PRINT : PRINT"YOU SELECTED ITEMS "OP$
```

The number of items the operator must select (no more and no less) is specified as "LN" in line number 10 - "3" in this case. In line 20, F$ gives the "acceptable" items with the dummy "X" in front. The string "OP$", used later, is initialised as LN (3) number of blank spaces. Line number 30 sets the outside loop for the number of acceptable entries the operator must make. Line number 40 is much the same as we had before except that here we have again allocated IN$ as being equal to INKEY$ so that we can use it later. If an "acceptable" key is pressed the program will exit line 40 and (skipping over line 50 for the moment) line 60 will print what key was accepted and then plant it in the first blank space of OP$. The NEXT at the end of line 60 returns you to line 30 to get another "acceptable" letter. But if on the next time around, you select the same "acceptable" key as you had done previously, then line number 50 detects that that letter is already in OP$, so it sends you back to line 40 to get a different "acceptable" letter. After having selected LN different acceptable letters, the outside loop is exited and then in line 70, you may print OP$ to show all the acceptable letters.

Needless to say in a practical program, rather than simply print OP$, you would use the information contained in OP$ to send you to the appropriate parts of the program. For example, if the LEFT$(OP$,1) equals say "B" then GOTO .. to do such and such and likewise with the MID$ (or RIGHT$) of OP$.

One final example of using INKEY$ in a FOR NEXT loop which could be of interest to those making up games programs is for a "timed" input i.e. where the operator must give an answer within a certain time. If he doesn't do so, he goes to the dungeons instead of getting the beautiful maiden - that is if there are any maidens left!

```
10 CLS : PRINT"YOU HAVE 50 CYCLES IN WHICH TO PRESS ANY KEY"
20 T=50 : FOR J=0 TO T : IN$=INKEY$ : J=J-T*(IN$>"") : NEXT
30 IF J=T+1 THEN PRINT"YOU MISSED OUT - OFF TO THE DUNGEONS" :
END
40 PRINT"YOU ONLY TOOK"J-T-1"CYCLES. YOU GET THE MAIDEN ?!!"
```

In this example, in line number 20 "T" is the number of "cycles and is set to 50. The term "cycle" is the time it takes to go once around the FOR NEXT loop, so adjust "T" to give you the time you set the operator for him to respond to your message. The logic term "(IN$>"")" inside the loop is false if no key is pressed so having a value of 0. Thus if no key is pressed "J" doesn't change from its normal incremented value as the loop goes around and the loop functions as any normal loop, exiting when "J" reaches the value of "T+1". However, if a key is pressed the logic term "(IN$>"")" becomes true with a value of -1 so that "J" becomes equal to whatever normally incremented value "J" had at that time, plus "T". This causes the loop to exit immediately and the arithmetic of "J-T-1" in line 40 tells you how many "cycles" elapsed before the operator responded.

The listings given above are little programs within themselves which can be keyed in to test the routines, and they will give you some indication of what you can do with these type of INKEY$ routines in developing your own programs.

[NORTHERN BYTES EDITOR'S NOTE: I'd like to add one comment about something that was not considered in the above article. Usually, when I receive a program for evaluation purposes that comes up saying "... PRESS ANY KEY TO CONTINUE ...", I proceed to make a royal pain of myself by taking the instructions literally and pressing any key - well, actually, one of two keys in particular. Those are the left and right SHIFT keys! Usually, pressing one of those two keys will do absolutely nothing! On a Model 4, pressing either the CTRL or the CAPS key usually has a similar non-effect. Or, if I am feeling REALLY nasty, I will press the BREAK key. This, of course, usually stops the program dead in its tracks.

Sure, I know better and am probably being a bit malicious by pressing those keys, but after all, it did say "PRESS ANY KEY..." The point is, for commercial software that may be used by people that don't realize that certain keys don't automatically fall under the classification of "any key", I much prefer to say something like "PRESS <ENTER> TO CONTINUE" and then test for the ENTER key only. This not only avoids the problems I have just mentioned (a user pressing the "wrong" key due to ignorance or malice) but also helps avoid problems that may be caused by someone accidentally pressing a key before they are really ready for the program to continue. The ENTER key is easy to find when you are really ready to continue, but doesn't leave the entire keyboard area vulnerable to an accidental key depression.

Of course, you could trap out the BREAK key and can test for ANY key being depressed by PEEKing at memory location 38FFH and testing for a non-zero value (on a Model I or III, or a Model 4 in the Mod III mode, memory location 38FFH will contain a non-zero value if ANY key is currently being depressed) but you may still experience undesired side effects. In my opinion, it's better to specify which key to press to continue!]

PROFILE 4 PLUS PAT©HES

compiled by Lewis E. Garrison

[This article is reprinted from READY, the newsletter of the Central Alabama Microcomputer Society.]

When using the "?" to display the value of a sort field in the title line when printing reports in PROFILE 4 PLUS, you get incorrect values and page breaks. To correct, apply the following patches to the PROFILE 4 PLUS Runtime diskette:

```
PATCH EFCA/CMD (X'61BE'=00 00 00)
PATCH EFCA/CMD (X'61C4'=C3 E2 61)
PATCH EFCA/CMD (X'61D3'=CA DA 61)
PATCH EFCA/CMD (X'61D9'=C9 E1 C3 9C 62 00 00 00 00 CC A4
                62)
PATCH EFCA/CMD (X'61EC'=00)
PATCH EFCA/CMD (X'61F0'=CC)
PATCH EFCA/CMD (X'61F3'=CD 40 6F)
PATCH EFCA/CMD (X'62E3'=FE 00)
PATCH EFCA/CMD (X'6F40'=2A CD 52 23 22 CD 52 CD C7 61 21
                95 7D C3 F6 61)
PATCH RM/CMD   (X'707E'=3031)
```

Apply the following patch to the PROFILE 4 PLUS Creation diskette:

```
PATCH CM/CMD   (X'707E'=3031)
```

After applying these patches the version will be 01.00.01.

When deleting records in PROFILE 4 PLUS, multiple deletions may occur. Also, while in Inquire, Update, and Add, pressing a number to go to another screen will send you out to TRSDOS. Also, using the SHIFT@ does not copy all 255 characters from one record to another while in the Add mode. Apply the following patches to the PROFILE 4 PLUS Runtime diskette:

```
PATCH EFC9/CMD (X'89C2'=2A 82 8B 2B 22 82 8B 3E FF 77 00
                B7 C9)
PATCH EFC9/CMD (X'8C68'=FF 03)
PATCH EFC9/CMD (X'F8C5'=C3 97 FC)
PATCH EFC9/CMD (X'FC96'=C9 E1 D1 2C 7D FE 50 C2 C8 F8 24
                2E 00 C3 C8 F8)
PATCH RM/CMD   (X'707E'=3032)
```

Apply this patch to the Creation diskette:

```
PATCH CM/CMD   (X'707E'=3032)
```

After applying these patches, the version will be 01.00.02.

Extended math calculations in PROFILE 4 PLUS may add one field to another field that contains a math formula even though the first field is not in the formula. Apply the following patches to a copy of the Runtime diskette:

```
PATCH EFC9/CMD (X'6FF0'=21 3B 88 3A 4D 8A 85 6F C9)
PATCH EFC9/CMD (X'8A04'=CD F0 6F)
PATCH EFC9/CMD (X'8A23'=CD F0 6F)
PATCH EFC9/CMD (X'8A3E'=01)
PATCH RM/CMD   (X'707E'=3033)
```

Apply the following patch to the Creation diskette:

```
PATCH CM/CMD   (X'707E'=3033)
```

After applying these patches the version will be 01.00.03.

BASIC PROGRAM VARIABLE NAMES LISTER

by Bob Koehler
R.D. #4, Box 174, Hopewell Junction, New York 12533

VARLST will list, on the screen, the names of all of the variables in a BASIC program residing in RAM. They will be listed in two groups: arrays and single-values. Any type designators used in the names, (! # $ %), will be included with those names. The names of variables defined with a DEF statement, or those defaulted to single-precision, will not have them.

It is designed to be loaded into upper, protected, memory. I chose the starting address shown, so that it will fit in with a group of other utilities that I use in conjunction with BASIC programming on my Model III. However, it can be assembled into any convenient location by changing the ORG address, (which is the entry point to the program). The OBJECT code occupies 369 bytes, and an additional 512 bytes, for storage buffers, must be provided at the end, for a total of 1181 bytes. (Before assembling, disk system users should delete line 324, and change line 325 to:

```
00325 BACK    JP      402DH
```

to return to DOS after program initialization or execution.)

You may execute the program from within a BASIC program, or in the Immediate Mode, with the command: NAME!. It is compatible with either cassette- or disk-based systems. If NAME is not immediately followed by an exclamation point, the normal response will occur: (L3 Error for cassette, or the DOS routine for disk). The procedure used to accomplish this was described by Hardin Brothers in his column, The Next Step, in the January 1985 issue of 80 Micro, titled: Teaching Old Basic New Tricks.

I have tried VARLST on a number of my own BASIC programs, and it performed correctly. If anyone runs across any bugs while using it, I would like to know about them.

```
                00001          ;VARIABLE NAME LISTER - (VARLST)
                00002          ;BY BOB KOEHLER
                00003          ;R.D. #4, BOX 174, LAKE WALTON ROAD
                00004          ;HOPEWELL JUNCTION, NEW YORK 12533
                00005          ;RELEASED TO THE PUBLIC DOMAIN 12/23/85
410E            00006 NAME     EQU     410EH           ;ADDR. OF 'NAME' VECTOR
E25F            00007          ORG     57951   ;MAY BE RE-ASSEMBLED ANYWHERE
E25F 2A8F41     00008 SETUP    LD      HL,(NAME+1)     ;GET ORIG. ADDRESS
E262 2276E2     00009          LD      (ORIG+1),HL     ;SAVE IT
E265 216EE2     00010          LD      HL,START        ;GET NEW ADDRESS
E268 228F41     00011          LD      (NAME+1),HL     ;SUBST. NEW FOR ORIG.
E26B C3B1E4     00012          JP      BACK            ;GO TO BASIC 'READY'
E26E F5         00013 START    PUSH    AF              ;SAVE STATUS FLAGS
E26F 7E         00014          LD      A,(HL)          ;GET VALUE
E270 FE21       00015          CP      '!'             ;THIS ROUTINE?
E272 2804       00016          JR      Z,NOW           ;YES
E274 F1         00017          POP     AF              ;ELSE RECOVER STATUS
E275 C30000     00018 ORIG     JP      $-$             ;GO TO ORIG. ROUTINE
E278 21FCE4     00019 NOW      LD      HL,ARBUF        ;SET BUFFER BOUNDARIES
E27B 2B         00020          DEC     HL
E27C 22F5E6     00021          LD      (AREND),HL
E27F 2160E5     00022          LD      HL,SNBUF
E282 2B         00023          DEC     HL
E283 22F7E6     00024          LD      (SNEND),HL
E286 01F401     00025          LD      BC,500          ;EMPTY BUFFERS
E289 21FCE4     00026          LD      HL,ARBUF
E28C 1620       00027 LOOP     LD      D,20H
E28E 72         00028          LD      (HL),D
E28F 23         00029          INC     HL
E290 0B         00030          DEC     BC
E291 78         00031          LD      A,B
E292 B1         00032          OR      C
E293 20F7       00033          JR      NZ,LOOP
E295 CDB7E4     00034          CALL    LINBEG          ;ZERO CHAR. COUNTER
E298 2AA440     00035          LD      HL,(40A4H)      ;START ADDR OF BASIC PROG
E29B 2B         00036          DEC     HL
E29C 23         00037 SRCH     INC     HL
E29D CD88E4     00038          CALL    CRCT            ;INCREMENT CHAR. COUNTER
E2A0 7E         00039          LD      A,(HL)          ;GET CHARACTER
E2A1 FE89       00040          CP      89H             ;INPUT?
E2A3 2839       00041          JR      Z,READ
E2A5 FE8B       00042          CP      8BH             ;READ?
E2A7 2835       00043          JR      Z,READ
E2A9 FE88       00044          CP      88H             ;DATA?
E2AB CA2FE3     00045          JP      Z,DATA
E2AE FE22       00046          CP      22H             ;QUOTATION MARK?
E2B0 CCD4E2     00047          CALL    Z,QUOTE         ;SKIP ANYTHING IN QUOTES
E2B3 FE05       00048          CP      05H             ;EQUALS SIGN?
E2B5 CA3CE3     00049          JP      Z,EQUAL         ;GET THE NAME
E2B8 FE00       00050          CP      0               ;LOOK FOR END
E2BA 2802       00051          JR      Z,END           ;END OF LINE OR PROGRAM?
E2BC 18DE       00052          JR      SRCH            ;KEEP GOING
E2BE 23         00053 END      INC     HL
E2BF 23         00054          INC     HL
E2C0 23         00055          INC     HL              ;END OF PROG. ADDR.?
E2C1 E5         00056          PUSH    HL              ;SAVE ADDRESS
E2C2 EB         00057          EX      DE,HL           ;NOW IN DE
E2C3 2AF940     00058          LD      HL,(40F9H)      ;E.O.P. ADDRESS HERE
E2C6 37         00059          SCF
E2C7 3F         00060          CCF                     ;ZERO CARRY FLAG
E2C8 ED52       00061          SBC     HL,DE
E2CA E1         00062          POP     HL
E2CB CA50E4     00063          JP      Z,PRINT         ;END OF PROGRAM
E2CE CDB7E4     00064          CALL    LINBEG          ;ZERO CHAR. COUNTER
E2D1 23         00065          INC     HL              ;SKIP LINE #'S
E2D2 18C8       00066          JR      SRCH            ;END OF LINE - CONTINUE
E2D4 23         00067 QUOTE    INC     HL
E2D5 CD88E4     00068          CALL    CRCT
E2D8 7E         00069          LD      A,(HL)
E2D9 FE22       00070          CP      22H             ;LOOK FOR CLOSE QUOTE
E2DB C8         00071          RET     Z               ;FOUND IT
E2DC 18F6       00072          JR      QUOTE           ;KEEP LOOKING
E2DE 23         00073 READ     INC     HL
E2DF 7E         00074          LD      A,(HL)          ;NEXT CHARACTER
E2E0 FE20       00075          CP      20H             ;SPACE?
E2E2 28FA       00076          JR      Z,READ          ;SKIP IT
E2E4 FE22       00077          CP      22H             ;QUOTATION MARK?
E2E6 2000       00078          JR      NZ,CONTIN       ;NO
E2E8 CDD4E2     00079          CALL    QUOTE           ;SKIP ANYTHING IN QUOTES
E2EB 23         00080 NXT      INC     HL              ;NEXT CHARACTER
E2EC 7E         00081          LD      A,(HL)
E2ED FE3B       00082          CP      3BH             ;SEMICOLON?
E2EF 28FA       00083          JR      Z,NXT           ;SKIP IT
E2F1 FE20       00084          CP      20H             ;SPACE?
E2F3 28F6       00085          JR      Z,NXT           ;SKIP IT
E2F5 E5         00086 CONTIN   PUSH    HL              ;SAVE 1ST CHAR. ADDRESS
E2F6 010000     00087          LD      BC,0            ;SET CHAR. CT. & VAR TYPE
E2F9 7E         00088 GETNM    LD      A,(HL)
E2FA FE2C       00089          CP      2CH             ;COMMA?
E2FC 2812       00090          JR      Z,SAVE          ;END OF NAME
E2FE FE3A       00091          CP      3AH             ;COLON?
E300 280E       00092          JR      Z,SAVE
E302 FE00       00093          CP      0
E304 280A       00094          JR      Z,SAVE          ;END OF LINE
E306 FE28       00095          CP      28H             ;OPEN PARENTHESIS?
E308 2804       00096          JR      Z,ARYNM         ;IT'S AN ARRAY
E30A 04         00097          INC     B               ;PART OF NAME
E30B 23         00098          INC     HL
E30C 18EB       00099          JR      GETNM           ;GET NEXT CHARACTER
E30E 0E01       00100 ARYNM    LD      C,1
E310 E1         00101 SAVE     POP     HL
E311 2B         00102          DEC     HL
E312 E5         00103          PUSH    HL              ;SAVE AGAIN
E313 CD76E3     00104          CALL    CLRBF           ;STORE IF NOT DUPLICATE
E316 E1         00105          POP     HL
E317 23         00106 LOOP7    INC     HL
E318 7E         00107          LD      A,(HL)
E319 FE20       00108          CP      20H             ;SPACE?
E31B 28FA       00109          JR      Z,LOOP7         ;SKIP IT
E31D FE2C       00110          CP      2CH             ;COMMA?
E31F 2808       00111          JR      Z,COMMA         ;GET NEXT NAME
E321 FE3A       00112          CP      3AH             ;COLON?
E323 CA9CE2     00113          JP      Z,SRCH          ;CONTINUE SEARCH
E326 FE00       00114          CP      0               ;END OF LINE?
E328 2894       00115          JR      Z,END           ;END OF PROGRAM?
E32A 18EB       00116          JR      LOOP7           ;KEEP LOOKING
E32C 23         00117 COMMA    INC     HL              ;SKIP COMMA
E32D 18C6       00118          JR      CONTIN          ;GET NEXT NAME
E32F 23         00119 DATA     INC     HL              ;SKIP OVER ALL DATA LINES
E330 7E         00120          LD      A,(HL)
E331 FE00       00121          CP      0
E333 2889       00122          JR      Z,END           ;END OF LINE
E335 FE22       00123          CP      22H             ;QUOTE?
E337 CCD4E2     00124          CALL    Z,QUOTE         ;SKIP ANYTHING IN QUOTES
E33A 18F3       00125          JR      DATA            ;KEEP LOOKING
E33C E5         00126 EQUAL    PUSH    HL              ;SAVE ADDRESS
E33D 010000     00127          LD      BC,0            ;B COUNTS CHAR'S IN NAME,
```

```
E340 2B       00128          DEC   HL          ;   C HOLDS VARIABLE TYPE
E341 7E       00129          LD    A,(HL)
E342 FE29     00130          CP    29H         ;CLOSE PARENTHESIS?
E344 200F     00131          JR    NZ,NAME1    ;NOT AN ARRAY
E346 CDC2E4   00132          CALL  DECCT       ;DECREMENT CHAR. CNTR.
E349 0E01     00133          LD    C,1         ;IT IS AN ARRAY
E34B 2B       00134 LOOP1    DEC   HL
E34C CDC2E4   00135          CALL  DECCT       ;DECREMENT CHAR. CNTR.
E34F 7E       00136          LD    A,(HL)
E350 FE28     00137          CP    28H         ;OPEN PARENTHESIS?
E352 20F7     00138          JR    NZ,LOOP1    ;NOT YET
E354 2B       00139 NAME2    DEC   HL          ;TEST THE CHARACTER
E355 CDC2E4   00140 NAME1    CALL  DECCT       ;DECREMENT CHAR. CNTR.
E358 3AFBE6   00141          LD    A,(NUMCH)   ;TEST FOR BEG. OF LINE
E35B FE00     00142          CP    0
E35D 2810     00143          JR    Z,BEGIN
E35F 7E       00144          LD    A,(HL)
E360 FE20     00145          CP    20H         ;SPACE?
E362 2008     00146          JR    Z,BEGIN     ;END OF NAME
E364 FE3A     00147          CP    3AH         ;COLON?
E366 2007     00148          JR    Z,BEGIN     ;END OF NAME
E368 FE58     00149          CP    58H         ;END OF ALPHABET
E36A 3003     00150          JR    NC,BEGIN    ;OUT OF RANGE
E36C 04       00151          INC   B
E36D 18E5     00152          JR    NAME2.
E36F CD76E3   00153 BEGIN    CALL  CLRBF
E372 E1       00154          POP   HL
E373 C39CE2   00155          JP    SRCH
E376 23       00156 CLRBF    INC   HL          ;IS NAME VALID?
E377 7E       00157          LD    A,(HL)
E378 FE41     00158          CP    41H         ;MUST BE ALPHA. CHAR.
E37A D8       00159          RET   C           ;NO - TOO LOW
E37B FE5B     00160          CP    5BH
E37D D0       00161          RET   NC          ;NO - TOO HIGH
E37E 2B       00162          DEC   HL
E37F 3E20     00163          LD    A,20H       ;CLEAR BUFFER
E381 32F0E6   00164          LD    (NMBUF),A
E384 32F1E6   00165          LD    (NMBUF+1),A
E387 32F2E6   00166          LD    (NMBUF+2),A
E38A 3E00     00167          LD    A,0
E38C 32F3E6   00168          LD    (NBPOS),A
E38F 32F4E6   00169          LD    (NBPOS+1),A
E392 23       00170 LOOP2    INC   HL
E393 7E       00171          LD    A,(HL)
E394 E5       00172          PUSH  HL
E395 11F0E6   00173          LD    DE,NMBUF    ;BEGINNING OF BUFFER
E398 2AF3E6   00174          LD    HL,(NBPOS)
E39B 19       00175          ADD   HL,DE       ;GET CHAR. POSITION
E39C 77       00176          LD    (HL),A      ;STORE CHARACTER
E39D 2AF3E6   00177          LD    HL,(NBPOS)
E3A0 23       00178          INC   HL
E3A1 22F3E6   00179          LD    (NBPOS),HL
E3A4 E1       00180          POP   HL
E3A5 10EB     00181          DJNZ  LOOP2
E3A7 3E01     00182          LD    A,1         ;TEST FOR TYPE
E3A9 B9       00183          CP    C
E3AA 2009     00184          JR    Z,ARRAY     ;IT'S AN ARRAY
E3AC 2AF7E6   00185          LD    HL,(SNEND)
E3AF 11E0E5   00186          LD    DE,SNBUF
E3B2 EB       00187          EX    DE,HL
E3B3 1807     00188          JR    GO
E3B5 2AF5E6   00189 ARRAY    LD    HL,(AREND)
E3B8 11FCE4   00190          LD    DE,ARBUF
E3BB EB       00191          EX    DE,HL
E3BC C5       00192 GO       PUSH  BC
E3BD E5       00193          PUSH  HL
E3BE D5       00194          PUSH  DE
E3BF DD21F0E6 00195          LD    IX,NMBUF
E3C3 0603     00196          LD    B,3
E3C5 ED52     00197          SBC   HL,DE
E3C7 3808     00198          JR    C,LOCATE    ;BUFFER NOT EMPTY
E3C9 D1       00199          POP   DE
E3CA E1       00200          POP   HL
E3CB 22F9E6   00201          LD    (UROST),HL
E3CE EB       00202          EX    DE,HL
E3CF 1857     00203          JR    FIRST       ;ENTER NAME
E3D1 D1       00204 LOCATE   POP   DE          ;LISTING END
E3D2 E1       00205          POP   HL
E3D3 22F9E6   00206 LOOP4    LD    (UROST),HL
E3D6 3E20     00207          LD    A,20H
E3D8 BE       00208          CP    (HL)
E3D9 2831     00209          JR    Z,STORE     ;END OF LIST
E3DB DD7E00   00210 LOOP3    LD    A,(IX)
E3DE BE       00211          CP    (HL)
E3DF 2816     00212          JR    Z,NXTCHR    ;CHARACTERS MATCH
E3E1 3029     00213          JR    C,STORE     ;PUT NEW NAME NEXT
E3E3 3E20     00214          LD    A,20H       ;END OF STORED NAME?
E3E5 BE       00215          CP    (HL)
E3E6 2819     00216          JR    Z,NXTNM     ;YES
E3E8 2AF9E6   00217 LOOP5    LD    HL,(UROST)
E3EB DD21F0E6 00218          LD    IX,NMBUF
E3EF 0603     00219          LD    B,3
E3F1 23       00220          INC   HL          ;GO FARTHER
E3F2 23       00221          INC   HL
E3F3 23       00222          INC   HL
E3F4 23       00223          INC   HL
E3F5 180C     00224          JR    LOOP4
E3F7 05       00225 NXTCHR   DEC   B
E3F8 2805     00226          JR    Z,DUPL      ;DUPLICATE NAME
E3FA 23       00227          INC   HL
E3FB DD23     00228          INC   IX
E3FD 18DC     00229          JR    LOOP3
E3FF C1       00230 DUPL     POP   BC          ;SEARCH FOR NEXT ONE
E400 C9       00231          RET
E401 2AF9E6   00232 NXTNM    LD    HL,(UROST)
E404 0603     00233          LD    B,3
E406 DD21F0E6 00234          LD    IX,NMBUF
E40A 18DC     00235          JR    LOOP5
E40C D5       00236 STORE    PUSH  DE          ;SAVE END ADDRESS
E40D 37       00237          SCF               ;SET CARRY FLAG
E40E 3F       00238          CCF               ;ZERO IT.
E40F 2AF9E6   00239          LD    HL,(UROST)  ;START. ADDR. OF NEW VRD.
E412 EB       00240          EX    DE,HL       ;START IN DE, END IN HL
E413 ED52     00241          SBC   HL,DE
E415 23       00242          INC   HL          ;# OF BYTES TO MOVE
E416 7C       00243          LD    A,H
E417 B5       00244          OR    L
E418 2800     00245          JR    Z,CONT      ;SKIP IF END OF LIST
E41A E5       00246          PUSH  HL          ;HOLD IT
E41B C1       00247          POP   BC          ;NOW IN BC
E41C E1       00248          POP   HL          ;END ADDRESS
E41D E5       00249          PUSH  HL          ;SAVE AGAIN
E41E 110400   00250          LD    DE,4
E421 19       00251          ADD   HL,DE       ;NEW END IN HL
E422 EB       00252          EX    DE,HL       ;NOW IN DE
E423 E1       00253          POP   HL          ;OLD END IN HL
E424 E5       00254          PUSH  HL          ;SAVE AGAIN
E425 ED88     00255          LDDR              ;MAKE ROOM FOR NEW NAME
E427 E1       00256 CONT     POP   HL          ;OLD LIST END
E428 010400   00257 FIRST    LD    BC,4
E42B 09       00258          ADD   HL,BC
E42C EB       00259          EX    DE,HL       ;NEW LIST END IN DE
E42D 2AF9E6   00260          LD    HL,(UROST)
E430 DD21F0E6 00261          LD    IX,NMBUF
E434 0603     00262          LD    B,3
E436 DD7E00   00263 LOOP6    LD    A,(IX)      ;PUT NEW NAME IN TABLE
E439 77       00264          LD    (HL),A
E43A DD23     00265          INC   IX
E43C 23       00266          INC   HL
E43D 10F7     00267          DJNZ  LOOP6
E43F C1       00268          POP   BC          ;GET VARIABLE TYPE
E440 EB       00269          EX    DE,HL
E441 3E01     00270          LD    A,1
E443 B9       00271          CP    C
E444 2005     00272          JR    Z,ARYEND    ;IT'S AN ARRAY
E446 22F7E6   00273          LD    (SNEND),HL  ;PUT IN NEW END
E449 1803     00274          JR    RTN
E44B 22F5E6   00275 ARYEND   LD    (AREND),HL
E44E EB       00276 RTN      EX    DE,HL
E44F C9       00277          RET
E450 CDC901   00278 PRINT    CALL  1C9H        ;CLS
E453 21C9E4   00279          LD    HL,ARMSG    ;ARRAY HEADING
E456 CD752B   00280          CALL  2B75H       ;PRINT HEADING
E459 3E00     00281          LD    A,0DH
E45B CD3300   00282          CALL  33H         ;LINE FEED
E45E CD3300   00283          CALL  33H
E461 2AF5E6   00284          LD    HL,(AREND)  ;END OF ARRAY LIST
E464 11FCE4   00285          LD    DE,ARBUF    ;BEGINNING
E467 ED52     00286          SBC   HL,DE
E469 23       00287          INC   HL          ;# OF BYTES IN LIST
E46A 7C       00288          LD    A,H
E46B B0       00289          CP    L
```

```
E46C 2808      00290        JR     Z,FEED          ;NO ARRAYS
E46E E5        00291        PUSH   HL
E46F C1        00292        POP    BC              ;NOW HERE
E470 2A4040    00293        LD     HL,(16416)      ;CURSOR LOCATION
E473 EB        00294        EX     DE,HL
E474 EDB0      00295        LDIR                   ;MOVE LIST TO SCREEN
E476 3E0D      00296 FEED   LD     A,0DH
E478 CD3300    00297        CALL   33H             ;3 LINE FEEDS
E47B CD3300    00298        CALL   33H
E47E CD3300    00299        CALL   33H
E481 21E2E4    00300        LD     HL,SNMSG        ;VARIABLE NAMES HEADING
E484 CD7528    00301        CALL   2875H           ;PRINT HEADING
E487 3E0D      00302        LD     A,0DH
E489 CD3300    00303        CALL   33H             ;2 LINE FEEDS
E48C CD3300    00304        CALL   33H
E48F 2AF7E6    00305        LD     HL,(SNEND)
E492 1160E5    00306        LD     DE,SNBUF
E495 ED52      00307        SBC    HL,DE
E497 23        00308        INC    HL
E498 E5        00309        PUSH   HL
E499 C1        00310        POP    BC
E49A C5        00311        PUSH   BC              ;SAVE IT AGAIN
E49B 1160E5    00312        LD     DE,SNBUF
E49E 2A4040    00313        LD     HL,(16416)      ;CURSOR LOCATION
E4A1 E5        00314        PUSH   HL              ;SAVE CURSOR LOCATION
E4A2 EB        00315        EX     DE,HL
E4A3 EDB0      00316        LDIR                   ;MOVE LIST TO SCREEN
E4A5 E1        00317        POP    HL              ;ORIG. CURSOR LOCATION
E4A6 C1        00318        POP    BC              ;# OF BYTES IN LIST
E4A7 09        00319        ADD    HL,BC           ;END OF LIST
E4A8 224040    00320        LD     (16416),HL      ;RELOCATE CURSOR AT END
E4AB 3E0D      00321        LD     A,0DH
E4AD CD3300    00322        CALL   33H             ;LINE FEED
E4B0 D1        00323        POP    DE              ;RESTORE STACK
E4B1 011B1A    00324 BACK   LD     BC,1A1BH
E4B4 C3AE19    00325        JP     19AEH           ;BACK TO BASIC READY
E4B7 08        00326 LINBEG EX     AF,AF'
E4B8 AF        00327        XOR    A
E4B9 08        00328        EX     AF,AF'
E4BA C9        00329        RET
E4BB 08        00330 CRCT   EX     AF,AF'
E4BC 3C        00331        INC    A
E4BD 32FBE6    00332        LD     (NUMCH),A
E4C0 08        00333        EX     AF,AF'
E4C1 C9        00334        RET
E4C2 08        00335 DECCT  EX     AF,AF'
E4C3 3D        00336        DEC    A
E4C4 32FBE6    00337        LD     (NUMCH),A
E4C7 08        00338        EX     AF,AF'
E4C8 C9        00339        RET
E4C9 4C        00340 ARMSG  DEFM   'LIST OF ARRAY VARIABLES:'
    49 53 54 20 4F 46 20 41 52 52 41 59 20 56 41 52
    49 41 42 4C 45 53 3A
E4E1 00        00341        DEFB   0
E4E2 4C        00342 SNMSG  DEFM   'LIST OF SINGLE VARIABLES:'
    49 53 54 20 4F 46 20 53 49 4E 47 4C 45 20 56 41
    52 49 41 42 4C 45 53 3A
E4FB 00        00343        DEFB   0
0064           00344 ARBUF  DEFS   100
0190           00345 SNBUF  DEFS   400
0003           00346 NMBUF  DEFS   3
0002           00347 NBPOS  DEFS   2
0002           00348 AREND  DEFS   2
0002           00349 SNEND  DEFS   2
0002           00350 WRDST  DEFS   2
0001           00351 NUMCH  DEFS   1
E25F           00352        END    SETUP
00000 TOTAL ERRORS
```

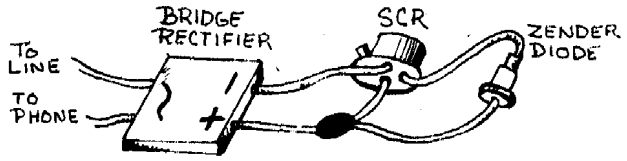
```
ARBUF   E4FC   AREND   E6F5   ARMSG   E4C9   ARRAY   E3B5   ARYEND  E44B
ARYNM   E30E   BACK    E4B1   BEGIN   E36F   CLRBF   E376   COMMA   E32C
CONT    E427   CONTIN  E2F5   CRCT    E4BB   DATA    E32F   DECCT   E4C2
DUPL    E3FF   END     E28E   EQUAL   E33C   FEED    E476   FIRST   E428
GETNM   E2F9   GO      E3BC   LINBEG  E4B7   LOCATE  E3D1   LOOP    E28C
LOOP1   E348   LOOP2   E392   LOOP3   E308   LOOP4   E3D3   LOOP5   E3E8
LOOP6   E436   LOOP7   E317   NAME    418E   NAME1   E355   NAME2   E354
NBPOS   E6F3   NMBUF   E6F0   NOW     E278   NUMCH   E6FB   NXT     E2EB
NXTCHR  E3F7   NXTNM   E401   ORIG    E275   PRINT   E450   QUOTE   E2D4
READ    E2DE   RTN     E44E   SAVE    E310   SETUP   E25F   SNBUF   E560
SNEND   E6F7   SNMSG   E4E2   SRCH    E29C   START   E26E   STORE   E40C
WRDST   E6F9
```

NO MORE MODEM CALL INTERRUPTIONS!
by Jack Decker

Have you ever been in the process of downloading a long file from CompuServe or a long distance BBS, and just as the last part of the file was being downloaded, you were knocked off the line by someone picking up an extension telephone? No, the solution is not to break the arm of the person who picked up the phone, nor to toss all the extension phones out the window (though you may have felt like both at the time). What you need is a privacy adapter. Install one on every phone in your home and no one can interrupt your modem calls. As a bonus, no one can listen in to your private calls from an extension. There is one drawback - you can't pick up a phone and yell at your teenage daughter to get off of her bedroom extension phone so you can make a call (unless you have a phone that does not have a privacy adapter installed).

The privacy adapter basically operates on telephone line voltage. When no phones are in use, the line voltage is about 48 volts D.C. However, when a phone is picked up, line voltage drops to about six volts. The privacy adapter takes advantage of this situation. When a phone is first taken off the hook, the privacy adapter (which is wired in series with the phone) will not make the connection to the phone line unless the higher on-hook voltage is present.

Here's the circuit, with a few warnings and other information first: 1) Don't laugh at the diagram too hard, I'm not a hardware hacker. I built two of these from junkbox parts and they both worked. If someone would care to re-draw this in correct schematic diagram format, I'll publish it in a future issue. In the meantime, this diagram should sufficient for all true hardware hackers to figure it out. 2) If you build this and somehow manage to louse up your phone line or offend your local phone company (or do any other mayhem), I am not responsible. This circuit should be assumed to be buggy, and probably has a serious design flaw that someone more knowledgeable than I will immediately point out. All I know is that it worked for me, and if you build it you should consider it an experimental circuit and test it carefully before putting it on an important telephone line. Also, you should check with your phone company to find out if (and under what conditions) you are permitted to attach a user-constructed device to your telephone line (this last sentence is included to keep me out of legal hot water). 3) This adapter may be installed inside the phone, or anywhere along the phone line in series with the phone (break the connection of either the red or the green line wire and insert this in series). You need one for each phone that needs to be restricted from accessing a line that is already in use. 4) There may be better ways to do this, and I am open to suggestions. As I have stated, I built this from available junkbox parts, which is why I cannot give exact part numbers.



PARTS LIST:

Bridge Rectifier (four silicon diodes wired in a bridge rectifier configuration may also be used) rated 1 amp at 400 PIV or greater.

Silicon Controlled Rectifier, rated 1 amp at 400 PIV or greater.

Zener Diode (lowest amperage unit you can get should be okay). Anything from 9 through about 24 volts should work. If the phone will break into a conversation in progress on another phone, use a higher voltage. If the phone will not access the line at all and you have used a fairly high voltage unit, try a lower voltage.

INSTALLING MEMDISK AS THE SYSTEM DRIVE IN LESS THAN 40 SECONDS

by Ronald A. Chill - Brooklyn, New York

[This article is reprinted from THE INTERFACE newsletter of the San Gabriel Valley TRS-80 Users Group.]

1. Initialize MEMDISK as drive 2.

```
SYSTEM (DRIVE=2,DRIVER="MEMDISK")
        A)  Which type of allocation ? D
        B)  Single or Double Density ? D
        C)  Do you wish to Format it ? Y
```

2. Move all system files to MEMDISK.

```
BACKUP S$/SYS:0 :2(S,I)
```

3. REMOVE SYS0/SYS and SYS13/SYS from MEMDISK, they are not needed.

```
REMOVE SYS0/SYS.LSIDOS:2
REMOVE SYS13/SYS.LSIDOS:2
```

4. MOVE QFB6/CMD TO MEMDISK

The file QFB6/CMD can be found on Radio Shack's TRSDOS 6.2 Utilities disk, catalog #26-315. This file is a QUICK FORMAT and BACKUP utility which doesn't require the files FORMAT/CMD or BACKUP/CMD to copy a disk.

```
COPY QFB6/CMD:0 :2
```

5. BUILD a file called NEWDRIVE/JCL as follows from DOS READY:

```
BUILD NEWDRIVE:2
SYSTEM (DRIVE=2,DRIVER="MEMDISK")
D
D
Y
BACKUP :1 :2
SYSTEM (DRIVE=2,WP)
SYSTEM (SYSTEM=2)
(press the BREAK KEY)
```

6. BUILD a file called OLDDRIVE/JCL as follows from DOS READY:

```
BUILD OLDDRIVE:2
SYSTEM (DRIVE=2,DRIVER="MEMDISK")
D
D
N
SYSTEM (DRIVE=2,WP)
SYSTEM (SYSTEM=2)
(press the BREAK KEY)
```

7. CREATING A BACKUP COPY OF MEMDISK:

```
Place a blank disk in drive 1.
FORMAT :1(Q=N,ABS)
QFB6 :2 :1
COPY DIR/SYS.LSIDOS:2 :1
```

You are creating an exact copy of the 14 tracks found on MEMDISK. If you look at the DIRectory of drive 1, before and after running QFB6/CMD, you will NOT see any changes. In order to find the DIRectory we fool TRSDOS by copying the MEMDISK DIRectory to the normal DIRectory location, track 20. The system in turn is fooled into thinking that the disk in drive 1 is ONLY 14 tracks. The disk created in drive 1 has two DIRectories, one for use by MEMDISK and one used by the system on track 20. If you have a disk zap utility, take a look at both tracks 1 and 20 to convince yourself.

USING NEWDRIVE AND OLDDRIVE

When you want MEMDISK to be installed as the SYSTEM, place the 14 track disk in drive 1, and type the following from DOS READY:

```
DO = NEWDRIVE
```

If it becomes necessary to RE-BOOT the SYSTEM and the second bank of 64K of memory was not used you can restore MEMDISK as the SYSTEM in even less time by typing:

```
DO = OLDDRIVE
```

One final note, this paper talks about installing MEMDISK as drive 2 initially. If you have more than a two drive system then change all references to drive 2. For a four drive system, all references to the 2 should be changed to a 4. Remember that when MEMDISK becomes the system, drive 0 becomes the drive number that was formally used by MEMDISK.

TIDBYTES ON VISICALC

[Reprinted from the K.C. South Computer Club newsletter.]

VisiCalc users with Model I/III have the ability to change printer format from within the VisiCalc program. This feature, although documented in the Model III VisiCalc manual, is obscure in its usage and syntax. If you have a compressed mode (16.5 cpi) on your printer, you probably want to take advantage of the extra columns that can be produced in this mode. You can set your printer parameters to the compressed mode before loading VisiCalc and when you PRINT, get condensed type output, OR, here is how to do it from within the VisiCalc program.

Type /PP" (don't omit the quote character) which will give you the prompt, "Setup or ENTER". Now press the <SHIFT> and <@> at the same time. A caret will appear on your Model III Edit line; Model I users will see a right arrow. Type H1B (H indicates that a hex value will follow, 1B is the hex value for an Escape <27 decimal>). Press <SHIFT> <@> again, and your caret or right arrow will appear again. Now type H50 and <ENTER>. You are prompted for the lower right cell designation for the print. Either type the cell you want followed by <ENTER>, or move the cursor to the cell with the arrow keys and then press <ENTER>.

Your output should be in the compressed mode if you are using an Epson printer. If you have some other printer, substitute the value you need to accomplish this task for the second hex value (the H50). This information can be found in the section of your printer's manual that discusses software control codes. Multiple commands can be entered by this method by continuing the sequence of <SHIFT> <@> followed by the next series of hex characters.

MODEL 4 TRSDOS 6 WRITE PROTECT

by Dick Rechlicz

[This article is reprinted from the Milwaukee Area TRS-80 User's Group (M.A.T.U.G) newsletter.]

When running business applications programs, we have a long standing practice to software write-protect the diskettes in our drive zero. We've experienced a few instances when a write-protect tab has been installed on a diskette, yet the computer has written to that disk - or killed files - without our first removing the tab. [NORTHERN BYTES editor's note: This is impossible with most brands of disk drive - the hardware simply won't permit it - but I suppose that there may be the oddball drive that works differently. Also, power line glitches can sometimes do strange things, even to a write-protected diskette.]

While anything is possible, we've had no problem using the DOS to write protect the diskette. This practice is commonly referred to as "software" write-protecting.

With DOSPLUS IV, the procedure to software write-protect drive zero as a default is relatively simple. Configure a disk, save the configuration file to another disk (on another drive), then copy the configuration file back to the original disk.

Recently when transferring to the TRSDOS 6 disk operating system, we quickly learned that - supposedly - it is not possible to write-protect drive zero and have it "SYSGEN" upon booting. We contacted Logical Systems, Inc., for help - and they provided the answer. We'd like to share that answer with you.

Locate the CONFIG/SYS file with DISKDUMP (or whatever monitor you're accustomed to using). Find the following sequence:

```
Ø1 52 ØØ 43 C3 ?? ?? 44
```

Change the "44" to "C4" and you've write-protected drive zero as a default condition.

If you should want to write to that disk, it is possible to bypass SYSGEN by pressing and holding <ENTER> when booting. Or use the SYSGEN command:

```
SYSGEN (DRIVE=Ø, WP=N)
```

SPEED UP YOUR PULSE-DIAL TELEPHONE
by Jack Decker

Okay, so this isn't exactly computer-related, but it might help enhance your standing with your wife (or others who see no practical value in your hardware tinkering).

Some of the new electronic telephones have pushbutton dialing but produce dial pulses rather than touch tones. This permits you to have the convenience of pushbutton dialing in areas where tone dialing is not available. However, many people buy these phones even in areas where tone dialing is available. The reason is simple - the telephone company charges extra to provide tone dial service on a telephone line (even though, in many areas, it doesn't cost them one cent more to provide the service since the tone decoders are part of their central office switching equipment anyway).

If you have such a phone, you may be able to speed up the pulse dialing speed and be able to dial numbers at near tone-dial speed, although your phone will still actually be producing the cheaper dial pulses. The big "if" is whether your local telephone exchange can handle the faster speed.

If your phone is connected to an electronic or to a crossbar central office, it should work. If, on the other hand, your phone is connected to an older-style step-by-step or to a panel central office (the latter are nearly extinct), you are pretty much stuck with the slower speed. How do you tell? If you're on friendly terms with the phone company, they might tell you. But here are some clues that, while not 100% accurate, can help you determine what type of office you're on (note: Custom Calling Services refer to services such as Call Waiting, Call Forwarding, and Three-Way Dialing):

1) Step-by-step: You will hear the noise of stepping relays (sounds similar to a playing card stuck in bicycle spokes) when dialing numbers, after some (but not all) digits are dialed. You may also get a relatively high percentage of wrong numbers or other phone problems on this type system. Custom calling services will not be available, but equal access to alternate long distance carriers and international direct dialing may be available. If you are serviced by a non-Bell phone company, and/or live in a rural area, the chances of you being on this type of system are much higher.

2) Crossbar: You will NOT hear relay noise while dialing (except as very low level background static), but should hear two or more clicks immediately after dialing the last digit. Custom calling services will not be available. Your exchange will be one of the last in your area to get equal access, and probably does not have international direct dialing (crossbar can't handle a varying number of digits in a phone number without added equipment). You may occasionally get a slow response on a dial tone, but rarely will you get a wrong number (when dialing in your same exchange).

3) Electronic: You will NOT hear relay noise while dialing (except as low-level static) but should hear two clicks after dialing the last digit of a phone number. These two clicks will be almost instantaneous after dialing another number on your exchange, but may be delayed by a few seconds when dialing long distance, or into an older exchange or a large business with direct inward dialing to extensions. You should almost always get dial tone within a fraction of a second after picking up the phone. Custom calling services, international direct dialing, and equal ("dial 1") access to alternate long distance carriers will all be available.

If you're still not sure, try the modification and if it doesn't work (you can't dial out or start getting lots of wrong numbers), you can always put everything back as it was.

How do you do it? Well, first you open up the phone in question and write down the manufacturer and part identification on each and every integrated circuit inside the phone (hopefully there will only be one or two). You then put the phone back together, and contact the manufacturer or the manufacturer's representative for data sheets on the integrated circuit(s) in question. In some cases you will be stopped right here, since some IC's bear no indication of the chip manufacturer.

But, assuming you're successful, one of the IC's should be identified as a pulse-dialer IC (or something similar). This is the one you want. Now check the data sheet and you should find that one pin controls the outpulsing speed. If connected to one side of the voltage supply, it will cause the chip to outpulse at 10 pulses per second (PPS). However, if connected to the opposite side of the voltage supply, the outpulse rate will be 20 pulses per second. Now, any hardware hacker worth his salt should be able to figure it out from that - you just break the connection to that one pin and connect the pin to the opposite voltage.

You may have a couple of other options, such as interdigital pause and/or make/break ratio that are controlled by different pins. The make/break ratio should NOT be changed - it usually won't make any difference, but if the phone was sold for use in the United States or Canada, it should be preset to the correct make/break ratio (normally between 58% and 64% "break"). Interdigital pause is another story - if it's not already set to the shortest value (usually expressed in milliseconds), change it. If you find you get wrong numbers, back off the interdigital pause first, since a too-short interdigital pause will sometimes cause problems in systems that would otherwise accept the faster dialing speed.

Now I'll save a few of you the trouble of having to write or call for the data sheets. If your telephone contains a Sharp LR-40992 pulse dialer IC (as do some of the Conair telephones sold by K-Mart and other stores), the pin to change is pin 10 (located at the lower right corner of the chip). It is now connected to V- (pin 6) and you want to change it to be connected to V+ (pin 1). Make sure you do NOT also change the make/break select ratio on pin 11 (pins 10 and 11 are tied together in the Conair phones, you must break this connection and then reconnect pin 11 to the V- supply).

If you are able to get data sheets for any other pulse dialer IC's, why not send me a copy of the pinout information and I may publish a follow-up article.

Some of you modem users may be wondering if you can speed up your auto-dial modem in a similar manner. If the modem uses a pulse-dialer chip, the answer is yes, and the procedure is the same. However, if the modem dials by operating a line-connect relay, you'd probably have to change the stored program within the modem's ROM to accomplish this. Again, if any of our readers have any information on speeding up the pulse dial rate of a specific modem, I'd like to hear about it!

TRSDOS 6.2 /HLP FILE FORMATS REVEALED
by Michael R. Johnston

The following is a description of the format the HELP/CMD utility expects /HLP files to be (from bottom to top):

NOTE: All /HLP files must have an LRL of one.

1) The last two records have the address of the HELP file directory in LSB/MSB format.

2) Moving to the start of the file directory (this is what is displayed on the screen when you enter for example, "HELP DOS" from TRSDOS READY) you will find each directory entry in ASCII FORMAT. The last character of the entry will have bit 7 set as an end of entry marker. The two bytes following the entry give the starting record number where the help text for that particular category can be found. These are also in LSB/MSB format. It is interesting to note that the categories MUST be written to the file in ALPHABETICAL ORDER! If this is not done, HELP/CMD will fail to access ALL entries after the first one that is out of order. They WILL however, be displayed in the file directory. One more caution; ALWAYS use UPPER CASE letters for the directory entries! Failure to do this will result in the same problem as described above.

3) The help text for each individual category within the file terminated with the HEX values 0D,0D,0D and 0C IN THAT ORDER. Within each category the following control codes are used:

a) The LAST letter of each word that is followed by a space may have bit 7 set. This eliminates the need to write the space.

b) For multiple blocks of spaces the SPACE COMPRESSION codes are utilized, i.e. a C0H value will produce ONE space, a C1H will produce two spaces etc.

c) To toggle INVERSE VIDEO on/off a code 7FH is used. The first use of the code will toggle it on, the second off and so on.

An example of a file follows:

> Main text of each category within the file, terminated by: 0DH,0DH,0DH,0CH
>
> The file directory containing each individual category with bit 7 of the last character of the category SET, followed by the two bytes pointing to the start of the individual category's text.
>
> The last two bytes of the file, pointing to the start of the file directory.

And that is about all there is to it! The hard part of all this is the typing involved to MAKE the help files. For those of you who hate typing, I have several HELP files that I have already created for my system, with more in the works. Just think of being able to COMPLETELY CLEAR your desk of those dog-eared manuals. For you TRSDOS programmers I have nearly completed a set of machine language reference files, for example: OPCODES/HLP, PORTS/HLP, SVCCALL/HLP, etc. If you are interested in obtaining these files, send a disk in a mailer with return postage and a self addressed mailing label to the following address:

Michael R. Johnston
E. Co. 4th Spt. Bn.
APO New York 09185

A SAD TALE WITH A HAPPY ENDING

by Tom Moody

[Reprinted from Data Important to Members Everywhere (DIME), the newsletter of the Northern Illinois Computer Owners League (NICOL).]

I have a few programs that I wrote for the company I work for. Most of them are used by several people that just 'love' to get into them and 'tinker', then all of a sudden I get a call that 'the program YOU wrote doesn't work anymore'. Well I got tired of this and Model 4 BASIC lets you save programs 'protected' with the command!

SAVE "FILENAME",P

This is SUPER. Now not everybody can get into the program and change it from what I wrote. Now my problems are all gone! Well, almost all gone.

After several months of joy, (no more 'it doesn't work' calls) I needed to modify one of the programs myself. No problem, I always keep a copy that is not protected. I hunt through piles and piles of disks. No copy to be found. Panic time. Wait, be calm now, there must be an answer to this dilemma. I thought of cutting my wrist but that had one problem, I can't stand the sight of blood. Oh well, I guess all that can be done is to write the program over from scratch, but that will take days of work and so I sat and thought about it for a while.

The more I thought about it the more I realized there had to be a way to get at that protected program after it was in memory. So, after a couple of hours with DEBUG looking around in BASIC, I came up with a way to do it.

First type DEBUG (ON) <ENTER>
Then type BASIC.BASIC <ENTER>

You will now see the DEBUG display on your screen but we are not ready to use DEBUG yet so we will continue by typing:

G<ENTER>

You will now get the signon-message for BASIC so you can go ahead and load your protected program by typing:

LOAD "filename"<ENTER>

Now we are going to recover the file so we can list it, save it or whatever we needed to do to it.

Hit the <BREAK> key and you should be back in DEBUG again

To Un-protect the program in memory we must change the byte at address 72CBH to 00. To do this from DEBUG,

Type H72CB<SPACE-BAR>
00<SPACE-BAR>
<ENTER>

Now we are ready to return to BASIC. Do this by typing

G<ENTER>

You should now be able to list your program. I would advise you to save it to another filename right away in a normal manner before you do anything else. That way if anything goes wrong you will not have to do this over again.

I hope none of you ever have to resort to this procedure, but it is nice to know there is a way out if you really need it.

NEW BOOT/SYS FOR DOSPLUS 3.50

by Oswald Cooper

[This article is reprinted from Bits and Pieces, the newsletter of The Northeast Computer Club.]

The file BOOT43/CIM is a replacement boot file for use on DOSPLUS 3.50 to be able to boot on the TRS-80 Model 4P in the Model III mode without the need to switch disks.

To create a usable disk, use the following procedure:

1. SYSGEN a new master disk
2. Copy the MODELA/III file to the new disk.
3. Run the DISKZAP Utility. Display Cylinder 0, Sector 1. Press "M" to enter modify mode. Type in the following data:

```
RELATIVE BYTE                                          ASCII EQUIVALENT
ADDRESS:                                               FOR REFERENCE ONLY
00: 00 FE 14 3E C3 32 49 40  21 E0 43 22 4A 40 11 02  ...>.2I@!.C"J@..
10: 00 21 00 52 CD A8 43 20  58 E5 DD E1 DD 56 0A 1E  .!.R..C X....V..
20: 04 24 CD A8 43 20 40 7E  2F E6 50 20 47 D9 2A 16  .$..C @~/.P G.*.
30: 53 55 7C 07 07 07 E6 07  28 07 47 AF D0 86 07 10  SU|.....(.G.....
40: FB 47 21 FF 53 D9 CD 82  43 67 CD 82 43 47 CD 82  .G!.S...Cg..CG..
50: 43 6F 25 28 10 25 20 05  CD 82 43 67 E9 05 28 E6  Co%(.% ...Cg..(.
60: CD 82 43 18 F8 CD 82 43  67 05 05 CD 82 43 77 23  ..C....Cg....Cw#
70: 10 F9 18 D2 21 F5 43 06  0B 7E 23 CD 33 00 10 F9  ....!.C..~#.3...
80: 10 FE D9 2C 20 1F E5 21  B6 43 CB A6 58 78 DD 96  ..., ..!.C..X{..
90: 05 38 03 CB E6 5F E1 CD  A8 43 20 D8 04 78 DD 96  .8..._...C ..x..
A0: 09 38 02 47 14 7E D9 C9  CD AC 43 CB C5 D5 E5 7B  .8.G.~....C....{
B0: D3 F2 7A D3 F3 3E 81 D3  F4 57 CB F2 1E 02 0E F3  ..z..>...W......
C0: 3E 13 CD EE 43 DB F0 0F  38 FB 3E 88 CD EE 43 3E  >...C...8.>...C>
D0: 80 D3 E4 7A D3 F4 DB F0  A3 28 F8 ED A2 C3 03 43  ...z.....(.....C
E0: F1 01 E4 00 ED 41 DB F0  E6 9C E1 D1 C1 C9 03 F0  .....A..........
F0: 06 0E 10 FE C9 44 49 53  4B 20 45 52 52 4F 52 21  .....DISK ERROR!
```

When finished, press <ENTER> to save the sector to disk. Enter BASIC, type in and run the following program:

```
5 'BOOT43/BAS
10 CLS
15 TOTAL#=0#
20 OPEN"R",1,"BOOT43/CIM:1",1
30 FIELD #1, 1 AS A$
40 FOR X=1 TO LOF(1):GET 1,X
50 REM PRINT HEX$(ASC(A$)),
55 PRINT USING"\    \";HEX$(ASC(A$));
60 TOTAL#=TOTAL#+ASC(A$)
100 NEXT
110 CLOSE
120 PRINT:PRINT TOTAL#
130 IF TOTAL#<>30360# THEN PRINT "ERROR"
      ELSE PRINT "VALUES CHECK"
```

The new boot has also been tested on a Model III and a Model 4, and works without problems.

CONFIGURE MODEM80 TO SUIT YOUR SYSTEM

by Leonard Yates

[Reprinted from SYDTRUG NEWS, P.O. Box 297, Padstow, New South Wales 2211, Australia.]

(a) Hard-Configure GRAPHICS ON: Since the Club-80 BBS now has graphics capabilities, you may like to zap your copy of MODEM80 as follows so that it initializes in the GRAPHICS ON mode:

MODEM80/CMD,05,B1 change E6 7F 21 to E6 FF 21
MODEM80/CMD,28,CB change 4F 46 46 0A to 4F 4E 20 0A

(Don't forget to reset GRAPHICS OFF if accessing a remote system which checks parity.)

(b) Configuration to Suit the AVTEK Multi-Modem: To set high the DTR and RTS signals of the RS-232C serial port; you cannot use MODEM80 with an AVTEK Modem unless you make the following changes:

MODEM80/CMD,04,3D change 55 6D 3E to 55 6C 3E
MODEM80/CMD,35,D6 change 3E 6D D3 to 3E 6C D3

(c) Configuration for NEWDOS/80 version 2.0 / Model III: In its original form MODEM80 does not allow use of the option '<C> = Accept DOS Command' when using NEWDOS/80 version 2.0 on a Model III. A way of accessing the DIR command is given on page 32 of the MODEM80 manual; the following method allows all DOS commands to be accessed:

MODEM80/CMD,12,58H change 3D 28 09 to 3D 20 09

Note: If you're modifying MODEM48/CMD or a DUMPed version of MODEM80, the relative byte in each ot the above zaps should be decreased by 8. For example, byte B1 would become A9, etc.

PRINTER SWITCH PROJECT

by Bill Baker

[This (somewhat abridged and edited) article is reprinted from The Cursor, newsletter of the K. C. South Computer Club. PLEASE NOTE that we at NORTHERN BYTES have not built or tested this project, thus we present it with the usual warning that we assume NO responsibility if you attempt to build or use this project and damage your computer, your printer, or any other part of your system.]

This project provides the materials required and some basic construction notes on how to build a printer switch for your Model 1, 3, 4, or 1000 Radio Shack computer.

The Supplies required are as follows:

1. 34 conductor flat ribbon cable, minimum 3 feet, maximum to suit your needs (approximately 50¢/foot)
2. One 34 pin Card Edge Connector ($4.00 at R.S.)
3. Two 34 pin Socket Connectors ($3.19 each at R.S.)
4. One 4PDT switch. Can be toggle, rotary, slide or other. You may substitute two DPDT switches if you want to (part #275-07, 2 for 79¢ at R.S.)
5. One mounting box approx 2-3/4 L x 2-1/8 W x 1-5/8 H (approximately $1.50 - BUD #CU 2100A, JMB #T-F770 or J-870, HAMMOND #1411B, etc.)

Several companies sell printer switches that allow you to switch one computer to two different printers. TAB has one available at $69.95, HF Signaling at $59.95, and Radio Shack at $119.95. They are all reliable products that do their job well.

My project of a home constructed switch can save you money. This project may not be for everyone, but is NOT difficult and if you can use a screwdriver, long nose pliers, and a soldering iron, you should have no problems building it. The project will probably require about two hours of your time. Materials can be obtained from several sources, including Radio Shack.

Lay the 3' cable out flat and measuring from the right edge, mark the cable at 6", 12", and 18" with a felt marker pen. See Fig. 1.

Install a 34 pin Socket Connector at lines B and D. Position connector B on the opposite side of the cable as connector D. See Fig. 2. Install a 34 pin Card Edge Connector at the open end of the cable. See Fig. 2.

The Red wire in ribbon cable is the #1 wire. Counting over from #1, locate and mark wires #21, 25, 28. Recount and identify these wires again to make sure you have the correct ones marked.

Use a sharp cutting device such as a utility knife or a single edge razor blade and carefully cut through the plastic grove in between the 4 marked wires, separating them from the rest of the cable. You will separate these 4 wires, each about 5" long, at 3 locations on the cable as identified in Fig. 1. From line A to the left side of Socket B; from the right side of Socket B to line C; and from line C to Socket D.

After separating these wires you need to cut one end free as follows:

1. Cut the 4 wires at the left side of Socket B
2. Cut the 4 wires at the left of line C
3. Cut the 4 wires at the left side of Socket D

The free ends of these 12 wires should be as shown in Fig. 3.

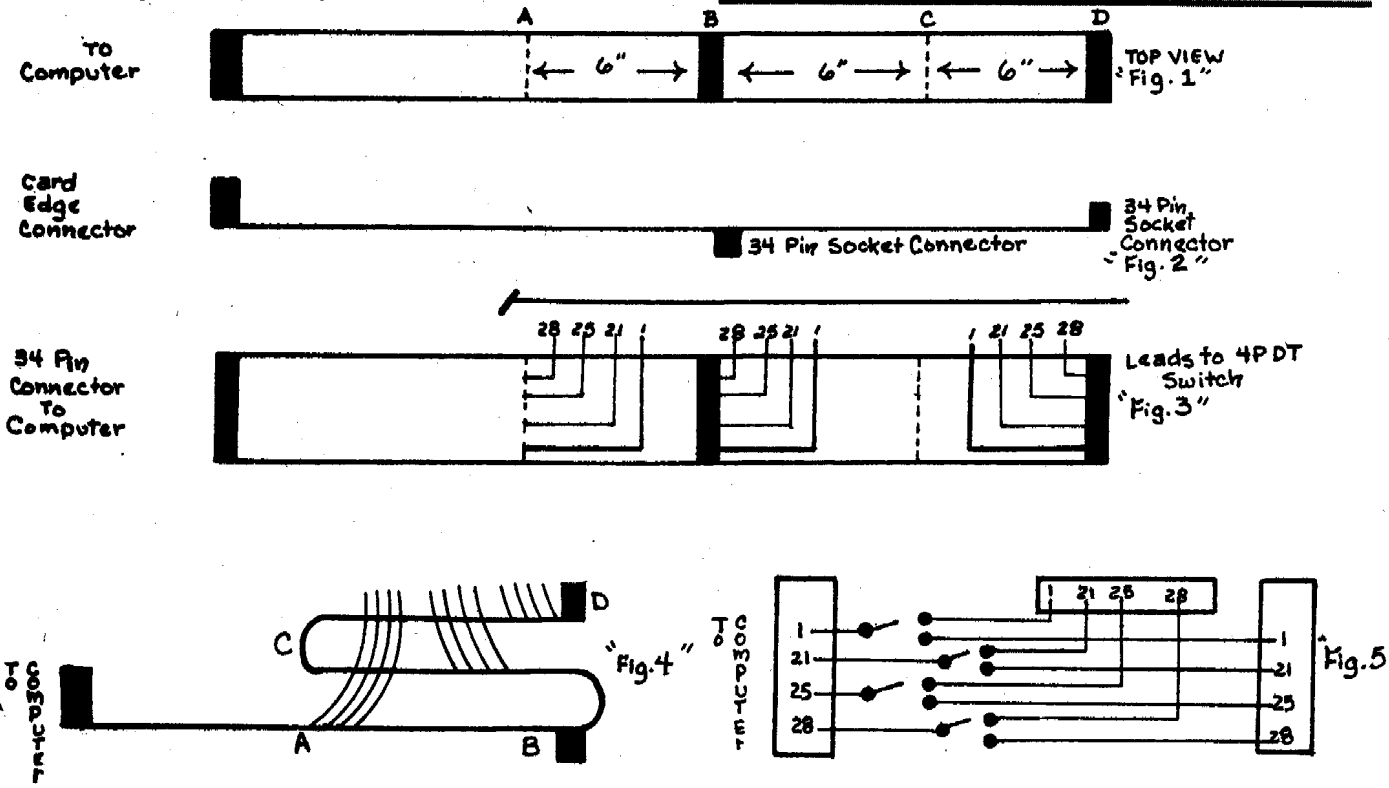
Fold the cable back on itself as shown in Fig. 4 and dress the free end of the 12 wires up through the slots left in the cable from the separation of the wires. These 12 ends will be connected to the 4PDT switch.

Connect and solder the 4 wires from line A to the 4 common contacts on the switch. Solder the wires from Socket B to one set of corresponding make contacts on the switch and solder the last 4 wires from Socket D to the other set of make contacts of the switch.

Before mounting the switch in the mini-box, check for continuity of the switched leads. Use an ohmmeter or other continuity checker, one side connected to pin 1 of the Card Edge connector and the other side to pin 1 of Socket B. If you don't get continuity, operate your switch to the other position. When you have established continuity, repeat the test for pins 21, 25, and 28. Then operate the switch and check these same 4 leads from the Card Edge Connector to the other Socket D.

Mount the Switch in the Mini-Box and the project is complete. You will also need to either modify your existing printer cables or make up new cables with a connector that will mate with the Socket connectors on the Printer Switch you constructed. [If you could locate female edge connectors that can be pressed onto a ribbon cable and that provide the same pinout as the printer port edge connector on your computer, you could use your original printer cables. However, this type of connector may be harder to come by than the 34 pin socket connectors specified for this project. - editor]

Your cost to construct this switch may vary [you can save money by using surplus cables or components if available], and could range up to $15-$17 if you purchase the components separately. Even at $17, it is still a far cry from the $60 to $120 commercial versions.



TABBING PAST 63 ON A MODEL I TRS-80 (OR PAST 127 ON A MODEL I OR III)

by Jack Decker

Believe it or not, I've seen this old chestnut discussed in a couple of exchange newsletters recently. The problem is that on most versions of the Model I TRS-80, the BASIC code in ROM limits the maximum argument for the TAB function to 63 - not at all adequate for today's printers which can use 132 or more columns. Later editions of the ROM change the limit to 127, where it remains throughout the Models III and 4 (Model III mode). This still may not be enough for some of today's wide-carriage printers.

Unfortunately, the newsletters I have read tend to use a much more complicated than necessary approach to solving this problem. So, here are five different methods that permit you to utilize TABs up to 255. Use the one that you prefer and that will work best in your particular hardware/software configuration:

1) THE USER-DEFINED FUNCTION: Near the start of your BASIC program, insert this line:

```
DEFFNA$(N%)=STRING$(N%-PEEK(16539),32)
```

Then use FNA$(n) in place of TAB(n) in any LPRINT statement, for example LPRINT FNA$(70) instead of LPRINT TAB(70). This has the advantage of working under Model I or III BASIC, using any DOS. If you are want your program to be so generic that even tape users can run it, simply repeat the function each time it is needed instead of pre-defining it - that is, use LPRINT STRING$(n-PEEK(16539),32) to replace each occurrence of LPRINT TAB(n) (note that n can be a constant, variable, or expression in any of these examples, though you may need to enclose an expression in parenthesis when making the conversion).

2) MACHINE LANGUAGE "PACKED STRING": Near the start of your program, place the following BASIC line (note that variables T$, X, Y, and Z may be replaced with any valid variable names of similar types, and that any or all of these variable names may be reused in subsequent lines):

```
10 T$ = "1234567": X = VARPTR (T$): Y = PEEK (X+1) + PEEK (X+2) * 256: POKE Y,231: POKE Y+1,252: POKE Y+2,5: POKE Y+3,43: FOR Z = 0 TO 2: POKE Y+Z+4, PEEK (16851+Z): POKE 16851+Z, PEEK (X+Z): NEXT: POKE 16851,195
```

The spaces are for readability only, and may be omitted with no ill effects. Note that after this line is executed, the string "1234567" will be replaced by machine language code, therefore, line 10 should not be EDITed and the program containing line 10 should not be SAVEd once line 10 has been run. The proper procedure is to SAVE the program to disk, then RUN the program and then, if any debugging is necessary (anywhere in the program), reLOAD the program and start editing. If you forget to do this and accidentally SAVE a version of the program where line 10 has already been executed, you may wind up having to re-enter line 10 from the keyboard.

This method works under any Disk Operating System, and even works for tape users. TAB arguments through 255 are permissible after the code in line 10 has been executed.

Here's a bit of technical information about this method, for those who are interested: The seven bytes of machine language code disassemble as follows:

```
E7       RST     20H       ;Check Number Type Flag
FC052B   CALL    M,2B05H   ;Re-evaluate TAB if NTF=integer
??????   ???               ;Original instruction at 41D3H
```

This code is patched into the DOS link at 41D3H (which contains only a RET instruction in a cassette-based machine, or a jump to some area of Disk BASIC when a Disk Operating System is used), and which is CALLed from the ROM at 2141H during processing of the TAB argument. At that point, the TAB argument has already been evaluated to a number in the range 0-63 or 0-127 (depending on the ROM version in use), and the result stored in the E register. But if the TAB argument was over the "allowable" range (63 or 127) but less than 256, no error message is given - instead the most significant bits are stripped off to yield a result below 64 or 128. However (and this is what makes this work), the original TAB argument is still stored as an integer in the arithmetic accumulator at 4121H-4122H!

The reason for the test of the Number Type Flag (NTF) is that the DOS link at 41D3H is actually called from two places in ROM - it's called at 2141H after evaluating the TAB function, and at 2108H while processing commas (used to tab to next print column). When called from the former, the NTF will contain a value of 2 indicating that an integer was just processed, and we can then make the CALL to 2B05H to re-evaluate the TAB function. However, if the DOS link was called from 2108H, the NTF contains a value of 3 indicating that a string was just processed. In that case we simply pass control to the original Disk BASIC link without further processing. All of this may make more sense to you if you have a commented disassembly of the ROM code to follow and/or you have a copy of my book "TRS-80 ROM ROUTINES DOCUMENTED."

3) PATCH YOUR DISK OPERATING SYSTEM: The idea here is to patch your Disk Operating System as follows: First, use a machine language monitor program to find out what instruction is currently stored at the Disk BASIC vector at 41D3H-41D5H. Next, place the same seven-byte machine language code segment given in (2) above somewhere in an unused section of your Disk BASIC's "patch area", making sure that the last instruction is the same as the one found at the Disk BASIC vector at 41D3H-41D5H. Finally, we modify the DOS vector at 41D3H-41D5H to jump first to our patched code segment (which will then jump to the original Disk BASIC vector). As an example, here are the patches that will work with the Model I version of NEWDOS/80 version 2:

```
Change BASIC/CMD,15,BE from: 00 00 00 00 00 00 00
                       to:   E7 FC 05 2B C3 3B 5F
Change BASIC/CMD,17,84 from: C3 3B 5F C3
                       to:   C3 B5 66 C3
```

Note that this is an "undocumehted" zap, so be sure to keep a record of it somewhere in case Apparat issues a zap that uses the same area of memory and you have to move (or remove) this one. Users of other DOSes may develop similar patches, but you are on your own (if you do, why not send it in so we can share it with other NORTHERN BYTES readers?). The advantage of this method is that it will permit all of your BASIC programs to use TABs up to 255 without further modification. The disadvantage is that since the patch is to DOS, you can't very well include it in an application program that is intended to be used by others (who may not have a similarly patched DOS).

4) BUY AND USE MULTIDOS: The above-mentioned machine-language code segment is already built into MULTIDOS - no patching required!

5) PATCH YOUR MODELA/III FILE (MODEL 4P ONLY): This works for Model 4P owners only, of course, and is only necessary if you have a need to TAB past 127. Simply change the byte that loads to 213BH, from 7FH to FFH. In my copy of MODELA/III this byte is found at File Relative Sector 33, byte C3. Once again, the disadvantage here is that if you're writing a program for someone else, you can't assume that their copy of MODELA/III will have this patch.

The bottom line: If you're wanting to be able to use TABs up to 255 in your own programs on your own system, one of methods 3-5 will probably be best for you. But, if you're writing a program for use by others (or if you use many different copies of a Disk Operating System, and don't want to patch them all) then either method #1 or method #2 should be used.

I hope this clears up some of the confusion regarding the use of wider TABs. As you can see, it is NOT necessary to use a 30 or 100 byte patch program to be able to TAB past 63!

REMOVE OR CHANGE PRINTER TIMEOUT IN TRSDOS 6

To remove the printer timeout capability completely:

```
PATCH BOOT/SYS.LSIDOS (D0C,2C=18:F0C.2C=20)
```

To increase the time before timeout,

```
PATCH BOOT/SYS.LSIDOS (D0C.24=NN:F0C.24=01)
```

where NN is some number greater than one. Each count greater than one should give you an additional 7.5 seconds.

These patches originally from --jjkd--, since passed around the LDOS SIG on CompuServe.

SPIKELESS MAINS SWITCHING

Hardware Project by Allan Dent

[Reprinted from the Adelaide Micro-User News, 36 Sturt Street, Adelaide, South AUSTRALIA 5000. Although this project was designed to work with 240 volt AC power circuits (the norm for household current in other parts of the world), the exact same circuitry can be used on 120 volt lines. Just substitute "120" wherever you see "240" in the article below.]

Soon after moving my Model I TRS-80 into the world of disk drives, I discovered that it had one tremendous disadvantage. If I turned off the power to the system with my disk still in the drive, I sometimes lost all the data on that disk. Needless to say I soon learnt to remove the diskette before powering down. The problem is caused by power spikes. These are created by the power transformers being turned off during a high voltage point (up to 340V peak) of the 240V AC waveform. It doesn't happen every time and when it does, the damage to the data on the disk is sometimes less catastrophic than on other occasions. If the head of the drive is over the directory at the instant of powering down then the directory track can be corrupted, whereas if it is over some other file, that file could be lost. Fortunately, Apparat came out with Newdos+ and included a utility called Superzap. This program, even though in Basic, allowed me with the help of Harv Pennington's book TRS-80 Disk And Other Mysteries to get into the disk and salvage most of my losses. Since then, improved versions of this marvelous original utility have made life even easier in repairing the damage caused by the occasional data destroying spike. Removing the disk before power down though is the best insurance against having to go through the arduous job of recovering the spiked data.

Even though I trained myself to remove the disk before power down I still had similar crashes caused by externally generated mains spikes. These were virtually eliminated by the installation of a mains filter and metal oxide varistor on my power distribution block. The only problem now was caused by my kids forgetting the power down sequence. Fortunately, they only ever crash games disks, and most of them are backed up on either cassette or another disk. I decided to try to overcome this problem by tackling the problem at the source, the switchoff timing of the mains waveform.

The project involves 240V mains and this should be kept in mind when wiring up the circuit. The heart of the circuit is a zero crossing solid state relay. This is basically an electronically controlled TRIAC which does the actual switching of the mains current. The input is a current limited infra-red Light Emitting Diode. This is optically coupled to an infra-red photo-transistor connected to a zero crossing detector. This construction gives complete electrical isolation between the input LED and the mains circuitry on the other side of the optical coupling. The photo-transistor, when illuminated by the LED, turns on the control circuitry. Then, when the control circuitry senses the mains supply voltage to be less than about 20 volts, (close enough to zero for 240V mains), it turns on the triac which passes the mains current through to the load. The characteristics of a triac are that once turned on, it will not turn off until the mains current flowing through it falls to zero. This point, depending on the reactance of the load is nominally also at the zero voltage point of the mains AC cycle. This means that instances of power on and power off of the load are both at the zero voltage point of the mains cycle virtually eliminating the transformer spike problem.

The circuit as can be seen is fairly simple and involves only a few parts. The small power transformer provides the current to drive the LED on the control side of the solid state relay and the other supervisory LEDs. These other LEDs are mounted along with the ON/OFF switch, on the edge panel of the jiffy box used to contain the circuit. I used a short extension cord that I cut in half for the mains in and out connections, others may prefer to mount a 3 pin socket on the side of the jiffy box. The whole circuit is mounted on a piece of experimenters matrix board, inside the box on a couple of standoffs. The supervisory LEDs are optional and almost any small, low voltage transformer will do, even a plug pack power supply, which will also eliminate the need for the rectifier and capacitor.

I installed the control box at a convenient point near the front of my desk and connected it into the power block. I then switched on all of the component parts of my system and powered the whole lot simultaneously from my little toggle switch. When I tested my circuit, I made ten copies of my DOS disk and added an AUTO DIR command. This ensured that the head stopped over the directory track and any problems would show up immediately. I must have switched my system on and off about 100 times that afternoon without a single failure. Every time the DOS booted and then displayed the directory. Since that day, about two months ago, I have not removed my disks at power down and have not lost any files or disks other than through my own stupidity, but then nobody's perfect.

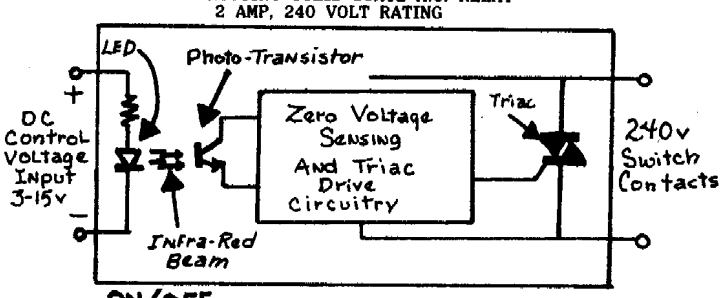
I cannot give a cast iron guarantee that this will work for every system, but I would be very surprised if it didn't and would like to hear of any failures. All I can do is report my success and hope that if you follow my example that you will be as happy with the results as I am. Don't forget to test the system using backup copies of your DOS and if you're as successful as me, you won't need the other nine copies and will be able to put them back in your disk box unused.

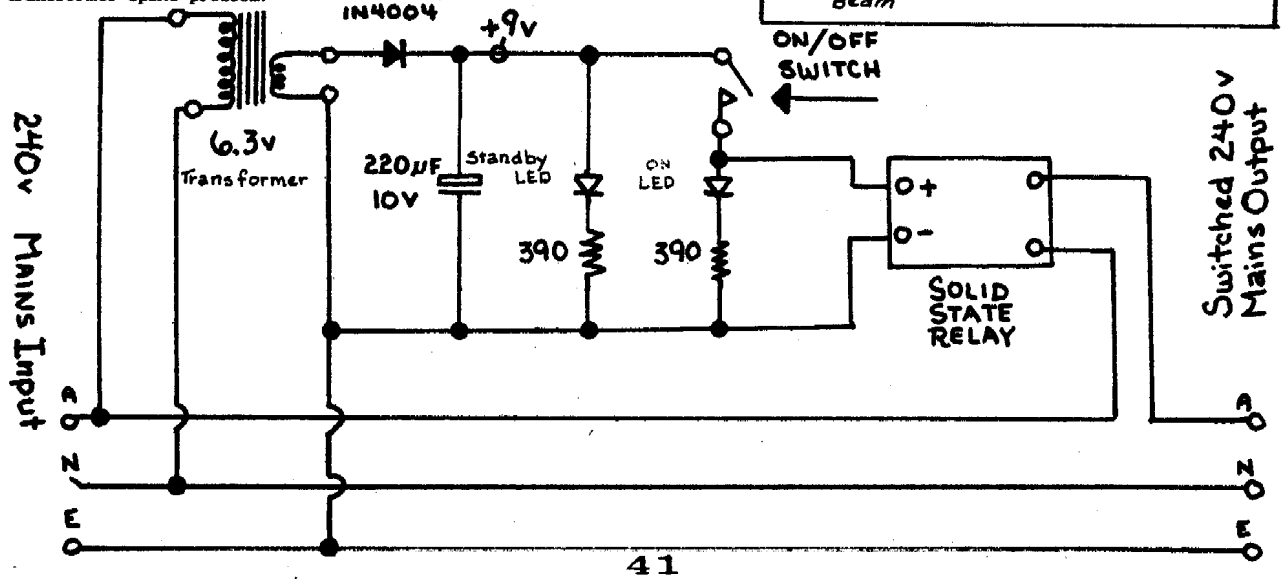
[Editor's note: In the second diagram below, the wires labelled A, N, and E would be connected to the normal 120 volt household wiring found in North American homes as follows (this assumes a 120 volt mains input):

A connects to the BLACK (hot) wire
N connects to the WHITE (neutral) wire
E connects to the BARE or GREEN (earth ground) wire

This assumes that you are using standard three-prong plugs and receptacles. Also, please keep in mind that this project works with household current which can be quite deadly if you don't know how to handle it properly. Unless you are absolutely certain that you know what you're doing, get help from a licensed electrician in building this project!]

ZERO CROSSING SOLID STATE A.C. RELAY
2 AMP, 240 VOLT RATING





SUPER UTILITY CONFIGURATION

by David Sutton

[Reprinted from SYDTRUG NEWS, P.O. Box 297, Padstow, New South Wales 2211, AUSTRALIA.]

One question overheard at the last club meeting was how can the Super Utility 3.2 (/CMD version) configuration table be amended and saved to disk, therefore avoiding the need to 'set-up' after each load?

This prompted me to delve into SU and come up with the following steps:

i) Load SU.
ii) Check SU configuration table.
iii) Examine memory used by the configuration table.
iv) Examine the disk sector used by the configuration table.
v) Amend configuration table as required.
vi) Note amended byte values.
vii) Zap the disk accordingly.

The following will lead you through.

1) CHECK CONFIGURATION TABLE
Load SU, select 9, <ENTER>. Note decimal values shown for drive 0 PTKS,RTKS,DIR and write down the equivalent HEX values -
e.g.. 35,35,17 = 23H,23H,11H

2) FIND MEMORY MATCH
Press shift/break, select 7, <ENTER>, select 9, <ENTER>.
Answer prompt with 1,FFFFH..<ENTER>.
Answer string prompt with Hex values per (1) above..SUFFIX H must be specified.
e.g.. 23H,23H,11H (or whatever)..<ENTER>, <ENTER>.
SU now shows memory location match, on my version 416BH. Write down this location. There may be more than one match but usually the first match refers to drive 0.

3) FIND SECTOR MATCH
Press shift/break, select 1, <ENTER>, select 9, <ENTER>.
Answer prompt with 0,0,0 (drive, track, sector) .. <ENTER>.
Answer prompt with 350 (35 track) 400 (40 track) etc..<ENTER>. Answer string prompt with your HEX values per (1) above making sure that suffix H is specified.
<ENTER>, <ENTER>.
Write down match details i.e. drive, track, sector, byte. Only the first one is relevant and refers to drive 0.

4) AMEND CONFIGURATION TABLE
Press shift/break, select 9, <ENTER>, and amend table as required.
Note decimal values for PTKS, RTKS, DIR and write down the equivalent HEX values.

5) NOTE BYTE VALUES FOR AMENDED TABLE
Press break, select 7, <ENTER>, select 1, <ENTER>.
Answer prompt with memory location per (2) above, <ENTER>.
The 13 bytes starting from the location apply to drive 0, the next 13 to drive 1 and so on.
Write down the 4 sets of 13 bytes (or SHIFT/CLEAR if you have a printer connected).

6) ZAP DISK
Press shift/break, select 1, <ENTER>, select 1, <ENTER>.
Answer prompt with drive,track,sector match details per (3) above...<ENTER>.
Press M and modify bytes starting at byte per (3) in line with byte values per (5) above...<ENTER>.
Finally press U and modifications will be written to disk.

Upon rebooting, the configuration table should reflect the amendments. As far as I can establish, the 13 bytes referred to in (5) reflect the drive configuration thus:

Byte 1	PTKS	Byte 2	RTKS
Byte 3	DIR	Byte 4	??????
Byte 5	STEP/WP/DLY	Byte 6	SINGLE/DOUBLE STEP
Byte 7	DENSITY	Byte 8	DOS TYPE
Byte 9	??????	Byte A	??????
Byte B	SECTORS/GRAN	Byte C	GRANS/TRACK
Byte D	SECTORS/TRACK		

All values being in HEX.

If any members can fill in the gaps via the newsletter, please do. One word of warning...for safety, practice on a backup copy.

MODEL 4 SCREEN DUMP SUBROUTINE

by Mike Zarowitz

In response from a member of the Twin Cities Tandy Users Group, I have written a subroutine that allows a screen dump from within a BASIC program. The variables are DUMP0% - the first line to be included in the screen dump, DUMP1% - the last line to be included, GRAPHICS% <> 0 will send all graphics characters to the printer without converting them to a period. If GRAPHICS%=0, then GRAPHIC$ can be set to the character you want the graphics characters converted to. The routine is fully error trapped for valid DUMPx% numbers, and except for the first entry they always default to 0 and 23 and GRAPHIC$=".". I'm always looking for new problems that involve accessing various DOS functions from BASIC. If anyone has any subroutines they would like to see, they can drop me a line at 830 Bransten Road, San Carlos, California 94070. No promises, but I'll consider everything.

```
10 'Front end processor for SCRNDUMP/BAS
20 INPUT "First line: ";DUMP0%:INPUT "Last line: ";DUMP1%
30 GRAPHICS%=0 ' no graphics characters
40 GRAPHIC$="#" ' set graphic char conversion to "#", instead of "."
50 GOSUB 1000:GOTO 20
60 END
1000 'SCRNDUMP/BAS
1010 IF DUMP0%>23 THEN DUMP0%=0 'if start line > 23, set to 0
1020 IF DUMP1%<DUMP0 OR DUMP1%>23 THEN DUMP1%=23 ' check validity of end
1030 'if GRAPHICS is on, save and set DOS flag
1040 DFLAG%=PEEK(&H6C):DSAVE%=DFLAG%:IF GRAPHICS% THEN DFLAG%=(PEEK(&H6D) OR 128)
1050 GOSUB 1090 ' poke values into DOS
1060 SCRNDUMP%=&H935:CALL SCRNDUMP% ' call the screen dump subroutine
1070 ' reset GRAPHICS flag, start, graphics char ("."), and end to defaults.
1080 DFLAG%=DSAVE%:DUMP0%=0:GRAPHIC$=".":DUMP1%=23 ' set values to DOS defaults
1090 ' POKE DOS Graphics flag, dump start, graphics char, and dump end.
1100 POKE &H6D,DFLAG%:POKE &H94A,DUMP0%:POKE &H95D,ASC(GRAPHIC$):POKE &H974,DUMP1%
1110 RETURN ' subroutine valid for TRSDOS 6.2.0 and 6.2.1
```

RECORD LENGTHS IN VARIOUS DOS's

by Bruce W. Tonkin
Abridged by Andy Chakires

[Reprinted from Micro Info Exchange, the newsletter of the Cabrillo Computer Society. The original article from which the following was excerpted originally appeared in Dynamic Memories, the newsletter of the South Bay TRS-80 Users Group. If the name Bruce W. Tonkin sounds familiar, you may recall that he is the author of THE CREATOR, a database program generator that first appeared in the January, 1983 issue of 80-MICRO.]

Contrary to my earlier belief, I found out that the TRS-80 Model 4 does not permit record lengths in excess of 256. The difficulty is not in the BASIC but in TRSDOS 6.x. It seems that LDOS (e.g. TRSDOS 6.x) keeps track of the record length in the directory, and allocates just one byte for the purpose. Thus the maximum possible record length is 256.

Can you believe it? CP/M, MSDOS, the Mac's FINDER, and UNIX manage to be competent operating systems, and none of them keep track of the record length. Why should they? They keep track of the file size, that's all. The actual record length is up to the user, and that's a good approach. After all, just because I once opened a file with a record length of 50, doesn't necessarily mean I wouldn't like to open it with a record length of 100 the next time. In fact, if I want to speed up disk accesses for particular purposes, I may well want to open the file with a record length of 5000.

Even if the DOS wanted to keep track of the record length, why must it report a fatal error when you try to open the same file with a different one?

Only TRSDOS-like operating systems prohibit that approach. It's a shame that Logical Systems (TRSDOS 6.x's creators) had to cripple Microsoft BASIC 5.2, especially since they make such a big deal of how sophisticated it is. For my money, it's not as good as CP/M, MSDOS, or UNIX. I'd rather run NEWDOS, DOSPLUS, or TRSDOS (I or III) than LDOS (TRSDOS 6.x.).

EEN AANGEPASTE LLIST (ANOTHER LLIST)

by Luch Klooster

Translated from Dutch to English by Paul Fransen

[This program is reprinted from REMARKS, the publication of the TRS-80 Gebruikers Vereniging (TRS-80 Users Group) in The Netherlands. Note that it was written to work with the Model I TRS-80, I do not know if it will work with the Model III.]

All LLIST will give you is a printout of your BASIC program. It knows only one page length: endless, so every listing is one page. This LLIST routine knows the page length and prints on every page a page number and the date and time.

How it works:

LLIST and LIST use the same ROM routine. Only the "output device code" at address 409C tells the computer where to send the output to. It appears that before printing a new line it will jump to the "DOS exit" at address 41E0. In Level 2 there is RET instruction at that address. In Disk BASIC there is a JUMP to another address in Disk BASIC. So, I place another address in 41E0, so the program jumps to the new routine. The routine first checks the "output device code". If it is not a "1", then it jumps back to the original routine in BASIC.

Page and line length

I use a page line length of 80 characters and a page length of 12 inches [a common paper size outside of the U.S. - readers in North America may wish to change this to 11 inches]. You can choose your own settings. Just one word of caution - the routine always prints an entire BASIC line without skipping, even if the line consists of 255 characters. So don't use a page length of 66 (when you have 12 inch paper). Make it 62.

I have used the following ROM-routines:

032A Send character in A to the Output device which is pointed to by the output device code. Counts the characters that have already been printed yet (in address 409B) and the number of lines already printed (in address 4029).

0FAF Converts integer in HL to ASCII and sends it to the output device.

1D9B Checks for SHIFT @ and if pressed waits for a another key to be depressed to go on.

1AFE Prints a Carriage Return (0D). Uses 032A.

2B75 Prints a string pointed by HL (using 032A). The string must be terminated with a 00H-byte.

2B7E Gets a BASIC line pointed to by HL and moves it to the address in 40A7H and changes the tokens into text.

The following DOS-routines are used:

446D Puts the time into the address in HL (Newdos 2.0)

4470 Puts the date into the address in HL (Newdos 2.0)

The following addresses are used:

4029 Line counter from printer DCB

402A Normally not used [in the Model I]. This routine uses it as a page counter [the Model III already uses this location for the same purpose].

409B Character counter. The number of characters printed (on one line).

409C Output device code (1=printer, 0=video, -1=cassette)

40A7 Contains address of keyboard buffer. Used as work area.

Patch for Newdos 2.0 Disk BASIC

By placing this routine into space which Apparat left for patches, we can make the routine permanent. You could do that with Superzap, Super Utility or a similar program [EDITOR'S NOTE: In the original article in REMARKS, this program was ORG'd at 6611H (in line 1200) and the method was given to install this program into formerly unused patch space in BASIC/CMD. Unfortunately, Apparat ZAPs 087 and 089 now make use of this same area, so I have changed the ORG address to FF55H to put the code into high memory (be sure to set HIMEM to FF54H or lower before calling this program, and make sure you don't already have another program in high memory unless you change the ORG address and re-assemble the program to accommodate the other high memory program). For the hackers in the crowd, when this program was installed in the Model I version of NEWDOS/80 Disk BASIC, two bytes in BASIC/CMD, sector 17, bytes 91 and 92 were modified to contain the starting address of this program.]

Changes for Level 2

The following few minor changes will make the routine work for level 2:

```
00900     JP        1A19H     ;TO BASIC
01200     ORG       high      ;HIGH IN MEMORY
01500     RET       NZ        ;REPLACES RET IN DOS-EXIT
```

Also delete lines 9200-9900. ROM BASIC doesn't know the date and time.

```
                  00100 ;*****************************************
                  00200 ;** LLIST with pagenumber, time and date **
                  00300 ;**                                       **
                  00400 ;** L Klooster                           **
                  00450 ;** Schoolstraat 28                      **
                  00460 ;** 9451 BG Ralde                       **
                  00470 ;** Holland                             **
                  00500 ;*****************************************
FD00              00600          ORG     0FD00H
FD00 2155FF       00700 START    LD      HL,PRNTR        ;new llist entry
FD03 22E041       00800          LD      (41E0H),HL      ;change BASIC entry
FD06 C32D40       00900          JP      402DH   ;back to caller
                  01000 ;
FF55              01200          ORG     0FF55H
FF55 3A9C40       01300 PRNTR    LD      A,(409CH)       ;current device to A
FF58 FE01         01400          CP      01H     ;test for printer
FF5A C2B95E       01500          JP      NZ,5EB9H        ;no, back to BASIC
FF5D F1           01600          POP     AF      ;get rid of RET address
FF5E AF           01700          XOR     A       ;zeroing A
FF5F 322A40       01800          LD      (402AH),A       ;init page counter
FF62 CDB3FF       01900          CALL    KOP     ;print head line
FF65 CD9B1D       02000 LOOP     CALL    1D9BH   ;test for SHIFT @
FF68 C5           02100          PUSH    BC      ;address line to print
FF69 4E           02200          LD      C,(HL)  ;LSB current line#
FF6A 23           02300          INC     HL
FF6B 46           02400          LD      B,(HL)  ;MSB current line#
FF6C 23           02500          INC     HL      ;HL points to 1st character
FF6D C5           02600          PUSH    BC      ;save line#
FF6E E3           02700          EX      (SP),HL ;exchange HL and Top Stack
FF6F EB           03000          EX      DE,HL   ;exchange DE and HL
FF70 DF           03300          RST     18H     ;test for all lines printed
FF71 C1           03400          POP     BC      ;pointer to 1st character
FF72 DA181A       03500          JP      C,1A18H ;if printing done, then input
FF75 E3           03600          EX      (SP),HL ;exchange HL and Top Stack
FF76 E5           03900          PUSH    HL      ;save address current line
FF77 C5           04000          PUSH    BC      ;save pointer 1st character
FF78 EB           04100          EX      DE,HL   ;exchange DE and HL
FF79 3A2940       04300          LD      A,(4029H)       ;count printed lines/page
FF7C FE3E         04400          CP      REGELS  ;test for page full
FF7E F4B3FF       04500          CALL    P,KOP   ;yes, then head-routine
FF81 CDAF0F       04600          CALL    0FAFH   ;print line# in Ascii
FF84 3E20         04700          LD      A,20H   ;space
FF86 E1           04800          POP     HL      ;address current line
FF87 CD2A03       04900          CALL    032AH   ;print space
FF8A CD7E2B       05000          CALL    2B7EH   ;current line to work area
                  05200 ;                        ;and tokens to text
FF8D 2AA740       05300          LD      HL,(40A7H)      ;start address work area
FF90 7E           05400 PRLOOP   LD      A,(HL)  ;get character
FF91 B7           05500          OR      A       ;test for end of line
FF92 280F         05600          JR      Z,PREND ;yes, then jump
FF94 3A9B40       05700          LD      A,(409BH)       ;count printed characters
FF97 FE4E         05800          CP      REGELL  ;test for line full
FF99 CCFE20       05900          CALL    Z,20FEH ;yes, then print CR and LF
FF9C 7E           06000          LD      A,(HL)  ;get character
FF9D CD2A03       06100          CALL    032AH   ;print character
FFA0 23           06200          INC     HL      ;point to next character
FFA1 18ED         06300          JR      PRLOOP  ;loop
FFA3 CDFE20       06400 PREND    CALL    20FEH   ;print CR and LF
FFA6 E1           06500          POP     HL      ;address pointer next line
FFA7 D1           06600          POP     DE      ;address last line
FFA8 4E           06700          LD      C,(HL)  ;LSB next line
FFA9 23           06800          INC     HL
FFAA 46           06900          LD      B,(HL)  ;MSB pointer next line
FFAB 23           07000          INC     HL      ;address next line#
FFAC 78           07100          LD      A,B     ;test if pointer to next
FFAD B1           07200          OR      C       ;line=nul (end program)
FFAE 20B5         07300          JR      NZ,LOOP ;no, then loop
FFB0 C3191A       07400          JP      1A19H   ;yes, then ready and to BASIC
                  07500 ;
```

```
FFB3 C5      00700 KOP   PUSH  BC      ;save registers
FFB4 E5      00710       PUSH  HL
FFB5 D5      00720       PUSH  DE
FFB6 3E0C    00730       LD    A,0CH   ;Formfeed to A
FFB8 CD2A03  00740       CALL  032AH   ;print Formfeed
FFBB 2AA740  00150       LD    HL,(40A7H)      ;start address work area
FFBE E5      00160       PUSH  HL      ;save start address
FFBF 0646    00170       LD    B,70D   ;init counter
FFC1 3620    00180 CLEAR LD    (HL),20H        ;space to work area
FFC3 23      00190       INC   HL      ;increment pointer
FFC4 10FB    00200       DJNZ  CLEAR   ;till 70 spaces
FFC6 EB      00210       EX    DE,HL   ;pointer to DE
FFC7 21FAFF  00220       LD    HL,STRING       ;text head line
FFCA 0E06    00230       LD    C,06D   ;length
FFCC ED80    00240       LDIR          ;text to work area
FFCE E1      00250       POP   HL      ;start address work area
FFCF E5      00260       PUSH  HL      ;save it
FFD0 0E32    00270       LD    C,50D   ;offset
FFD2 09      00280       ADD   HL,BC   ;base + offset
FFD3 CD7044  00290       CALL  4470H   ;date
FFD6 23      00300       INC   HL
FFD7 23      00310       INC   HL      ;2 spaces
FFD8 CD6D44  00320       CALL  446DH   ;time
FFDB E1      00330       POP   HL      ;start address work area
FFDC CD752B  00340       CALL  2B75H   ;print head line
FFDF 212A40  00350       LD    HL,402AH        ;page counter
FFE2 34      00360       INC   (HL)    ;increment counter
FFE3 2A2A40  00370       LD    HL,(402AH)      ;page counter to HL
FFE6 2600    00380       LD    H,00    ;zeroing H
FFE8 CDAF0F  00390       CALL  0FAFH   ;print page#
FFEB CDFE20  00400       CALL  20FEH   ;print CR and LF
FFEE 3E20    00410       LD    A,20H   ;space
FFF0 CD2A03  00420       CALL  032AH   ;print space
FFF3 CDFE20  00430       CALL  20FEH   ;print CR and LF
FFF6 D1      00440       POP   DE      ;get registers again
FFF7 E1      00450       POP   HL
FFF8 C1      00460       POP   BC
FFF9 C9      00470       RET
             00480 ;
FFFA 50      00490 STRING DEFM  'Page '
     61 67 65 20
FFFF 00      00500       DEFB  00
003E         00510 REGELS EQU   62D     ;Number of lines/page
004E         00520 REGELL EQU   78D     ;Max. line-length
FD00         00530       END   START
00000 TOTAL ERRORS

CLEAR  FFC1   KOP    FFB3   LOOP   FF65   PREND  FFA3   PRLOOP FF90
PRNTR  FF55   REGELL 004E   REGELS 003E   START  FD00   STRING FFFA
```

DO SOMEONE A FAVOR-PLEASE POST THIS ON YOUR LOCAL BBS:
WANTED - YOUR UNUSED COPY OF NEWDOS/80

Many people have purchased NEWDOS/80 and, for one reason or another, it is now sitting on a shelf gathering dust. However, Apparat has recently discontinued NEWDOS/80, and many folks who waited too long have found that they are now unable to purchase a copy.

We still feel that NEWDOS/80 is a superior operating system (despite the manual), and although we have been accused by one newsletter editor of "trying to fragment the TRS-80 market" by keeping NEWDOS/80 alive, we still feel that for many people (especially those with 80 track and/or double sided drives in their systems), NEWDOS/80 is one of the best operating systems available. As for that newsletter editor, maybe if the folks in his group hadn't all rallied around LDOS and TRSDOS 6, we'd all be united in using NEWDOS/80, which after all was available (in an earlier version) before LDOS was!

More to the point, there are still many useful application programs around that are designed to run under NEWDOS/80. So (here comes the part you might wish to upload to your local BBS), The Alternate Source will BUY your copy of NEWDOS/80 for resale purposes. If you don't want it, others probably do, so why not make a little money and free up some shelf space to boot?

The only conditions are that both the original NEWDOS/80 disk and manual must be included, and both must be in good condition. Contact The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906-5319 or telephone (517) 482-8270 and ask for Marianne or Charley.

Of course, if you're in the market for a copy of NEWDOS/80, you may also contact TAS and ask about current availability.

MODEL 4 DESKMATE (26-1608)
by Oswald Cooper

[This is a combination of two articles that originally appeared in Bits and Pieces, the newsletter of The Northeast Computer Club.]

WRITING ALARM/ARM TO ANOTHER DISK DRIVE

I have applied the following patches, without problems, to write to the ALARM/ARM file on drive 1 instead of drive 0 as I prefer to keep my drive 0 program disks write protected. The file may also be kept on drive 2, 3, 4, 5, 6, or 7 by changing the underlined character in the patches below to the appropriate drive number:

```
PATCH DMRES2/CMD (D09,5F=31:F09,5F=30)
PATCH DMRES2/CMD (D09,90=31:F09,90=30)
PATCH DMALARM/CMD (D1E,DB=31:F1E,DB=30)
PATCH DMCALNDR/CMD (D31,7D=31:F31,7D=30)
```

Copy the file ALARM/ARM from drive 0 to drive 1 and then remove it from drive 0.

NOTE: Apply these patches only to backup copies of Deskmate program files. These patches are only for Version 01.00.00.

If you have applied these patches to Deskmate, you may want to add the following patches to change the version number from 01.00.00 to 01.00.01.

```
PATCH DMMENU/CMD (D20,D2=31:F20,D2=30)
PATCH DM/CMD (D00,AD=31:F00,AD=30)
```

Additional notes: DeskMate will not load ASCII files "as is". The following procedure can be used:

1. Copy: FILENAME/TXT TO FILENAME/DOC
2. Use a file dump utility such as FED or DISKDUMP to change the last byte +1 in the file to 1AH.
3. The file should now be loadable with TEXT in DeskMate, as long as there is sufficient memory available.
4. This procedure has been found to work with BASIC files saved in ASCII (,A), Allwrite text files, and VisiCalc print (/PRT) files.

The following procedure can be used if you use Allwrite: Load the document and type <CTRL><E> to move to the end. Then type <CTRL><:> 0 2 6 . [See the Allwrite manual section on "Special Symbols" if you are not familiar with this use of the "Control-:" key sequence.] This will put a "Control Z" (1AH) at the end of the file which is what DeskMate Text looks for at the end of file byte.

BOB BRUMLEY'S HACKER HINTS

This information was provided by Bob Brumley during recent telephone conversations:

SPEEDING UP AN OLDER-STYLE MODEL 4

We haven't tried this and present it for experimental purposes only. If you try it and damage your computer as a result, don't come crying to us. We DO NOT guarantee that these mods will work in any manner whatsoever, and we DO NOT guarantee that by installing them you will not damage your computer or software or other data. We specifically disclaim any responsibility for any damage that may result from anyone's use of this information. (Even Judge Wapner should be satisfied with THAT statement - I hope!).

How would you like to speed up your older-style Model 4 to 5.07 MHz. Or maybe 6.335 MHz? Or maybe even 10.141 MHz? Well, here is a very simple modification. One warning, though - you lose the ability to change speeds or to shift back to either of the original clock speeds (more on that later).

Here's the instructions for the modification:

1. Remove PAL IC chip U3 from its socket.
2. Bend out pin 19 of the PAL chip.
3. Add a jumper wire between pin 19 of the U3 socket (NOT the PAL IC) and one of the following points:
   a. Pin 17 of the U3 socket for 5.07 MHz.
   b. Pin 11 of the U4 socket (the other PAL chip) for 6.335 MHz.
   c. Pin 16 of the U3 socket for 10.141 MHz.
4. Plug the PAL chip U3 back into the socket, making sure that pin 19 does NOT get plugged back in.
5. If you have selected the 10.141 MHz speed-up, you will have to replace the Z-80 microprocessor with a high-speed Z-80H chip. You will probably also have to speed up the RAM timing as described in the next step. Even then, it may not work. Let the hardware hacker beware!
6. If you feel the need to speed up RAM timing (approximately 2 or 3 times), replace your RAM chips with 120 ns. units. Then make the following mods: Cut pin 7 of U18 loose from the circuit board. Jumper pin 6 of U18 to pin 13 of U18. Remove U58 (a PAL chip) from its socket, bend out pin 6 and plug it back in. Then jumper the pin you just bent out (on the PAL chip itself, NOT the socket this time) to U22 pin 8.
7. Don't forget to set any appropriate options in your Disk Operating System (such as SYSTEM option BJ of NEWDOS/80) to help it cope with your new fast clock speed.

One last point - installation of this mod disables your ability to switch between high and low speed (normally 4 MHz vs. 2 MHz). Bob suggests that a hardware hacker could easily build up a small printed circuit board and add a couple of IC's (such as a 74LS00 and a 74LS74) to permit switching between any of FOUR different speeds (the original 2 or 4 MHz at pin 19 of PAL chip U3, which is selected using bit 6 of port 0ECH, and any two higher speeds. Bit 7 of port 0ECH is unused, so if bits 6 and 7 of port 0ECH are used for speed control, up to four speeds could be selected). I will leave the design of such a board as a challenge for our readers (if you design one, please send it in so we can publish it!).

UNDOCUMENTED SUPERZAP COMMAND

Bob Brumley also points out that SUPERZAP has an undocumented command. It's RRT, which stands for "Read Raw Track", which is just what it does - it reads an entire disk track to memory, then tells you where it put it so you can examine it using the "DM" (Display Memory) function. Unfortunately, there is no companion command to write the track back out to disk, but this command could still prove useful, such as when you are trying to get some easily-recognizable data (a text file, perhaps) off of an alien disk.

FASTER ACCESS TO NEW OVERLAY MODULE

If you have installed Joachim Kelterbaum's modification to NEWDOS/80 that gives you a JKL routine with TRS-80 graphic blocks for use with the Epson RX-80/FX-80 printers (from NORTHERN BYTES Volume 5, Number 4, starting on page 6) you can speed up access to this routine. The article instructs you to patch the start of the normal JKL routine so that it will call the new overlay and jump to it. What that means is that overlay SYS3/SYS is loaded, then it immediately calls in the new overlay containing the JKL routine. You can skip the loading of SYS3/SYS and go directly to the new overlay by making the following patch in SYS1/SYS, which was provided by Bob Brumley:

In file SYS1/SYS, relative sector 3, starting at byte 18H (in the Model I) or 2EH (in the Model III), you will find the byte sequence:

4A 4B 4C 80 A5 10

Change the underlined byte (the A5) to 9D and when you use the JKL routine, it should go directly to the new overlay, without calling in SYS3/SYS first.

If you have any questions about any of these hints (or about the inner workings of NEWDOS/80 in general), Bob Brumley's address is: 8522 Alden Lane, Windsor California 95492. If you write, it would be only normal courtesy to enclose a stamped, self-addressed envelope.

LISTING OF CHRISTIAN BULLETIN BOARD SYSTEMS

Compiled by Neil Trudel, SYSOP, Fishermen's Scroll BBS
(voice (705) 253-5514, B.B.S. (705) 253-5366 - 300 baud)

Downloaded from the Fishermen's Scroll Christian BBS, (705) 253-5366, 35 Ontario Avenue, Sault Ste. Marie, Ontario, Canada (postal code P6B 1E2). If you notice that any of the information below has changed, or if you know of any other Christian BBS's that are not listed below, please leave a message to Sysop Neil Trudel. Thank You!

The last update of this list was April 28, 1986.

(Modem code A=300, B=300/1200, C=300/1200/2400)

Phone #,	Modem,	Country,	State/Province,	City,	Name of BBS
011-4439-527-2611		UK,	England,	Exmouth,	#9 Comp. For Christ
215-932-8829	A,	USA,	PA,	Oxford,	Penn Church Board
301-828-4572	B,	USA,	MD,	Baltimore,	#3 Computers For Christ
301-596-0123	C,	USA,	MD,	Washington,	#8 Computers For Christ
301-995-0032	,	USA,	MD,	?,	Harvester Baptist Church
303-426-4052	,	USA,	CO,	Denver,	Christian Protocol
312-362-7875	A,	USA,	IL,	Chicago,	#11 Computers For Christ
312-422-3326	,	USA,	IL,	Oaklawn,	Bible On Line
312-738-2785	,	USA,	IL,	Chicago,	Life Savior
312-879-2751	,	USA,	IL,	Batavia,	Stewards Ship
312-934-6060	,	USA,	IL,	?,	Gospel Outreach
313-213-1214	,	USA,	MI,	?,	One Way
313-459-8375	A,	USA,	MI,	Plymouth,	Good News
313-736-8031	,	USA,	MI,	?,	City Funt
314-947-0895	,	USA,	MO,	St. Louis,	Christian
408-732-0312	,	USA,	CA,	?,	Christian
408-997-2790	C,	USA,	CA,	San Jose,	HQ Computers For Christ
412-344-1315	A,	USA,	PA,	Pittsburg,	#12 Computers For Christ
412-391-0395	,	USA,	PA,	?,	#12 Computers For Christ
412-836-8407	,	USA,	PA,	Greensburg,	S.O.A.R.
501-864-0899	C,	USA,	AR,	El Dorado,	#7 Computers For Christ
502-361-4774	,	USA,	KY,	?,	Terrain
512-926-5414	,	USA,	TX,	Austin,	The Dove Works
515-576-5091	,	USA,	IA,	Fort Dodge,	Fishnet
516-878-8319	,	USA,	NY,	Long Island,	The Electronic Angel
603-644-2255	,	USA,	NH,	Manchester,	Narrow Way
612-929-6530	,	USA,	MN,	Minneapolis,	ICHTHUS
616-891-8488	,	USA,	MI,	Caledonia,	Christ I Serve
616-957-1397	,	USA,	MI,	Grand Rapids,	Christian Computer
702-457-4710	,	USA,	NV,	John	The Baptist
705-253-5366	A,	CANADA,	ON,	Sault Ste. Marie,	Fishermen's Scroll
707-964-7114	A,	USA,	CA,	Fort Bragg,	Commodore Christian
713-721-0888	B,	USA,	TX,	Houston,	J.L. Christian
714-983-9923	,	USA,	CA,	#?	Computers For Christ
714-974-4294	,	USA,	CA,	?,	Prayer In Home
716-694-7351	,	USA,	NY,	Tonawanda,	City Gates
804-226-0441	B,	USA,	VA,	Richmond,	The Resistance Board
804-286-2929	,	USA,	VA,	?,	Kamas
805-687-2754	A,	USA,	CA,	Santa Barbara,	Kingdom
816-532-0984	,	USA,	MO,	Kansas City,	Dove Christian
817-483-1264	,	USA,	TX,	Fort Worth,	Good News
817-595-2620	A,	USA,	TX,	Hurst,	Lu-ach Yeshua
904-625-2737	A,	USA,	FL,	Silver Springs,	The Light
907-345-6661	,	USA,	AK,	Anchorage,	Alaskan Christian Exchange
913-537-7173	,	USA,	NY,	Manhatten,	What's The Good News

syslink™

Powerful. Simple. Professional.

SYSLINK offers powerful features not found in other bulletin board software. Menu driven. Simple to use. Professional style and format.

USER FEATURES

- Supports 9000 public messages. Sectionable.
- Separate private mail feature with reply.
- Additional BBS database with search capability.
- File transfer with ASCII and XMODEM protocols.
- Online advertising and direct product ordering.
- MicroMatch - Versatile online dating section.
- SuperVote - Powerful voting/surveying area. It supports multiple topics too.
- Terminal configuration options for each user.
- Have fun with the expandable Trivia section.
- TeleLink - Syslink's hottest feature. Send messages and files to other Syslinks on the network. Automatic cost accounting included.

SYSOP/OVERALL FEATURES

- Powerful text editor with Notepad.
- Complete online help facility.
- Set user time limits according to user priority.
- All user activity is recorded in the Job Log file.
- Full-featured menu-driven SYSOP programs allow you to customize Syslink as well as maintain and repair files. Also generates mailing labels.
- Syslink is smooth in operation and easy to use.
- Turn Syslink into a profitable venture. An optional marketing support package is available.
- We offer full support and some customization.
- TRS-80 (III/IV, NEWDOS version available now. Hayes compatible modem required. IBM PC/MSDOS version coming Fall 1986!

Trade in your original BBS software disks and get $25 off.

The Alternate Source
Information Outlet
704 North Pennsylvania Avenue
Lansing, Michigan, 48906-5319, USA

Phone: (517) 482-8270
MCI Mail ID: 109-7407
TELEX: 6501097407
Answerback: 6501097407 MCI
Compuserve EasyPlex: 72167,161
Delphi Mail: TASIO

Powerful. Simple. Professional.

syslink™

Yes. I want to get involved with Syslink's expanding Network.

☐ Please send me more information on Syslink.
☐ Send me the complete Syslink package for $125. (TRS-80 Model III/IV version)
☐ Enclosed are my original BBS software diskettes. Send me Syslink for $100.

☐ Check ☐ Money Order ☐ Visa ☐ Mastercard

Name _____ Computer type _____

Street _____ Telephone _____

City _____ BBS Phone _____

State _____ Zip _____ Signature _____

Account No. _____ Expiration date _____

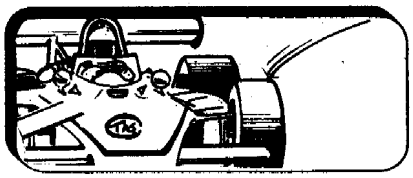
NEW!

THE ALTERNATE SOURCE

704 North Pennsylvania Avenue
Lansing, Michigan, 48906-5319, USA
Phone: (517) 482-8270
MCI Mail ID: 109-7407
TELEX: 6501097407
Answerback: 6501097407 MCI
Compuserve EasyPlex: 72167,161
Delphi Mail: TASIO

Toll Free Ordering:
US: 800-253 3200 Ex 700

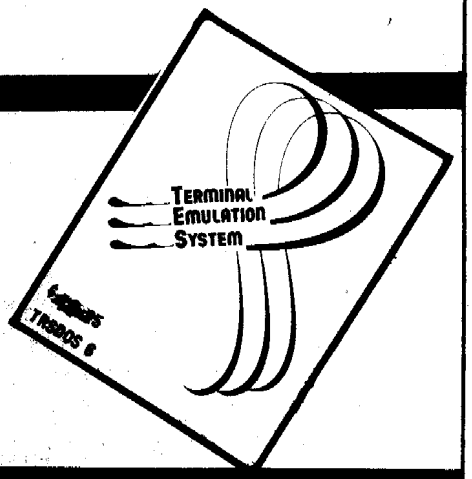
TURBO CHARGE



"Designed by an automotive engineer and guaranteed NOT to be RECALLED!" Bruce Hansen, author of our popular TASMON and ADVENTURE SYSTEM programs, is now an automotive engineer. Amidst an MSDOS world he has still found time to develop a superb package for the TRS-80 I, III(4)(P)(D). Turbo Charge displays the curving, winding roads of the racetrack. Glance in your rear-view mirror and you can see your opponents as they challenge your position or eat your dust. YOU control the shifting, the acceleration, the steering and the braking.

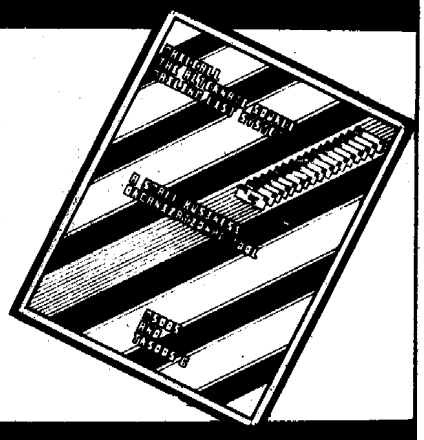
D8TURBO: Bruce Hansen's Turbo Charge $29.95

TERMINAL EMULATION SYSTEM (TES)



Why invest in a dumb terminal when you can use a comparable priced TRS-80 Model 4 or 4P and have the power of a full computer available to your operators? Whether you are talking with a mainframe or a dumb terminal, TES will train your TRS-80 TRS-80 to properly respond to control codes from over one hundred different terminals! As the world of telecommunications expands, your TRS-80 can grow with it! TES includes a "terminal emulation editor for defining your own protocol and adding new ones"(up to 300) as you meet them.

D6TES: Terminal Emulation System for TRSDOS 6 ..... $79.95



MAILCALL can manage one or more of your mailing lists with minimum intervention and maximum performance. MAILCALL supports names and addresses from around the world. You will no longer need to give your foreign customers second class service. Complete setup instructions and a friendly operator interface are included to get you organized fast. Once organized, you stay that way, because your master MAILCALL list is never sorted, yet you can instantly access your list in both name and zip code order. A conversion utility is included to assist you with transferring your current files to MAILCALL. Unprotected and hard disk compatible. MAILCALL also integrates with the optional TAS ORDER ENTRY SYSTEM.

DMMC: MAILCALL for MSDOS $79.95
D4MC: MAILCALL for TRSDOS 6 ..... $79.95

USE THIS TO ORDER

Name: _____ Total $: _____
Address: _____ Tax: _____
Address: _____ Shipping: $3.00
Address: _____ COD: _____
Charge Card: _____ Expiry: _____ Total: _____
Special Instructions: _____
Please add $3 shipping/handling per order; COD $2 additional

VISA

NORTHERN BYTES

Subscription Information

Northern Bytes is edited by Jack Decker and published on an irregular basis by The Alternate Source Information Outlet. Back issues are available starting with Volume 5, Number 1. Issues prior to that are not available. Some of the most valuable articles from earlier issues may be reprinted in future issues of Northern Bytes. Currently there are eight back issues available for Volume 5, as well as all issues from Volume 6. All back issues are $2 each.

It is very easy to be placed on the Northern Bytes REGULAR list. Simply place your address, Visa or MasterCard number and expiration date on file with us. We will start with the issue you request. We do not bill you for ANY ISSUE until that issue has been mailed. This way, we can continue to offer you top quality information with absolutely no risk to you. There's no question of what to do about unfulfilled issues if we decide to quit publishing. Unless otherwise requested, we presume your subscription will extend through the month of your expiration date.

Don't have a charge card, huh? We understand the myriad of reasons for not having them and we feel that a "To-Be-Invoiced" policy could help increase the demand for Northern Bytes. If you'll do it for us, we'll do it for you. Would you like to be placed on a regular list TO BE BILLED for each issue? You could then send a check for the issues as they are mailed. If you didn't send a check, we would presume that your interest has died and discontinue your subscription. The only requirement for getting onto the list is to pay for the first issue up front; the next will be mailed automatically, if you request. You are insured that you will receive top of the line TRS-80 information as it is released. Ask to be placed on the NO RISK "To-Be-Invoiced Northern Bytes List".

Call or write, but SIGN UP TODAY!

The Alternate Source Information Outlet
704 North Pennsylvania Avenue
Lansing, Michigan 48906-5319

Telephone (517) 482-8270, or use our toll-free line for ORDERS ONLY:
(800) 253-3280 ext. 780 in U.S.A.

EMAIL: CompuServe: 72167,161 / Delphi: TASIO / MCI Mail: 109-7407 / Telex: 6501097407 MCI

NORTHERN BYTES

c/o Jack Decker
1804 West 18th Street # 155
Sault Ste. Marie, Michigan 49783-1268
U.S.A.

Bulk Mail
U.S. Postage Paid
Permit 815
Lansing, MI

POSTMASTER: Address Correction Requested.
Send address changes, USPS Form 3579, and undeliverable copies to: The Alternate Source Information Outlet, 704 North Pennsylvania Avenue, Lansing, Michigan 48906-5319

To:

ELECTRONIC MAIL ADDRESSES:

Compuserve EasyPlex:	72167,161
Delphi Mail:	TASIO
MCI Mail:	109-7407
Telex:	6501097407
Answerback:	6501097407 MCI