NORTHERN BYTES

Double Issue: Volume 7 Number 4

Greetings! and Farewell!! This is the last issue of NORTHERN BYTES that will be produced under my editorship. Future issues of NORTHERN BYTES will be edited by Charley Butler of The Alternate Source. As most of our readers are aware, TAS has published NORTHERN BYTES since Volume 5, Number 1 (prior to that it was a computer club publication, and was nothing at all like the NORTHERN BYTES of today). And Charley was the co-editor of The Alternate Source Programmer's Journal, one of the first "hacker" publications for the TRS-80 enthusiast.

If you're a regular reader of NORTHERN BYTES, you are probably already aware of some of the reasons for my decision to abdicate the editor's chair, so I won't go into them here. I had originally figured that this would be the last issue of NORTHERN BYTES, until I discovered that Charley had enough interest in continuing this publication to take over the editor's position. It may well be that NORTHERN BYTES will improve when the entire operation is brought "under one roof"; the over-200 mile distance between Sault Ste. Marie and Lansing has certainly presented many difficulties in the production of this newsletter. This WOULD have been a perfect project for "telecommuting" - that is, doing the work here and sending it by modem to Lansing - but unfortunately, Sault Ste. Marie is still in the stone age when it comes to communications. Someday firms like Tymnet and Telenet and MCI and Sprint and CompuServe and GEnie are going to have to realize that intelligent life exists in cities with less than 50,000 population, if true telecommuting is ever going to become a reality. In a way, I sort of hope they all go broke if they continue to restrict the availability of their services to those big cities they love so much. In the meantime, we just can't afford to "telecommute" at 15 to 20 cents per minute (that's the late night rate!), PLUS packet network charges, PLUS online service charges as may be applicable (depending on which online $ervice we are trying to access). But I digress...

Once this issue is off the press, my participation in NORTHERN BYTES will be finished. This means that if your club or user group is exchanging newsletters with NORTHERN BYTES, you should address all future copies to Charley Butler, The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906-5319. If you feel that you can spare the extra postage and would like to continue sending copies of your newsletter to me as well, I would be most happy to receive them. But your group's official "exchange" copy should be sent to Charley, not to me.

Of course, this also implies that any and all correspondence for NORTHERN BYTES should now go to TAS. Any NORTHERN BYTES related correspondence that is sent to me by mistake will (in most cases) be forwarded directly to TAS, without intervention on my part.

There is much I could say in closing, but I think it's just best to say that I have learned a lot while doing NORTHERN BYTES, and have appreciated very much the support and encouragement I have received from many of you. No matter what form future issues of NORTHERN BYTES may take, it is satisfying to know that it has been regarded by many as one of the best TRS-80 publications ever. I'm happy to have been a part of that.

I want to encourage you to help the <u>new</u> editor by sending him your articles, tips, techniques, etc. just as you have done in the past. Remember that NORTHERN BYTES will dry up very quickly without reader participation. I wish Charley the best of success in keeping this publication going, and want to say "Thank You!" to all of you that have been so supportive to this point in time. May God bless you all!

Jack Decker, former editor

P.S. I can't quit without telling my crazy mail story of the month. Courts of law generally tend to assume that when something has been put into the mail, it will be received by the recipient. They shouldn't. Recently I received a mailing from a small computer supply company. It was one sheet of paper, folded over and stapled (once) at the bottom. It practically sang out "I am junk mail!!!!!" But, being the curious sort, I opened it. And what should fall out but a bill payment from a lady in Huber Heights, Ohio, to the Dayton Power and Light Company! The stamp on it had never even been cancelled. What probably happened was that when this lady dropped her payment in the mailbox (or perhaps somewhere in the mail collection process), her payment envelope slipped inside the advertisement. Of course, I dropped the payment back into the mail and (hopefully) Dayton Power and Light got their money, albeit a few days later than they would have otherwise.

This is the second time something like this has happened in recent months. A few months back I received a newsletter from a computer user group in New York and when I opened that one up, there was (what appeared to be) a city tax payment, mailed from a local Veterinarian's office. That one also went back into the mail.

Now you are probably thinking that most people, in a similar circumstance, would do as I did and drop the misdirected mail back into the mailbox. That may or may not be true (certainly the point is open to question with today's moral standards). But if you want an eye-opening experience, stand around the post office lobby (in the area of the P.O. Boxes) right after the post office opens on any business day. Watch the people collect their mail from their P.O. Boxes, and then walk over to the nearest table, begin sorting their mail, and pitch anything that resembles an advertisement into the trashcan, WITHOUT EVEN OPENING IT!! Wait an hour or two and peer into the trashcan, and (if the janitor hasn't been around already) you will see that it is overflowing with UNOPENED "junk" mail. This scene isn't confined to the post office, either - it occurs in the mail rooms of many corporations and at the desks of many businessmen. Those who get lots of mail often feel no obligation to open every piece. So, what happens if your important mail slides into one of those pieces that wind up in the trash?

It's an interesting fact that international mail is supposed to be sent inside an envelope (or at least sealed on all sides), presumably to avoid just this sort of thing (other pieces of mail slipping inside). We used to get copies of Northern Bytes returned as unmailable by the post office when we failed to adhere to this regulation - yet there is no such regulation for domestic mail! Maybe there should be. In the meantime, about the only protection I can see against losing mail in this manner is to mail anything really important in as large an envelope as possible - at least a #10 business envelope.

And don't send your mail to the trashcan unopened - there might really be something important inside!

THE EXTERMINATOR

Some BUGS manage to hide for a long time before anyone discovers them! Such is the case with the program REROUTE (a NEWDOS/80 utility that routes printer output to a disk file) that appeared back in NORTHERN BYTES Volume 5, Number 1 (pages 4-6) and on TAS Public Domain Library Disk #001 (very recent copies of this disk have a corrected version, so check the opening credits of the source code before you start making changes).

The problem was that although the program ran fine on a Model I, it did not get the HIMEM value from the correct location on the Models III or 4. So Model III/4 users could not count on the program operating, at least not reliably. The code below fixes the problem and (with the exception of lines 570, 580, and 790, which are changes to existing lines) should be ADDED TO the source code listing that was printed in NORTHERN BYTES (or to the REROUTE/ASM file on the PD disk, if you have the uncorrected version). Then reassemble the source code to produce a new /CMD file that will operate properly on both the Model I and the Model III/4.

```
00105 ;Model I/III compatible version 06/20/86

00561 INTLZE LD   A.(54H)          ;Get byte from ROM
00562        DEC  A                ;Determine if Mod 1 or 3
00563        JR   NZ,MOD3          ;Go if Model III DOS
00564        LD   HL,4049H         ;Mod I DOS top-of-memory
00565        LD   (MEMSIZ+2),HL    ;Self-modify program
```

```
00566           LD     (STRMEM+1),HL ;  memory pointers
00570 MOD3      LD     HL,END         ;End of unrelocated pgrm
00580 MEMSIZ LD     DE,(4411H)     ;End of unprotected mem

00790 STRMEM LD     (4411H),HL     ;Save new memory size
```

Be sure to read the letters in this issue for other bug fixes!

LETTERS DEPARTMENT

Persons sending letters intended for publication should send them on magnetic media or via Compuserve [72167,161] or MCI Mail [109-7407] (especially if longer than a couple of paragraphs). If 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! Also, please keep in mind that due to the change in editors (see page 1), all mail for NORTHERN BYTES should now be sent to Charley Butler at The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906-5319.

Dear Jack,

In Volume 7, Number 2 there is a program named NEWMAP. I typed it into my Model 4 and found that it did not quite work. Once I began to study the program I found that I really wanted it, so I not only repaired it but I enhanced it. I believe it is now more user friendly and provides a better print out. The following are the changes I made with a explanation.

The major error is in line 570 where the program is trying INT and integer number. The fix takes place in Line 100. Instead of DEFINTA-Y the program should read DEFINTB-Y. Without the fix A will always equal zero.

The next change I made in the program allows the user to type either in lower or upper case. The program will convert the entry to upper case and then make its decision. This is accomplished with the function defined in line 100. I named it UC$ because it will always give the user the uppercase of the letter typed in. Lines 770 and 810 use the function.

The third change is a correction. It too is in line 100. The author used CMD"T" for a command to turn off the real-time clock display. My version of NEWDOS80 does not recognize that command. I changed it to CMD"clock n". In line 160 the author used CMD"R" to turn the clock on, again I changed it to CMD"clock y" for the same reason.

The last change I made was to have the program page and print a header if it there are going to be more than 55 printed lines. These changes start with line 100 where I set the variable LP to the location of the Model I/III location of the LPRINT line counter. It is continued in line 145 where I make sure the counter starts with a one, and lines 160 & 190 where I send a FORM FEED if needed. Line 700 sends a FORM FEED for the second page if needed and prints a second page title.

The following are the changes that I suggest be made:

```
100 CLEAR2000: DEFINTB-Y: DEFSTRS: DEFFNUC$(A$)=CHR$(ASC(A$)+(3
2*(ASC(A$)>97ANDASC(A$)<123))): DIMF$(8,13),Q$(30,8): FORI=5TO8: S(I)=
STRING$(I,32): NEXTI: CMD"clock n": LP=16425

145 POKELP,1: PRINT"Set the printer paper to the top of the page an
d press any key": A$=INKEY$

147 A$=INKEY$: IFA$=""THEN147ELSEA$=""

160 GOSUB770: E$=A$: IFASC(E$)<48ORASC(E$)>51THENCMD"clock y": IF
PEEK(LP)<>1THENLPRINTCHR$(12);: CLEAR50: ENDELSECLEAR50: END

190 IFA$="Y"THEN PRINT"Yes": CMD"ROUTE,PR,PR,DO": IFPEEK(LP)<>1TH
ENLPRINTCHR$(12);: GOTO210ELSEGOTO210

570 C1=1: A=B-INT(B/5)*5: U(0)=A+((A=0)*-5)-1: U(1)=4: FORGG=9TO13: I
FGG<13THENIFASC(F$(F,GG+1))=255THENC1=0

700 NEXTF: LPRINTSTRING$(63,"-"): IFPEEK(LP)>55THENLPRINTCHR$(12)
: LPRINT" ": LPRINT"Map of 705 NEXTM "TAB(23)"#SEC's"TAB(32)"Exten
t"TAB(40)"TRK: TRS"TAB(50)"DSK REL SECS"

705 NEXTM

770 A$=INKEY$: IFA$=""THEN770ELSEIFASC(A$)=31THEN770ELSEA$=FNUC
$(A$): RETURN

810 GL=2: DG=10: TD=0: PRINT: LINEINPUT"Single <S> or double <D> si
ded drive (Default=<S>) ": SX: SX=FNUC$(SX): IFSX="D",TD=2
```

Jack, I found the program excellent. I don't want the author to think I am attacking his program, it is something that I have always wanted but was too lazy to do the research to write.

I have enjoyed NORTHERN BYTES. Keep up the good work. You are filling a void. Without a NORTHERN BYTES the work and play of hackers would be unknown.

Thank you,
Frank Blunda, 3717 Sundown Road, Gaithersburg, Maryland 20879

[Frank, I ran the original program prior to publication and didn't seem to have any problems, but perhaps there is some difference in our systems (hardware or software) that makes a difference. Please see the next letter and my reply.]

Dear Jack:

[Regarding] NEWMAP (Northern Bytes volume 7, number 2, page 17): What a great utility! I could have used this many times in the past few years to sort out problems with diskettes which went "funny".

I discovered a few (very minor) difficulties with the source as published. These difficulties wouldn't cause problems for your typical reader ... who I imagine as a "medium to advanced" grade hacker. Possibly, though, you have many less experienced TRS-80 users as readers. They might get hung up by one or more of these small traps, so I'll mention them while they're fresh in my mind.

References are BASIC line numbers as published.

<u>100</u> - Change DEFINTA-Y to DEFINTA-Z. There is no need to Z variables to default to single precision, which is the result in the program as published.

<u>Lines 100 & 160</u> - CMD"T" and CMD"R" are replaced by CLOCK,N and CLOCK,Y in the Model III version of NEWDOS/80. In NEWMAP, neither performs any useful function as far as I can see. I eliminated them.

<u>110</u> - P = 26949. I wonder if it's sufficiently clear what this is all about? Gash says "... line 110 POKEs a machine language routine directly into the file buffer area for file 3." True enough, IF you're using NEWDOS/80. If you're not, you'll have to refer to Rosenfelder's <u>"BASIC Faster and Better & Other Mysteries"</u>, Appendix 3 to discover what you should set P to.

<u>350 & 680</u> - The string literal, "to", should be changed to "thru". All sector ranges reported by NEWMAP are INCLUSIVE, thus "thru" is more explicit.

<u>330</u> - PR = VARPTR(D$) returns a result in signed integer format. Thus, in a 32k/48k device, PR is likely to be negative. The algorithm published works ONLY if PR is positive. Here's an algorithm which works when PR is EITHER positive OR negative:

```
330 Z = VARPTR( D$ ): ZL = Z AND 255
332 ZM = ( Z AND -256 )/ 256
334 IF ZM <0 THEN ZM = ZM + 256
336 POKE P +  9, ZL
338 POKE P + 10, ZM
```

<u>420</u> - This line is part of a sequence which offers the user some control over NEWMAP output. While it's cumbersome to use, it does work if this change is made in 420:

```
Change IF PEEK( 14400 ) = 128 THEN ...
-to-    IF PEEK( 14400 ) >= 128 THEN ...
```

This change introduces a pause in program execution when either the <SPACE> key OR the <SPACE> + <ENTER> keys are held down at the proper time during NEWMAP execution.

Well, Jack, so much for NEWMAP. Thanks again for publishing it!

Cordially, Gil Spencer
Box 300, Spit Junction, New South Wales 2088, AUSTRALIA

[Gil, what's interesting to me is that both you and Frank (see preceding letter) felt the need to make changes to NEWMAP, yet (except for the changes to CMD"T" and CMD"R") the two of you did not seem to have much agreement on which changes should be made! When I prepared NEWMAP for inclusion on TAS Public Domain Library disk #020, I made several minor changes (like replacing implied THEN statements - indicated by a comma - with the actual keyword THEN), and a few not-so-minor changes, mostly based on suggestions from both of your letters (although I did not include <u>all</u> of your suggested changes in the PD Library version). The lines that had more than just cosmetic changes are printed below. Changes are <u>underlined</u> (where blank space is underlined, it means that something was deleted from that line at that point):

```
100 CLEAR10000:DEFINTB-Z:DEFSTRS:DEFFNUC$(A$)=CHR$(ASC(A$)+(32*
(ASC(A$)>97ANDASC(A$)<123))):DIMF$(8,13),Q$(30,8):FORI=5TO8:S(I)=STRIN
G$(I,32):NEXT_______
160 GOSUB770:E$=A$:IFASC(E$)<48ORASC(E$)>51THEN_______CLEAR50:EN
D
330 PR=VARPTR(D$):POKEP+10,(PR+(PR<0)*-65536)/256:POKEP+9,PRAND25
5
420 DS=2:IFPEEK(14400)>127THEN420ELSEPRINT:PRINT"WHICH DISK SECT
OR ( 2 TO"DG-1")TO START FROM ";:INPUTDS:DS=DS+1
770 A$=INKEY$:IFA$=""THEN770ELSEIFASC(A$)=31THEN770ELSEA$=FNUC$
(A$):RETURN
810 GL=2:DG=10:TD=0:PRINT:LINEINPUT"SINGLE <S> OR DOUBLE <D> SID
ED DRIVE (DEFAULT=<S>) ";SX:SX=FNUC$(SX):IFSX="D"THENTD=2
```

You'll note that I used a bit simpler method than the suggested one to extract the proper POKE values in line 330 - although I have never had a problem with line 330 in the original version of NEWMAP, the changes shown should insure that no error can occur there. The only problem I've ever personally had with NEWMAP is running out of string space (while mapping an 80-track double sided disk) and that is corrected by the change in the CLEAR statement in line 100.

Again, thanks to both Frank and Gil for their comments on NEWMAP. If anyone else has had similar problems with the program, perhaps the above suggestions will help.]

Dear Jack,

First, I will join the growing chorus of praise for the "new look" of Northern Bytes. The new print quality is great. Then I will hit on several other topics that come to mind as a result of reading Northern Bytes for about a year.

For the year that I have subscribed to Northern Bytes, it has been full of letters and articles about NEWDOS, patches for NEWDOS, praise for NEWDOS, etc. Even you, Jack, admit to being a NEWDOS lover. If that is not bias then I will eat my aging 1963 copy of Webster's New Collegiate Dictionary. On page 82 it offers the following as definition for bias -- "an inclination of temperament or outlook", "prejudice", "bent", or "tendency". But bias is not all bad. After all, without it our little semi-conductors would not be typing this now! (I will confess up front that my favorite operating system on my Model III is LDOS, though I will use any DOS necessary to run a software package if it is choosey.) I like its versatility and I have not found it difficult to learn as many people indicate.

Now to more serious matters, specifically Roy Soltoff. I am not acquainted with Mr. Soltoff personally (and I can understand how his condemnation of hardware manufacturers that make disk drives that run "exactly on speed" might seem like "arrogance" to many people), but on the several occasions that I have spoken to him on the phone he has been most helpful. This applies both to answering queries about his software prior to purchase and to post sales support. In addition he was friendly enough on some occasions to enter into discussion with me on software and the computer BUSINESS in general, not having anything directly to do with sales to me. If he had any inclination to be arrogant or superior, he could certainly have been justified in being so with me, as I am a rank amateur at hacking.

More importantly, in this time of decreasing support for Z80 machines generally and TRS-80's specifically, we should feel very glad that Misosys is sticking with us (Apparat didn't). Actually, there have been few companies put out software that is as smooth operating, professional and sophisticated as Misosys, LSI and Roy Soltoff have, for our machines, especially along the lines of "serious" utilities.

Jack, I know that several times you have stated that the "bias" toward NEWDOS in Northern Bytes is mainly produced because the material that you receive has been mostly from NEWDOS users. I hereby resolve to do my best to change that trend. To that end, I have [sent] HIMAP, an assembly language program that produces a directory and map of high memory usage under LDOS (provided that all high memory residents have the standard header as specified by LSI) and WPFLT, a printer filter for LDOS that adds some "word processing" features to the comment portion of an assembler listing (or to any printer output with a minor modification indicated in the listing). These are programs that I wrote and are in the public domain. You may publish them in Northern Bytes or file them in the trash can as you see fit. I will also be submitting these programs to the Alternate Source for possible inclusion on one of their public domain disks.

I want to express my appreciation for the work you are doing to support the Z-80 TRS-80's. As far as I know 80-Micro is the only other magazine supporting us and with the recent re-introduction of the color computer material and its big shift to the MSDOS machines, it looks like we are on our own. There is still life in my Z-80 and I sure don't want it to be lonely. So keep up the good work!

Sincerely, Bill McQueen

[Thanks for the submissions, Bill. Despite the amount of ribbing that Roy Soltoff has taken in these pages for his comment about the "fiasco" of a disk drive that always runs at exactly the correct speed, we should all remember that Roy has contributed a great deal to the community of TRS-80 users. I think the basic problem is that Roy (and some others who are devoted LDOS/TRSDOS 6 users) think that we should all share their opinions on which DOS to use, how to program (always use SVC's, never use PEEKs and POKEs, etc.), and some of the rest of us don't appreciate being told how to think. Granted that Roy is a professional programmer and has written some very fine programs, but that does not mean that the rest of us are unprofessional if we choose not to use the same DOS or programming techniques that Roy uses, or that LSI recommends that we use (on the other hand, it should be noted that there are often very good reasons for LSI's suggested programming techniques). I guess I just resent the underlying implication that anyone who prefers a DOS other than LDOS/TRSDOS 6 is something less than a "real" programmer. So maybe I am a bit "biased", but only against the "superiority complex" I sometimes detect from LDOS users.

Of course, if you happen to agree that LDOS is the best DOS, etc. you probably will get along famously with Roy and other LDOS devotees. But we like you, too!]

Jack,

On page 18 of volume 7, number 3 there is a question from the M.A.T.U.G. newsletter about a BASIC program that yields different results depending whether it is run in the III or 4 mode. The problem probably is caused by lines 170 and 180 in the listed program. These lines execute single precision-to-integer conversions, which are handled differently under the two BASICs. Model III BASIC truncates (2.5 -> 2), and Model 4 BASIC rounds (2.5 -> 3).

I did not test this out, but I've seen this problem crop up numerous times.

Mike Zarowitz

[No doubt that's the problem, Mike. I wasn't aware of the difference in the way the two versions of BASIC handle the INT function (it's probably in the manual, but who reads manuals?).]

Dear Jack,

... Your comments about Fast80 and where you got that information instantly became logical, i.e., you got it from reading the newsletters from other clubs. I am guilty of that myself once, I took aim at Prosoft and was written off as a dummy by those who had the program and liked it. So much for being a responsible editor myself (maybe I should practice what I preach). As for background on Fast80, you are correct in your assumption that it was designed to fit into memory. There were a number of reasons for this and I'll at least try to point some of them out.

I have been running a BBS system since 1979 when I started with the first generation of EBBS. It ran in BASIC with a ML driver in high mem and had about 8 overlays. To say that it was slow, would be an understatement. I still get about 80-100 calls a day (when I am not using the system for programming on) and about 40% of those were long distance. Back in '79 a diskette lasted about 30 days. Then you could start getting I/O errors. I was only providing a public service, so I tried to cut down on the cost (which at that time was more expensive than now). I re-wrote the BASIC portion and Online 80 was born out of it. I ran that for another 2 years and then, changed to DOSPLUS. Again the BASIC portion was redone and the machine language driver was also redone. I run that on the final days of my Model 1 and finally on my new Model 3 (summer '83). I still was going through too many diskettes and the access time was still too slow.

In '84, I bought a nice new Model 4 (couldn't wait for a real computer with an 80 column screen). And tried to get the BBS to run on it. Naturally the RST's fouled me up for a while, and I was trying to use TRSDOS 6.1.0 and it sucked. I eventually went to DOSPLUS 4 and got a version running and it was all in machine language. You must keep in mind that I have always only had floppy disk systems.

That last version was the forerunner to Fast80 at present. I had about 5 years of feedback from users to extract from and I sat down and wrote exactly what they suggested. About the same time I met SOTA systems and they liked what they saw in the initial version and encouraged me to continue with development. Which brings us up to date.

Fast80 had certain design limitations to meet. I must allow only MINIMUM diskette wear. Message retrieval and entry must be FAST, no indexes could be created. It must use the total 128K ram to store ALL the information it required and provide a programmable update rate where needed. I tried to keep it user friendly and think I did at least that. I had to provide some form of file transfer. I completely agree with anyone who says that the current version lacks in the transfer area. When I wrote it, I must admit that XMODEM was about as clear as San Francisco fog to me. I understood only the XON/XOFF protocols. Hence XMODEM was left out.

However since then I have written a terminal program (which should have been on that disk I sent you "FT/CMD" and was released as shareware this spring). I learned about XMODEM etc. and will be using that information later on.

To continue, as of May 1 this year there are 136 Fast80 Systems in operation. From those Sysops (most of which are hard pressed to type in the date and time), I receive feedback. I have started with a new version of the BBS again on the guidelines of the information received, i.e., xmodem, fast access, etc. Only this time, there will be up to 16 sub-boards. The message base will still be limited, but I am using encoding techniques to store the messages in memory so more will fit. The download menu has been increased from a maximum of 50 files, to 300. Well as you may be able to see, it was completely redone from scratch. Even the driver will be changed. Modem support will be increased and on and on.....But the final version will still be limited by available memory.

I am still writing it at this point and hope to have a version up by the end of the summer. As I mentioned, I have always used a floppy based system for a BBS. It turns out that about 40% of Fast80 owners use a Hard Disk and want more features. I have been looking at sort of trying to keep both families happy to best extent but it's not easy. There are choices to make that will no doubt make some unhappy. I have been examining other BBS offerings like TBBS, SYSLINK, FRENCH CONNECTION, etc. and I think for a floppy based system, those are pretty useless. A hard disk is required. If I am wrong, I would like to know the technique used to store 9000 messages of 1K each on a standard 2 drive Model 4. Granted they use what they have to fullest, but they are even more limited than Fast80 in some areas.

Mel Patrick

[This letter is actually one out of a short exchange of letters between Mel and myself, which was prompted by Gil Spencer's letter in Northern Bytes Volume 7, Number 3 and by my comments which followed that letter. I plead guilty to judging Fast80 by a couple of comments that I read in other newsletters. One thing that must be remembered is that each BBS program has its own unique design criteria. Fast80 was designed with minimizing disk wear as a primary consideration. SYSLINK was designed to permit networking between BBS's, something previously only done by the Fido BBS program (which runs on MS-DOS machines) - and SYSLINK permits both file and message transfer. TBBS was, I think, designed for message base handling and irritating BBS users, particularly when it drops callers without warning when their time has expired, even if they are typing the last line of a 20 line termination message to the SYSOP (I don't care how much the designers of TBBS may protest that this doesn't happen - it DOES, or at least it used to. It happened to me and to MANY other users, and SYSOPs complained about it for years, to no avail. Maybe the problem's been fixed in a recent version but I swore off TBBS's about a year ago, after being disconnected without warning just once too often!).

In any case, potential purchasers of a BBS program should compare features and if possible, call a BBS using each program under consideration and try out all of the features that are important to you.]

Jack,

I have some information about MCI mail you might find interesting.

Seems I have become a member for life. I joined when it was free (no annual fee) and when they raised the rates and tacked on the annual fee I sent them a letter explaining why I wanted off and paid my last bill (minus the annual fee).

This was last December. And since I received no reply and no past due notices I assumed all was fine.

Boy was I wrong. You don't respectfully decline MCI mail. I received a letter from their lawyer. He is taking me to court for the annual fee.

I don't know yet what is going to happen. I suppose the annual fee is cheaper then a lawyer, unless they never let me quit and I have to pay it forever. That's silly you say, but should I refuse to pay they can play havoc with my credit rating. So I'm over the proverbial barrel.

This is not the first time I have had problems with MCI. And it is not the first time they pulled something I considered shaky. I am going to fire off letters to the BBB, them and their lawyer, and I'm going to quit the MCI long distance service. This is not the kind of company I want to be associated with (or owe money to).

One word of warning. The BBB said I should have insisted on a final bill, so that six months down the line they couldn't claim I still owed them.

Dave Bower

[After receiving the above letter, I sent Dave a reply containing the following suggestions:

1) Call their 800 number - I think it's (800) MCI-2255 and/or (800) 424-6677. Talk to them on <u>their</u> nickle. <u>Write down</u> the name of each person you talk to. If you feel you are getting nowhere with the person you're talking to, ask to talk to that person's supervisor. However, I have found that the people who answer MCI Mail's 800 number are usually able to resolve problems such as this (we've had a few ourselves). <u>Keep calling back</u> until the problem is resolved.

2) <u>Don't</u> pay the $18 mailbox fee and don't worry about your credit rating. No reputable lender is going to refuse to loan you money because of an $18 disputed bill. Should you ever apply for credit, you can always tell the lender up front that there may be an $18 disputed charge from MCI appearing on your credit rating, but it was for service that you did not order and you are waiting for MCI to either sue you (which would be REAL interesting) or drop their demand for payment. Incidentally, I seriously doubt that anything will ever appear in your credit rating because of this. It takes a lot more to damage a credit rating than an $18 disputed bill.

3) DON'T be intimidated by letters from attorneys. In some cases the "attorney" may only exist in the mind of a collection agency computer (there may be a real attorney with that name somewhere, but dollars to doughnuts he's never seen the letter you've received - it probably came straight from computer printer to you). On the off chance that I am wrong, keep in mind that you still have the right to tell your side of the story in court.

The Alternate Source had a similar situation, where one of their unused accounts was cancelled but MCI Mail still billed them for the annual mailbox fee, so we KNOW that MCI Mail is capable of making such errors. Fortunately, in that case a telephone call to the 800 number resolved the problem. I think large companies would be a lot better off if they would invest a little time and money to call their customers to find out whether there is a problem, instead of firing off letters from attorneys as part of their routine correspondence, thereby alienating the customer. Especially if the customer gets mad enough to start writing letters, which (when printed in a publication like ours) probably wind up costing the company a lot more than the money they're trying to collect, due to the chilling effect such letters have on potential new customers.

One further word of advice when dealing with any large company of this type - if communications seem to be breaking down, tell your problem to someone fairly high up on the

managemen
incompeten
chain (rem
if you can
if it's just
more weig
than if it
I'm s
such error
similar sit
attorneys.
reason, wi

Dear Jack
First
the patch
full chara
and recor
You
default S
parity to
record 2,
rate rem
incidental
LSI when
other gro
Her
bits inste
byte 54 (
The
called
buy/trad
efunct.
Tha

Dep

[An
Rod Ste
patch to
at recor
I'm not
thing as
versions
able to

Dear Jac
I
LOAD-80
Categor
program
publicat
out wit
program
interest
Do
UltraTer

[T
UltraTe
index,
should
like a
might
address

[
assista
any of
and, if
send a

management ladder. Lower level flunkeys are sometimes borderline incompetent, and tend not to pass messages up the management chain (remember NASA and the Space Shuttle?). On the other hand, if you can reach someone in the company president's office (even if it's just his secretary), you can bet that your message will carry more weight if it comes down the management chain from above than if it has to work its way up.

I'm sure that MCI Mail isn't the only large company to make such errors, so the above hints may be useful to other readers in similar situations. Above all, don't be intimidated by letters from attorneys. Remember that, while anybody can sue anybody for any reason, winning the case is something else altogether.]

Dear Jack,

First, here are three more patches. The first was inspired by the patches on page 18 of [Volume 7, Number 2]. To display the full character set in LS-FEDII (not FED), find record 16, byte D7 and record 1E, byte C0 of the program; change 3E84 to 0000.

You may already have published the second. To change the default SETCOM in TRSDOS 6.2 from 7 bit word length and even parity to the more common 8 bit word length and no parity, find record 2, byte B1 of COM/DVR.DRIVER; change A5 to ED. The baud rate remains at 300, and the stop bit at 1. (This change, incidentally, was given to me over the phone several months ago by LSI when I called them and asked how to do it -- I know of no other group that supportive or knowledgeable about its programs!)

Here is the third patch. If you want to use COMM with 8 bits instead of 7 as the default, you need to find record 4, relative byte 54 (hex) of COMM/CMD.UTILITY and change 00 to 01.

The second item: I am looking for the disk version of a game called Volcano Hunter. Does anyone know where I can buy/trade/acquire this program? The manufacturer appears to be defunct.

Thanks a bunch and keep up the good work...

Gary W. Shanafelt
Department of History, McMurry College, Abilene, Texas 79697

[And thank you for the patches. Here's a related one, from Rod Stevenson of the Adelaide (Australia) Micro User Group: The patch to display all characters in the TRSDOS 6 version of FED is at record 16, byte D2. The original value is 38, change it to C9. I'm not sure if this is an alternate method to accomplish the same thing as your first patch, or if we are talking about two different versions of FED, but I'm sure that loyal TRSDOS 6 hackers will be able to figure it out!]

Dear Jack,

I wrote a program called LOAD80 INDEX. It has files on all LOAD-80 programs from 1981 thru March 1986 listed by File Name, Category, Month/Year/Page, and a comment line describing the program. The files can be searched by category or by date of publication or scanned as a total file. The listings can be printed out with LeScript or other large capacity word processor. The program is entirely menu driven. Let me know if you are interested and I'll send you a copy on disk.

Do you ever recall seeing any patches for modifying UltraTerm comm package to run in the Model 4 mode?

William E. (Bill) Baker
2419 Queen Ridge Drive, Independence, Missouri 64055

[To answer the last question first, I've never seen any UltraTerm patches of the kind mentioned. As for the LOAD-80 index, I'd rather not get involved in redistributing it for (what should be) obvious reasons, but would suggest that readers who'd like a copy of this index contact Bill directly (if you write, it might be wise to enclose a blank diskette and a stamped, self addressed disk mailer).]

[NOTE TO READERS: The next four letters are asking for assistance, and I don't have the answers for these folks. So, if any of you can help, please contact the letter writer directly - and, if it's not too much trouble, we'd appreciate it if you'd also send a copy of the answer to us here at NORTHERN BYTES!]

Dear Jack,

NEWDOS/80 version 2 has been a Godsend for my TRS-80. Is there an equivalent for MS-DOS machines? I put GW-BASIC in the same class as Apple BASIC. I especially need good BASIC commands such as REF, RENUM (with ALL operands), CMD"O", the editing commands and line listing commands, etc., and the ease of "FF" random access files with record lengths over 256 bytes.

Dave McGlumphy, 4429 Paula Lane, Chattanooga, Tennessee 37415

Dear Jack,

Communications Problems - Question #1: Recently I purchased the communications package VIDEOTEX PLUS from Radio Shack to run on a Model 4. In the manual mode, the program communicates through the modem to anyone, however, in AUTO-LOG, the modem will not "wake up". Radio Shack has been unable to help. The modem is a Signalman MARK XII which is a Bell System 212 and Hayes compatible. Can anyone out there help me determine what codes I must send through AUTO-LOG in order to store my numbers and issue commands automatically?

Question #2: I own WINDOW-COMM by Pacific Software Consultants and was sent an incomplete manual. I have been completely unable to reach the vendor by phone or mail. If anyone has this software with a good manual, I would like to arrange to copy sections one through three of the manual. I have section four and the Appendix A.

G. R. (Ranny) Robertson, Jr.
2314 Hilliard Road, Richmond, Virginia 23228

Dear Jack,

I am generally using one of the many calendars that are frequently appearing in the magazines and newsletters, all of them coming from your country.

For us, Europeans and Latins, they are written in a different way, because you always present them like:

Sun Mon Tue Wed Thr Fri Sat

but we always start each week by a Monday, that is:

Mon Tue Wed Thr Fri Sat Sun

For the benefit of your subscribers in Europe, could you correct any of the calendars to be read this way and publish it in Northern Bytes? We shall appreciate it very much.

Gabriel Domínguez, Magistrado M. Artime 6, 15004 La Coruña, SPAIN

Hi there.

I have a TRS80 MODEL 4P and a STAR SG-10 printer, and am trying to find a decent RECIPE catalog program. I have my own word processor/editor (Lescript) - just in case you were going to ask.

I would like to be able to keep track of recipes by category (i.e. deserts, breads, meat dishes, fish dishes, etc), and can be flexible as to what the defined categories are. The entry editor should be fairly easy to make corrections with (or perhaps I can use my own), and I should be able to view the recipes on my screen or printer.

Unless the program is really outstanding I usually don't like to buy Model III software due to the screen size limitation. If you have a 'good' Model III program, though, please let me know.

Thanks for your time. Feel free to answer by mail or easyplex. I am:

Ken Kornblum, 6751 Kingswood West, Delton, Michigan 49046
CompuServe EasyPlex [74706,506]

WANTED TO BUY

Two (2) 80 track double sided disk drives for use with a Model 4 TRS-80. Must be reliable as they will be used with a BBS. Contact Neil Trudel, 35 Ontario Avenue, Sault Ste. Marie, Ontario, Canada P6B 1E2 (note: drives can be sent to a U.S. address if you do not wish to ship them to Canada), or telephone (705) 253-5514 (voice line) or (705) 253-5366 (BBS line - 300 bps only).

[redacted]

by Bill McQueen

This is the source code for HIMAP/CMD, in MISOSYS EDAS or MRAS listing file format. The documentation for this program is in the source code comment lines.

```
                  00001 ;                  ===== HIMAP VER 3.0.1 =====
                  00002 ;                          6/2/86
                  00003 ;
                  00004 ;                  by:  Bill McQueen
                  00005 ;                          805 Ely Road
                  00006 ;                          Hixson, Tn
                  00007 ;                          (615) 875-0731
                  00008 ;
                  00009 ;
                  00010 ;                     === NOTICE ===
                  00011 ;                         ------
                  00012 ;
                  00013 ;This program placed in the PUBLIC DOMAIN and submitted to
                  00014 ;NORTHERN BYTES and THE ALTERNATE SOURCE" by Bill McQueen.
                  00015 ;It may be used, modified, given away etc. by anyone.
                  00016 ;Please give credit to the author and NORTHERN BYTES or
                  00017 ;THE ALTERNATE SOURCE
                  00018 ;-------------------------------------------------------
                  00019 ;
                  00020 ;This program is a utility for use with LDOS 5.1.x. (tm).
                  00021 ;It will display a directory and map of all modules
                  00022 ;resident in high memory, provided all are identified with
                  00023 ;the LDOS (tm) standard header.
                  00024 ;All system modules, filters, drivers etc. by Logical
                  00025 ;Systems, Inc (tm) and MISOSYS (tm) intended to
                  00026 ;be resident in high memory have this header.  (If you need
                  00027 ;info on this header and can't find it in your LDOS
                  00028 ;documentation, contact LSI, INC or the author).
                  00029 ;
                  00030 ;This source file was written with SAID, the text editor
                  00031 ;by MISOSYS to be assembled on EDAS (tm), the MISOSYS
                  00032 ;macro assembler.
                  00033 ;
                  00034 ;        NOTE:   VERSION 3.0.1 is the same program as
                  00035 ;                Ver 3.0.0 except that it has been
                  00036 ;                re-arranged so that it contains the
                  00037 ;                appropriate portions of the FBUFF
                  00038 ;                library in its source code. Therefore
                  00039 ;                it can be assembled with EDAS or other
                  00040 ;                non-relocatable assembler.
                  00041 ;
                  00042 ;If you have questions or want a copy of the object code
                  00043 ;of this program, contact the author.  Improved versions
                  00044 ;are planned.
                  00045 ;
                  00046 ;-------------------------------------------------------
0003              00047 ETX      EQU     03H              ;end of text mark (ASCII)
000D              00048 CRTN     EQU     0DH              ;ASCII carriage return
000A              00049 LNFEED   EQU     0AH              ;ASCII end of line mark
4467              00050 @DSPLY   EQU     4467H            ;LDOS display routine
402D              00051 @EXIT    EQU     402DH            ;Return to DOS
000F              00052 D0_MSK   EQU     00001111B        ;Digit 0 mask
00F0              00053 D1_MSK   EQU     11110000B        ;Digit 1 mask
                  00054 ; MODEL III EQUATES
4411              00055 HIGH3$   EQU     4411H
                  00056 ;----------------------------------------------
                  00057 ;
                  00059 ;
5200'             00060          ORG     5200H
5200' 0000        00061 BEGIN    DW      0                ;address of beginning
                  00062                                   ;of code segment.
5202' 0000        00063 END      DW      0                ;AND ITS END.
                  00064 ;
                  00065 ENTRY
5204' 2A1144      00066          LD      HL,(HIGH3$)      ;Fetch and
5207' 220252'     00067          LD      (END),HL         ;save HIGH$
520A' 2B          00068          DEC     HL
520B' F9          00069          LD      SP,HL            ;Set Stack
520C' 219053'     00070          LD      HL,BANNER        ;Clear screen and
520F' CD6744      00071          CALL    @DSPLY           ;print banner.
5212' CD3D52'     00072          CALL    SHO_HI           ;display high$
                  00073 LOOP1
5215' 2A0252'     00074          LD      HL,(END)         ;HL = High$
5218' 23          00075          INC     HL               ;HL = First byte of hi mem
5219' 220052'     00076          LD      (BEGIN),HL       ;save it
521C' 7C          00077          LD      A,H              ;Check to see if we are
521D' FE00        00078          CP      0                ;at top of physical memory
521F' 2005        00079          JR      NZ,SKIP1         ;Jump if not.
5221' 7D          00080          LD      A,L
5222' FE00        00081          CP      0
5224' 2814        00082          JR      Z,NO_HI          ;If so jump --------->
                  00083 SKIP1                             ;There is some high mem
                  00084                                   ;storage left.
5226' 23          00085          INC     HL
5227' 23          00086          INC     HL               ;HL-->Begining of segment
5228' E5          00087          PUSH    HL               ;Transfer it
5229' DDE1        00088          POP     IX               ;to IX.
522B' DD5E00      00089          LD      E,(IX)           ;Then get address of
522E' DD5601      00090          LD      D,(IX+1)         ;end of segment
5231' ED530252'   00091          LD      (END),DE         ;and save it.
5235' CD6952'     00092          CALL    SHO_SEG          ;Display the data on CRT
5238' 18DB        00093          JR      LOOP1            ;Then loop.
                  00094 ;
                  00095 ;----------------------------------------------
                  00096 NO_HI
                  00097 ;
                  00098 ;        There is no more High Memory so ...
523A' C32D40      00099          JP      @EXIT            ; return to LDOS
                  00100 ;----------------------------------------------
                  00101 SHO_HI
                  00102 ;
                  00103 ;        Display the current HIGH$ value.
                  00104 ;
523D' DD21E953'   00105          LD      IX,BCB           ;IX --> Buffer control block
5241' CDA552'     00106          CALL    CLRB             ;Clear work buffer
5244' 11D453'     00107          LD      DE,L1A           ;DE --> txt to be printed
5247' 3E00        00108          LD      A,0              ;No chr count, stop on ETX
5249' CDC552'     00109          CALL    ASC_BUF          ;Put it in buffer
524C' 3E10        00110          LD      A,16             ;Tab val
524E' CDB552'     00111          CALL    TABA             ;Move cursor
5251' ED4B0252'   00112          LD      BC,(END)         ;BC = HIGH$
5255' CDCD52'     00113          CALL    HEX_BUF          ;Print hex val to buffer
5258' 3E0A        00114          LD      A,LNFEED         ;Let's add a Line Feed
525A' CDAD52'     00115          CALL    WR_BYT           ;Writhe that byte to buffer
525D' 3E0D        00116          LD      A,CRTN           ;Also a Carriage Return
525F' CDAD52'     00117          CALL    WR_BYT           ;Mark end of work buffer
5262' 21F853'     00118          LD      HL,BUFFER        ;HL --> buffer
5265' CD6744      00119          CALL    @DSPLY           ;Print to screen
5268' C9          00120          RET
                  00121 ;
                  00122 ;----------------------------------------------
                  00123 ;
                  00124 SHO_SEG
                  00125 ;
                  00126 ;        DISPLAY THE SEGMENT DATA
                  00127 ;
5269' DD21E953'   00128          LD      IX,BCB           ;IX --> Buffer
526D' CDA552'     00129          CALL    CLRB             ;Clear text format buffer
5270' 23          00130          INC     HL
5271' 23          00131          INC     HL               ;HL--> length of name
5272' 7E          00132          LD      A,(HL)           ;Nr of bytes to move
5273' E5          00133          PUSH    HL               ;Transfer it to DE
5274' D1          00134          POP     DE
5275' 13          00135          INC     DE               ;DE--> name
5276' CDC552'     00136          CALL    ASC_BUF          ;Move name to buffer
5279' 3E10        00137          LD      A,16             ;Tab value
527B' CDB552'     00138          CALL    TABA             ;Tab to pos 16
527E' ED4B0052'   00139          LD      BC,(BEGIN)       ;BC = start add of seg
5282' CDCD52'     00140          CALL    HEX_BUF          ;Put seg start add to buffer
5285' 3E15        00141          LD      A,16+5           ;Tab value
5287' CDB552'     00142          CALL    TABA             ;Tab
528A' 3E00        00143          LD      A,0              ;No char count
528C' 11E153'     00144          LD      DE,LXB           ;DE --> txt
528F' CDC552'     00145          CALL    ASC_BUF          ;Move text to buffer
5292' ED4B0252'   00146          LD      BC,(END)         ;BC = Seg Ending Address
5296' CDCD52'     00147          CALL    HEX_BUF          ;Write hex value to buffer
5299' 3E0D        00148          LD      A,CRTN           ;Lets put a Carriage Ret
529B' CDAD52'     00149          CALL    WR_BYT           ;Mark end of txt
529E' 21F853'     00150          LD      HL,BUFFER        ;HL --> Buffer
52A1' CD6744      00151          CALL @DSPLY              ;Display to CRT
52A4' C9          00152          RET
                  00153 ;
                  00154 ;==============================================
              00155 ;         The following are portions of the
              00156 ;         ASCII Buffer Formatting Library (FBUFF)
              00157 ;==========================================================
              00158 SAVE_REGISTERS  MACRO
              00159           PUSH    HL
              00160           PUSH    DE
              00161           PUSH    BC
              00162           ENDM
              00163 ;
              00164 RESTORE_REG     MACRO
              00165           POP     BC
              00166           POP     DE
              00167           POP     HL
              00168           ENDM
              00169 ;
              00170 ;=========================================================
              00171 CLRB
52A5'         00172           SAVE_REGISTERS
52A5' E5      00173+          PUSH    HL
52A6' D5      00174+          PUSH    DE
52A7' C5      00175+          PUSH    BC
52A8' CD0952' 00176           CALL    $CLRB
52AB' 1828    00177           JR      $EXIT
              00178 WR_BYT
52AD'         00179           SAVE_REGISTERS
52AD' E5      00180+          PUSH    HL
52AE' D5      00181+          PUSH    DE
52AF' C5      00182+          PUSH    BC
52B0' CDF152' 00183           CALL    $WR_BYT
52B3' 1820    00184           JR      $EXIT
              00185 TABA
52B5'         00186           SAVE_REGISTERS
52B5' E5      00187+          PUSH    HL
52B6' D5      00188+          PUSH    DE
52B7' C5      00189+          PUSH    BC
52B8' CD0453' 00190           CALL    $TABA
52BB' 1818    00191           JR      $EXIT
              00192 TABR
52BD'         00193           SAVE_REGISTERS
52BD' E5      00194+          PUSH    HL
52BE' D5      00195+          PUSH    DE
52BF' C5      00196+          PUSH    BC
52C0' CD1C53' 00197           CALL    $TABR
52C3' 1810    00198           JR      $EXIT
              00199 ASC_BUF
52C5'         00200           SAVE_REGISTERS
52C5' E5      00201+          PUSH    HL
52C6' D5      00202+          PUSH    DE
52C7' C5      00203+          PUSH    BC
52C8' CD2F53' 00204           CALL    $ASC_BUF
52CB' 1808    00205           JR      $EXIT
              00206 HEX_BUF
52CD'         00207           SAVE_REGISTERS
52CD' E5      00208+          PUSH    HL
52CE' D5      00209+          PUSH    DE
52CF' C5      00210+          PUSH    BC
52D0' CD0653' 00211           CALL    $HEX_BUF
52D3' 1800    00212           JR      $EXIT
              00213 ;
              00214 $EXIT
52D5'         00215           RESTORE_REG
52D5' C1      00216+          POP     BC
52D6' D1      00217+          POP     DE
52D7' E1      00218+          POP     HL
52D8' C9      00219           RET
              00220 ;
              00221 ;==========================================================
              00222 $CLRB
52D9' DD6E00  00223           LD      L,(IX)          ;Fetch buffer address
52DC' DD6601  00224           LD      H,(IX+1)        ;(HL-->buffer)
52DF' DD4602  00225           LD      B,(IX+2)        ;Fetch buffer length
52E2' 3E20    00226           LD      A,20H           ;A = space
52E4' 77      00227 $LOOP1    LD      (HL),A          ;set char to space
52E5' 23      00228           INC     HL              ;Bump ctr
52E6' 10FC    00229           DJNZ    $LOOP1          ;Repeat til buffer clear
52E8' DD7E05  00230           LD      A,(IX+5)        ;Fetch ETX mark value
52EB' 77      00231           LD      (HL),A          ;Mark end of buffer
52EC' AF      00232           XOR     A
52ED' DD7703  00233           LD      (IX+3),A        ;Set "cursor" to zero
52F0' C9      00234           RET
```

```
              00235 ;
              00236 ;-----------------------------------------------------
              00237 $WR_BYT
52F1' DD6E00  00238           LD      L,(IX)          ;Fetch buffer address
52F4' DD6601  00239           LD      H,(IX+1)
52F7' DD5E03  00240           LD      E,(IX+3)        ;Fetch cursor postion
52FA' 1600    00241           LD      D,0
52FC' 19      00242           ADD     HL,DE           ;HL--> next chr in buff
52FD' 77      00243           LD      (HL),A          ;put byte in buffer
52FE' DD7E03  00244           LD      A,(IX+3)
5301' 3C      00245           INC     A
5302' 18B1    00246           JR      TABA            ;Advance cursor
              00247 ;
              00248 ;-----------------------------------------------------
              00249 $TABA
5304' 4F      00250           LD      C,A             ;Save argument
5305' 2600    00251           LD      H,0
5307' 6F      00252           LD      L,A             ;HL = argument (tab)
5308' 1600    00253           LD      D,0
530A' DD5E02  00254           LD      E,(IX+2)        ;DE = buffer length
530D' AF      00255           XOR     A               ;clear flags
530E' ED52    00256           SBC     HL,DE           ;Subtract buffer length
5310' FA1553' 00257           JP      M,GOOD_TAB      ;Jp if tab within buffer
5313' 1813    00258           JR      TAB_ERR         ;Else report an error
              00259 GOOD_TAB
5315' 79      00260           LD      A,C             ;Restore argument
5316' DD7703  00261           LD      (IX+3),A        ;Set cursor
5319' AF      00262           XOR     A               ;Set zero flg
531A' 180F    00263           JR      EXIT_TAB
              00264 ;
              00265 ;-----------------------------------------------------
              00266 $TABR
531C' 2600    00267           LD      H,0
531E' 6F      00268           LD      L,A             ;HL = argument
531F' 1600    00269           LD      D,0
5321' DD5E03  00270           LD      E,(IX+3)        ;DE = current cursor position
5324' 19      00271           ADD     HL,DE           ;HL = requested position
5325' 7D      00272           LD      A,L             ;A = requested position
5326' 188D    00273           JR      TABA
              00274 ;
              00275 ;--------------------------------------------------------
              00276 TAB_ERR
5328' AF      00277           XOR     A               ;Clear A
5329' C601    00278           ADD     A,1             ;Set non-zero Condition
              00279 EXIT_TAB
532B' DD7E03  00280           LD      A,(IX+3)        ;Restore A to resulting
              00281                                   ;cursor position
532E' C9      00282           RET
              00283 ;
              00284 ;--------------------------------------------------------
              00285 $ASC_BUF
532F' DD7707  00286           LD      (IX+7),A        ;Save count
5332' DD7708  00287           LD      (IX+8),A        ;Twice
              00288 LOOP2
5335' DD7E08  00289           LD      A,(IX+8)        ;Fetch original Count
5338' FE00    00290           CP      0               ;Was it zero?
533A' 280A    00291           JR      Z,SKIP2         ;If so skip count check
533C' DD7E07  00292           LD      A,(IX+7)        ;Fetch count
533F' FE00    00293           CP      0
5341' 281A    00294           JR      Z,END_TXT       ;Max chr count, quit
5343' DD3507  00295           DEC     (IX+7)          ;Decrement chr cntr
              00296 SKIP2
5346' 1A      00297           LD      A,(DE)          ;Fetch Character
5347' FE00    00298           CP      0               ;Is it a zero byte?
5349' 2812    00299           JR      Z,END_TXT       ;If so get out
534B' FE0D    00300           CP      0DH             ;Is it a carriage return?
534D' 280E    00301           JR      Z,END_TXT
534F' FE03    00302           CP      03H             ;Is it an ETX mark?
5351' 280A    00303           JR      Z,END_TXT
5353' D5      00304           PUSH    DE
5354' CDAD52' 00305           CALL    WR_BYT          ;put char in buffer
5357' D1      00306           POP     DE
5358' 2003    00307           JR      NZ,END_TXT      ;Quit if end of buffer
535A' 13      00308           INC     DE              ;Bump string pointer
535B' 18D8    00309           JR      LOOP2           ;and do it again
              00310 ;
              00311 END_TXT
535D' AF      00312           XOR     A               ;clear flags
535E' 18CB    00313           JR      EXIT_TAB
              00314 ;
```

```
               00315 ;------------------------------------------------------
               00316 ;
000F           00317 D0_MSK  EQU     00001111B       ;Digit 0 mask
00F0           00318 D1_MSK  EQU     11110000B       ;Digit 1 mask
               00319 ;
               00320 $HEX_BUF
               00321         ;enter with hex number in BC
               00322 ;
5360' 78       00323         LD      A,B             ;get msb
5361' CD7553'  00324         CALL    CNVT_DIG
5364' 78       00325         LD      A,B
5365' E60F     00326         AND     D0_MSK
5367' CD7D53'  00327         CALL    CNVT_DIG2
536A' 79       00328         LD      A,C
536B' CD7553'  00329         CALL    CNVT_DIG
536E' 79       00330         LD      A,C
536F' E60F     00331         AND     D0_MSK
5371' CD7D53'  00332         CALL    CNVT_DIG2
5374' C9       00333         RET
               00334 ;------------------------------------------------------
               00335 CNVT_DIG
5375' CB3F     00336         SRL     A
5377' CB3F     00337         SRL     A
5379' CB3F     00338         SRL     A
537B' CB3F     00339         SRL     A
               00340 CNVT_DIG2
537D' C5       00341         PUSH    BC
537E' CD8653'  00342         CALL    ADJUST          ;cnvt to ASCII Hex digit
5381' CDAD52'  00343         CALL    WR_BYT          ;put digit in buffer
5384' C1       00344         POP     BC
5385' C9       00345         RET
               00346 ;
               00347 ;------------------------------------------------------
               00348 ADJUST
               00349 ;       Enter with binary digit in A.  Adjust A to
               00350 ;       an ASCII Hex digit
               00351 ;
5386' C630     00352         ADD     A,30H
5388' FE3A     00353         CP      3AH
538A' FA8F53'  00354         JP      M,SKIP3
538D' C607     00355         ADD     A,7H
538F' C9       00356 SKIP3   RET
               00357 ;=====================================================
               00358 ;
5390' 1C       00359 BANNER  DB      1CH,1FH
      1F
5392' 48       00360         DB      'High Memory Map          Ver 3.0.1',LNFEED
      69 67 68 20 4D 65 6D 6F 72 79 20 4D 61 70 20 20
      20 20 20 20 20 20 56 65 72 20 33 2E 30 2E 31 0A
53B3' 2D       00361         DB      '---------------------------------',CRTN
      2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
      2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 0D
               00362 ;
53D4' 28       00363 L1A     DB      '(HIGH$)  =  ',ETX
      48 49 47 48 24 29 20 20 3D 20 20 03
53E1' 3C       00364 LXB     DB      '<----> ',ETX
      2D 2D 2D 2D 3E 20 03
               00365 ;------------------------------------------------------
               00366 ;
               00367 BCB
53E9' F853'    00368 BADD    DW      BUFFER
53EB' 40       00369 BLEN    DB      40H
53EC' 00       00370 BCUR    DB      0
53ED' 00       00371         DB      0
53EE' 0D       00372 BETX    DB      0DH
53EF' 2D       00373 BFILL   DB      '-'
53F0' 00       00374 T123    DB      0,0,0
      00 00
53F3' 00       00375 BTAB    DB      0
53F4' 10       00376         DB      10H
53F5' FF       00377         DB      0FFH
53F6' 0000     00378         DW      0
               00379 BUFFER
53F8'          00380         DS      50H
               00381 ;------------------------------------------------------
               00382 ;
5204'          00383         END     ENTRY
```

[redacted]

by Mel Patrick

[Reprinted from SMUG News, the newsletter of the Surrey (British Columbia, Canada) Microprocessors Users Group:]

If you do any amount of programming in BASIC you should be aware of the command which you can use while saving a BASIC program which will protect it, i.e. SAVE "filespec",P

What this command does is encode the program in a binary form and enable only the commands RUN, LOAD, MERGE, and CHAIN. Any attempt to EDIT or LIST the program won't work. Using the LIST command from DOS will not work either.

While there are those of us who would take Tandy's word at this, there are others amongst us who balk at not being privy to view something we aren't supposed to. It's kind of like saying, "Here is the safe, bet you can't open it!". Close but no cigar.

Indeed you can open the safe and should be allowed for one very good reason. If you save a version of the BASIC Program you are writing and forget to save an unprotected one, how are you ever going to edit your program again? Right, you can't. So that 23K BASIC program you just spent 3 1/2 months on is gone. Hardly fair, is it?

So for this very reason I will tell you how to unprotect those protected BASIC programs.

First you have to find out which version of BASIC you are using. Type BASIC from the TRSDOS READY mode and have a look at the version number. Use the technique outlined for your BASIC version.

a) BASIC 1.1.0 (earliest version) - from BASIC type in:
    SYSTEM "MEMORY (ADD=X'6247',WORD=X'0000')"
b) BASIC 1.1.1 (newest version) - from BASIC type in :
    SYSTEM "MEMORY (ADD=X'72CB',WORD=X'0000')"

Once you have used the proper patch, simply use the LIST and EDIT commands as you always have. They will work again normally.

SIMPLE MODEL 4/4D/4P CPU SPEEDUP

Information provided by David G. Huffman

If you would like to have a FAST Model 4, 4D or 4P and are an experienced hardware hacker, you can do the following: Open up the case, remove the Z-80 CPU chip from its socket, bend out pin 6 so that it sticks straight out, reinsert the Z-80 into its socket, run a piece of wire-wrap wire from the bent-out Z-80 pin 6 to pin 22 of the graphics connector (connector J7, a 34 pin header), solder both ends of the wire with a low wattage soldering iron, and put everything back together. This will give a CPU clock speed of 5.0688 MHz.

Now you might be saying "big deal" because the normal TRSDOS 6 clock speed is 4 MHz, BUT consider this - this modification does NOT affect the interrupt timing that controls the system clock. What this means is that if you boot up using a Model III DOS, the CPU still runs at 5 MHz but the interrupts are running at the "slow" speed, so the clock is okay. However, if you boot up TRSDOS 6, the CPU speed stays at 5 MHz (still a noticable improvement) but now the interrupt timing changes to the "fast" speed, so again the system clock works properly.

In other words, changing the "speed select" bit of port 84H now changes only the interrupt timing speed, not the CPU speed. A drawback to this mod is that there is no way to select a "normal" CPU speed, so if you have some Model III "shoot-em-up" type arcade games, you probably will not want to make this mod. Timing loops in other programs would be affected in a similar manner. Of course, you could always install a SPDT switch somewhere on the chassis to switch pin 6 of the Z-80 between pin 6 on the Z-80 socket (normal speed) and pin 22 of the graphics connector (high speed), or perhaps you could use some sort of port-controlled hardware switching arrangement.

One other note - when using this mod you may have to press the RESET switch twice when booting a disk. This is because the timing loops in the boot ROM are not quite long enough for the higher speed.

MODEL 4/4

13
Telephon

[EDITOR
topics. One
NEWDOS/80 c
memory (even
hardware mod
board. Althou
recommend t
making this
this modifica
experimental.
will NOT be r
a result of yo
How ab
2 Meg? Sou
chips, a roll
(Yeah, I know
speed up the
NEC RAMs a
$3.54 each, s
This mo
do not have
that you hav
the chip nu
modification
installed.
We will
leave two de
control port.
The wa
the alternate
change afte
which 64K g
the alternate
have five b
pages of 64
upper thre
modification
is replaced
Number 3, p
CPU is run
is to be m
modification
of Port 94H
compatibility
To be
must have a
you set 94H
addressing i
RAM bank
machine, yo
you are goi
could also
LD A,1 ins
might be p
automaticall
hacker coul
some code
this article)
To ge
and bend u
line and wi
the 4P, ho
shield so th
without cut
4P without
are careful
to stick a
where it m
part of t
Interferenc
Commission
sort of me
order to ke

MODEL 4/4D/4P RAMDISK PROGRAM PLUS [redacted]ARDS

by Dave Huffman
1345 Williams Street, Des Moines, Iowa 50317
Telephone: (515) 266-1759 (7-10 P.M. Central time only!)

[EDITOR'S NOTE: This article really covers two related topics. One is a RAMDISK program that can be used with NEWDOS/80 on any Model 4/4D/4P that has more than 64K of memory (even if it's just a stock 128K machine). The other is a hardware modification to the Model 4P with gate array type circuit board. Although this modification is not extremely complicated, we recommend that you attempt it only if you are experienced in making this type of hardware modification. We have not tested this modification, and because of this it should be considered experimental. The author of this article and/or NORTHERN BYTES will NOT be responsible for damages of any kind that may occur as a result of your attempting to build and use this modification.]

How about expanding your Model 4P to 512K RAM or even to 2 Meg? Sound expensive? Well, for the 4P we will need ten IC chips, a roll of wire wrap wire and tool, and 120ns 256K RAMs (Yeah, I know 150ns will work just as well, but... if you want to speed up the machine faster that 4 Mhz then get the 120ns). I use NEC RAMs and have had good luck. The last set I bought were $3.54 each, so a 512K system should cost you $75.

This modification is for the GATE ARRAY board ONLY. If you do not have a gate array board but have schematics for the board that you have, the logic that is used here will also apply, however, the chip numbers will change. Actually, the original memory modification was first done on my old Model 4, which has 1 Meg installed.

We will need to add a new port and it was nice of Tandy to leave two decoded address on U28. I used U28-10 (94H) as the RAM control port.

The way this modification works is simple. In the Model 4/4P the alternate 64K RAM is accessed using port 84H, and this will not change after the modification is installed. Port 94H will select which 64K group within the 2048K (maximum) RAM will be used as the alternate 64K RAM bank that is accessed by port 84H. So we have five bits in port 94H (D0-D4). This will give us one to 31 pages of 64K RAM that we can select for port 84H to access (the upper three bits of port 94H are used by other hardware modifications designed by the author, primarily when the Z-80 CPU is replaced by a Hitachi HD64180 [see NORTHERN BYTES Volume 7, Number 3, page 6]. Bit D5 is used for speed selection when the CPU is run at more than 4 MHz, and Bit D7 selects whether memory is to be mapped in 64K or 256K segments. These additional modifications are not discussed here, but the additional bit usage of Port 94H is mentioned here in the hope that possible software compatibility problems can be forestalled).

To be fully compatible with all Model 4 software, port 94H must have at least one of the lower five bits set. For example, if you set 94H to a 1 then this will enable the second group of 64K addressing in the first 256K RAM to be used for the alternate 64K RAM bank (32K high & 32K low). When you first power up the machine, you can go to BASIC and issue a POKE &H94,1 command if you are going to run a program that requires 128K of memory. You could also AUTO a short machine language program that contains a LD A,1 instruction followed by an OUT (94H),A instruction, or it might be possible to patch the boot sector of your DOS to automatically issue this command (the really ambitious hardware hacker could probably copy the boot ROM to an EPROM and add some code to flip a bit of port 94H, but that's beyond the scope of this article).

To get more than 512K RAM we will need to stack RAM chips and bend up pin 15 of the stacked chips. This is the CAS* control line and will be connected to the 74S08 chip in the drawings. In the 4P, however, you will need to cut away some of the metal shield so the RAMs will clear. You can get one meg in a Model 4 without cutting any metal. You may be able to get one meg in a 4P without any cutting, if you keep everything nice and tight and are careful that no pins touch the metal shield (it might not hurt to stick a strip or two of insulating tape on the shield at the point where it might otherwise contact the IC pins). If you do cut any part of the shield away, it may permit Radio Frequency Interference leakage in violation of Federal Communications Commission rules and regulations, so you should try to fashion some sort of metal box-like enclosure to fit over the cut-away part in order to keep the shielding effect intact.

I don't own a Model 4P (don't get worried!) so Jerry Johnson's 4P was the first guinea pig. Jerry says it nice to have 352 Grans (512k) for a system disk.

The installation for the 4P is as follows (refer to the diagrams):

1. Open up the machine and remove the main CPU board.
2. Mount the following IC chips on the board. We will piggyback them, and only solder the pins that we don't bend up:

#1 74LS32 - U105P - bend all pins except 7&14
#2 7486 - U24P - bend all pins except 7&14
#3 74S08 - U129P - bend all pins except 7&14
#4 74LS32 - U87P - bend all pins except 7&14
#5 74S08 - U114P - bend all pins except 7&14
#6 74157 - U111P - bend all pins except 8,15&16
#7 74157 - U110P - bend all pins except 1,8,15&16
#8 74LS244- U125P - bend all pins except 3,5,7,9,10,12,14,16,18,20
#10 7442 - U116P - bend all pins except 8,16
(#9 is not mounted until step 4!)

3. Connect the following:

U85-16 to U24P-2
U85-12 to U105P-9
U85-15 to U105P-10

4. Place the 74LS273 (#9) on U85 with all pins bent up except 1,3,4,7,8,10,13,14,17,18,20.
5. Make all connections shown in Figure 1 (only!) except U114P-3&6 and U129P-6 (also don't add any circuitry associated with over 512K of memory yet). Then power up the board in the machine and check for operation. At this point the only change noticed should be a new port 148 (94H). Set and reset the bits and read the port to verify operation. NOTE: If you have removed the drives from the case, the 4P may not boot because the drives are not properly terminated. In this case, try holding down the . (period) key while pressing RESET. This will run the built-in memory test of the boot ROM. If this runs correctly, everything is probably okay.
6. Then remove the Main CPU board and do the following:

a. Pull all RAMs.
b. Wire all pin 1 of the RAM sockets together and connect to U129P-6.

7. Remove R21.
8. RAS MOD - Lift one end of R23 that goes to U115-11 (see drawing). Add jumper from R23 on board side to pad on R21 opposite to U115 (see drawing). Connect lifted end of R23 to U114P-11. Connect remaining pads of R21 & R23 to U114P-12&13. MAKE SURE IT IS AS SHOWN IN THE DRAWING!!!!!!
9. CAS MOD - Find the feedthrough next to RP1 between pins 15 & 16. This should be the CAS line for the RAMs. MAKE SURE - it should connect to pin 15 of the RAMS (use digital ohmmeter only or LOOK). Turn board over to the non-component side (bottom) and follow the trace to another feedthrough about the same distance from the RAMs as the first feedthrough. Cut the trace that connect the two feedthroughs. The feedthroughs will now be the RAM CAS connections. U133-140 will be "CAS1" & U153-160 will be "CAS2". Now follow the trace from the first feedthrough on the bottom of the board to U136 & U156-15. Cut the trace that connects U136-15 & U156-15. Follow the trace to a feedthrough next to pin 16 of U136. Cut this trace near the feedthrough. Now we should have separated the two rows of RAMs on pin 15. Check with a digital ohmmeter that the two rows are separated. Make sure you understand which traces to cut!!!!
10. Now connect U114P-3 to a 47 Ohm resistor and place the other end in the feedthrough "CAS1". Connect U114P-6 to a 47 Ohm resistor and place the other end in the feedthrough "CAS2."
11. Refresh is found on pin 5 of U87. Connect it (you can run the wire through any available feedthrough) to U114P-2&4.
12. Replace the main board and power up. If you get trash on the screen, check the wiring - you GOOFED. If it runs but is unreliable, you will need a good scope and will have to look at the RAS line. Check for ringing. Add resistance to the series resistor if you have ringing. If the transitions are a good 4.75V then reduce the resistance. Check the CAS line also - 47 Ohms may be too much. On my Model 4 I have no resistors in the RAS or CAS.

Now, to add more RAM you will have to piggyback one or two more 74S08 IC's as U96P and/or U117P (bend up all pins except 7 and 14 before piggybacking). One input of each of the AND gates (example: pins 1,4,9,12) must be connected to the refresh line (pin 5 of U87). The other input for each of the AND gates (example: pins 2,5,10,13) should be connected to one of the CASnA lines (3 through 8) at U116P. You must then stack the added RAM and bend pin 15 out and connect all pin 15's per row of RAMs to a 47 Ohm resistor and to one of the CAS lines (3 through 8) at the output of one of the added AND gates (example: pins 3,6,8,11). See the detail in the lower left corner of Figure 1. Walla - 2 Meg. I recommend you run 512K for a month before you stack RAMs.

I have modifications for the Model 4 and have one Meg in mine, but the design incorporates the use of a HD64180. This makes the modification a little complicated. The Model 4 board version that I have is one of the older non-Green Screen type and I had to redo the RAS/MUX/CAS circuits in order to run the one Meg RAMs. However, if there is some interest for the Model 4's I will send in the modifications I have done.

RAMDISK PROGRAM

If you use NEWDOS/80 then I have a RAMDISK program that will add a 5th drive and use the memory we just added. The RAMDISK uses the fourth PDRIVE, so if you have 512K RAM then set PDRIVE #4 as follows:

```
PDRIVE 0 4=0
PDRIVE 0 4 tc=99
```

Then type:

```
RAMDISK 4    'I renamed it to RM so it would be " RM 4 "
```

If the NEWDOS/80 4P system has two directories then we need two steps (I calculate the directory starting and ending points so I can return Directory Protected Status):

```
COPY 0 4,,FMT CBF NDMW /SYS
COPY 0 4,,NFMT CBF USR NDMW
```

If, on the other hand, you made the MODELA/III a SYS file then a full disk copy will work:

```
COPY 0 4,,FMT CBF NDMW
```

Then, to mount as drive 0:

```
RAMDISK 0
```

That's it (yes, Jack, we can CHAIN through this version).

The RAMDISK program will support a standard Model 4 or 4P with just 128K if you set the switches (at the start of the assembly language program) accordingly.

[Editor's note: The assembly language listing shown below does not include the machine language hexadecimal byte output field because this will vary according to how the switches are set at the start of the program. The code is written to be assembled using the ZEUS editor-assembler, but it should be possible to use other editor-assembler programs if the proper changes in syntax are made. For those who do not wish to type in this listing, both source and object code files should appear on a TAS Public Domain Library disk sometime in the near future.]

SOURCE CODE FOR MODEL [illegible] RAMDISK PROGRAM

```
; RAMDV362/ASM - Ram Disk Driver
;       LNW Team - HD64180ROP CPU With 256K Ram
;       Model 4/4P  - Z80 CPU 128K Ram
;
;       UPDATED 9/15/86
; ADDED Support for 2048K Model 4
;
        ORG     5200H
;*****************************************************
;               Type Flags
LNW     EQU     0       ; 1=LNW Computer
;                         0=Model 4 or 4P Computer
MOD4    EQU     LNW-1
;
HD64180 EQU     0       ; 1=HD64180 Processor 256K Ram
;                          MMU will be Used
;                         0=Z80 Cpu With 128K Ram Banked
;                          for Low address 0-8000H
;                          Model 4 Supported only
Z80     EQU     HD64180-1
MUXMEM  EQU     0       ; Added MUX Memory - Port 148
;                         1=Yes 0=No
;                         Memory Support to 2048k
```

Model 4P Memory Modification for Gate Array Boards

Figure 1

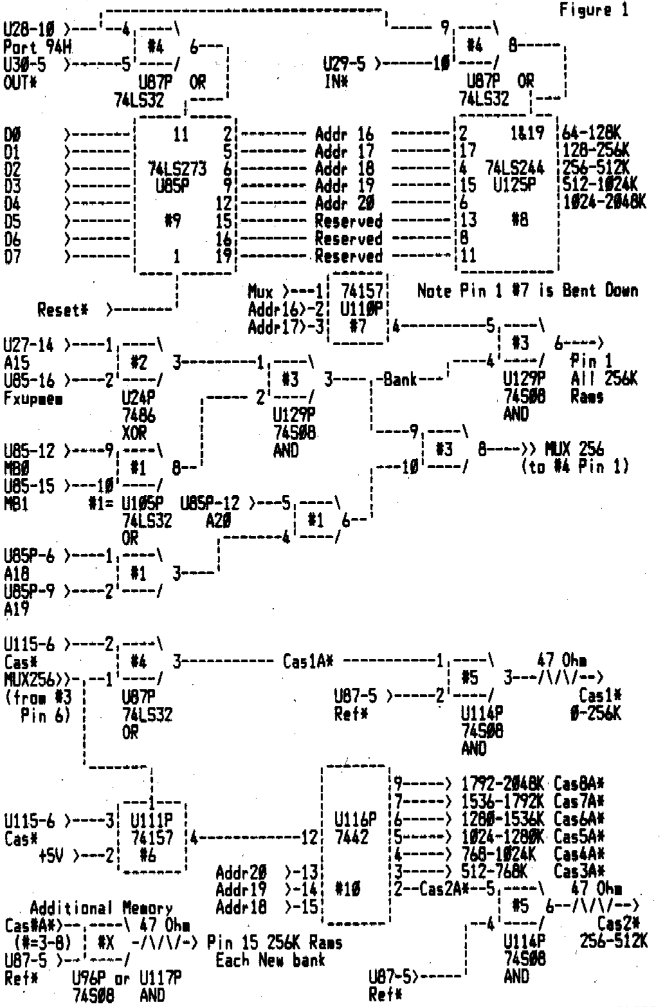


Figure 2

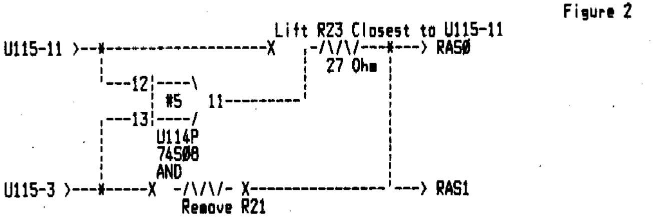


Table of I.C. Designations

1	U105P	-	74LS32
2	U24P	-	7486
3	U129P	-	74S08
4	U87P	-	74LS32
5	U114P	-	74S08
6	U111P	-	74S157 or 74157
7	U110P	-	74S157 or 74157
8	U125P	-	74LS244
9	U85P	-	74LS273
10	U116P	-	7442

X	U96P }
	and/or U117P}
		-	74S08 (1 or 2
			if required)

```
;
DISP    EQU     0       ; Display Patch # when called
;                         Used for Debuging Program
;*****************************************************
;               [illegible] Disk Driver
;       For NewDos/80 Version 2.0
;
;   This driver patches into the disk Sector Read/Write &
;drive selects. An additional patch checks for SYS6/SYS
;to be present. If SYS6 is present and we are Formatting,
;Copying then a patch is added to SYS6/SYS to bypass the
;format routine and just write BOOT/SYS & DIR/SYS Sectors
;
;   The Ram disk can be installed in drive Slots 0-4. The
;pdrive for the Ram Disk will ALWAYS be #4. If we select
;Slot 0-3 then the physical drive will select when Slot 4
;is requested
;  THIS MEANS THAT WE WILL HAVE 5 DRIVES POSSIBLE ONLINE
;
;  This Version for the HD64180 moves 1 byte at a time.
;Next update will include a buffer area for moving 256
;bytes at a time as an option. The Model 4/4P  use Hi
;memory and I HATE to lose memory unless the benifit
;is great. The HD64180 would be most efficent using the
;DMA controller however, I didn't want to lose possible
;compatability for "Ramdisk boards".
; Thus each sector is read to and from the buffer 1 byte
;at a time. That is for each byte read/written a memory
;map switch must take place. Ramdisk I/O speed can be
;dramatically increased at a cost of 256 bytes of himem
;on the model 4/4P ( A full Buffer option will be added
;later as a option )
;
;  The Routine CALPAGE does the Page Calculating so any
;combination of port decoded ram & MMU ram controlled by
;the HD64180 can be used.
;
;   Several patch areas were necessary because of the way
;SYS6/SYS operates. In all patches a test for the current
;drive is made. If it is the Ramdisk and SYS6 is resident
;and we have not patched SYS6 yet, then we patch in some
;bytes for FORMAT. Patch 1,2 are basically to detect if
;the Ramdisk is to be POWERED UP and or SELECTED. Patch
;3 is the actual Sector Read/Write. Patch 4 & 5 is for
;swapped drives. If we install the Ramdisk at drive 1 then
;the physical drive 1 becomes drive 4 so we don't have to
;give up a drive slot. Patch 6 is mainly for SYS6/SYS in
;case we got an error condition and exit Via 4409H. The
;purpose is to reset the Patch Flag (SYS6FLG). Once we
;patch SYS6/SYS we don't want any drive selects to
;repatch SYS6/SYS because SYS6/SYS calls other SYS
;modules that overlay SYS6 in the normal overlay area.
;
;                  Operators Instructions
;
; Pdrive #4 will Always be the Defination for the Ramdisk
; There is a check for the Maximun # of sectors so you
;can't write over the Rom image in the Model 4/4P
;
; To install type " RamDisk Dn " where dn = Drive #
; Once installed the Ramdisk will operate like the Floppy
;disk drives. You Must format it to use it.
; Format dn,,,,ndmw
; To remount the ramdisk as another drive at any time
;type Ramdisk dn
;
; Legal slots for the Ramdisk = 1to4
;
; When the Ramdisk is mounted over an existing drive the
;physical drive will become Drive #4
;
; If you want to change the Pdrive for Slot # 4 then
;you MUST reinstall the Ramdisk.
;
;*******************************************************
;                Equates for Each Type Machine
;
         IF      LNW ; Team Model 1 Compatable
;*******************************************************
;                LNW TEAM - With HD64180/Z80H
;*******************************************************
PATAD0  EQU     47E0H   ; Patch address for Patch 1
;
PATAD1  EQU     4776H   ; Patch address for Patch 1
PATAD1A EQU     PATAD1+1
;
PATAD2  EQU     47ADH   ; Patch address for Patch 2
PATAD2A EQU     PATAD2+1
;
PATAD3  EQU     465CH   ; Patch address for Patch 3
;
PATAD4  EQU     47A3H   ; Patch address for Patch 4
PATAD4A EQU     PATAD4+1
;
PATAD5  EQU     4798H   ; Patch address for Patch 5
PATAD5A EQU     PATAD5+1
;
PATAD6  EQU     4409H   ; patch address for Patch 6
PATAD6A EQU     PATAD6+1
;
SYAL    EQU     477AH   ; System AL Option - Max Drives
SYAL1   EQU     439FH   ; System - Max # Drive searched
;
CURDRV  EQU     4308H   ; Current drive selected
CURDCMD EQU     4309H   ; Current FDC Select Bits 1-2-4-8
;
CONFLG1 EQU     4369H   ; Condition Flags - Debug/Overlay
;                               Ect.
SYS6ID  EQU     4D09H   ; Byte ID to Check if SYS6
;                         Overlayed its self.
;
SY6BF   EQU     6886H   ; System 6 Bypass format Patch
;
PAT0RCM EQU     4776H   ; Patch 0 End Replace command
PAT1END EQU     4779H   ; Patch 1 Replaced Command
PAT2END EQU     47DDH   ; Patch 2 End Jump address
PAT3END EQU     4773H   ; Patch 3 End Jump address
PAT3CON EQU     471DH   ; Patch 3 Continue address
PAT6END EQU     440CH   ; Patch 6 End Jump address
;
P6REPCM EQU     4C1EH   ; Patch 6 Replaced cmd Address
DIRLMP  EQU     430AH   ; Directory Starting Lump for
;                         Current Drive
DIRGRN  EQU     430FH   ; Grans/Lump Value for Current
;                         Drive
PDRTBL  EQU     4371H   ; Address of Pdrive Table Drive 1
;
PDRADO  EQU     4399H   ; Address of current Pdrive
;
CURFDC  EQU     46C4H   ; Current FDC cmd
LDRV0TK EQU     4300H   ; Last track Accessed by Drive 0
LDRV4TK EQU     4304H   ; Last track Accessed by Drive 4
;
;*******************************************************
         ENIF
;
         IF      MOD4
;*******************************************************
;                Model 4 - 128K Ram
;*******************************************************
PATAD0  EQU     4792H   ; Patch address for Patch 1
;
PATAD1  EQU     4723H   ; Patch address for Patch 1
PATAD1A EQU     PATAD1+1
;
PATAD2  EQU     475EH   ; Patch address for Patch 2
PATAD2A EQU     PATAD2+1
;
PATAD3  EQU     4607H   ; Patch address for Patch 3
;
PATAD4  EQU     4754H   ; Patch address for Patch 4
PATAD4A EQU     PATAD4+1
;
PATAD5  EQU     4749H   ; Patch address for Patch 5
PATAD5A EQU     PATAD5+1
;
PATAD6  EQU     4409H   ; patch address for Patch 6
PATAD6A EQU     PATAD6+1
;
SYAL    EQU     4727H   ; System AL Option - Max Drives
SYAL1   EQU     42BFH   ; System - Max # Drive searched
;
CURDRV  EQU     427EH   ; Current drive selected
CURDCMD EQU     427FH   ; Current FDC Select Bits 1-2-4-8
;
CONFLG1 EQU     4289H   ; Condition Flags - Debug/Overlay
;                               Ect.
SYS6ID  EQU     4D09H   ; Byte ID to Check if SYS6
;                         Overlayed its self.
;
```

```
00194 SY68F   EQU     6888H   ; System 6 Bypass format Patch
00195 ;
00196 PAT0RCM EQU     4723H   ; Patch 0 End Replace command
00197 PAT1END EQU     4726H   ; Patch 1 Replaced Command
00198 PAT2END EQU     4783H   ; Patch 2 End Jump address
00199 PAT3END EQU     4720H   ; Patch 3 End Jump address
00200 PAT3CON EQU     46CDH   ; Patch 3 Continue address
00201 PAT6END EQU     440CH   ; Patch 6 End Jump address
00202 ;
00203 P6REPCM EQU     4289H   ; Patch 6 Replaced cmd Address
00204 ;
00205 DIRLMP  EQU     4280H   ; Directory Starting Lump for
00206 ;                         Current Drive
00207 DIRGRN  EQU     4285H   ; Grans/Lump Value for Current
00208 ;                         Drive
00209 PDRTBL  EQU     4291H   ; Address of Pdrive Table Drive 1
00210 ;
00211 PDRADD  EQU     4289H   ; Address of current Pdrive
00212 ;
00213 CURFDC  EQU     4687H   ; Current FDC cmd
00214 LDRV0TK EQU     4276H   ; Last track Accessed by Drive 0
00215 LDRV4TK EQU     4279H   ; Last track Accessed by Drive 4
00216 ;
00217 LDDE    EQU     5BEDH   ; LD DE,(XX) - OPCODE
00218 LDIX    EQU     2ADDH   ; LD IX,(XX) - OPCODE
00219 STHL    EQU     22H     ; LD (XX),HL - OPCODE
00220 ;
00221 HIMEM   EQU     4411H   ; HIMEM FOR MODIII/IV
00222 PORTMSK EQU     4CFFH   ; PORT MASK of Port 84H
00223 ;
00224 ;******************************************************
00225         ENIF
00226 ;******************************************************
00227         IF      HD64180
00228 ;       Find Type of Processor We are Running
00229 ;               HD64180/Z80H Cpu
00230 ;
00231 ;
00232 HDSTART DEFB    0EDH,38H,18H ; HD64180 INO (18H),A
00233         DEFB    3H           ; IF HD64180 THEN INC BC
00234         JP      START   ; HD64180 SYSTEM
00235 ;
00236         JP      Z80CPU  ; Z80 SYSTEM
00237 ;
00238 ;
00239 ;
00240 Z80CPU  LD      HL,MTYPEER
00241         CALL    4467H   ; DISPLAY ERROR
00242         JP      402DH
00243 ;
00244 MTYPEER DEFM    '***** This MUST be a HD64180ROP Processor ****',0DH
00245 ;
00246         ENIF
00247 ;
00248 ;*********************************************************
00249         DEFM    'Written By D. Huffman ',0DH
00250 ;*********************************************************
00251 ;       Get Drive # for Ramdisk
00252 ;
00253 START   LD      A,(HL)  ; GET Command
00254         CP      0DH
00255         JR      Z,CMDER         ; ERROR
00256 GETCMD1 LD      A,(HL)
00257         INC     HL
00258         CP      ' '
00259         JR      Z,GETCMD1
00260         CP      ','
00261         JR      Z,GETCMD1
00262         CP      ':'
00263         JR      Z,GETCMD1
00264 ; Ah Finaly
00265         SUB     30H     ; CONV TO BIN
00266         CP      5       ; Drive 0-4
00267         JR      C,GETCMD2       ; GO IF OK
00268 CMDER   LD      HL,MESER
00269         CALL    MESOUT
00270         JP      402DH
00271 ;
00272 NOMEMER LD      HL,MESMER
00273         CALL    MESOUT
00274         JP      402DH
00275 ;
00276 ;
00277 GETCMD2 LD      (DRIVE),A       ; SET IN DRIVE VALUES
00278 ; JUST for My Sanity We will Attempt to set 148
00279         IN      A,(148) ; Get Port if exists
00280         SET     0,A
00281         OUT     (148),A
00282 ;
00283 ;
00284         IF      MOD4
00285         XOR     A               ; Zero a
00286         LD      (PORTMSK),A     ; Setup Port Mask
00287 ;                                 to zero
00288 ;
00289         IF      MUXMEM
00290         IN      A,(148)         ; Check for more memory
00291 ;                                 Mux is Installed ?
00292         CP      255             ; Assume No Memory
00293         JR      Z,NOMEMER; NOP
00294         ENIF
00295 ; Setup and relocate to top of himem
00296         LD      IX,(HIMEM)      ; GET HIMEM
00297 ; BUT FIRST Check if we are already installed...
00298         LD      A,(IX+4)
00299         CP      0CDH            ; Check for CALL 4723H
00300         JR      NZ,GETLA        ; NOP USE NEW ADDRESS
00301         LD      A,(IX+5)
00302         CP      23H             ; ?
00303         JR      NZ,GETLA
00304         LD      A,(IX+6)
00305         CP      47H             ; ? 4723H
00306         JR      NZ,GETLA
00307 ; OK IS US.....
00308         LD      L,(IX+2)
00309         LD      H,(IX+3)        ; Old Himem value
00310         JR      GETLB
00311 GETLA   LD      HL,(HIMEM)
00312 GETLB   LD      DE,PGMEN-PGM    ; PROGRAM LENGTH
00313         LD      (HIMEM1),HL     ; Save for Next boot
00314         AND     A
00315         SBC     HL,DE           ; HL=NEW START OF PGM
00316         LD      (NEWLOC),HL     ; SAVE IT
00317         DEC     HL
00318         DEC     HL              ; SUB 2
00319         LD      (HIMEM),HL      ; SAVE HIMEM
00320         INC     HL
00321         INC     HL              ; PUT IT BACK
00322         LD      DE,PGM
00323         AND     A
00324         SBC     HL,DE           ; HL = OFFSET
00325         LD      (OFFSET),HL     ; SAVE IT
00326         LD      A,(RELNM)       ; GET # IN TABLE
00327         LD      B,A
00328         LD      IX,RELTBL       ; GET TABLE
00329 GETLP1  PUSH    IX              ; SAVE IT
00330         LD      (IXBYTE),IX
00331         DEFW    LDIX            ; LD IX,(XX) - OPCODE
00332 IXBYTE  DEFS    2
00333         LD      (LOCBYTE),IX    ; SAVE
00334         LD      (LOCBYT1),IX
00335         DEFW    LDDE            ; LD DE,(LOCBYTE)
00336 LOCBYTE DEFS    2
00337         LD      HL,(OFFSET)
00338         ADD     HL,DE           ; ADD OFFSET
00339         DEFB    STHL            ; LD (LOCBYT1),HL
00340 LOCBYT1 DEFS    2
00341         POP     IX              ; GET IT BACK
00342         INC     IX
00343         INC     IX              ; BUMP +2
00344         DJNZ    GETLP1          ; DO LOOP
00345 ;
00346         JP      GETLPEN         ; CONTINUE
00347 ;
00348 ;---------------------------------------------------------
00349 ;                       Relocation Table
00350 ;
00351         IF      MUXMEM
00352 RELNM   DEFB    36      ; # OF ENTERYS IN TABLE
00353         ENIF
```

```
00354          IF      NOT,MUXMEM
00355 RELNM    DEFB    34
00356          ENIF
00357 ;
00358 RELTBL   DEFW    ADDR1+1             ; Relocation address
00359          DEFW    ADDR2+1
00360          DEFW    ADDR3+1
00361          DEFW    ADDR4+1
00362          DEFW    ADDR5+1
00363          DEFW    ADDR6+1
00364          DEFW    ADDR7+1
00365          DEFW    ADDR8+1
00366          DEFW    ADDR9+1
00367          DEFW    ADDR10+1
00368          DEFW    ADDR11+1
00369          DEFW    ADDR12+1
00370          DEFW    ADDR13+1
00371          DEFW    ADDR14+1
00372          DEFW    ADDR15+1
00373          DEFW    ADDR16+1
00374          DEFW    ADDR17+1
00375          DEFW    ADDR18+1
00376          DEFW    ADDR19+1
00377          DEFW    ADDR20+1
00378          DEFW    ADDR21+1
00379          DEFW    ADDR22+1
00380          DEFW    ADDR23+1
00381          DEFW    ADDR24+1
00382          DEFW    ADDR25+1
00383          DEFW    ADDR26+1
00384          DEFW    ADDR27+1
00385          DEFW    ADDR28+1
00386          DEFW    ADDR29+1
00387          DEFW    ADDR30+1
00388          DEFW    ADDR31+1
00389          DEFW    ADDR32+1
00390          IF      MUXMEM
00391          DEFW    ADDR33+1
00392          DEFW    ADDR34+1
00393          ENIF
00394 ;
00395          DEFW    SEC3+1
00396          DEFW    PATCH2A+1
00397 ;
00398 ;
00399 GETLPEN NOP                          ; CONTINUE HERE
00400          ENIF
00401 ;
00402 ; Get Fourth Pdrive for New Drive #4
00403 ; This will be the prive for the Ram Driver no matter
00404 ;where we install it.
00405 ;
00406          LD      HL,BOOTSYS         ; GET BOOT/SYS Sector
00407          LD      DE,FCB
00408          CALL    441CH              ; Extract File spec
00409          JP      NZ,BOOTER          ; Got error
00410          LD      DE,FCB
00411          LD      HL,BUFFER
00412          LD      B,0                ; 256 Byte sectors
00413          CALL    4424H              ; Open Boot/sys
00414          JP      NZ,BOOTER
00415          CALL    4436H.             ; Sector 1
00416          JP      NZ,BOOTER
00417          CALL    4436H              ;        2
00418          JP      NZ,BOOTER
00419          CALL    4436H              ;        3
00420          JP      NZ,BOOTER
00421          CALL    4428H              ; Close it
00422          JP      NZ,BOOTER
00423 ; Now get Fourth Pdrive
00424          DI                 ; For safety
00425 ;
00426          IF      MOD4
00427 ;********************************************************
00428 ;***** Ok now Move New Copy of Ramdisk to Correct Loc **
00429          LD      HL,PGM             ; START
00430          LD      DE,(NEWLOC)        ; NEW ADDRESS LOCATION
00431          LD      BC,PGMEN-PGM
00432          LDIR                       ; MOVE IT
00433          ENIF
```

```
00434 ;
00435          LD      HL,BUFFER+40H   ; Start of buffer
00436 ADDR1    LD      DE,PDRIVE
00437          LD      BC,16
00438          LDIR                    ; MOVE IT
00439 ;
00440          IF      Z80     ; MODEL 4 WITH Z80
00441          IF      MOD4
00442          IF      MUXMEM  ; MUX MEMORY ADDED
00443          IN      A,(148)         ; Check for more memory
00444 ;                                  Mux is Installed ?
00445          CP      255             ; Assume No Memory
00446          JP      Z,NOMEMER; NOP
00447 ; AH We do have more memory
00448          RES     7,A     ; Enable Z80 Port Addressing
00449          SET     0,A     ; Enable First bank of Mux
00450          OUT     (148),A
00451          LD      HL,7936         ; 2048K  2 MEG
00452          LD      HL,MESS4M       ; Message for disp
00453          LD      (MESS44P),HL    ; SAVE FOR DISPLAY
00454 ;                                Maximum # of sectors .
00455 ;                                 For Safety in case the
00456 ;                                 Pdrive is set wrong
00457          JR      ADDR2.
00458          ENIF
00459 ; Now Check if Model 4 or 4P
00460 NOMORE   LD      A,(0)
00461          LD      B,A     ; SAVE THE BYTE
00462          LD      A,1
00463          OUT     (84H),A
00464          LD      (0),A
00465          XOR     A
00466          OUT     (84H),A
00467          LD      A,(0)   ; GET THE BYTE
00468          CP      1
00469          JR      Z,MODEL4P ; IS A 4P
00470 ;
00471 MODEL4   LD      HL,MESS4
00472          LD      (MESS44P),HL    ; SAVE FOR DISPLAY
00473          LD      HL,320          ; # OF SECTORS MAX 4
00474          JR      ADDR2           ; GO CHECK IF MORE MEM
00475 ;
00476 MODEL4P  LD      A,1
00477          OUT     (84H),A ; BANK IN RAM
00478          LD      A,B     ; GET BYTE
00479          LD      (0),A   ; PUT IT BACK
00480          XOR     A
00481          OUT     (84H),A ; BANK IN ROM/WRITE PROTECT
00482 ;
00483          LD      HL,MESS4P
00484          LD      (MESS44P),HL    ; SAVE FOR DISPLAY
00485          LD      HL,256          ; MAX # OF SECTORS 4
00486          ENIF
00487          ENIF
00488          IF      HD64180
00489          LD      HL,768          ; MAX # OF SECTORS
00490          ENIF
00491 ;
00492          IF      LNW
00493          LD      HL,768          ; MAX # OF SECTORS
00494          ENIF
00495 ;
00496 ADDR2    LD      (SECTOR+1),HL   ; SAVE IT
00497 ;
00498          IF      HD64180
00499 ;********************************************************
00500 ;                HD64180 MMU
00501 ;        Config MMU for 32K Boundrys 0000H-FFFFH
00502 ;
00503          LD      A,128            ; CBAR VALUE
00504          DEFB    0EDH    ; OPCODE  OUTO (m),g
00505          DEFB    39H
00506          DEFB    3AH     ; PORT 3AH CBAR OF MMU
00507 ;
00508 ;********************************************************
00509          ENIF
00510 ;patch Ramdrive into DOS
00511 PATCH    LD      A,(DRIVE)       ; Get Drive # for Patch
00512 ADDR3    LD      (PAT0+1),A      ; SET IN DRIVE VALUES
00513 ADDR4    LD      (PAT1+1),A
```

```
00514 ADDR5   LD      (PAT2+1),A
00515 ADDR6   LD      (PAT3+1),A
00516 ADDR7   LD      (PAT4+1),A
00517 ;
00518 ADDR8   LD      HL,PATCH0
00519         LD      (PATAD0),HL
00520 ;
00521 ADDR9   LD      HL,PATCH1       ; FIRST PATCH
00522         LD      A,0C3H          ; JP
00523         LD      (PATAD1),A
00524         LD      (PATAD1A),HL
00525 ;
00526 ADDR10  LD      HL,PATCH2
00527         LD      A,0CDH          ; CALL BYTE
00528         LD      (PATAD2),A
00529         LD      (PATAD2A),HL
00530 ;
00531 ADDR11  LD      HL,PATCH3
00532         LD      (PATAD3),HL
00533 ;
00534 ; Is Drive 4 - Now Patch Pdrive Check routine
00535 ;
00536         LD      A,0CDH          ; CALL
00537         LD      (PATAD4),A
00538 ADDR12  LD      HL,PATCH4
00539         LD      (PATAD4A),HL    ; CALL PATCH4
00540 ;
00541         LD      A,0CDH
00542         LD      (PATAD5),A
00543 ADDR13  LD      HL,PATCH5
00544         LD      (PATAD5A),HL
00545 ;
00546         LD      A,0C3H          ; Patch 6 for SYS6TES
00547         LD      (PATAD6),A
00548 ADDR31  LD      HL,PATCH6
00549         LD      (PATAD6A),HL
00550 ;
00551         LD      A,5             ; 4 DRIVES
00552         LD      (SYAL),A
00553         LD      (SYAL1),A
00554 ;
00555         EI
00556 ;
00557 ; Check if Ramdisk will be drive 0 ?
00558         LD      A,(DRIVE)       ; get current drive
00559         CP      0               ; Are we setting up Dr 0
00560         JR      NZ,PATCHDN      ; Nop
00561 ; Yes it is .. So we need to copy the Pdrive for
00562 ; the Physical drive 0 to the Ram Disk
00563 ;
00564 ; BUFFER Still has Physical Drive 0 Rel Sector 3
00565 ;
00566 ; Open Boot/sys on Ramdisk
00567         LD      HL,BOOTSYS      ; GET BOOT/SYS Sector
00568         LD      DE,FCB
00569         CALL    441CH           ; Extract File spec
00570         JP      NZ,BOOTER       ; Got error
00571         LD      DE,FCB
00572         LD      HL,BUFFER0      ; NEW BUFFER
00573         LD      B,0             ; 256 Byte sectors
00574         CALL    4424H           ; Open Boot/sys
00575         JR      NZ,BOOTER
00576         CALL    4436H           ; Sector 1
00577         JR      NZ,BOOTER
00578         CALL    4436H           ;       2
00579         JR      NZ,BOOTER
00580         CALL    4436H           ;       3
00581         JR      NZ,BOOTER
00582 ; Ok now copy over old Pdrive to Ramdisk
00583         LD      HL,BUFFER       ; Source
00584         LD      DE,BUFFER0      ; Dest
00585         LD      BC,16           ; 16 Bytes
00586         LDIR                    ; Move them
00587 ;
00588         LD      DE,FCB
00589         CALL    4445H           ; Backup 1
00590         JR      NZ,BOOTER       ; Error recovery
00591 ;
00592         LD      HL,BUFFER0
00593         CALL    4439H           ; Write the Sector
```

```
00594         JR      NZ,BOOTER
00595 ;
00596         CALL    4428H           ; Close it
00597         JR      NZ,BOOTER
00598 ;
00599 PATCHDN LD      A,(LDRV0TK)     ; Get Last Track Accessed
00600         LD      (LDRV4TK),A     ; Move to Drive # 4 Slot
00601 ;                                 This is so Dos can find
00602 ;                                 the correct track the
00603 ;                                  first time
00604 ;
00605         LD      HL,MESS1        ; MESSAGE INIT
00606         CALL    MESOUT
00607         IF      Z80
00608         LD      HL,(MESS44P)    ; GET CORRECT MESSAGE
00609         CALL    MESOUT
00610         ENIF
00611         IF      HD64180
00612         LD      HL,MESSHD
00613         CALL    MESOUT
00614         ENIF
00615         LD      HL,MESS2
00616         CALL    MESOUT
00617         JP      402DH           ; Done
00618 ;
00619 BOOTER  JP      4409H           ; ERROR EXIT
00620 ;
00621 NEWLOC  DEFS    2               ; NEW ADDRESS FOR RAMDRV
00622 OFFSET  DEFS    2               ; OFFSET STORAGE
00623 DRIVE   DEFS    2               ; Patch Drive #
00624 FCB     DEFS    50              ; FCB BUFFER
00625 BUFFER  DEFS    256             ; DOS BUFFER - PDRIVE
00626 BOOTSYS DEFM    'BOOT/SYS',0DH
00627 ;
00628 BUFFER0 DEFS    256             ; DOS BUFFER - PDRIVE
00629 ;
00630 MESS44P DEFS    2               ; Model 4/4P Display Mess
00631 ;
00632 MESS1   DEFM    '**********************************************',0DH
00633         DEFM    '*        Ramdisk Version 3.62 Installed      *',0DH
00634         DEFM    '*              Written By D. Huffman         *',0DH
00635         IF      MOD4
00636         DEFM    '*  Bit Mask for Port 84H is at Address 4CFFH  *',0DH
00637         ENIF
00638         DEFB    0       ; END OF MESSAGE SEGMENT
00639 ;
00640 MESS4   DEFM    '*        Model 4 - 320 Sectors Maximum       *',0DH,0
00641 ;
00642 MESS4P  DEFM    '*        Model 4P - 256 Sectors Maximum      *',0DH,0
00643 MESS4M  DEFM    '* Model4/4P-Mux Memory Added-7936 Sectors Max *',0DH,0
00644 MESSHD  DEFM    '*                768 Sectors Maximum          *',0DH,0
00645 MESS2   DEFM    '**********************************************',0DH
00646         DEFB    0DH,0
00647 ;
00648 MESER   DEFM
'*********************************************************',0DH
00649         DEFM    '*                    Illegal Drive Specifyied
*',0DH
00650         DEFM    '*  Parameters  --->  RAMDISK (Dn)
*',0DH
00651         DEFM    '*  Example     --->  RAMDISK 4
*',0Dh
00652         DEFM    '*     The RamDisk uses Pdrive #4 in the Pdrive table
*',0DH
00653         DEFM
'**********************************************************'
00654         DEFB    0DH,0
00655 ;
00656 MESMER  DEFM    '*************PORT 148 (94H) RETURNS 255 (FFH)
**************',0DH
00657         DEFM    ' I Must Assume that the Memory Expansion Doesnt exist
',0DH
00658         DEFM    '                 I Cannot Proceed ',0DH,0
00659 MESOUT  LD      A,(HL)  ; GET BYTE
00660         CP      0
00661         RET     Z       ; DONE
00662         CALL    033H
00663         INC     HL
00664         JR      MESOUT
00665 ;
```

```
00666 ;*****************************************************
00667         IF      LNW     ; Lnw has Ram from 3000H-37DFH
00668 ;                       ; But the Illegal Emulator code
00669 ;                       ; Starts at 3500H
00670         ORG     3000H
00671         ENIF
00672         IF      MOD4
00673         ORG     8000H
00674         ENIF
00675 ;
00676 PGM     EQU     $       ;****** START OF RAMDISK
00677 ;*****************************************************
00678 ;************* Do Not Change the next 6 Bytes ***********
00679 HIMEM1  DEFS    2       ; OLD HIMEM LOCATION
00680 ;*****************************************************
00681 ;               47ECH           4792H
00682 ; * Bypass Drive Select & Test routine IF Ram Disk
00683 ;  This Routine is called by SYS6/SYS Directly
00684 PATCH0  CALL    PAT0RCM
00685         RET     NZ
00686         IF      DISP
00687         LD      A,'0'
00688         LD      (3FF7H),A       ; Display Patch #
00689         ENIF
00690         LD      A,(CURDRV)      ; GET CURRENT DRIVE
00691 PAT0    DEFB    0FEH            ; CHECK FOR MEMDRIVE
00692         DEFS    1               ; CP DN#
00693         JR      NZ,RET0         ;jmp if not Ramdrive
00694 ; Now check if system 6 is active - We may be Formatting
00695 ADDR14  CALL    SYS6TES         ; If system 6 then patch
00696 ;
00697         POP     AF              ;pop call off of stack
00698 RET0    XOR     A
00699         RET
00700 ;*****************************************************
00701 ;               4776H           4723H
00702 ; * Bypass Drive Select If Ram Disk - Normal Call
00703 PATCH1  PUSH    AF              ; SAVE IT
00704         LD      A,(CURDRV)      ; GET CURRENT DRIVE
00705 PAT1    DEFB    0FEH            ; CHECK FOR MEMDRIVE
00706         DEFS    1               ; CP DN#
00707         JR      NZ,RET1         ;jmp if not Ramdrive
00708         IF      DISP
00709         LD      A,'1'
00710         LD      (3FF8H),A       ; Display Patch #
00711         ENIF
00712 ; Now check if system 6 is active
00713 ADDR15  CALL    SYS6TES         ; If system 6 then patch
00714 ;
00715 RET1    POP     AF              ;pop AF back ...
00716         PUSH    HL              ; Replace cmd
00717         PUSH    DE
00718         PUSH    BC
00719         JP      PAT1END         ; GO we replaced the cmds
00720 ;*****************************************************
00721 ;       47ADH           475EH
00722 ;* Bypass Controller Init for the Read/Write IF Ram Disk
00723 PATCH2  LD      A,(CURDRV)      ; GET CURRENT DRIVE
00724 PAT2    DEFB    0FEH            ; CHECK FOR MEMDRIVE
00725         DEFS    1               ; CP DN#
00726 ;
00727         IF      LNW
00728         LD      HL,37ECH        ; Replaced Cmd
00729         ENIF
00730 ;
00731         IF      MOD4
00732         LD      A,(4286H)       ; Replaced Cmd
00733         ENIF
00734 ;
00735         RET     NZ      ;ret if no Ramdrive
00736 ;
00737         IF      DISP
00738         LD      A,'2'
00739         LD      (3FF9H),A       ; Display Patch #
00740         ENIF
00741         IF      MOD4
00742         IN      A,(0F0H)        ; Get current FDC state
00743         BIT     7,A             ; Is it active
00744         JR      NZ,PATCH2A
00745 ; AH Drives are running - shut them off
00746         OUT     (0F4H),A        ; Reset drive latches
00747         ENIF
00748 ;
00749         IF      LNW
00750         XOR     A
00751 ;       LD      (37E0H),A       ; Reset drive latches
00752         ENIF
00753 ;
00754 ; Is Ram Disk - now check if Drive 0  If so dont patch
00755 ;
00756 ;
00757 PATCH2A CALL    SYS6TES ; If system 6 then patch
00758 ;
00759 RET20   POP     AF      ;pop calls & pushes off stack
00760         POP     AF
00761         POP     AF
00762         JP      PAT2END         ;continue
00763 ;
00764 ;       SYS6/SYS - Format bypass
00765 ; Called if Current drive to be accessed to Ramdisk
00766 SYS6TES LD      A,(CURDRV)      ; Get current drive agn
00767         CP      0
00768         RET     Z               ; Dont patch if 0
00769         LD      A,(CONFLG1)     ; Overlay Address Clt
00770         BIT     3,A
00771         JR      Z,SYS6A         ; Not SYS6
00772         LD      A,(SYS6ID)      ; Check if sys is called
00773 ;                                 another overlay
00774         CP      68H
00775         RET     NZ              ; OOPS Don't patch
00776         LD      A,(626CH)       ; GET Secondary Check PT
00777         CP      20H             ; Has SYS6 Bin Reloaded ?
00778         JR      NZ,SYS6T1       ; Yes Repatch
00779 ADDR16  LD      A,(SYS6FLG)     ; PATCH SYS6 FLAG
00780         CP      10
00781         RET     Z               ; RET IF PATCHED
00782 ; Ah is active so patch it for Ramdrive
00783 SYS6T1  LD      A,0C9H  ; By pass Source Dest Check.
00784         LD      (4E6FH),A
00785 ;
00786         LD      A,94H   ; Change Check Flag
00787         LD      (4E86H),A
00788         LD      A,20H   ; Patch Bypass Secondary Check
00789         LD      (626CH),A
00790 ;
00791         LD      A,18H   ; By pass Prompts
00792         LD      (558FH),A
00793 ;
00794 ;       LD      A,0A0H  ; Force Option DDND
00795 ;       LD      (5994H),A
00796 ;
00797         XOR     A       ; By Pass Format Verify
00798         LD      (6655H),A
00799         LD      (6656H),A
00800         LD      (6657H),A
00801 ;
00802         LD      A,0C9H  ; By Pass Formatting
00803         LD      (SY6BF),A
00804 ;
00805         IF      DISP
00806         LD      A,'P'
00807         LD      (3FFEH),A
00808         ENIF
00809         LD      A,10
00810 ADDR17  LD      (SYS6FLG),A     ; SET PATCH FLAG
00811 ;
00812         RET
00813 SYS6A   XOR     A
00814 ADDR18  LD      (SYS6FLG),A     ; Reset flag
00815         IF      DISP
00816         LD      A,'U'
00817         LD      (3FFEH),A
00818         ENIF
00819         RET
00820 ;
00821 ;*****************************************************
00822 SYS6FLG DEFS    2       ; System 6 Patch flag
00823 ;*****************************************************
00824 ;       465CH           4607H
00825 ;* This The Actual Ramdisk Sector Read/Write Patch
```

```
00826 PATCH3  LD      A,(CURDRV)      ; GET CURRENT DRIVE
00827 PAT3    DEFB    0FEH            ; Check for Ramdisk
00828         DEFS    1               ; CP DN#
00829         JP      NZ,PAT3END      ;jmp if not ramdisk
00830 ;
00831         IF      DISP
00832         LD      A,'3'
00833         LD      (3FFAH),A       ; Display Patch #
00834         ENIF
00835         POP     AF              ;pop call off of stack
00836         PUSH    BC              ;save registers
00837         PUSH    DE      ; Rel sector
00838         PUSH    HL      ; Buffer area
00839         DI                      ;disable interrupt
00840         EX      DE,HL   ;HL=relative sector to read
00841 SECTOR  DEFB    11H             ; LD DE,XXXXX
00842         DEFS    2               ; SECTOR MAX
00843         PUSH    HL
00844         AND     A
00845         SBC     HL,DE
00846         POP     HL
00847 ADDR22  JP      NC,BADREC;jmp if sector not in Ramdisk
00848 ; Check IF We are accessing  Directory Sectors
00849         PUSH    HL              ; Save it
00850         POP     DE              ; DE=HL
00851         PUSH    DE
00852         LD      A,(DIRLMP)      ; Get Dir Lump
00853         LD      C,A
00854         XOR     A               ; Zero a
00855         SBC     HL,HL
00856         LD      B,A
00857         ADD     HL,BC           ; Multiply Dir Lump * 5
00858         ADD     HL,HL
00859         ADD     HL,HL
00860         ADD     HL,BC
00861         LD      A,(DIRGRN)      ; Grans per Lump
00862         DEC     A
00863         PUSH    HL
00864         POP     BC
00865 SECLP1  ADD     HL,BC           ; Multiply dir Lump/Sec
00866         DEC     A
00867         JR      NZ,SECLP1
00868 ; HL=Starting Sector of DIR ( DL * GRANS/LUMP * 5 )
00869         RST     18H             ; Compare HL TO DE
00870         JR      Z,SECDIR        ; IS DIR sector
00871         JR      NC,SEC3A        ; Before Dir
00872 ; Sector is beyond the start of the Directory
00873         PUSH    HL
00874         PUSH    DE
00875         LD      DE,9            ; Offset in Pdrive
00876         LD      HL,(PDRADD)     ; Address of current pdr
00877         ADD     HL,DE           ; Point to DDGA
00878         LD      A,(HL)          ; Get it
00879         POP     DE
00880         POP     HL
00881         LD      B,A             ; Multiply * 5
00882         ADD     A,A
00883         ADD     A,A
00884         ADD     A,B
00885         LD      C,A
00886         LD      B,0
00887         ADD     HL,BC
00888         RST     18H             ; Compare HL to DE
00889         JR      Z,SEC3A         ; Past end
00890         JR      C,SEC3A         ; Past end
00891 SECDIR  LD      A,32            ; Read Protect
00892         JR      SEC3
00893 SEC3A   LD      A,0
00894 SEC3    LD      (STATUS+1),A    ; Reset Read Protect Sec
00895         POP     HL              ; Get it back
00896 ; Cal bank value
00897         LD      DE,128  ; # Sectors/Bank
00898         LD      C,1     ; set first bank
00899 SEC1    AND     A
00900         SBC     HL,DE   ; Sub 128
00901         JR      C,SEC2  ; Go if less than
00902         INC     C       ; bump bank ptr
00903         JR      SEC1
00904 SEC2    LD      A,C             ; A = BANK # 1 to 14
00905 ADDR19  CALL    CALPAGE         ; Find Byte for bank Cmd
```

```
00906 ADDR20  LD      (READ+1),A      ; A = Command for MMU
00907         IF      MOD4
00908         LD      A,(PORTMSK)     ; Get Bit Mask
00909 ADDR27  LD      (PORT84),A      ; Save it
00910         ENIF
00911 ; Now Cal Rel Sec in bank
00912         AND     A
00913         ADD     HL,DE           ; Get Rel Sec
00914 ; HL=Rel Sector in Bank
00915 ; C=Bank #
00916 ; Now set HL=to address of Sector in Bank
00917 ; HL Cannot be > 128
00918         XOR     A               ; Zero a
00919         CP      L               ; is 0 ?
00920         JR      Z,SEC5          ; Is start of next bank
00921         LD      B,L             ; # of Sectors
00922         LD      DE,256          ; # of Bytes/Sector
00923         LD      HL,00H          ; Base Address
00924         AND     A
00925 SEC4    ADD     HL,DE
00926         DJNZ    SEC4            ; HL=B*256
00927 ;
00928         IF      HD64180         ; Bank High by MMU
00929 SEC5    LD      DE,8000H        ; Offset to 8000H
00930         AND     A
00931         ADD     HL,DE
00932         ENIF
00933         IF      NOT,HD64180
00934 SEC5    NOP
00935 ;HL=Rel address from 0000H
00936         ENIF
00937 ;HL=Address of Sector
00938         POP     DE              ;pop buffer address
00939         PUSH    DE
00940 ;
00941         LD      A,(CURFDC)      ; BYTE FOR FDC CMD
00942         AND     20H
00943         JR      NZ,WRITE        ;jmp if write sector
00944 ;
00945 ;**************************************************
00946 ;                       READ A SECTOR
00947         LD      B,0
00948 READ    LD      A,0     ;A=bank to read/write from
00949 ;--------------------------------------------------
00950         IF      HD64180
00951         DEFB    0EDH    ; OPCODE  OUT0 (m),g
00952         DEFB    39H
00953         DEFB    38H     ; PORT 38H CBR OF MMU
00954 ;                         Bank in Ram
00955         ENIF
00956         IF      Z80
00957         OUT     (84H),A ; BANK IN RAM
00958         ENIF
00959 ;--------------------------------------------------
00960         LD      C,(HL)          ;get character
00961         INC     HL      ;add 1 to Ramdrive address
00962         XOR     A               ; PAGE IN 0
00963 ;--------------------------------------------------
00964         IF      HD64180
00965         DEFB    0EDH    ; OPCODE  OUT0 (m),g
00966         DEFB    39H
00967         DEFB    38H     ; PORT 38H CBR OF MMU
00968 ;                        Bank it out
00969         ENIF
00970         IF      Z80
00971         IF      MOD4
00972 ADDR28  LD      A,(PORT84)      ; Normal Mask
00973         ENIF
00974         OUT     (84H),A ; BANK OUT RAM
00975         ENIF
00976 ;--------------------------------------------------
00977         LD      A,C
00978         LD      (DE),A          ;save byte to buffer
00979         INC     DE              ;add 1 to buffer address
00980         DJNZ    READ    ;continue  256 bytes read
00981 ;**************************************************
00982 ;
00983 STATUS  LD      A,0     ;read protect status if directory
00984         JR      RET3            ;continue in DOS
00985 BADREC  LD      A,0C1H          ;Error code
```

```
00986         JR      RET3            ;continue in DOS
00987 ;
00988 ;****************************************************
00989 ;       WRITE A SECTOR
00990 WRITE   LD      B,0
00991 WRITE1  XOR     A
00992 ;---------------------------------------------------
00993         IF      HD64180
00994         DEFB    0EDH    ; OPCODE  OUT0 (m),g
00995         DEFB    39H
00996         DEFB    38H     ; PORT 38H CBR OF MMU
00997 ;                         Bank it out
00998         ENDIF
00999         IF      Z80
01000         IF      MOD4
01001 ADDR29  LD      A,(PORT84)      ; Normal Mask
01002         ENDIF
01003         OUT     (84H),A ; BANK OUT RAM IF IN
01004         ENDIF
01005 ;---------------------------------------------------
01006         LD      A,(DE)          ;get character to write
01007         LD      C,A
01008         INC     DE              ;add 1 to buffer
01009 ADDR23  LD      A,(READ+1)
01010 ;---------------------------------------------------
01011         IF      HD64180
01012         DEFB    0EDH    ; OPCODE  OUT0 (m),g
01013         DEFB    39H
01014         DEFB    38H     ; PORT 38H CBR OF MMU
01015 ;                         Bank it in
01016         ENDIF
01017         IF      Z80
01018         OUT     (84H),A ; BANK IN RAM
01019         ENDIF
01020 ;---------------------------------------------------
01021         LD      (HL),C          ;write character
01022         INC     HL      ;add 1 to Ramdrive address
01023         DJNZ    WRITE1  ;continue write 256 times
01024         XOR     A
01025 ;---------------------------------------------------
01026         IF      HD64180
01027         DEFB    0EDH    ; OPCODE  OUT0 (m),g
01028         DEFB    39H
01029         DEFB    38H     ; PORT 38H CBR OF MMU
01030 ;                         Bank it out
01031         ENDIF
01032         IF      Z80
01033         IF      MOD4
01034 ADDR30  LD      A,(PORT84)      ; Normal Mask
01035         ENDIF
01036         OUT     (84H),A ; BANK OUT RAM
01037         ENDIF
01038 ;
01039 ;---------------------------------------------------
01040 ;
01041 ;
01042 RET3    NOP                     ; Patch Area for Debug
01043         IF      Z80
01044         IF      MUXMEM  ; Added Memory
01045         PUSH    AF      ; SAVE READ STATUS FLAG
01046 ADDR34  LD      A,(PORT148)     ; GET ORIG BITS
01047         OUT     (148),A
01048         POP     AF
01049         ENDIF
01050         ENDIF
01051         JP      PAT3CON         ;continue in DOS
01052 ;
01053         IF      Z80
01054         IF      MOD4
01055         IF      NOT,MUXMEM      ; ADDED MUX MEMORY
01056 ;****************************************************
01057 ;               CALPAGE - SETUP PAGE FOR MMU
01058 ;                  96K 1 to 3  Pages for Model 4
01059 ;                  64K 1 to 2  Pages for Model 4P
01060 ;       ENTER   A = PAGE # 1T03
01061 ;       EXIT    A = Command Byte for Port 84H
01062 ;
01063 ;
01064 CALPAGE PUSH    HL
01065         LD      L,63H   ; Bank 1
01066         CP      1
01067         JR      Z,CALEN
01068         LD      L,73H   ; Bank 2
01069         CP      2
01070         JR      Z,CALEN
01071         LD      L,3H    ; Bank 3
01072 CALEN   LD      A,(PORTMSK)     ; GET PORT  84 bit mask
01073         OR      L
01074         POP     HL
01075         RET
01076         ENDIF
01077         ENDIF
01078         ENDIF
01079 ;****************************************************
01080         IF      HD64180
01081 ;****************************************************
01082 ;               CALPAGE - SETUP PAGE FOR MMU
01083 ;                  256K =  1 TO 14 PAGES
01084 ;       ENTER   A = PAGE # 1T014
01085 ;       EXIT    A = Command Byte for MMU
01086 ;
01087 ;
01088 CALPAGE PUSH    HL
01089         LD      L,A
01090         LD      H,8             ; FACTOR
01091         DEFB    0EDH            ; MLT - MULTIPLY
01092         DEFB    6CH             ; H X L
01093 ; HL HAS RESULTS
01094         LD      A,L             ; GET LSB
01095 ;
01096         POP     HL
01097         RET
01098 ;
01099 ;****************************************************
01100         ENDIF
01101         IF      Z80
01102         IF      MUXMEM  ; ADDED Mux Memory Port 148
01103 ;****************************************************
01104 ;               CALPAGE - SETUP PAGE FOR MMU
01105 ;          1 TO 31 Pages 2,031,616 Bytes of Ramdisk
01106 ;
01107 ;
01108 ;       ENTER   A = PAGE # 1T03
01109 ;       EXIT    A = Command Byte for Port 84H
01110 ;
01111 ;
01112 CALPAGE PUSH    HL
01113         PUSH    BC
01114         LD      L,A     ; SAVE IT
01115         IN      A,(148) ; GET BITS
01116 ADDR33  LD      (PORT148),A ; Save Bits for later
01117         LD      H,A     ; SAVE H
01118         LD      A,L     ; PUT A BACK
01119 ;
01120         LD      L,63H   ; Bank 1
01121         CP      1
01122         JR      Z,CALEN
01123         LD      L,73H   ; Bank 2
01124         CP      2
01125         JR      Z,CALEN
01126 ; OK Its more that 64K We will assume extra Memory
01127 ;
01128         LD      L,73H   ; Even page
01129         BIT     0,A     ; Is it an even # ?
01130         JR      Z,CALEVEN
01131 ; IS AN ODD #
01132         LD      L,63H   ; Odd page
01133         INC     A       ; BUMP UP TO EVEN #
01134 CALEVEN AND     31      ; Mask off upper bits
01135                         ; Just incase we screwed up
01136         AND     A       ; Clear Carry if set
01137         RR      A       ; Divide / 2
01138         LD      C,A     ; Save bottom 5 Bits
01139         LD      A,H
01140         AND     224     ; Mask off lower bits
01141         OR      C       ; Mix together
01142 ; Ah .... about time ....
01143         OUT     (148),A ; Set Extra Memory Mux
01144         IF      MOD4    ; Is Z80 & MOD4
01145 CALEN   LD      A,(PORTMSK)     ; GET PORT  84 bit mask
```

```
01146          OR      L        ; Mix A & L
01147          POP     BC
01148          POP     HL
01149          RET
01150          ENIF
01151          IF      NOT;MOD4
01152 CALEN    LD      A,L
01153          POP     BC
01154          POP     HL
01155          RET
01156          ENIF
01157 ;
01158 ;*******************************************************
01159          ENIF
01160          ENIF
01161 ;
01162 ;*******************************************************
01163 ;        47A3H           4754H
01164 ;* Patch in Pdrive Prameters IF Ramdisk or Swaped Drive
01165 ;
01166 PATCH4   PUSH    AF       ; Save AF
01167          LD      A,(CURDRV)      ; Check Current Drive
01168 PAT4     DEFB    0FEH            ; Check if Ramdisk
01169          DEFS    1               ; Patch byte
01170          JR      NZ,PATCH41      ; Nop
01171 ;
01172          IF      DISP
01173          LD      A,'4'
01174          LD      (3FFBH),A       ; Display Patch #
01175          ENIF
01176 ; Is us - Load in ower Pdrive for Mem Drive
01177 ADDR21   LD      HL,PDRIVE       ; SET HL=DRIVE 4 PDRIVE
01178 PATCH42 POP      AF
01179          LD      (PDRADD),HL
01180          RET                     ; CONTINUE
01181 ;
01182 PATCH41 CP       4               ; IS DRIVE 4 ?
01183          JR      NZ,PATCH42      ; NOP
01184 ; Is Drive # 4 - AND This is not the Ramdisk
01185 ; - Now set Pdrive for Replaced drive
01186          LD      HL,PDRTBL       ; Pdrive Table
01187 ADDR24   LD      A,(PAT1+1)      ; Get patched drive
01188          CP      0               ; Is it drive 0 ?
01189          JR      Z,PATCH42       ; HL=DRIVE 0 PDRIVE
01190          PUSH    BC              ; Save it
01191          PUSH    DE
01192          LD      DE,10           ; # of enterys
01193          LD      B,A             ; Set up counter
01194 PAT4LP1 ADD      HL,DE           ; BUMP 10
01195          DJNZ    PAT4LP1
01196          POP     DE
01197          POP     BC
01198          JR      PATCH42
01199 ;*******************************************************
01200 ;        4798H                   4749H
01201 ;* Patch in Physical Drive Latches IF Swaped Drive
01202 PATCH5   LD      L,A      ; Save it
01203 ADDR25   LD      A,(PAT1+1)      ; Get current Mem drive
01204          CP      4               ; Swap flag
01205          JR      Z,PATCH51       ; NOTHING TO DO
01206 ;
01207          IF      DISP
01208          LD      A,'5'
01209          LD      (3FFCH),A       ; Display Patch #
01210          ENIF
01211 ; Now check current drive
01212          LD      A,(CURDRV)      ; Get current drive
01213          CP      4
01214          JR      NZ,PATCH51      ; DO NOTHING
01215          PUSH    BC
01216          LD      A,(CURDCMD)     ; Get Current bit setting
01217          AND     0F0H            ; Mask of low bits 0-3
01218          LD      C,A             ; save
01219 ADDR26   LD      A,(PAT1+1)      ; Get Swaped Drive
01220          LD      B,A             ; Save to B for loop
01221          CP      0               ; Is it dr0 ?
01222          LD      A,1
01223          JR      Z,PATCH54       ; IS drive 0
01224 PATCH52 RL       A               ; ROTATE 1 2 4 8
01225          DJNZ    PATCH52
01226 PATCH54 OR       C               ; OR in bits 4-7
01227          POP     BC
01228          JR      PATCH53
01229 PATCH51 LD       A,L
01230 PATCH53 LD       (CURDCMD),A     ; DO INST
01231          RET
01232 ;
01233 ;*******************************************************
01234 ;        4C19H           4B8EH
01235 ; Patch for SYS6 incase we get an error and reload SYS6
01236 ;
01237 PATCH6   PUSH    AF
01238          IF      DISP
01239          LD      A,'6'
01240          LD      (3FFDH),A
01241          ENIF
01242          XOR     A
01243 ADDR32   LD      (SYS6FLG),A
01244 ;
01245          LD      A,26H
01246 ;
01247          JP      PAT6END
01248 ;
01249 ;*******************************************************
01250          IF      MOD4
01251 PORT84   DEFS    2        ; Port Mask for Model 4
01252          IF      MUXMEM
01253 PORT148 DEFS     2        ; Port Mask for Extra Memory bank
01254          ENIF
01255          ENIF
01256 PDRIVE   DEFS    17       ; Pdrive Table for Drive 4
01257          DEFS    3
01258 PGMEN    NOP              ; END OF PROGRAM FOR RELOCATION
01259 ;*******************************************************
01260          IF      HD64180
01261 ;                         MMU CHART
01262 ;*******************************************************
01263 ;                         CBAR = 128
01264 ;                  0 TO 32K = BASE AREA
01265 ;                  32 TO 64K = COMMON AREA 1
01266 ;                         BBR = 0
01267 ;                  BANK AREA - NOT USED
01268 ;
01269 ;                         CBR
01270 ;
01271 ; 256 128 64K 32K
01272 ; 64  32  16  8  4  2  1   MEM RANGE    BYTE   BANK #
01273 ;  0   0   0  0  0  0  0   32 TO 64K    0      1
01274 ;  0   0   0  1  0  0  0   64 TO 96K    8      2
01275 ;  0   0   1  0  0  0  0   96 TO 128K   16     3
01276 ;  0   0   1  1  0  0  0   128 TO 160K  24     4
01277 ;  0   1   0  0  0  0  0   160 TO 192K  32     5
01278 ;  0   1   0  1  0  0  0   192 TO 224K  40     6
01279 ;  0   1   1  0  0  0  0   224 TO 256K  48     7
01280 ;
01281          END     HDSTART
01282          ENIF
01283          IF      Z80
01284          END     START
01285          ENIF
```

TA

Ba
offered
P1351C (
DMP-210
their do
folks wh
a font
newly-c
program.
creation

The
converte
format f
includes
I.D. bloc
standard
way?).
different.
their fon
editor ca
(even if
that the
all the b
nothing
from a P
install th

For
hexadeci
second a

Tandy:
Prosoft: 1

A g
files sim
Prosoft f
use the d

Tandy : 42
Prosoft: 42

I w
What doe
nybbles.
selection
file. The
of zero t
of 30-37H
significan
value for

As
Custom F
of the fo
the font
15H, or 8
but not i
any of th
allowed in
contains
bits are
character
and some
loaded int
future ref

So,
follows:

wher
Tandy ext
set if the
illegal in t

One
program th
you to spe
font 5, an

TANDY DMP-2100P FONT CONVERSION PROGRAM REVISITED

by Jack Decker

Back in NORTHERN BYTES Volume 7, Number 2 (page 8) I offered a program called FONTCONV/BAS, which converted Toshiba P1351C (or P351C) printer fonts to the format used by the Tandy DMP-2100P printer (actually, to the format used by Prosoft for their downloadable fonts). Well, it turns out that there are many folks who have purchased the Tandy Custom Font System, designed a font for the DMP-2100P, and then wanted to use their newly-created font from within the Allwrite word processing program. This requires a minor conversion of the font file, and the creation of a width table (with the /TAB extension).

The reason that the Custom Font System font file needs to be converted is that while both Toshiba and Prosoft use a standard format for their font files (with the sole exception that Toshiba includes a "dummy" first sector of 256 bytes, which serves as an I.D. block), Tandy Custom Font System font files do NOT use the standard format (since when did Tandy ever do things the easy way?). The big problem is that the very start of the font files are different. Tandy supplies their own loader program for downloading their fonts, which means that fonts created using the TANDY font editor cannot be directly used as a "downloadable font" by Allwrite (even if the proper width table file is present). It seems to me that the Prosoft format is the most logical one, since it consists of all the bytes that must be sent to the printer to install a font - nothing more, nothing less. ANY program that can read the bytes from a Prosoft format file and sling them out to the printer can install the font.

For illustrative purposes, here are the first few bytes (in hexadecimal) of the same font file, first as supplied by Tandy, and second as converted to run under Allwrite ("Prosoft" format):

```
Tandy:                         24 0F 20 31 33 30 30 0F 21 ...
Prosoft: 1B 26 34 30 34 30 30 30 0F 20 31 33 30 30 0F 21 ...
```

A glance at the ends of the files shows that the Tandy font files simply end with pattern data for the last character, while the Prosoft format includes some control codes to set up the printer to use the downloaded font:

```
Tandy  : 42 40 40 40 40 40 40 40 40 40 40 40 40 40 40
Prosoft: 42 40 40 40 40 40 40 40 40 40 40 40 40 40 40 00 00 1B 3F 34 1B 11
```

I wondered about that first byte of the Tandy font file. What does it indicate? Well, it appears that it divides into two nybbles. Bit 3 appears to be unused, but bits 0-2 are the pitch selection and are equivalent to the fifth byte of a Prosoft font file. The only difference is that the three bits only allow a value of zero through seven, whereas the printer expects an ASCII value of 30-37H. The conversion is obvious, you just extract three least significant bits, ADD to or AND with 30H, and you get the proper value for byte 5.

As far as I can tell, bits 6 and 7 of the first byte of a Custom Font System font file are unused (at least they are in all of the fonts supplied with the Custom Font System). Bit 4 is set if the font file contains any of the "IBM Extended Characters" 14H or 15H, or 80H-9FH, which are allowable characters in the IBM mode but not in the Tandy mode. Bit 5 is set if the font file contains any of the "Tandy Extended Characters" C0H-DFH, which are not allowed in the IBM mode. If neither bit is set, then the font file contains no characters that are illegal in either mode. If BOTH bits are set, it means the user has created a font file with characters that are illegal in both modes (that is, it has some IBM and some Tandy extended characters) and while this file cannot be loaded into the printer in either mode, it can be saved to disk for future reference.

So, the first byte of a Tandy font file might be pictured as follows:

```
U   U   T   I   U   P   P   P
```

where the U's are unused, T is set if the file contains any Tandy extended characters that are illegal in the IBM mode, I is set if the file contains any IBM extended characters that are illegal in the Tandy mode, and P P P are the allowable pitch bits.

One other note - when you use the Tandy FLOAD (font loader) program that comes with the Tandy Custom Font System, it forces you to specify which font to use as font 4 and which to use as font 5, and the font and pitch that the printer should be using after the FLOAD program is run. None of this information, therefore, is included in the Tandy font file itself.

I have revised FONTCONV/BAS so that it will now convert both Toshiba and Tandy font files to the Prosoft format, so that any of these files can be used with Allwrite. The conversion program now asks whether you are converting a Tandy Custom Font System format file, or a Toshiba P1351C/P351C format file, and then makes the necessary conversions. As presently written, the program will not attempt to convert a Tandy format font file that contains "IBM extended characters" (14H or 15H). To convert such a file, you must use the FEDIT program of the Tandy Custom Font System to delete the offending characters, then convert the resulting file.

I would like to gratefully acknowledge the assistance of Rabbi Naftoli M. Eisemann (7432 Malvern Avenue, Philadelphia, Pennsylvania 19151) in developing the latest version of this conversion program. Rabbi Eisemann has developed several Hebrew character sets using the Tandy Custom Font System, and some of these (converted to Prosoft format) appear on TAS Public Domain Library Disk #020, along with the following program (FONTCONV/BAS), Arne Rohde's user-supported font editor program (which lets you create character sets for the Tandy DMP-2100P <u>and</u> the Toshiba P1351C/P351C printers), and other public domain font files and printer utilities. If you own a Tandy DMP-2100P or a Toshiba P1351C or P351C printer, you will almost certainly want to get a copy of TAS Public Domain Library disk #020 ($10 plus shipping and handling from The Alternate Source). By the way, early purchasers of PD disk #020 may have received versions of P351C and P351C1 (Arne Rohde's font editor) that either would not load Tandy format font files at all, or if they did, they did not erase the old font information in memory before loading a new font, thus intermixing character sets. The current version has this bug fixed. If you have the old versions of P351C and P351C1, you may send your master PD disk (as purchased from TAS) and a <u>stamped</u>, <u>self-addressed</u> disk mailer to me (<u>not</u> to TAS) and I will copy the revised programs onto your PD Library disk (my address is: 1804 West 18th Street #155, Sault Ste. Marie, Michigan 49783-1268).

Here's the revised version of FONTCONV/BAS:

```
10 CLEAR1000:DEFINTA-Z:INPUT"CONVERT FROM <1> TANDY CUSTOM
 FONT OR <2> TOSHIBA FONT";F:IFF<1ORF>2THEN10
20 LINEINPUT"FILE TO CONVERT: ";A$:IFINSTR(A$,"/FNT")>0ORI
NSTR(A$,"/TAB")>0THENPRINT"DUPLICATE FILE NAMES":END
30 OPEN"I",1,A$:CLOSE:ONERRORGOTO310:OPEN"R",1,A$,1:ONERRO
RGOTO0:T=INSTR(A$,"/"):IFT>0THENA$=LEFT$(A$,T-1)
40 INPUT "OUTPUT TO DRIVE #";T:IFT<0ORT>9THEN40ELSEB$=":"+
MID$(STR$(T),2,1)
50 OPEN"R",2,A$+"/FNT"+B$,1:OPEN"R",3,A$+"/TAB"+B$
60 FIELD1,1ASA$:FIELD2,1ASB$:FIELD3,1ASX$,255ASZ$
70 LSETX$="2":Y$=STRING$(255,0):FORX=1TO255:MID$(Y$,X,1)=C
HR$(X):NEXT:LSETZ$=Y$:PUT3:Y$=STRING$(255,0)
80 IFF=1THEN260ELSEFORX=1TO256:GET1:NEXT
90 FORX=1TO2:GET1:LSETB$=A$:PUT2:NEXT
100 GET1:LSETB$=A$:PUT2:D$=A$
110 GET1:LSETB$="0":PUT2:P$=A$
120 GET1:LSETB$=A$:PUT2:P=VAL(A$)
130 FORX=1TO3:GET1:LSETB$=A$:PUT2:NEXT
140 GET1:LSETB$=A$:PUT2:T=ASC(A$):IFT=0THEN190ELSEIF(T<14)
OR(T>15)THEN320
150 GET1:LSETB$=A$:PUT2:IFT=15THENT=ASC(A$)ELSET=ASC(A$)+1
28
160 T$="":FORX=1TO2:GET1:LSETB$=A$:PUT2:T$=T$+A$:NEXT:MID$
(Y$,T-31,1)=CHR$(VAL(T$)+64)
170 T$="":FORX=1TO2:GET1:LSETB$=A$:PUT2:T$=T$+A$:NEXT
180 IFVAL(T$)=0THEN140ELSEFORX=1TO4*VAL(T$):GET1:LSETB$=A$
:PUT2:NEXT:GOTO140
190 GET1:LSETB$=A$:PUT2:IFA$<>CHR$(0)THEN320
200 IF(PAND1)THENLSETB$=CHR$(27):PUT2:LSETB$=CHR$(61):PUT2
:LSETB$=D$:PUT2
210 IF(PAND2)THENLSETB$=CHR$(27):PUT2:LSETB$=CHR$(62):PUT2
:LSETB$=D$:PUT2
220 IF(PAND4)THENLSETB$=CHR$(27):PUT2:LSETB$=CHR$(63):PUT2
:LSETB$=D$:PUT2
230 LSETB$=CHR$(27):PUT2:IFP$="N"THENLSETB$=CHR$(18)ELSEIF
P$="E"THENLSETB$=CHR$(29)ELSELSETB$=CHR$(17)
240 PUT2:IFMID$(Y$,96,1)=CHR$(0)THENMID$(Y$,96,1)=LEFT$(Y$
,1)
250 MID$(Y$,193,31)=STRING$(31,82):LSETZ$=Y$:PUT3:CLOSE:EN
D
```

```
260 GET1:P=ASC(A$):IF(P/16)AND1THENPRINT"FONT FILE CONTAIN
S IBM EXTENDED CHARACTERS - CANNOT CONVERT":ENDELSEP=(PAND
7)
270 LINEINPUT"DOWNLOAD AS FONT NUMBER <4> OR <5>? ";D$:IFD
$<>"4"ANDD$<>"5"THEN270
280 PRINT"SELECT PITCH TO LEAVE PRINTER IN AFTER DOWNLOAD
-"
290 LINEINPUT"<N>ORMAL (10), <E>LITE (12), OR <P>ROPORTION
AL: ";P$:IF(P$<>"N")AND(P$<>"E")AND(P$<>"P")THEN280
300 LSETB$=CHR$(27):PUT2:LSETB$=CHR$(38):PUT2:LSETB$=D$:PU
T2:LSETB$="0":PUT2:LSETB$=MID$(STR$(P),2):PUT2:LSETB$="0":
FORX=1TO3:PUT2:NEXT:GOTO140
310 IFERR=162THENPRINT:PRINT"ERROR - YOU MUST SPECIFY TWO
VARIABLE LENGTH FILES WHEN":PRINT"ENTERING BASIC (I.E. "CH
R$(34)"BASIC 2V"CHR$(34)")":PRINT:ENDELSEONERRORGOTO0:OPEN
"R",1,A$,1:END
320 PRINT"INVALID CHARACTER DETECTED":CLOSE:END
```

CONVERTING MODEL III SUPERSCRIPSIT FILES FOR USE WITH THE MODEL 4 VERSION OF SUPERSCRIPSIT

[Reprinted from the TIDBITS column of the SIG-80 newsletter of the Chicago Area Computer Hobbyist Exchange:]

This issue's TIDBIT comes from a conversation I had with Peter Hru at our last meeting. He's no beginner, but was having trouble with something that Radio Shack didn't explain and no one else knew the answer to.

For the many of us who have upgraded from Model III's to Model 4's or 4P's, the big headache is - "Do we want to retype or recreate all those SuperScripsit and VisiCalc files, or is there an easier way?"

Of course there's an easier way. It's called CONVERT (CONV/CMD), a handy program supplied with the Model 4 (-4P) to 'convert' your Model III files to your new computer. With your NEW Model 4 TRSDOS Disk in Drive :0 and your OLD Model III data disk in Drive :1, you type "CONV:1" at TRSDOS Ready and answer the questions and "poof" - your old files on Drive :1 are moved to Drive :0 in the new Model 4 'mode'.

Nice, until you try to open a SuperScripsit file under your new operating system and encounter "FILE NOT FOUND" as an error message. What the heck does that mean? I know that file is there ... because I can see it in the Directory.

It's not the data file that the program can't find ... it's your print driver. "My WHAT?...?" Your Print Driver, that little sub-program that tells your printer how to handle all those wonderful control codes that you try to use in this word processor.

You'll notice when you open a document under SuperScripsit, you must specify which printer you will be using. If you have a DMP2100 and print a document, the program will call a control program called "DMP2100/CTL" which is on your program disk and use it to print your document.

The problem here is that you may be using a Radio Shack printer like the DWP-210. You specify that printer in your opening menu on your Model III. And when you convert your file to the Model 4, you see the same print driver specified at the opening menu. But instead of seeing the Model 4 character for "no character" or "blank" which is an UNDERLINE, you see the carry-over character for no-space, which is a small block SQUARE. These "square" blanks translate during the CONVERT process to the same blank-squares. Unfortunately, the Model 4 Superscripsit looks at those blank-squares and tries to match them to a printer named "DWP210 PLUS 2 BLANK SQUARES". It can't find such a printer and quits right there, giving you the error message.

So how do you fix this problem? After CONVERT'ing your Model III program files to Model 4, and after opening that favorite SuperScripsit file, before you jump into the document, delete the two or three or four square spaces that follow the name of your printer. Move the cursor past the printer name and press F-2 (Delete) or Control-D. You'll see underlines (blanks) in place of the squares and you can then open the file under your Model 4 word processor with no problem.

Why doesn't Tandy/Radio Shack tell you this trick? I don't know. I found it out by trial and error and have never seen it in print anywhere. I would love to be proven wrong by seeing something official from Tandy, in print, that explains this process. I don't think it has ever been published (another SIG-80 EXCLUSIVE).

SU 4/4P: THE FILE

by Mark McDougall

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

This article details a method for the creation of a /CMD file of the Model 4/4P version of Super Utility.

1. Boot the SU 4/4P disk as normal.
2. Change the following bytes at 69E4H - 77 BE C4 to C3 00 E0.
3. Enter the following bytes at E000H -

   21 FF 8D 11 FF DF 01 00 8E ED B8 C3 0B E0.
4. Test Memory (8F00H,8FFFH).
5. After a few seconds, boot (MODELA/III) NEWDOS/80.
6. Enter the command - DUMP SU4/CMD:1 5200H,DFFFH,402DH.
7. Reboot the SU 4/4P disk.
8. Boot (MODELA/III) NEWDOS/80 again.
9. Enter the command -

   DUMP TITLE/CMD:1 9201H,9A00H,402DH.
10. Relocate TITLE/CMD to E000H.
11. Using SU32 or similar, display file sectors SU4/CMD.
12. Display FRS 107, and <M> for modify.
13. Change the following bytes at SRB 90 - C3 00 E0 to 77 BE C4.
14. Return to NEWDOS/80.
15. Using EDPLUS or similar, assemble the following /ASM file:

```
E800            00100          ORG     0E800H
E800            00110 START    EQU     $
E800 F3         00120          DI
E801 3100E9     00130          LD      SP,0E900H
E804 ED56       00140          IM      1
E806 3E60       00150          LD      A,60H
E808 ED47       00160          LD      I,A
E80A 3E82       00170          LD      A,82H
E80C D384       00180          OUT     (84H),A
E80E 210052     00190          LD      HL,5200H
E811 110000     00200          LD      DE,0000H
E814 01008E     00210          LD      BC,8E00H
E817 EDB0       00220          LDIR
E819 2100E0     00230          LD      HL,0E000H
E81C 1100F8     00240          LD      DE,0F800H
E81F 010008     00250          LD      BC,0800H
E822 EDB0       00260          LDIR
E824 CD2C19     00270 KEY      CALL    192CH
E827 B7         00280          OR      A
E828 28FA       00290          JR      Z,KEY
E82A 31B101     00300          LD      SP,01B1H
E82D C3A203     00310          JP      03A2H
0000            00320          END
00000 TOTAL ERRORS

KEY      E824      START    E800
```

16. Assemble the file under SUEK/CMD.
17. From NEWDOS, enter the commands.

```
LOAD SU4/CMD:1
LOAD TITLE/CMD:1
LOAD SUEK/CMD:1
DUMP SU4x/CMD:1 5200H,E82FH,E800H
```

Super Utility 4/4P is now executable from any Model III/4 DOS.

Hard Configuring the SU 4/4P File

1. Execute the file and configure system.
2. Display Memory at 1B1H and dump it & the next page to printer.
3. Take note of:

   The byte at 00BFH - the Printer Config byte

   The byte at 037BH - the User Key.
4. Display File Sectors SU4x/CMD.
5. FRS 00 BYTE C3H - the printer config byte.
6. FRS 01 - The drive configuration starts after the FILENAME/EXT.PASSWORD, start zapping the table as is on the printout. BE CAREFUL of the sector data near the beginning of file relative sector 2.
7. FRS 03 BYTE 8BH - the user key byte.
8. FRS 37 BYTE 78H - 01 enables the sound prompt (F2).
9. Save the modified sector to disk.

Super Utility 4/4P is now hard configured.

[redacted]

by Arne Rohde

Box 82-211, Highland Park, Auckland, New Zealand

Command line edit is a short program for NEWDOS/80 version 2 on the TRS-80 Model I or III, and TRSDOS 6 on Model 4 systems, allowing the last executed command line to be recalled and edited. With NEWDOS/80 the complete line is recalled, with TRSDOS 6 the first two characters will have to be retyped.

The program is found in three separate versions, I/CMD for Model I NEWDOS/80, E/CMD for Model III/4 NEWDOS/80, and R/CMD for Model 4 TRSDOS 6. The Model I and III versions require no user memory, and the Model 4 version resides under location 3000H and should not overlay any user programs. The source code is found in I/ASM, E/ASM and R/ASM.

When the program is executed, the screen will be cleared, and the last command line will be displayed on the top screen line. For NEWDOS/80 the complete line will be displayed, and the cursor will be positioned after the last character. For TRSDOS the first two characters will be blank, and the cursor is placed on the first character.

The command line editor starts in replace mode, but can be toggled between replace and insert mode with the down arrow key. The left and right arrow keys can be used for moving the cursor, unshifted to move one position at a time, and shifted to move to the beginning or end of the line. Characters will be inserted to the left of the cursor position. The CLEAR (SHIFT-CLEAR for TRSDOS) key can be used to delete the character hidden by the cursor. The BREAK key will terminate editing and return to DOS ready mode.

The ENTER key is used to execute the edited command line. The position of the cursor in the line is immaterial; the complete line will be passed to the DOS for execution.

It should be possible to convert the program for use on other operating systems. The location of the command line buffer and the system I/O buffer must be known, as must the entry point for executing a DOS command.

If you have any comments or suggestions, please send them to the author at the address given at the top of this article.

MODEL I VERSION - I/ASM

```
              00100 ;I/ASM - edit dos command line 86-07-28
              00110 ;programmed by arne rohde, box 82-211, auckland, nz
              00120 ;must fit within 1 sector to use DOS sector buffer
              00130 ;for Newdos/80 v2 Model I (for M III change nxt 3 lines)
4318          00140 BUFF    EQU     4318H           ;command buffer(MIII=4225)
4200          00150         ORG     4200H           ;dos buffer area(MIII=4300)
4200 2AA743   00160 BEGIN   LD      HL,(43A7H)      ;2 chars (MIII=42C7)
4203 221843   00170         LD      (BUFF),HL       ;restore first 2 chars
4206 3E0D     00180         LD      A,0DH           ;set last to cr
4208 326743   00190         LD      (BUFF+79),A
420B 211843   00200         LD      HL,BUFF         ;buffer start
420E C3A942   00210         JP      LINEND          ;cursor to line end
4211          00220 BEGLIN  EQU     $
4211 211843   00230         LD      HL,BUFF         ;buffer start
4214          00240 DSPLC   EQU     $
4214 CDC901   00250         CALL    01C9H           ;clear screen
4217 DD2A2040 00260         LD      IX,(4020H)      ;cursor pos
421B          00270 DSPLB   EQU     $
421B 4E       00280         LD      C,(HL)          ;buffer char
421C 7D       00290         LD      A,L
421D FE67     00300         CP      BUFF+79&0FFH    ;check at buffer end
421F 2814     00310         JR      Z,RPLNOC        ;yes, no cursor
4221 79       00320         LD      A,C             ;char at cursor
4222 FE0D     00330         CP      0DH             ;check if end of line
4224 380F     00340         JR      C,RPLNOC        ;line feed, no cursor
4226 2003     00350         JR      NZ,RPLCUR       ;no, replace cursor
4228 23       00360         INC     HL              ;to next char
4229 77       00370         LD      (HL),A          ;set to line end
422A 2B       00380         DEC     HL              ;cursor pos back
422B          00390 RPLCUR  EQU     $
422B 368F     00400         LD      (HL),08FH       ;set cursor
422D 3A9542   00410         LD      A,(MODFLG)      ;check insert mode
4230 B7       00420         OR      A
4231 2802     00430         JR      Z,RPLNOC        ;replace mode
4233 3680     00440         LD      (HL),080H       ;alternative cursor
4235          00450 RPLNOC  EQU     $
4235 EB       00460         EX      DE,HL           ;save pos
4236 211843   00470         LD      HL,BUFF         ;start for display
4239 CD6744   00480         CALL    4467H           ;displ line
423C DD222040 00490         LD      (4020H),IX      ;restore cursor pos
4240 EB       00500         EX      DE,HL           ;pos back to hl
4241 71       00510         LD      (HL),C          ;restore char
4242 CD4900   00520         CALL    0049H           ;get key in
4245 FE18     00530         CP      24              ;shift left
4247 28C8     00540         JR      Z,BEGLIN        ;line begin
4249 FE01     00550         CP      1               ;break key
424B CA2D40   00560         JP      Z,402DH         ;return to dos
424E FE09     00570         CP      9               ;right arrow
4250 282D     00580         JR      Z,MVRGT
4252 FE08     00590         CP      8               ;left arrow
4254 2831     00600         JR      Z,MVLFT
4256 FE0A     00610         CP      10              ;down arrow
4258 283A     00620         JR      Z,INSMDE        ;set insert mode
425A FE0D     00630         CP      0DH             ;enter key
425C 283F     00640         JR      Z,ENTRKY
425E FE19     00650         CP      25              ;shift right
4260 2847     00660         JR      Z,LINEND
4262 FE1F     00670         CP      31              ;clear key
4264 284B     00680         JR      Z,DELCHR        ;delete character
4266 FE20     00690         CP      20H             ;check ctrl char
4268 38B1     00700         JR      C,DSPLB         ;yes, ignore
426A 08       00710         EX      AF,AF'          ;save key code
426B 7D       00720         LD      A,L             ;get addr lsb
426C FE67     00730         CP      BUFF+79&0FFH    ;check if end
426E 28AB     00740         JR      Z,DSPLB         ;yes, ignore
4270 7E       00750         LD      A,(HL)          ;get char
4271 FE0D     00760         CP      0DH             ;check current end
4273 CCC442   00770         CALL    Z,MBUFRG        ;yes, move buffer right
4276 3A9542   00780         LD      A,(MODFLG)      ;check insert mode
4279 B7       00790         OR      A
427A C4C442   00800         CALL    NZ,MBUFRG       ;insert character
427D 08       00810         EX      AF,AF'          ;get char again
427E 77       00820         LD      (HL),A          ;insert char
427F          00830 MVRGT   EQU     $
427F 7E       00840         LD      A,(HL)          ;check if end of line
4280 FE0D     00850         CP      0DH
4282 2897     00860         JR      Z,DSPLB         ;at end, no move right
4284 23       00870         INC     HL              ;else incr cursor pos
4285          00880 JDSPLB  EQU     $
4285 1894     00890         JR      DSPLB
4287          00900 MVLFT   EQU     $
4287 7D       00910         LD      A,L             ;current pos lsb
4288 FE18     00920         CP      BUFF&0FFH       ;check at beginning
428A 288F     00930         JR      Z,DSPLB         ;yes, exit
428C 7E       00940         LD      A,(HL)          ;char at cursor
428D 2B       00950         DEC     HL              ;else cursor back
428E FE0D     00960         CP      0DH             ;check line end
4290 2089     00970         JR      NZ,DSPLB        ;no, no screen clear
4292          00980 JDSPLC  EQU     $
4292 1880     00990         JR      DSPLC           ;else clear screen
4294          01000 INSMDE  EQU     $
4294 3E00     01010         LD      A,0             ;current mode flag
4295          01020 MODFLG  EQU     $-1
4296 EEFF     01030         XOR     0FFH            ;invert flag
4298 329542   01040         LD      (MODFLG),A
429B 18E8     01050         JR      JDSPLB
429D          01060 ENTRKY  EQU     $
429D CDC901   01070         CALL    01C9H           ;clear screen
42A0 211843   01080         LD      HL,BUFF         ;command buffer start
42A3 CD6744   01090         CALL    4467H           ;redisplay
42A6 C30544   01100         JP      4405H           ;execute dos command
42A9          01110 LINEND  EQU     $
42A9 3E0D     01120         LD      A,0DH
42AB BE       01130         CP      (HL)            ;find line end
42AC 28E4     01140         JR      Z,JDSPLC        ;found, display new
42AE 23       01150         INC     HL
42AF 18F8     01160         JR      LINEND          ;to next char
42B1          01170 DELCHR  EQU     $
42B1 7E       01180         LD      A,(HL)          ;char at cursor
42B2 FE0D     01190         CP      0DH             ;check end of line
42B4 28CF     01200         JR      Z,JDSPLB        ;yes, exit
42B6 54       01210         LD      D,H             ;addr in buffer
42B7 5D       01220         LD      E,L
42B8          01230 DELNCH  EQU     $
42B8 7B       01240         LD      A,E             ;addr lsb
42B9 FE67     01250         CP      BUFF+79&0FFH    ;check end of buff
42BB 2805     01260         JR      Z,JDSPLC        ;yes, exit to clr screen
42BD 13       01270         INC     DE              ;next source char
```

```
42BE 1A        01280        LD     A,(DE)
42BF 1B        01290        DEC    DE            ;to dest
42C0 12        01300        LD     (DE),A
42C1 13        01310        INC    DE            ;ready for next move
42C2 18F4      01320        JR     DELNCH        ;delete next
42C4           01330 MBUFRG EQU    $
42C4 016843    01340        LD     BC,BUFF+80    ;end addr + 1
42C7 116743    01350        LD     DE,BUFF+79    ;end addr
42CA           01360 MBUFCH EQU    $
42CA 7B        01370        LD     A,E           ;check if at end
42CB BD        01380        CP     L             ;current cursor addr
42CC C8        01390        RET    Z             ;yes, exit
42CD 0B        01400        DEC    BC            ;else back one
42CE 1B        01410        DEC    DE
42CF 1A        01420        LD     A,(DE)
42D0 02        01430        LD     (BC),A
42D1 18F7      01440        JR     MBUFCH        ;repeat move
4200           01450        END    BEGIN
00000 TOTAL ERRORS

BEGIN    4200    BEGLIN  4211    BUFF     4318    DELCHR  42B1    DELNCH  42B8
DSPLB    421B    DSPLC   4214    ENTRKY   429D    INSMDE  4294    JDSPLB  4285
JDSPLC   4292    LINEND  42A9    MBUFCH   42CA    MBUFRG  42C4    MODFLG  4295
MVLFT    4287    MVRGT   427F    RPLCUR   422B    RPLNOC  4235
```

MODEL III VERSION – E/ASM

```
               00100 ;E/ASM - edit dos command line 86-07-10
               00110 ;programmed by arne rohde, box 82-211, auckland, nz
               00120 ;must fit within 1 sector to use DOS sector buffer
               00130 ;for Newdos/80 v2 Model III (for M I change nxt 3 lines)
4225           00140 BUFF   EQU    4225H         ;command buffer(MI=4318)
4300           00150        ORG    4300H         ;dos buffer area(MI=4200)
4300 2AC742    00160 BEGIN  LD     HL,(42C7H)    ;2 chars (MI=43A7)
4303 222542    00170        LD     (BUFF),HL     ;restore first 2 chars
4306 3E0D      00180        LD     A,0DH         ;set last to cr
4308 327442    00190        LD     (BUFF+79),A
430B 212542    00200        LD     HL,BUFF       ;buffer start
430E C3A943    00210        JP     LINEND        ;cursor to line end
4311           00220 BEGLIN EQU    $
4311 212542    00230        LD     HL,BUFF       ;buffer start
4314           00240 DSPLC  EQU    $
4314 CDC901    00250        CALL   01C9H         ;clear screen
4317 DD2A2040  00260        LD     IX,(4020H)    ;cursor pos
431B           00270 DSPLB  EQU    $
431B 4E        00280        LD     C,(HL)        ;buffer char
431C 7D        00290        LD     A,L
431D FE74      00300        CP     BUFF+79&0FFH  ;check at buffer end
431F 2814      00310        JR     Z,RPLNOC      ;yes, no cursor
4321 79        00320        LD     A,C           ;char at cursor
4322 FE0D      00330        CP     0DH           ;check if end of line
4324 380F      00340        JR     C,RPLNOC      ;line feed, no cursor
4326 2003      00350        JR     NZ,RPLCUR     ;no, replace cursor
4328 23        00360        INC    HL            ;to next char
4329 77        00370        LD     (HL),A        ;set to line end
432A 2B        00380        DEC    HL            ;cursor pos back
432B           00390 RPLCUR EQU    $
432B 36BF      00400        LD     (HL),0BFH     ;set cursor
432D 3A9543    00410        LD     A,(MODFLG)    ;check insert mode
4330 B7        00420        OR     A
4331 2802      00430        JR     Z,RPLNOC      ;replace mode
4333 36B0      00440        LD     (HL),0B0H     ;alternative cursor
4335           00450 RPLNOC EQU    $
4335 EB        00460        EX     DE,HL         ;save pos
4336 212542    00470        LD     HL,BUFF       ;start for display
4339 CD6744    00480        CALL   4467H         ;displ line
433C DD222040  00490        LD     (4020H),IX    ;restore cursor pos
4340 EB        00500        EX     DE,HL         ;pos back to hl
4341 71        00510        LD     (HL),C        ;restore char
4342 CD4900    00520        CALL   0049H         ;get key in
4345 FE18      00530        CP     24            ;shift left
4347 28C8      00540        JR     Z,BEGLIN      ;line begin
4349 FE01      00550        CP     1             ;break key
434B CA2D40    00560        JP     Z,402DH       ;return to dos
434E FE09      00570        CP     9             ;right arrow
4350 282D      00580        JR     Z,MVRGT
4352 FE08      00590        CP     8             ;left arrow
4354 2831      00600        JR     Z,MVLFT
4356 FE0A      00610        CP     10            ;down arrow
4358 283A      00620        JR     Z,INSMDE      ;set insert mode
435A FE0D      00630        CP     0DH           ;enter key
435C 283F      00640        JR     Z,ENTRKY
435E FE19      00650        CP     25            ;shift right
4360 2847      00660        JR     Z,LINEND
4362 FE1F      00670        CP     31            ;clear key
4364 284B      00680        JR     Z,DELCHR      ;delete character
4366 FE20      00690        CP     20H           ;check ctrl char
4368 38B1      00700        JR     C,DSPLB       ;yes, ignore
436A 08        00710        EX     AF,AF'        ;save key code
436B 7D        00720        LD     A,L           ;get addr lsb
436C FE74      00730        CP     BUFF+79&0FFH  ;check if end
436E 28AB      00740        JR     Z,DSPLB       ;yes, ignore
4370 7E        00750        LD     A,(HL)        ;get char
4371 FE0D      00760        CP     0DH           ;check current end
4373 CCC443    00770        CALL   Z,MBUFRG      ;yes, move buffer right
4376 3A9543    00780        LD     A,(MODFLG)    ;check insert mode
4379 B7        00790        OR     A
437A C4C443    00800        CALL   NZ,MBUFRG     ;insert character
437D 08        00810        EX     AF,AF'        ;get char again
437E 77        00820        LD     (HL),A        ;insert char
437F           00830 MVRGT  EQU    $
437F 7E        00840        LD     A,(HL)        ;check if end of line
4380 FE0D      00850        CP     0DH
4382 2897      00860        JR     Z,DSPLB       ;at end, no move right
4384 23        00870        INC    HL            ;else incr cursor pos
4385           00880 JDSPLB EQU    $
4385 1894      00890        JR     DSPLB
4387           00900 MVLFT  EQU    $
4387 7D        00910        LD     A,L           ;current pos lsb
4388 FE25      00920        CP     BUFF&0FFH     ;check at beginning
438A 288F      00930        JR     Z,DSPLB       ;yes, exit
438C 7E        00940        LD     A,(HL)        ;char at cursor
438D 2B        00950        DEC    HL            ;else cursor back
438E FE0D      00960        CP     0DH           ;check line end
4390 2089      00970        JR     NZ,DSPLB      ;no, no screen clear
4392           00980 JDSPLC EQU    $
4392 1880      00990        JR     DSPLC         ;else clear screen
4394           01000 INSMDE EQU    $
4394 3E00      01010        LD     A,0           ;current mode flag
4395           01020 MODFLG EQU    $-1
4396 EEFF      01030        XOR    0FFH          ;invert flag
4398 329543    01040        LD     (MODFLG),A
439B 18E8      01050        JR     JDSPLB
439D           01060 ENTRKY EQU    $
439D CDC901    01070        CALL   01C9H         ;clear screen
43A0 212542    01080        LD     HL,BUFF       ;command buffer start
43A3 CD6744    01090        CALL   4467H         ;redisplay
43A6 C30544    01100        JP     4405H         ;execute dos command
43A9           01110 LINEND EQU    $
43A9 3E0D      01120        LD     A,0DH
43AB BE        01130        CP     (HL)          ;find line end
43AC 28E4      01140        JR     Z,JDSPLC      ;found, display new
43AE 23        01150        INC    HL
43AF 18F8      01160        JR     LINEND        ;to next char
43B1           01170 DELCHR EQU    $
43B1 7E        01180        LD     A,(HL)        ;char at cursor
43B2 FE0D      01190        CP     0DH           ;check end of line
43B4 28CF      01200        JR     Z,JDSPLB      ;yes, exit
43B6 54        01210        LD     D,H           ;addr in buffer
43B7 5D        01220        LD     E,L
43B8           01230 DELNCH EQU    $
43B8 7B        01240        LD     A,E           ;addr lsb
43B9 FE74      01250        CP     BUFF+79&0FFH  ;check end of buff
43BB 28D5      01260        JR     Z,JDSPLC      ;yes, exit to clr screen
43BD 13        01270        INC    DE            ;next source char
43BE 1A        01280        LD     A,(DE)
43BF 1B        01290        DEC    DE            ;to dest
43C0 12        01300        LD     (DE),A
43C1 13        01310        INC    DE            ;ready for next move
43C2 18F4      01320        JR     DELNCH        ;delete next
43C4           01330 MBUFRG EQU    $
43C4 017542    01340        LD     BC,BUFF+80    ;end addr + 1
43C7 117442    01350        LD     DE,BUFF+79    ;end addr
43CA           01360 MBUFCH EQU    $
43CA 7B        01370        LD     A,E           ;check if at end
43CB BD        01380        CP     L             ;current cursor addr
43CC C8        01390        RET    Z             ;yes, exit
43CD 0B        01400        DEC    BC            ;else back one
43CE 1B        01410        DEC    DE
43CF 1A        01420        LD     A,(DE)
```

```
43D0 02
43D1 18F7
4300
00000 TOTAL ER

BEGIN    4300
DSPLB    431B
JDSPLC   4392
MVLFT    4387

0420
0016
0002
000A
0001
0018
000F
2600
2600
2600 3E0F
2602 0604
2604 EF
2605 221327
2608 211527
260B 0606
260D 3E0F
260F EF
2610 3E02
2612 0E0F
2614 EF
2615 212104
2618 3620
261A 2B
261B 3620
261D 3E0D
261F 326F04
2622
2622 212004
2625
2625 0E1C
2627 3E02
2629 EF
262A 0E1F
262C 3E02
262E EF
262F
262F 46
2630 7D
2631 FE6F
2633 2814
2635 78
2636 FE0D
2638 380F
263A 2003
263C 23
263D 77
263E 2B
263F
263F 36BF
2641 3AAB26
2644 B7
2645 2802
2647 36B0
2649
2649 3E02
264B 0E1C
264D EF
264E E5
264F 212004
2652 3E0A
2654 EF
2655 E1
2656 70
2657 3E01
2659 EF
```

```
4300 02        01430         LD      (BC),A
43D1 18F7      01440         JR      MBUFCH          ;repeat move
4300           01450         END     BEGIN
00000 TOTAL ERRORS

BEGIN    4300   BEGLIN  4311   BUFF    4225   DELCHR  43B1   DELNCH  43B8
DSPLB    4318   DSPLC   4314   ENTRKY  439D   INSMDE  4394   JDSPLB  4385
JDSPLC   4392   LINEND  43A9   MBUFCH  43CA   MBUFRG  43C4   MODFLG  4395
MVLFT    4387   MVRGT   437F   RPLCUR  432B   RPLNOC  4335
```

MODEL 4 VERSION – R/ASM

```
               00100 ;R/ASM - edit TRSDOS 6.2 command line 86-07-10
               00110 ;programmed by arne rohde, box 82-211, auckland, nz
               00120 ;for TRSDOS 6.2 Model 4 TRS-80
0420           00130 BUFF    EQU     420H            ;command buffer
0016           00140 @EXIT   EQU     16H
0002           00150 @DSP    EQU     02H
000A           00160 @DSPLY  EQU     0AH
0001           00170 @KEY    EQU     01H
0018           00180 @CMNDI  EQU     18H
000F           00190 @VDCTL  EQU     0FH
2600           00200         ORG     2600H           ;dos buffer area
2600           00210 BEGIN   EQU     $
2600 3E0F      00220         LD      A,@VDCTL
2602 0604      00230         LD      B,4             ;get cursor pos
2604 EF        00240         RST     28H
2605 221327    00250         LD      (CURP),HL       ;save pos
2608 211527    00260         LD      HL,SCRN         ;save screen info
260B 0606      00270         LD      B,6             ;screen to mem
260D 3E0F      00280         LD      A,@VDCTL
260F EF        00290         RST     28H
2610 3E02      00300         LD      A,@DSP
2612 0E0F      00310         LD      C,0FH           ;cursor off
2614 EF        00320         RST     28H
2615 212104    00330         LD      HL,BUFF+1       ;second buffer char
2618 3620      00340         LD      (HL),20H        ;first two to spaces
261A 2B        00350         DEC     HL
261B 3620      00360         LD      (HL),20H
261D 3E0D      00370         LD      A,0DH           ;set last to cr
261F 326F04    00380         LD      (BUFF+79),A
2622           00390 BEGLIN  EQU     $
2622 212004    00400         LD      HL,BUFF         ;buffer start
2625           00410 DSPLC   EQU     $
2625 0E1C      00420         LD      C,1CH           ;home cursor
2627 3E02      00430         LD      A,@DSP
2629 EF        00440         RST     28H
262A 0E1F      00450         LD      C,1FH           ;clr screen
262C 3E02      00460         LD      A,@DSP
262E EF        00470         RST     28H
262F           00480 DSPLB   EQU     $
262F 46        00490         LD      B,(HL)          ;buffer char
2630 7D        00500         LD      A,L
2631 FE6F      00510         CP      BUFF+79&0FFH    ;check at buffer end
2633 2814      00520         JR      Z,RPLNOC        ;yes, no cursor
2635 78        00530         LD      A,B             ;char at cursor
2636 FE0D      00540         CP.     0DH             ;check if end of line
2638 380F      00550         JR      C,RPLNOC        ;line feed, no cursor
263A 2003      00560         JR      NZ,RPLCUR       ;no, replace cursor
263C 23        00570         INC     HL              ;to next char
263D 77        00580         LD      (HL),A          ;set to line end
263E 2B        00590         DEC     HL              ;cursor pos back
263F           00600 RPLCUR  EQU     $
263F 36BF      00610         LD      (HL),0BFH       ;set cursor
2641 3AAB26    00620         LD      A,(MODFLG)      ;check insert mode
2644 B7        00630         OR      A
2645 2802      00640         JR      Z,RPLNOC        ;replace mode
2647 3680      00650         LD      (HL),080H       ;alternative cursor
2649           00660 RPLNOC  EQU     $
2649 3E02      00670         LD      A,@DSP
264B 0E1C      00680         LD      C,1CH           ;home cursor
264D EF        00690         RST     28H
264E E5        00700         PUSH    HL              ;save pos
264F 212004    00710         LD      HL,BUFF         ;start for display
2652 3E0A      00720         LD      A,@DSPLY        ;display line
2654 EF        00730         RST     28H
2655 E1        00740         POP     HL              ;pos back to hl
2656 70        00750         LD      (HL),B          ;restore char
2657 3E01      00760         LD      A,@KEY          ;wait for key
2659 EF        00770         RST     28H
265A FE18      00780         CP      24              ;shift left
265C 28C4      00790         JR      Z,BEGLIN        ;line begin
265E FE80      00800         CP      80H             ;break key
2660 CA0A27    00810         JP      Z,RETDS         ;return to dos
2663 FE09      00820         CP      9               ;right arrow
2665 282D      00830         JR      Z,MVRGT
2667 FE08      00840         CP      8               ;left arrow
2669 2831      00850         JR      Z,MVLFT
266B FE0A      00860         CP      10              ;down arrow
266D 283B      00870         JR      Z,INSMDE        ;set insert mode
266F FE0D      00880         CP      0DH             ;enter key
2671 2840      00890         JR      Z,ENTRKY
2673 FE19      00900         CP      25              ;shift right
2675 2853      00910         JR      Z,LINEND
2677 FE1F      00920         CP      31              ;shift clear key
2679 2857      00930         JR      Z,DELCHR        ;delete character
267B FE20      00940         CP      20H             ;check ctrl char
267D 38B0      00950         JR      C,DSPLB         ;yes, ignore
267F 08        00960         EX      AF,AF'          ;save key code
2680 7D        00970         LD      A,L             ;get addr lsb
2681 FE6F      00980         CP      BUFF+79&0FFH    ;check if end
2683 28AA      00990         JR      Z,DSPLB         ;yes, ignore
2685 7E        01000         LD      A,(HL)          ;get char
2686 FE0D      01010         CP      0DH             ;check current end
2688 CCE526    01020         CALL    Z,MBUFRG        ;yes, move buffer right
268B 3AAB26    01030         LD      A,(MODFLG)      ;check insert mode
268E B7        01040         OR      A
268F C4E526    01050         CALL    NZ,MBUFRG       ;insert character
2692 08        01060         EX      AF,AF'          ;get char again
2693 77        01070         LD      (HL),A          ;insert char
2694           01080 MVRGT   EQU     $
2694 7E        01090         LD      A,(HL)          ;check if end of line
2695 FE0D      01100         CP      0DH
2697 2896      01110         JR      Z,DSPLB         ;at end, no move right
2699 23        01120         INC     HL              ;else incr cursor pos
269A           01130 JDSPLB  EQU     $
269A 1893      01140         JR      DSPLB
269C           01150 MVLFT   EQU     $
269C 7D        01160         LD      A,L             ;current pos lsb
269D FE20      01170         CP      BUFF&0FFH       ;check at beginning
269F 288E      01180         JR      Z,DSPLB         ;yes, exit
26A1 7E        01190         LD      A,(HL)          ;char at cursor
26A2 2B        01200         DEC     HL              ;else cursor back
26A3 FE0D      01210         CP      0DH             ;check line end
26A5 2088      01220         JR      NZ,DSPLB        ;no, no screen clear
26A7           01230 JDSPLC  EQU     $
26A7 C32526    01240         JP      DSPLC           ;else clear screen
26AA           01250 INSMDE  EQU     $
26AA 3E00      01260         LD      A,0             ;current mode flag
26AB           01270 MODFLG  EQU     $-1
26AC EEFF      01280         XOR     0FFH            ;invert flag
26AE 32AB26    01290         LD      (MODFLG),A
26B1 18E7      01300         JR      JDSPLB
26B3           01310 ENTRKY  EQU     $
26B3 CDF426    01320         CALL    RSTSCR          ;restore screen
26B6 212004    01330         LD      HL,BUFF         ;command buffer start
26B9 3E0A      01340         LD      A,@DSPLY
26BB EF        01350         RST     28H             ;display line
26BC 111527    01360         LD      DE,SCRN         ;save command line
26BF 015000    01370         LD      BC,80           ;max 80 chars
26C2 EDB0      01380         LDIR
26C4 211527    01390         LD      HL,SCRN         ;command line
26C7 3E18      01400         LD      A,@CMNDI
26C9 EF        01410         RST     28H
26CA           01420 LINEND  EQU     $
26CA 3E0D      01430         LD      A,0DH
26CC BE        01440         CP      (HL)            ;find line end
26CD 28D8      01450         JR      Z,JDSPLC        ;found, display new
26CF 23        01460         INC     HL
26D0 18F8      01470         JR      LINEND          ;to next char
26D2           01480 DELCHR  EQU     $
26D2 7E        01490         LD      A,(HL)          ;char at cursor
26D3 FE0D      01500         CP      0DH             ;check end of line
26D5 28C3      01510         JR      Z,JDSPLB        ;yes, exit
26D7 54        01520         LD      D,H             ;addr in buffer
26D8 5D        01530         LD      E,L
26D9           01540 DELNCH  EQU     $
26D9 7B        01550         LD      A,E             ;addr lsb
26DA FE6F      01560         CP      BUFF+79&0FFH    ;check end of buff
26DC 28C9      01570         JR      Z,JDSPLC        ;yes, exit to clr screen
```

```
26DE 13       01580         INC    DE              ;next source char
26DF 1A       01590         LD     A,(DE)
26E0 1B       01600         DEC    DE              ;to dest
26E1 12       01610         LD     (DE),A
26E2 13       01620         INC    DE              ;ready for next move
26E3 18F4     01630         JR     DELNCH          ;delete next
26E5          01640 MBUFRG  EQU    $
26E5 017004   01650         LD     BC,BUFF+80      ;end addr + 1
26E8 116F04   01660         LD     DE,BUFF+79      ;end addr
26EB          01670 MBUFCH  EQU    $
26EB 7B       01680         LD     A,E             ;check if at end
26EC BD       01690         CP     L               ;current cursor addr
26ED C8       01700         RET    Z               ;yes, exit
26EE 0B       01710         DEC    BC              ;else back one
26EF 1B       01720         DEC    DE
26F0 1A       01730         LD     A,(DE)
26F1 02       01740         LD     (BC),A
26F2 18F7     01750         JR     MBUFCH          ;repeat move
26F4          01760 RSTSCR  EQU    $
26F4 211527   01770         LD     HL,SCRN         ;saved screen
26F7 0605     01780         LD     B,5             ;back to screen
26F9 3E0F     01790         LD     A,@VDCTL
26FB EF       01800         RST    28H
26FC 0603     01810         LD     B,3             ;cursor pos set
26FE 2A1327   01820         LD     HL,(CURP)
2701 3E0F     01830         LD     A,@VDCTL
2703 EF       01840         RST    28H
2704 3E02     01850         LD     A,@DSP
2706 0E0E     01860         LD     C,0EH           ;cursor on
2708 EF       01870         RST    28H
2709 C9       01880         RET
270A          01890 RETDS   EQU    $
270A CDF426   01900         CALL   RSTSCR          ;restore screen
270D 3E16     01910         LD     A,@EXIT
270F 210000   01920         LD     HL,0
2712 EF       01930         RST    28H
2713 0000     01940 CURP    DEFW   0               ;save cursor pos
0800          01950 SCRN    DEFS   2048            ;screen data
2600          01960         END    BEGIN
00000 TOTAL ERRORS

@CMNDI  0018   @DSP    0002   @DSPLY  000A   @EXIT   0016   @KEY    0001
@VDCTL  000F   BEGIN   2600   BEGLIN  2622   BUFF    0420   CURP    2713
DELCHR  26D2   DELNCH  26D9   DSPLB   262F   DSPLC   2625   ENTRKY  2683
INSMDE  26AA   JDSPLB  269A   JDSPLC  26A7   LINEND  26CA   MBUFCH  26EB
MBUFRG  26E5   MODFLG  26AB   MVLFT   269C   MVRGT   2694   RETDS   270A
RPLCUR  263F   RPLNOC  2649   RSTSCR  26F4   SCRN    2715
```

BACKING UP "HITCHHIKER'S GUIDE"

by Alan Morrison 011+61+2+625-5869

[Excerpted from the column <u>ALAN'S SECTOR #5</u>, which appeared in SYDTRUG NEWS (P.O. Box 297, Padstow, New South Wales 2211, Australia):]

Hi hackers, welcome to another month of info and ideas for you to try. First up, we'll take a look at the new Infocom rage "Hitchhikers Guide to the Galaxy". Infocom has outdone themselves once more in another action-packed adventure, this time starting on Earth and ending up on the "Heart of Gold", a stolen space-craft.

When making a backup copy of this work of art and you decide to do so on another DOS, a certain password is lost in the process and the message "INSERT DISK #2 AND PRESS <ENTER>" appears. The user then presses <ENTER>, the drives whirr and lights flash, smoke pours from the screen, terrible screaming noises are heard from the CPU and then.... the question appears again. If the user is very persistent, the message will appear until the end of time (or until your system carks it!).

This is all over a little password! This little password does not get backed up along with the file "HITCHHIK/CMD" so you must put it back there. You see, the computer is looking for "HITCHHIK/CMD.password" and not "HITCHHIK/CMD". There is a difference. To correct all of this you must load your Hitchhikers Guide to the Galaxy disk in drive 1, your current system disk into drive 0 and DON'T PANIC!! Then you type this:

```
ATTRIB HITCHHIK/CMD:1 (ACC=,UPD=SMC)<ENTER>
```

You have now protected this file with a password, thus changing its filespec and it should therefore execute.

(STILL) MORE ON DON McKENZIE'S PBUFF KIT

by Darrel Hegarty 011+61+2+560-9681

[This information is excerpted from the SECRETARY'S SAYINGS column of SYDTRUG NEWS, P.O. Box 297, Padstow, New South Wales 2211, Australia:]

... I have also had another letter from Tony Briggs in the Land of the Long White Cloud [New Zealand], and ... he has also given some information on the PBUFF kits. Those of you who have the Revision 3 board or earlier, AND have a Gemini Star printer, may notice that the printer goes "OFFLINE", i.e. DESELECTS, when the PBUFF is reset. A 470 ohm resister from the Z80 address line A13 to ground should cure this problem. The best place to put this resistor is at the Z80 itself - from pin 3 to ground. Don McKenzie has added this resistor on the Revision D board as a 1k resistor, but Tony Briggs has found that 1k is NOT low enough to cure the problem, hence the 470 ohm value.

There is also a method to bypass the PBUFF without turning it off and unplugging - cut the track to Z80/16 (INT*), and solder a 4.7k resistor to +5 volts, and a switch or push-button to ground. When the bypass option is needed, simply RESET the PBUFF, and press the BYPASS switch or button. PBUFF will then go into a continuous loop and pass all incoming data straight out to the printer. To regain normal operation, press RESET, or switch off and back on again.

The option to load PBUFF offline, may not work with some printers (Gemini Star is one) - if so, leave wire 12 connected as normal, and use wire 13 instead.

A letter from Don McKenzie has bought up a modification to facilitate a HARDWARE pause (such may be required if you have the printer sharer board, whilst changing printers), simply put a single-pole switch in series with the BUSY line of the OUTPUT port (this line goes to pin 11 of the 8255). When this switch is open, the buffer will stop sending to the printer. Make sure that you cut the track such that R6 is still connected to pin 11 of the 8255.

[NORTHERN BYTES Editor's Note: I have just received a new price list from Don McKenzie that has special prices for U.S.A. residents. These are for "short form" kits, which means that you get the printed circuit board and any EPROMs required for the project (with the code already burned in, of course). You then purchase the remainder of the parts from a local (or mail-order) supplier, thus saving even more money (since the parts are not being sent from the U.S.A. to Australia and back again). The PBUFF short form printer buffer kit (which includes a parallel I/O board) costs $26.00. Add-on boards that can connect to the PBUFF board include a serial I/O board (this board also permits serial/parallel or parallel/serial conversions if necessary) at $12.00, a P2C1 printer sharer board (lets you connect two printers to one computer, but if a serial board is used only <u>one</u> of the printers may be serial) at $8.00, and a C2P1 computer sharer board (lets you connect two computers to one printer) at $8.00. All prices are in <u>U.S. dollars</u> and you should add $5.00 for postage and handling per order. This seems like an inexpensive way for a hardware enthusiast to solve any computer/printer connection problems you may have! For more information contact Don McKenzie, 29 Ellesmere Crescent, Tullamarine, Victoria 3043, Australia (you may also telephone 011+61+3+338-6286 for more information).]

[redacted]

This patch disables password checking under TRSDOS 6.2 (from Bert's Ramblings, newsletter of the Tandy Hobart Users' Group):

```
PATCH SYS2/SYS.LSIDOS (D02,33=18:F02,33=28)
```

The patch has not been tested under TRSDOS 6.2.1.

The second patch is in the form of a /JCL file and is self-explanatory:

```
. Patch to SYS6/SYS.LSIDOS to default DEVICE (B=Y)
. For TRSDOS 6.2.1
L61
X'2580'=FF FF
. End of Patch
```

MODEL 4 VIVACE WANTED

A Model 4 user in the Netherlands wants to obtain a copy of the Model 4 Version of Vivace (the BASIC compiler from WittSoft). Since WittSoft no longer markets this product, a pre-owned copy would be about the only other possible option. So, if you have a new or used copy of the <u>Model 4 version</u> of Vivace for sale, please contact P.J. Plomp, Linnaeusparkweg 59, Amsterdam, Holland.

SOFTWARE WRITE PROTECT FOR DOSPLUS 4

by Dick Rechlicz
P.O. Box 403, Brookfield, Wisconsin 53005

[Editor's note: In NORTHERN BYTES Volume 7, Number 3 (page 36) we published an article by Dick Rechlicz entitled "MODEL 4 TRSDOS 6 WRITE PROTECT". In this article, Dick stated that the procedure to software write-protect drive zero as a default under DOSPLUS 4 is relatively simple, but didn't really elaborate. Bill Baker of Independence, Missouri wrote to ask what the procedure is, I forwarded the letter to Dick, and the letters started flying. Anyway, here are Dick's instructions, as extracted and edited from a couple rounds of correspondence:]

Here is the information on write protecting Drive :0 using DOSPLUS 3.5 and 4.1.

Our computers are used in an office environment, and are nothing more than tools which permit us an easier and more rapid processing of our communications and accounting programs. It is our belief that computer operators in a commercial/industrial environment need to be no more knowledgeable about the computer than they are expected to be about a typewriter.

Since we've evolved from the Model I through the Model III to the Model 4, <u>many</u> of our programs are RUN in the Model III mode....others were developed for the Model 4 or converted. When DOSPLUS 3.5 became available, we began to write protect Drive :0.

Preventive maintenance treats operator error (whenever possible). A write protect tab removed from a diskette is not always replaced. Where and when feasible, this instance of possible error can be corrected by write protecting the drive through software.

Additionally, under some circumstances, write protect tabs may not offer the protection their use assumes will be furnished. The first letter to Bill Baker was saved to a diskette in our Model 4's Drive :0 which had a <u>black</u> write protect tab installed. We can - and do - kill and save files on this drive, supposedly safeguarded by the <u>black</u> write protect tab. Tabs of other colors force the Model 4 to behave properly (drives 1, 2, and 3 operate appropriately regardless of the color of the tab).

The article published in Northern Bytes detailed the procedures to software write-protect Drive :0 using TRSDOS 6. DOSPLUS 4 requires a different approach.

The CONFIG feature of DOSPLUS provides the user with the method to "configure" the system. It, in fact, describes and establishes the protocols for the drives to the operating system (DOSPLUS) so that the computer can function properly.

Our Model 4 has four drives. We inform DOSPLUS 4 of the additional two drives - that they have a 30 millisecond step rates by typing:

```
CONFIG :2 (Step=3)
CONFIG :3 (Step=3)        (See Page 2-35)
```

Additional commands might be:

```
CONFIG :0 (Write Protect=Y)      or, (WP=Y)
```

A problem arises at this point, however. When the computer is turned off, the "CONFIG" instructions are lost. The "SYSTEM" command can "SAVE" them for you (they would become the default parameters upon BOOT up) <u>excepting</u> that you cannot save to a write protected disk! (Or to a software write protected disk drive) (See Pages 2-129 to 2-136).

The answer is simple. "Save" the configuration instructions to a non-write protected drive. Here is an example to help clarify it for you:

```
CONFIG :2 (STEP=3)
CONFIG :3 (STEP=3)
SYSTEM (T=N, E=Y, CL=Y, SA=Y)
CONFIG :0 (WP=Y)
SYSTEM STEP1/CFG:1
CONFIG :0 (WP=N)
COPY STEP 1/CFG:1 :0
AUTO STEP1/CFG
RE-BOOT the system

Remove or kill STEP1/CFG on Drive :1
```

The first two CONFIG statements establish STEP rates for Drives :2 and :3. "SYSTEM" commands turn the time (T) prompt off; turn the beep (E) and the click (CL) on; and "Save" (SA) these "SYSTEM" configurations to disk. Use the combination of "SYSTEM" defaults that you prefer, but be sure to include the "SA=Y".

Next, the CONFIG command to apply the software write protect status is entered, and becomes effective immediately. At that point, nothing can be "saved" to that drive (Drive :0). Neither a file nor the desired defaults established by the "SYSTEM" command or the CONFIG commands can be saved to any but an un-write protected disk.

Therefore, the final "SYSTEM" command saves <u>all</u> configuration instruction to a user-named file on Drive <u>:1</u> (See Page 2-134) <u>Our</u> filename is STEP1/CFG. You can give this file any name you wish to.

Immediately after you "save" your configurations with the SYSTEM STEP1/CFG:1 command, you must issue a command to un-write protect Drive :0, and then copy the STEP1/CFG file from Drive :1 to Drive :0.

The AUTO command executes this STEP1/CFG file when next the computer is BOOTED. This command then follows any further instruction. In our case, we use the statement

```
AUTO STEP1/CFG; FAST/CMD; DO L
```

which turns on the high speed clock in the Model III mode (FAST/CMD) then loads Lazy Writer (L), our word processing program.

If you choose not to AUTO the CONFIG file, you can have a different CONFIG file for each user of the system if you wish, that can be different from the "standard" STEP1/CFG file. You need only type the name of the desired CONFIG file to install the configuration contained therein. If you are using a data base such as PROFILE that requires that Drive :0 not be software write protected, you can simply not use STEP1/CFG.

Should you need to revise or add a program to Drive :0, type:

```
CONFIG :0 (WP=N) <ENTER>
```

When completed, type:

```
CONFIG :0 (WP=Y) <ENTER>
```

We have used DOSPLUS ever since it became available for the Model I. In our opinion, it was (and is) a most superior DOS for the TRS-80 Models I, III, and 4. However, we became aware of new programs for the Model 4 which were developed specifically for use with TRSDOS 6. Their numbers became greater. It was our decision, reluctantly, to transfer our base to the TRSDOS 6.

We've not been dissatisfied. TRSDOS 6 is a most powerful DOS, with features which place it in a premier position. We no longer know if some of the newer programs work with other DOSes....they <u>do</u> work well with TRSDOS 6.

By the way, we've added the DOSPLUS BASIC shorthand overlays to TRSDOS 6 for an operating system of even greater excellence. In the Model 4 mode, the SYSTEM files are in MEMDISK which becomes :0, and which is software write protected. No SYSTEM files reside in any of the four physical drives (1, 2, 3, and 4).

We <u>like</u> our Model 4 and TRSDOS 6! Though a great many of our applications are now on MS-DOS based equipment (Tandy 1000 and IBM PC-XTS) we look forward to using the Model 4!

Maybe the future belong to the XTs and ATs, but there'll be a Model 4 in our future for a long, long time.

[Editor's note: Both I and Bill Baker were initially skeptical of Dick's comments about writing to a disk that had a write protect tab on it (I even stated my doubts in an editorial comment inserted into Dick's previous article). Then Bill came across some more information (in RAMparts Volume 4, Number 5) that suggests that perhaps there is more to this than just an oddball hardware problem. Here are excerpts from Bill's article on the subject (which in turn contains excerpts from the RAMparts article), which appeared in <u>The Cursor</u> (newsletter of the Kansas City South Computer Club):

"...The first report of anyone accidentally writing to a disk protected with a write protect tab was noted in an article by Dick Rechlicz that appeared in Northern Bytes Volume 7, Number 3. Dick reported that he had lost some data on disks protected with the common sticky write protect tabs on the disks. I had occasion to correspond with Dick on another matter and he again mentioned that he had accidentally written to disks protected with black tabs but never with silver, gold or white.

"I brushed his statement off at the time as a fluke, impossible to occur, at least by any standards that I knew of then, but after reading RAMparts, I now believe Dick was on to a new, bewildering problem. Have drives become color conscious? May sound silly, but read on. It may be just that.

"In the June 24th edition of PC WEEK, the alleged problem appeared again when Peter Norton, who wrote the report, told RAMparts columnist Terry Kepner that he had discovered the 'problem' when he accidentally wrote onto write protected disks with the COMPAC 286 DESKPROS in his office. These units were equipped with MITSUBISHI D.S.D.D. drives built to meet IBM standards.

"Norton reported that he suspected the trouble was caused by a change in the design of the write protect system -- going from the industry standard 'transmissive' system, to a rather strange 'reflective' system.

"In drives using the 'transmissive' detector, an infrared emitter sits on one side of the disk's write protect notch, and on the other side is an IR photo-transistor. If the notch is uncovered, the IR LED illuminated the photo-transistor, telling the drive it is okay to write to the disk. When the IR photo-transistor CANNOT be illuminated (because the notch is either not there or is covered by tape), the drive assumes the floppy is protected..."

It seems like I recall reading a report of a major diskette manufacturer who decided to include some snazzy red translucent write protect tabs with their diskettes. As you can imagine, the infrared light passed right on through, and thousands of boxes of diskettes had to be recalled, so even the "transmissive" detector may not be foolproof if you use a translucent or transparent write protect tab! Anyway, back to Bill's explanation:

"...But the 'reflective' system reverses the principle. Both the IR emitter and the photo-transistor are on the same side of the disk, and the drives rely on a reflected signal to tell when the drive is protected. When the photo-transistor IS illuminated, the drive assumes that the disk is protected. The problem is that some protected disks are manufactured without a notch, and the disk's textured black surface won't reflect the IR signal. Black write protect tabs, commonly used to cover the notch, would also fail to bounce the IR light back to the photo-transistor.

"Mitsubishi engineers are denying they use a reflective notch detector and COMPAC personnel say they are investigating the report. Meanwhile, we now have two reports of the same condition from different parts of the country, so something may be amiss somewhere. The best advice to owners of new drives may well be 'when in doubt, check it out'. Try to write to a disk protected with a black covered notch. If a write is allowed, use gold or silver reflective foil tabs over the notch. If that fixes the trouble, you may have one of the problem drives."

Well... it appears that I owe Dick an apology for my initial skepticism. Whether the reflective system is a good or bad thing depends, of course, on whether it will properly recognize the write protect tab on a TRSDOS 6/LDOS master disk. If it does, it's (presumably) great, if not, it's another "fiasco"! (Sorry about that, I just couldn't resist!) -Jack]

IBM AND THE CUSTOMER DRESS CODE

by Nick Baran

[With all the recent publicity about the Tandy Computer Center dress code (even though Tandy certainly has a right to enforce these regulations, I must say that I would not want to work for Tandy because of the necktie (noose?!) requirement, if nothing else), I thought you might be interested in this view from the other side. At least Tandy enforces the dress code on their salesmen, and not on their customers... yet...

This article is reprinted from the CSRA Computer Society, Inc. Newsletter and they got it from SF Blue Notes, April 1986:]

It's a well-known "fact" that to succeed in the business world, you have to dress appropriately -- dress for success. You may not know, however, that you also have to dress appropriately to be a successful customer of IBM's.

A case in point. I used to work downtown and always wore a coat and tie. In fact, I bought my PC at the IBM Product Center back in 1982. I'd drop by periodically to see what's new and was always treated with respect and the utmost attention. Salespersons would eagerly hand me their card, give me a demo of the latest IBM printer, and would exchange niceties about the weather or the 49ers.

Now, I am working for myself as a consultant and writer, and I don't wear a coat and tie every day. I often wear blue jeans, no necktie, and even my shoes are sometimes a little scruffy. Nevertheless, IBM still has me on their customer list and sent me a "Personalized Validation Form" a while back which made me eligible for a grand prize of an IBM PC AT "Super System" or some other prizes such as a Proprinter or an IBM PC Cross Pen.

All I had to do was go down to the Product Center and see if the number on my personalized form matched the numbers posted at the Product Center. If there was a match, I would win a prize. If not, IBM would give me a free LCD digital stopwatch if I sat through a demonstration of the PC AT and sent my validated form to Rochester, New York.

This all sounded pretty good. I figured, odds are I won't win a prize but at least I'll get a demo of the AT and a free stopwatch. On the form it said that the demo "will show you how the advanced technology in this IBM system can raise high performance -- and your productivity -- to new heights." Why not? I hadn't seen an AT in action anyway, and this way, I could get a free watch in the bargain.

So I headed down to the San Francisco IBM Product Center. I had on a pair of blue jeans, a vest sweater, a "flight jacket", and a pair of beat up old loafers. Aside from the shoes, I looked perfectly respectable, although I clearly was not a VP at Crocker Bank. But I've seen bankers with beat up loafers before.

I walked into the Product Center. All the salespeople were busy. Aha! There was the list of Prize Numbers posted prominently on a wall. I hurried over, and, to my enormous disappointment, found that none of the numbers matched mine.

OK, no prize, not even a Cross Pen. The next best thing was the AT demo and the forthcoming free LCD digital stopwatch! I walked up to the cashier counter, showed the woman my Personalized Validation Form, and politely asked if I could have an AT demo. She was very nice, and hailed a passing salesman and asked him if he could give me a demo. No, he was very busy with a client. The woman at the cashier counter turned to help another customer. I stood there for about five minutes and finally decided to go sit down at the AT on display. Maybe that would give off a signal that I want my AT demo.

I sat down, and after several minutes, a woman walked by and asked if I'd been helped. I said no, I was waiting for an AT demo. She said she wasn't qualified to give demos but would find someone who was and walked off.

A few minutes later, a salesman wearing a vest two sizes too small came rushing by, saying he would be with me in a moment. Fine. Then I saw him conferring with another salesman and they were gesturing in my direction. Finally, one of the salesmen, but not the one with the vest, walked over and introduced himself. I was seated next to the AT and there was a chair next to mine right in front of the keyboard. I was expecting the salesman to sit down and proceed with the demo. Instead, he remained standing, glanced down at my shoes and said, "Are you interested in buying an AT?"

I answered: "Well, I have a PC and was interested in seeing the AT demo."

Salesman: "What do you want to know about the AT?"

I thought this was a demo! He was waiting for me to say something. So I asked, feigning ignorance: "What's different about the AT?"

Salesman: "It's faster and it can store more data."

Silence. I realized it was my turn to say something. This was very awkward and was not helping my productivity reach new heights.

I said: "The keyboard is different, isn't it?"

Salesman: "Yeah, now you have to learn typing all over again, just when you'd gotten used to the backslash key!"

Another silence. This time, he didn't wait for me to say something.

Salesman: "Well! Let me get you that validation sticker!"

The salesman headed for the cashier's counter. I followed obediently. He gave me the validation sticker and I said, "Thank you very much", being a polite sort of fellow. He said, "No problem!"

I was relieved that I hadn't caused the salesman too much trouble. I dropped my validation card in the mailbox. Fortunately, they won't see my shoes in Rochester, New York.

A COLLECTION OF NEWDOS/80 ZAPS
Compiled by Bob Seaborn

Note: In the following zaps, the notations (I) and (III) indicate that the File Relative Sector and Byte given are for the Models I and III versions of NEWDOS/80, respectively. Note that not all of the zaps are applicable to both Models. The final five zaps (1045 through 1049) were added after the original article was received (zaps 1048 & 1049 are by K. Hemstra of Oosthem, Holland).

Zap #1001 - This zap converts the slash (/) and colon (:) in the DATE/TIME input at DOS powerup to periods. This allows easier use of the numeric keypad. Slashes and colons are required and displayed at all other points in the system.

```
SYS0/SYS,12,62 (I)      from     3E2F
        12,25 (III)     to       3E2E
SYS0/SYS,12,75 (I)      from       3E 3A
        12,39 (III)     to         3E 2E
SYS0/SYS,13,88 (I)      from     4D4D 2F44 442F 5959
        13,4E (III)     to       4D4D 2E44 442E 5959
SYS0/SYS,13,98 (I)      from       48 483A 4D4D 3A53 53
        13,61 (III)     to         48 482E 4D4D 2E53 53
```

Zap #1002 - This zap allows the time to be bypassed with <ENTER> thus inserting either 00:00:00 or to the previous value. Caution must be exercised as there is no error checking as to proper value.

```
SYS0/SYS,12,78 (I)      from       4F 20ED C9
        12,20 (III)     to         4F 0000 C9
```

Zap #1003 - This zap forces only the date to be required on re-boot, skipping the TIME prompt, thus setting time to either 00:00:00 or to the previous value.

```
SYS0/SYS,12,68 (I)      from     ED21 76
        12,20 (III)     to       EDC9 76
```

Zap #1004 - This zap allows <ENTER> to default the DATE prompt to either the previous value or to 00/00/00. Caution must be exercised to ensure that valid values are entered as there is no error checking performed.

```
SYS0/SYS,12,65 (I)      from     4F20 ED
        12,29 (III)     to       4FC9 ED
```

Zap #1005 - This zap eliminates the BASIC banner when BASIC is first entered.

```
BASIC/CMD,14,14 (I)     from     CD67 44
BASIC/CMD,13,ED (III)   to       CDC9 01
```

Zap #1006 - This zap eliminates the DOS banner upon re-boot.

```
SYS0/SYS,11,E5 (I)      from      CD 6744
        11,AD (III)     to        00 0000
```

Zap #1007 - This zap eliminates the TIME/DATE display on re-boot.

```
SYS0/SYS,12,0B (I)      from      CD 6744
        11,CF (III)     to        00 0000
```

Zap #1008 - This zap causes the AUTO command not to be displayed when the system disk is booted.

```
SYS0/SYS,12,50 (I)      from     CD67 44
        12,13 (III)     to       0000 00
```

Zap #1009 - This zap enables the "-ƏX" function in SofTrend's BREV-T package to be valid upon re-boot or powerup.

```
SYS0/SYS,12,0B (I)      from      CD 6744
                        or        00 0000
                        to        CD B050
SYS0/SYS,13,CD (I)      from      all zeros
                        to        F5 3E00 32F0 41F1 C900 00
                        or        F5 3E00 32F0 41F1 C367 44
```

Zap #1010 - This zap eliminates the double line feed at DOS READY.

```
SYS1/SYS,00,EF (I)      from      3E 0DCD 3300
SYS1/SYS,04,1D (III)    to        00 0000 0000
```

Zap #1011 - This zap sets the default number of files when entering BASIC.

```
BASIC/CMD,14,46 (I)     from     3E 03 32
BASIC/CMD,14,1F (III)   to       3E yy 32
```

The value used to replace "yy" should be the default number of files to be allocated when BASIC is invoked.

Zap #1012 - This zap enables more than the 15 files that Radio Shack and Apparat allow.

```
BASIC/CMD,16,66 (I)     from     FE 10 30
BASIC/CMD,16,3F (III)   to       FE xx 30
```

The value used to replace "xx" must be one more than the maximum number of files required and cannot be greater than 81H in a 48K system and, for all practical purposes should not be greater than 64H, because it restricts programming space.

Zap #1013 - This zap changes DOS to display the current time setting under the NEWDOS/80 READY prompt when the prompt appears. It does not update the time in this location.

```
SYS1/SYS,00,F6 (I)      from     CC67 4421
                        to       CCC8 5121
SYS1/SYS,04,DC (I)      from     all zeros
                        to                                  CD67 44D9
                                 2157 433E 0D32 5F43 E5CD 6D44 E1CD 6744
                                 D9C9
```

Zap #1014 - When the Model I is equipped with a Percom type double density adaptor and the last disk access was to a double density diskette the system will not re-boot properly on a HALT or JP 0000H instruction. Since this is the way Apparat has chosen to reset the computer under two conditions the following zaps were constructed. They will allow proper reset to occur under double density from the BOOT command or from the "Fatal DOS Error. Key <R> to Reset" error message.

```
SYS9/SYS,00,CF (I)      from       02 CD60 0076
                        to         02 C3D7 5176
SYS9/SYS,04,EA (I)      from     all zeros
                        to       00CD 6000 21EC 3736 FE76
```

Zap #1015 - This zap allows SUPERZAP to exit to DIRCHECK if a SHIFT key is pressed as the T in EXIT is entered.

```
SUPERZAP/CMD,04,D4      from     C32D 4047  to  C3E2 6F47
SUPERZAP/CMD,28,F7      from     all zeros
                        to       003A 8038 FE00 CA2D 40
SUPERZAP/CMD,29,00      from     all zeros
                        to       21F0 6FC3 0544 4449 5243 4845 434B 2F43
                                 4D44 0D00
```

Zap #1016 - This zap allows DIRCHECK to exit to SUPERZAP if a SHIFT key is pressed as the N is entered at the main menu.

```
DIRCHECK/CMD,07,10      from     CA2D 40FE  to  CAF7 60FE
DIRCHECK/CMD,14,E0      from     all zeros
                        to       003A 8038 FE00 CA2D 4021 0561 C305 4453
                                 5550 4552 5A41 502F 434D 4400
```

Zap #1017 - This zap allows changing PDRIVE parameters without writing to disk. Syntax - Same as using parameter "A" except use "B". Example "PDRIVE,0,1=7,B". This would set PDRIVE for drive 1 to equal 7 until a reset, or another PDRIVE command with either "A" or "B" is called. It will not change the PDRIVE table on the disk, only that in memory.

```
SYS16/SYS,02,F7 (I)     from       7E FE41 1220 0623
                        to         7E C3C0 5100 0023
SYS16/SYS,02,E8 (III)   from     7EFE 4112 20
                        to       7EC3 CB51 20
SYS16/SYS,04,D3 (I)     from     all zeros
                        to            00 FE41 1220 03C3 F14F FE42 C2F7
                                 4F3E 4112 77C5 E501 0007 21E2 4071 2310
                                 FCE1 C1C3 F14F 00
SYS16/SYS,04,DF (III)   from     all zeros
                        to                                            FE
                                 4112 2003 C3E0 4F3D FE41 20F8 C5E5 0100
                                 0721 D34D 7123 10FC E1C1 18E3
```

In SUPERZAP/CMD, DEBUG, and the JKL screen print module non-displayable or non-printable characters are changed to periods (2EH). In the first two items, a space (20H) is defined as non-displayable or non-printable. As well, in the JKL module, graphics are changed to periods (2EH), if the SYSTEM parameter AX indicates that the printer is not capable of printing graphics. Some printers, (i.e. Radio Shack LP4 - Centronics 737) will print very limited graphics for certain ASCII codes. The following zaps allow you to define the character to be displayed or printed instead of the period, if a replacement is to take place. Also given, is the byte to zap to change the lowest displayable or printable character as well as the high value of the displayable characters in SUPERZAP/CMD and DEBUG.

Zap #1018 - This is a zap to SUPERZAP/CMD to change the character displayed when a non-displayable character is encountered.

```
SUPERZAP/CMD,12,DF (I)   from     3E 2E FE
SUPERZAP/CMD,12,E5 (III) to       3E xx FE
```

(replacing "xx" with the Hex value of the character to be displayed.)

Zap #1019 - This is a zap to SUPERZAP/CMD to set the high value of the displayable characters (stay under "DF"H).

```
SUPERZAP/CMD,12,D4 (I)   from     0E 5F 1A
```

```
SUPERZAP/CMD,12,D0 (III) to        3E xx FE
```

(replacing "xx" with the Hex value of the highest displayable character.)

Zap #1020 - This is a zap to SUPERZAP/CMD to change the character printed when a non-printable character is encountered.

```
SUPERZAP/CMD,16,DC (I)   from      3E 2E FE
SUPERZAP/CMD,16,E5 (III) to        3E xx FE
```

(replacing "xx" with the Hex value of the character to be printed.)

Zap #1021 - This is a zap to SUPERZAP/CMD to allow the space (20H) to be displayed as such and not changed to a period (2EH).

```
SUPERZAP/CMD,12,D7 (I)   from      D6 21 B9
SUPERZAP/CMD,12,E0 (III) to        D6 20 B9
SUPERZAP/CMD,16,DE (I)   from      FE 20 30
SUPERZAP/CMD,16,E7 (III) to        FE 1F 30
```

Zap #1022 - This is a zap to SYS5/SYS (DEBUG) to change the character displayed when a non-displayable character is encountered.

```
SYS5/SYS,03,75               from      3E 2E E3
                             to        3E xx E3
```

(replacing "xx" with the Hex value of the character to be displayed.)

Zap #1023 - This is a zap to SYS5/SYS (DEBUG) to allow the space (20H) to be displayed as such and not changed to a period (2EH).

```
SYS5/SYS,03,6D               from      D6 21 B9
                             to        D6 20 B9
```

Zap #1024 - This is a zap to SYS5/SYS to set the high value of the displayable characters (stay under "BF"H).

```
SYS5/SYS,03,69 (I)           from      C6 3F 4A
                             to        00 0E xx
```

(replacing "xx" with the Hex value of the highest displayable character.)

Zap #1025 - This is a zap to SYS3/SYS (JKL) to change the character printed when a non-printable character is encountered.

```
SYS3/SYS,04,C1 (I)           from      3E 2E CD
SYS3/SYS,04,9D (III)         to        3E xx CD
```

(replacing "xx" with the Hex value of the character to be printed.)

Zap #1026 - This zap allows the interchanging of the period and slash in filespec definition. Thus a valid filespec would be "filespec.ext/password:dn". One known inconsistency is in the COPY function where the parameter "/ext" is used to define a group of files to be copied which must still be stated as "/ext", not ".ext". Any programs that do not use the NEWDOS/80 system calls for I/O or filespec handling will need changes to work with this zap.

```
SYS1/SYS,02,5E (I)          from      3E2F 12     to   3E2E 12
SYS1/SYS,02,75 (III)        from        3E 2F12   to     3E 2E12
SYS2/SYS,01,48              from      FE2F 06     to   FE2E 06
SYS2/SYS,01,4F              from        FE 2ECD   to     FE 2FCD
SYS3/SYS,03,7B (I)          from        FE 2F20   to     FE 2E20
SYS3/SYS,03,58 (III)        from      FE2F 20     to   FE2E 20
SYS3/SYS,04,17 (I)          from        3E 2FCD   to     3E 2ECD
SYS3/SYS,03,F0 (III)        from      3E2F CD     to   3E2E CD
SYS6/SYS,11,64              from      3E2F C4     to   3E2E C4
SYS8/SYS,00,B6              from      FE2F C2     to   FE2E C2
SYS8/SYS,01,62              from      3E2F C4     to   3E2E C4
```

Zap #1027 - This zap will do the following on a RESET:

a) Check to see if Alpha Products Newclock-80 is present and functional.

b) If it is, zap SYS0 temporarily with the two zaps outlined in Alpha Products Newclock-80 instruction sheet.

c) If a printer is on-line and ready a series of bytes will be sent to the printer for initialization provided a SHIFT key was not pressed.

d) If left arrow is held during reset and the clock is not present the old time and date will be used regardless of the setting of SYSTEM option AY and AZ.

```
SYS0/SYS,11,BF (I)          from      CD 6744 3A
                            to        CD E650 3A
SYS0/SYS,14,07 (I)          from      all zeros
                            to                          D9 0800 E5FD E50E 80ED
                                      78FE FF28 4B21 0951 1130 4F01 1800 ED B0
                                      11CD 4401 1E00 ED B0 1836 F3CD C044 2E44
                                      01BC 01CD D544 042E 44CD D544 04CD D344
                                      FBC9 2143 4001 8503 1603 ED78 00A2 160F
                                      0777 0707 B677 ED78 A200 B677 2B10 ECC9
                                      3A80 38FE 0020 2400 21E8 37FD 218A 5100
                                      7E00 FE3F 2015 06xx FD7E 00F5 DD7E 00FE
                                      3F20 F9F1 DD77 00FD 2310 ED3A 4038 CB6F
                                      2B0F 3EA5 32AB 433A F942 CB87 CB8F 32F9
                                      42FD E1DD E108 09CD 6744 C9yy
```

In the above zap you must replace "xx" with the hex number of bytes to be sent to the printer and start inserting the bytes to be sent at "yy"

Zap #1028 - This zap converts the "CLEAR" command to respect HIMEM and only clear the memory from 5200H to HIMEM on the DOS command "CLEAR" while the DOS command "CLEAR *" will clear all memory from 5200H to FFFFH and resets all user routing.

```
SYS14/SYS,03,70 (I)         from      FE 0D 28
                            to        C3 75 51
SYS14/SYS,03,82 (III)       from      FE 0D 28
                            to        C3 87 51
SYS14/SYS,04,88 (I)         from      all zeros
                            to                             FE 2ACA 7950 FE0D
                                      C276 502A 4940 228C 50E5 D121 FFFF C394
                                      50
SYS14/SYS,04,98 (III)       from      all zeros
                            to                                       FE2A CA0B
                                      50FE 0DC2 7650 2A11 4422 AE50 E501 21FF
                                      FFC3 A650
```

Zap #1029 - This zap changes the "+" and "-" keys required by SUPERZAP/CMD to the left and right arrow keys.

```
SUPERZAP/CMD,03,BC (I)   from       FE3B
            05,23 (III)  to         FE09
SUPERZAP/CMD,04,16 (I)   from       FE2D
            05,7C (III)  to         FE08
SUPERZAP/CMD,04,34 (I)   from       FE2D
            05,9A (III)  to         FE08
```

Zap #1030 - This zap removes the '?' prompt at the end of a directory page (Model I only), clears the screen when the next page is displayed (Model I only), and allows any key to be pressed when the next page is requested, instead of only <ENTER>. <BREAK> is still used to abort to DOS.

```
SYS8/SYS,03,86 (I)          from                        3E3F CDA5 50CD 4900 3DCA
                                      2040 FE0C 20F5
                            to                          CD49 0030 CA2D 40CD C901
                                      C3A5 5000 0000
SYS8/SYS,02,B6 (III)        from      FE0D
                            to        0000
```

Zap #1031 - This zap tidies up the directory display when in the expanded (,A) mode.

```
SYS8/SYS,04,C8              from      2E2E 2E2E
                            to        2020 2020
SYS8/SYS,04,F3              from        2E 2E2E 2E
                            to          20 2020 20
```

Zap #1032 - This zap zaps NEWDOS/80 so it will not clobber RAM when used with VIDEO4/CMD (an 80 by 24 video driver utility). This relocates the "bit bucket" area used during disk I/O from RAM to 3800H which is still protected ROM.

```
SYS0/SYS,03,2A (I)          change    26 01 CD
SYS0/SYS,02,A8 (III)        to        28 38 CD
SYS6/SYS,13,DC              change    00 00 00
                            to        00 38 00
SYS6/SYS,30,E5 (I)          change    00 00 CD
SYS6/SYS,30,EE (III)        to        00 38 CD
SYS19/SYS,04,29             change    11 00 CD
                            to        11 38 CD
```

Zap #1033 - This zap fixes the directory display to display filenames 5 across and to use all 24 lines of the video display when using VIDEO4/CMD.

```
SYS8/SYS,01,0C              change    0D    to   15
SYS8/SYS,01,12              change    05    to   06
SYS8/SYS,02,0E              change    04    to   05
SYS8/SYS,02,99 (I)          change    0F
SYS8/SYS,02,96 (III)        to        17
```

Zap #1034 - This zaps DISASSEM/CMD and EDTASM/CMD to use all 24 lines during screen paging when using VIDEO4/CMD.

```
EDTASM/CMD,10,72 (III)   change    10    to    18
DISASSEM/CMD,01,75 (III) change    10    to    18
```

Zap #1035 - This zap fixes the "LIB" display to display either 8 names across, or 10 names across when using VIDEO4/CMD in the 80 column mode.

```
SYS1/SYS,03,E1 (III)        change    0E 0006
                            to        0E 2F06
```

Zap #1036 - This is an optional zap for the Model 4 (& Model III DOS) that add SYSTEM flag CB. This tells the system that it is a Model 4 and the fast clock speed is activated. Note: SYSTEM option BJ must be set to 2.

```
SYS0/SYS,11,12 (4)          from      453A A543 87
```

```
                         to      45CD A850 B7
SYS0/SYS,13,84 (4)       from    all zeros
                         to      3AFF 43CB 7F2B 0521 1042 CBF6 3AA5 43C9
SYS17/SYS,02,57 (4)      from      FE FFFF FFFF FFFF FFFF FFFF FFFF FFFF
                                 FF
                         to        FE FEFE FEFE FEFE FEFE FEFE FEFE FE7F
                                 FF
```

Zap #1037 - These zaps document the changes made to the DOS by SofTrends' BREV-T. Note SYS28/SYS and SYS29/SYS as well as ABBREV/B and ABBREV/D must be installed, with the /SYS files in their proper places in the directory. The patch in SYS0/SYS is loaded at 41F1H.

```
SYS0/SYS,14,E9 (I)       from    all zeros
                         to                              01 11F1 41CD 4000
                                 F57E FE2D 2802 F1C9 233E 3FEF
SYS1/SYS,01,16 (I)       from    4000
                         to      F141
SYS18/SYS,00,1C (I)      from    CB53
                         to      8766
SYS18/SYS,00,68 (I)      from    CB53
                         to      8766
BASIC/CMD,15,90 (I)      from    all zeros
                         to      CDF4 41F5 3A00 52FE BFCA CC53 E521 9B66
                                 E33E 34EF AFE5 470E F07E FE00 2804 2304
                                 18F7 E1F1 C3CB 53
```

Zap #1038 - This zap changes the default argument of the BASIC "&" routine to default to Hex rather than Octal. Now "&O" must be specified for Octal and "&" or "&H" specified for Hex.

```
SYS20/SYS,02,D1          from      D7 4F11 0000 79FE 4820 2207 EBD6 30FE
                                 0A38 0806 11FE 0630 22C6 0A29 3807 2938
                                 0429 3801 290A B207 856F EB18 D90E 4FB9
                         to        4F 0711 0000 79FE 4F28 280E 4889 2801
                                 2B07 EBD6 30FE 0A38 08D6 11FE 0630 1CC6
                                 0A29 3807 2938 0429 3801 290A B207 856F
SYS20/SYS,03,00          from    2801 2B07 EBD6 0100 FA54 30FE 0838 E3CD
                                 9A0A EBC9
                         to      EB18 D3D7 EBD6 0100 FA54 30FE 0838 E9CD
                                 9A0A EBC9
```

Zap #1039 - This zap loads SYSTEM option BI (cursor character) into 4023H if the user programmable cursor character option is blasted into the EPROM replacements for the ROM.

```
SYS0/SYS,11,40 (I)       from    8728 03
                         to      3223 40
```

Zap #1040 - This zap makes CAPS,Y all uppercase and CAPS,N both upper and lower case with the new EPROMs installed.

```
SYS3/SYS,00,A1 (I)       from      32 B445 AFC9 CDEF 4028 F53E C918
                         to        32 1940 AFC9 CDEF 4028 F53E 0118
```

Zap #1041 - With the EPROMs installed, the BASIC caps lock control is changed to POKE 16409,0 to set mixed (upper & lower) case and POKE 16409,1 to set upper only (the same as Model III BASIC). This location replaces the 17844 (4584H) location that Apparat documented (reluctantly) in Model I zap # 082. You must also set SYSTEM option BF=N, BG=N - to prevent a conflict between the ROM lowercase driver and that of NEWDOS/80 for the SHIFT-0 caps lock function.

Zap #1042 - This zap corrects the date and time on a warm boot on a Model 4.

```
SYS0/SYS,02,DF (4)       from    1E 200D 361E 21
                         to      19 200D 3619 21
```

Zap #1043 - This zap enables the <BREAK> key in the FORTRAN package.

```
EDIT/CMD,02,DA           from    AF32 1E53
                         to      C3A8 7C00
EDIT/CMD,43,54           from    all zeros
                         to      AF32 1E53 3EC9 3278 44C3 0254
F80/CMD,00,04            from    C3A1 5700 0000 0000
                         to      3EC9 3278 44C3 A157
L80/CMD,00,07            from      AF 3215 43
                         to        C3 E674 00
L80/CMD,35,72            from    3000 0000 0000 0000 0000 0000
                         to      3EC9 3278 44AF 3215 43C3 0752
```

Zap #1044 - This zap will update the Newclock-80 every second, and get the date [illegible] each re-boot. SYSTEM options AY and AZ must be set to "Y".

```
SYS0/SYS,01,BA (I)       from                            2141 40E5 11AC
                                 4301 0600 EDB0 117C 40E1 0603 341A 96D0
                                 7113 2310 F723 34C9
                         to                              2143 4001 B503
                                 1603 ED78 0DA2 160F 0777 0707 8677 ED78
                                 A20D 8677 2B10 EBC9
SYS0/SYS,12,56 (I)       from                  44 2163 50CD 644F 019C 5011
                                 4640 3E2E CD6F 4F20 ED21 7650 CD64 4F
                         to                    44 F3CD CD44 2E44 018C 01CD
                                 D544 042E 46CD D544 04CD D344 FBC9 4F
SYS0/SYS,00,0A (III)     from    C318 3001
                         to      C3B6 4C01
SYS0/SYS,09,9D (III)     from    all zeros
                         to                                        F5 E5D5
                                 C53A 3003 EC21 1A42 018C 0116 0F18 1A3E
                                 168D 282C 042E 1C3E BAB9 280D 2DD6 02B9
                                 2805 0D2D 2D04 0416 03ED 780D A216 0D07
                                 7707 0786 77ED 78A2 0D86 772B 10EB 18CF
                                 C1D1 E1F1 C318 30
```

Zap #1045 - This zap permits use of labels starting with the "@" character in source code assembled by NEWDOS/80's EDTASM/CMD program.

```
EDTASM/CMD,01,78         from    FE41 38    to   FE40 38
EDTASM/CMD,01,8A         from    FE41 38    to   FE40 38
EDTASM/CMD,02,BA         from    FE41 30    to   FE40 30
EDTASM/CMD,08,18         from    FE41 D8    to   FE40 D8
EDTASM/CMD,12,44         from    FE41 20    to   FE40 20
EDTASM/CMD,28,38 (I)     from      FE 4138
EDTASM/CMD,28,37 (III)   to        FE 4038
```

Zap #1046 - This zap to NEWDOS/80's EDTASM/CMD program disables the handling of octal constants during assembling, and replaces it by the more useful binary number decoding facility. The zapped EDTASM/CMD will no longer recognize octal numbers (450, 0340 etc.) but it will now handle binary numbers. These must be written with an ending "B", e.g. 11110000B, 11B, 1111000011110000B.

```
EDTASM/CMD,15,B7         from        0608 FE4F 28   to   0602 FE42 28
```

Zap #1047 - This zap modifies NEWDOS/80's BASIC to allow TAB arguments up to 255.

```
BASIC/CMD,15,BE (I)      from    0000 0000 0000 00
                         to      E7FC 0528 C338 5F
BASIC/CMD,15,C2 (III)    from    0000 0000 0000 00
                         to      E7FC 0528 C314 5F
BASIC/CMD,17,84 (I)      from    C338 5FC3
                         to      C3B5 66C3
BASIC/CMD,17,5D (III)    from      C3 145F C3
                         to        C3 B566 C3
```

Zap #1048 - NEWDOS/80 currently ignores memory below 5200H when accessing user-defined asterisk (*name) routines (see NEWDOS/80 manual sections 3.31 & 3.32). Some Model I owners have installed a hardware modification that adds memory at (formerly unused) locations 3000H-37D0H. This zap allows the *name routines to access this added memory.

```
SYS9/SYS,03,D4 (I)       from      52   to   30
```

Zap #1049 - Some Model I owners have purchased a double sided drive zero and have performed a hardware modification to the drive so that side one is accessed as drive zero, and side two is accessed as drive one. The problem with this arrangement (which is seldom used) is that when the disk head is moved to a new track on one side, the head on the other side is "dragged along" and the next time that disk side is accessed, NEWDOS/80 finds that the head is not on the same track it had been left on and moves the head back to track zero and then up to the correct track. This wastes a lot of time and the following patch will speed things up considerably (please note that in most double-sided installations, both sides of drive zero are accessed as a single volume, and in that situation this zap should NOT be applied).

```
SYS0/SYS,04,84 (I)       from    3AED 37    to   C35B 43
SYS0/SYS,14,04 (I)       from    F7         to   E6
SYS0/SYS,14,EB (I)       from    all zeros
                         to        01 0F5B 43E5 7DEE 016F 3AED 3777 E1C3
                                 8E47
```

Zap#	Program	Reference - Indexed by Zap Number
1001	SYS0/SYS	TIME/DATE entry by numeric keypad
1002	SYS0/SYS	bypassing the TIME prompt with <ENTER>
1003	SYS0/SYS	forcing date only to be required on reboot
1004	SYS0/SYS	bypassing the DATE prompt with <ENTER>
1005	BASIC/CMD	eliminates the BASIC banner
1006	SYS0/SYS	eliminates the DOS banner on reboot
1007	SYS0/SYS	eliminates the TIME/DATE display on reboot
1008	SYS0/SYS	eliminates display of the AUTO command on boot
1009	SYS0/SYS	enables the "-@X" function in BREV-T automatically
1010	SYS1/SYS	eliminates double line feed at DOS READY
1011	BASIC/CMD	sets the default number of files available in BASIC

1012	BASIC/CMD	allows more than 15 files in BASIC
1013	SYS0/SYS	displays current time under DOS READY
1014	SYS9/SYS	allows proper reboot under double density
1015	SUPERZAP/CMD	allows automatic exit to DIRCHECK/CMD
1016	DIRCHECK/CMD	allows automatic exit to SUPERZAP/CMD
1017	SYS16/SYS	allows changing PDRIVE tables in memory only
1018	SUPERZAP/CMD	changes non-displayable character
1019	SUPERZAP/CMD	changes high value of displayable characters
1020	SUPERZAP/CMD	changes non-printable character
1021	SUPERZAP/CMD	allows space to be displayed
1022	SYS5/SYS	changes non-displayable character in DEBUG
1023	SYS5/SYS	allows space to be displayed in DEBUG
1024	SYS5/SYS	changes high value of displayable character in DEBUG
1025	SYS3/SYS	changes non-printable character in JKL
1026	SYS1/SYS	reverses "/" and "." in filespecs
1026	SYS2/SYS	reverses "/" and "." in filespecs
1026	SYS3/SYS	reverses "/" and "." in filespecs
1026	SYS6/SYS	reverses "/" and "." in filespecs
1026	SYS8/SYS	reverses "/" and "." in filespecs
1027	SYS0/SYS	checks for Newclock-80
1028	SYS14/SYS	changes CLEAR to respect HIMEM
1029	SUPERZAP/CMD	changes scrolling control keys
1030	SYS8/SYS	changes DIR prompt, etc
1031	SYS8/SYS	cleans up DIR display
1032	SYS0/SYS	changes "bit bucket" location
1032	SYS6/SYS	changes "bit bucket" location
1032	SYS19/SYS	changes "bit bucket" location
1033	SYS8/SYS	changes DIR display for 80 x 24 video
1034	EDTASM/CMD	changes display to use 24 lines
1034	DISASSEM/CMD	changes display to use 24 lines
1035	SYS1/SYS	changes LIB display for 80 x 24 video
1036	SYS0/SYS	adds SYSTEM option CB
1036	SYS17/SYS	adds SYSTEM option CB
1038	SYS0/SYS	changes required by BREV-T
1037	SYS1/SYS	changes required by BREV-T
1037	SYS18/SYS	changes required by BREV-T
1037	BASIC/CMD	changes required by BREV-T
1038	SYS20/SYS	changes BASIC "&" defaults
1039	SYS0/SYS	loads SYSTEM option BI into user cursor
1040	SYS3/SYS	caps lock processing with EPROMs
1041	BASIC/CMD	caps lock processing with EPROMs
1042	SYS0/SYS	corrects DATE/TIME processing
1043	EDIT/CMD	corrects (BREAK) key function in FORTRAN
1043	F80/CMD	corrects (BREAK) key function in FORTRAN
1043	L80/CMD	corrects (BREAK) key function in FORTRAN
1044	SYS0/SYS	Newclock-80 patches
1045	EDTASM/CMD	allows labels starting with "@" character
1046	EDTASM/CMD	changes to handle binary instead of octal constants
1047	BASIC/CMD	allows TAB arguments to 255 in BASIC
1048	SYS9/SYS	allows *name routines to be located at 3000H-37D0H
1049	SYS0/SYS	speeds up access to modified double sided drive zero

Zap#	Program	Reference - Indexed by Filespec
1005	BASIC/CMD	eliminates the BASIC banner
1011	BASIC/CMD	sets the default number of files available in BASIC
1012	BASIC/CMD	allows more than 15 files in BASIC
1037	BASIC/CMD	changes required by BREV-T
1041	BASIC/CMD	caps lock processing
1047	BASIC/CMD	allows TAB arguments to 255 in BASIC
1016	DIRCHECK/CMD	allows automatic exit to SUPERZAP/CMD
1034	DISASSEM/CMD	changes display to use 24 lines
1043	EDIT/CMD	corrects (BREAK) key function in FORTRAN
1034	EDTASM/CMD	changes display to use 24 lines
1045	EDTASM/CMD	allows labels starting with "@" character
1046	EDTASM/CMD	changes to handle binary instead of octal constants
1043	F80/CMD	corrects (BREAK) key function in FORTRAN
1043	L80/CMD	corrects (BREAK) key function in FORTRAN
1015	SUPERZAP/CMD	allows automatic exit to DIRCHECK/CMD
1018	SUPERZAP/CMD	changes non-displayable character
1019	SUPERZAP/CMD	changes high value of displayable characters
1020	SUPERZAP/CMD	changes non-printable character
1021	SUPERZAP/CMD	allows space to be displayed
1029	SUPERZAP/CMD	changes scrolling control keys
1001	SYS0/SYS	TIME/DATE entry by numeric keypad
1002	SYS0/SYS	bypassing the TIME prompt with (ENTER)
1003	SYS0/SYS	forcing date only to be required on reboot
1004	SYS0/SYS	bypassing the DATE prompt with (ENTER)
1006	SYS0/SYS	eliminates the DOS banner on reboot
1007	SYS0/SYS	eliminates the TIME/DATE display on reboot
1008	SYS0/SYS	eliminates display of the AUTO command on boot
1009	SYS0/SYS	enables the "-@X" function in BREV-T automatically
1013	SYS0/SYS	displays current time under DOS READY
1027	SYS0/SYS	checks for Newclock-80
1032	SYS0/SYS	changes "bit bucket" location
1036	SYS0/SYS	adds SYSTEM option CB
1038	SYS0/SYS	changes required by BREV-T
1039	SYS0/SYS	loads SYSTEM option BI into user cursor
1042	SYS0/SYS	corrects DATE/TIME processing
1044	SYS0/SYS	Newclock-80 patches
1049	SYS0/SYS	speeds up access to modified double sided drive zero
1010	SYS1/SYS	eliminates double line feed at DOS READY
1026	SYS1/SYS	reverses "/" and "." in filespecs
1035	SYS1/SYS	changes LIB display for 80 x 24 video
1037	SYS1/SYS	changes required by BREV-T
1026	SYS2/SYS	reverses "/" and "." in filespecs
1025	SYS3/SYS	changes non-printable character in JKL
1026	SYS3/SYS	reverses "/" and "." in filespecs
1040	SYS3/SYS	caps lock processing with EPROMs
1022	SYS5/SYS	changes non-displayable character in DEBUG
1023	SYS5/SYS	allows space to be displayed in DEBUG
1024	SYS5/SYS	changes high value of displayable character in DEBUG
1026	SYS6/SYS	reverses "/" and "." in filespecs
1032	SYS6/SYS	changes "bit bucket" location
1026	SYS8/SYS	reverses "/" and "." in filespecs
1030	SYS8/SYS	changes DIR prompt, etc
1031	SYS8/SYS	cleans up DIR display
1033	SYS8/SYS	changes DIR display for 80 x 24 video
1014	SYS9/SYS	allows proper reboot under double density
1048	SYS9/SYS	allows *name routines to be located at 3000H-37D0H
1028	SYS14/SYS	changes CLEAR to respect HIMEM
1017	SYS16/SYS	allows changing PDRIVE tables in memory only
1036	SYS17/SYS	adds SYSTEM option CB
1037	SYS18/SYS	changes required by BREV-T
1032	SYS19/SYS	changes "bit bucket" location
1038	SYS20/SYS	changes BASIC "&" defaults

[redacted]

by Andy Levinson

[Reprinted from THE INTERFACE newsletter of the San Gabriel Valley TRS-80 Users Group:]

As you have probably heard by now, TRSDOS 6 and LDOS have excellent type-ahead but with a problem: If the drives are running at exactly 300 R.P.M. (the usual recommended speed), TRSDOS 6 and LDOS may "go to sleep" or experience "silent death". The solution is to set the drives to run slightly faster, say 301 or 302 R.P.M.

I have mentioned the problem but it never bothered me on my Model 4. I figured that since I had to use TRSDOS 6 for Model 4 mode, I might as well enjoy it. All my disks were SYSGEN'ed for SYSTEM (SMOOTH=NO). It worked great. But then I knew that my drive was running slow (about 296 R.P.M.). No big deal since I had three drives and never wrote to drive 0. Then came my Model 4P.....

I knew the 4P booted slower but something more seemed wrong. I finally listened to myself and removed all those SYSTEM (SMOOTH=NO) commands and restored the default SYSTEM (SMOOTH) command. The result: My non-scientific estimate is that my 4P now runs about three to six times faster when it has to access the system disk. The type-ahead does not work as well but I'd rather get to the disks faster. If your system seems to be running slow then check to see that SMOOTH is set to [illegible] not off.

The second tip comes from Northern Bytes, Volume 7, Number 2. TRSDOS 6 treats MemDisk just like any other disk drive. This means that normal delay values (wait a moment before accessing a disk) also apply. When configuring MemDisk, TRSDOS 6 assigns a long delay factor even though MemDisk does not need any delay at all. To make MemDisk even faster, use the command: SYSTEM (DRIVE=#,DELAY=NO). Although this is not mentioned in the Northern Bytes article, you can include this in the SYSTEM command that invokes MemDisk. The improvement is worth the extra three words. Next time, try something like:

```
SYSTEM (DRIVE=2[redacted]DRIVER="MEMDISK")
```

LNW USER GROUP BBS ONLINE

LNW computer users can communicate with each other via the LNW User Group BBS. The number is (515) 285-4531 and it operates at 300 or 1200 baud, 24 hours a day.

NEWDOS/80 INPUT@nnnn

by Eddy Schouten, Nieuwendijk 116, 7311 RM Apeldoorn, Holland
Telephone 011+31+55+212298
Translated from Dutch to English by Bert Guffens

[Reprinted from REMARKS, the publication of TRS-80 Gebruikers Vereniging (TRS-80 Users Group), Postbus 551, 2070 AN Santpoort Noord, Holland:]

If you are working with DOSPLUS you'll know the statement. NEWDOS/80 users may not, but they will now.

Although it does not have all the niceties of INPUT@ from DOSPLUS such as allowing for numeric or alphanumeric input and automatic continuation after the field length is reached, I gather someone out there will come up with the 'rest of the story' [If you do, please send it in so we can print it in Northern Bytes].

You can use the function in a BASIC program by means of a USR call. The following line will see to it.

```
DEFFNED$(L,P)=CHR$(L)+MKI$(&H3C00+P)+STRING$(-(L-3>0)*(L-3),32)
```

You hereby define the length of ED$ and reserve string space whereby L=length of the input field and P=the starting screen position. (P must be between 0 and 1023)

To call the routine:

```
PRINT@500,"NAME :":A$=FNED$(10,507):X=USR1(VARPTR(A$))
```

You may also predefine length and position:

```
L=10:P=507:PRINT@500,"NAME :":A$=FNED$(L,P):X=USR1(VARPTR(A$))
```

At position 500 you will see

```
NAME ..........
```

with the cursor flashing at the first position. You must save the returned string (A$) before calling the function again (example: B$=A$).

To cancel a previous field use the next function:

```
DEFUSR2=&HF0D4:X=USR2(X)
```

Cancelling a field does not need any parameters.

The assembler listing, KEYIN/ASM is the actual program while the Basic program KEYDEMO/BAS is a demonstration.

```
10 'KEYDEMO/BAS
20 '
30 CLS
40 CMD"HIMEM F000H":CMD"LOAD KEYIN/CMD"
50 CLEAR500:DEFUSR1=&HF001:DEFUSR2=&HF0D4
60 DEFFNED$(L,P)=CHR$(L)+MKI$(&H3C00+P)+STRING$(-(L-3>0)*(L-3),32)
 'fix length of string
70 PRINT@450,"Type in a string :"
80 A$=FNED$(20,467) '20 dots
90 X=USR1(VARPTR(A$)) 'pass address A$
100 B$=A$:PRINT@590,"B$= ";B$
110 INPUT"Enter ...";q$
120 X=USR2(X) ' erase A$
```

```
               00010 ;         KEYIN/ASM
               00020 ;
               00030 ;         E SCHOUTEN  (COPY FREE)
               00040 ;         NIEUWENDIJK 116
               00041 ;         7311 RM  APELDOORN, NETHERLANDS
               00042 ;         TEL=055-212298
               00043 ;
F001           00050       ORG     0F001H
F001 CD7F0A    00060 START CALL    0A7FH         ;ARGUMENT IN HL
F004 22E8F0    00070       LD      (BUF3),HL     ;VARPTR (A$)
F007 E5        00080       PUSH    HL            ;
F008 DDE1      00090       POP     IX            ;
F00A DD6E01    00100       LD      L,(IX+1)      ;STOP ADDRESS
F00D DD6602    00110       LD      H,(IX+2)      ;A$ IN
F010 22E9F0    00120       LD      (BUF2),HL     ;BUFFER 2
F013 DD21E6F0  00130       LD      IX,BUF0       ;POINT TO BUF0
F017 0603      00140       LD      B,3           ;
F019 7E        00150 LOOP  LD      A,(HL)        ;GET 3 BYTES
F01A DD7700    00160       LD      (IX),A        ;BYTE1=LENGTH
F010 23        00170       INC     HL            ;BYTE2,3=POSITION
F01E DD23      00180       INC     IX            ;ON SCREEN
F020 10F7      00190       DJNZ    LOOP          ;
F022 2AE7F0    00200       LD      HL,(BUF1)     ;PRINT@nnnn
F025 E5        00210       PUSH    HL            ;
F026 D1        00220       POP     DE            ;HL IN DE
F027 13        00230       INC     DE            ;
F028 362E      00240       LD      (HL),'.'      ;PUT NUMBER
F02A 3AE6F0    00250       LD      A,(BUF0)      ;OF DOTS ON
F02D 3D        00260       DEC     A             ;SCREEN
F02E FE00      00270       CP      0             ;
F030 2804      00280       JR      Z,CONT        ;
F032 4F        00290       LD      C,A           ;AS LONG AS
F033 A8        00300       XOR     B             ;BUF1
F034 EDB0      00310       LDIR                  ;
F036 23        00320 CONT  INC     HL            ;
F037 3620      00330       LD      (HL),' '      ;CLOSE OFF THE
F039 23        00340       INC     HL            ;FIELD WITH
F03A 3620      00350       LD      (HL),' '      ;2 BLANKS
F03C 3C        00360       INC     A             ;
F030 47        00370       LD      B,A           ;
F03E 2AE9F0    00380       LD      HL,(BUF2)     ;
F041 DD2AE7F0  00390       LD      IX,(BUF1)     ;PRINT@nnnn POS
F045 D9        00400 KBDCH EXX                   ;CHANGE REGIST.
F046 DD7E00    00410 SUB0  LD      A,(IX)        ;
F049 EE8E      00420       XOR     142           ;
F04B DD7700    00430       LD      (IX),A        ;
FD4E 010002    00440       LD      BC,200H       ;
F051 CD2B00    00450 SUB1  CALL    2BH           ;ROM INPUT KBRD
F054 FE00      00460       CP      0             ;
F056 2007      00470       JR      N2,LOOP0      ;
F058 0B        00480       DEC     BC            ;
F059 78        00490       LD      A,B           ;
F05A B1        00500       OR      C             ;
F05B 20F4      00510       JR      NZ,SUB1       ;
F05D 18E7      00520       JR      SUB0          ;
F05F D9        00530 LOOP0 EXX                   ;RESTORE REGIST
F060 F5        00540       PUSH    AF            ;
F061 3E2E      00550       LD      A,'.'         ;
F063 DD7700    00560       LD      (IX),A        ;
F066 F1        00570       POP     AF            ;
F067 FE08      00580       CP      8             ;
F069 2012      00590       JR      NZ,LOOP1      ;
F06B 3AE6F0    00600       LD      A,(BUF0)      ;
F06E B8        00610       CP      B             ;
F06F 2804      00620       JR      Z,KBDCH       ;
F071 04        00630       INC     B             ;
F072 2B        00640       DEC     HL            ;
F073 DD2B      00650       DEC     IX            ;
F075 362E      00660       LD      (HL),'.'      ;
F077 DD36002E  00670       LD      (IX),'.'      ;
F07B 18C8      00680       JR      KBDCH         ;
F070 FE0D      00690 LOOP1 CP      13            ;
F07F 2013      00700       JR      NZ,LOOP3      ;
F081 DD360020  00710       LD      (IX),' '      ;
F085 DD23      00720       INC     IX            ;
F087 04        00730       INC     B             ;
F088 78        00740       LD      A,B           ;
F089 FE00      00750       CP      0             ;
F08B 2820      00760       JR      Z,SLOT        ;
F08D 23        00770       INC     HL            ;
F08E 23        00780       INC     HL            ;
F08F 181C      00790       JR      SLOT          ;
F091 05        00800 LOOP2 DEC     B             ;
F092 18B1      00810       JR      KBDCH         ;
F094 FE20      00820 LOOP3 CP      32            ;
F096 38AD      00830       JR      C,KBDCH       ;
F098 FE7F      00840       CP      127           ;
F09A 30A9      00850       JR      NC,KBDCH      ;
F09C 77        00860       LD      (HL),A        ;
F090 007700    00870       LD      (IX),A        ;
F0A0 23        00880       INC     HL            ;
F0A1 DD23      00890       INC     IX            ;
F0A3 78        00900       LD      A,B           ;
F0A4 FE00      00910       CP      0             ;
F0A6 20E9      00920       JR      NZ,LOOP2      ;
F0A8 2B        00930       DEC     HL            ;
F0A9 DD2B      00940       DEC     IX            ;
F0AB 1898      00950       JR      KBDCH         ;
F0AD 3620      00960 SLOT  LD      (HL),' '      ;CLOSE A$
F0AF DD2B      00970       DEC     IX            ;WITH BLANK
```

```
F0B1 DD360020 00980          LD      (IX),' '        ;
F0B5 DD23     00990          INC     IX              ;
F0B7 DD36008C 01000          LD      (IX),140        ;CLOSE WITH
F0BB DD23     01010          INC     IX              ;GRAPHIC BLOCK
F0BD 05       01020          DEC     B               ;
F0BE 3AE6F0   01030          LD      A,(BUF0)        ;
F0C1 90       01040          SUB     B               ;
F0C2 2AEBF0   01050          LD      HL,(BUF3)       ;
F0C5 77       01060          LD      (HL),A          ;
F0C6 78       01070 LOOP4    LD      A,B             ;WIPE LEFTOVER
F0C7 FE00     01080          CP      0               ;DOTS OFF
F0C9 2808     01090          JR      Z,RET           ;SCREEN
F0CB DD360020 01100          LD      (IX),' '        ;
F0CF DD23     01110          INC     IX              ;
F0D1 10F3     01120          DJNZ    LOOP4           ;
F0D3 C9       01130 RET      RET                     ;TO BASIC
F0D4 2AE7F0   01140          LD      HL,(BUF1)       ;CALL ERASE
F0D7 3AE6F0   01150          LD      A,(BUF0)        ;ROUTINE
F0DA C602     01160          ADD     A,2             ;
F0DC 3620     01170 ERASE    LD      (HL),' '        ;
F0DE 23       01180          INC     HL              ;
F0DF 3D       01190          DEC     A               ;
F0E0 FE00     01200          CP      0               ;
F0E2 28EF     01210          JR      Z,RET           ;
F0E4 18F6     01220          JR      ERASE           ;
0001          01230 BUF0     DEFS    1               ;LENGTH
0002          01240 BUF1     DEFS    2               ;PRINT@ POS.
0002          01250 BUF2     DEFS    2               ;ADDRESS A$
0002          01260 BUF3     DEFS    2               ;VARPTR A$
0002          01270          END     START           ;ENTRY POINT
```

NEW PROGRAM REVIEW

by Peter Goed

[Peter Goed is the Program Co-ordinator for the TRS-80 System 80 Computer Group, Brisbane, Australia. This article was reprinted from their newsletter Bits & Bytes. Glen McDiarmid (author of the program under review) sent me a copy of this program for review purposes a while back, but as I have been extremely pressed for time I have not been able to give the program a decent going-over, so I hope this reprint will serve the purpose (my apologies to Glen for not getting to it myself, but a good review takes a lot of time!). Please note that this program runs on the Model 4P only at the time of this writing, however, Glen has indicated to me that he is working on a hardware modification that will make a Model 4 emulate a 4P, so you may wish to contact him if you are interested in performing such a modification.]

*** TIME MACHINE FOR THE 4P -- Disk and Manual $50 [Australian] +$5 postage & packing anywhere in Australia. Obtainable from Glen McDiarmid, 28 Marginson Street, Ipswich, Queensland 4305, Australia. Telephone (07) 281 7057 [international direct dial from North America 011+61+7+281-7057].

In July last year, Glen McDiarmid, one of our club members wrote an assembly-language program that is unlike any other program that I have ever seen. Glen spent nearly six months perfecting the program and then gave me a copy to Beta Test. After another six months of changes, improvements and de-bugging, the program has now been released.

The name of the program is Time Machine™, and, while it may seem a fanciful name for a program, it does accurately describe one of its functions. How could a time machine help computing? Well, for one thing, if you just made a mistake in inputting data, and you had already pressed <ENTER>, it would be very helpful if you could go back in time to when you were just starting to input the data, so that you could retype it correctly, without having to RESET, load the program again, and input responses to get you to the same point in the program that you were before. In fact, when you know that you can go back in time whenever you wish, you are able to experiment with programs like you have never been able to before.

At the moment, Time Machine runs on Model 4P computers only, in Model III mode. Once installed, any Model III program may be used, as Time Machine resides wholly in low memory (in fact it resides in the ROM area of the 4P). You only have to run Time Machine once, when you first turn on your computer, and it stays there, even if you RESET, except when you go to Model 4 mode.

This program has opened my eyes to new, faster ways of doing things, and some things that simply couldn't be done at all in the past become simplicity itself. For instance, you can save a moment in time, then save it to disk and send it to a friend with Time Machine, so that he/she could also share that moment you were experiencing on your computer. When you are having troubles with software, this would be VERY handy indeed. Or, if you start something that will take a long time to complete, you may like to save to disk the moment just before you wish to switch the computer off at night, then load that moment again the next day, and continue as if nothing had happened!!

To me the greatest boon has been in the development of assembly language programs, where I am saving about 40% of normal development time, because of the ease of running the assembled program, then doing an immediate return to the editor/assembler for alterations without having to reload. If you are into assembly language at all, Time Machine is a definite must for your shopping list.

As Time Machine can interrupt virtually any program without 'damaging' it, other functions were added, that may be handy while running some other program. By running two separate moments alternately, you can have two DOS's running concurrently, each using full memory (48K). For those that are new to concurrent DOS, it is like having two computers, running only one at a time, while the other one simply waits to be used again.

Other Time Machine functions that can be used DURING virtually any Model III program are:- Switch between slow and fast CPU speeds; Take photos of screen contents (up to six); Examine any one of the six screen photos; Toggle reverse video ON/OFF, Toggle alternate character set ON/OFF, Immediate return to DOS READY; Software RESET; Pass control to a monitor program, with all registers intact -- about 26 functions in all.

The PHOTO function can be used instead of JKL. These photos can be stored on disk. Those that do not have a printer will definitely welcome these solid-state JKL's!!

[redacted]

by Jack Decker

While preparing an issue of NORTHERN BYTES for publication, I ran into the problem of producing a Editor-Assembler output listing from a source code file that had labels that started with the "@" character (it was a Model 4 program and the labels defined Supervisory Calls [SVC's] in TRSDOS 6). The Editor-Assembler program I usually use came up with an error message whenever it hit a label with an "@" character.

Then it dawned on me - somewhere in the Editor-Assembler's code (probably in several places) there was most likely a test for a character lower than an uppercase "A", which would cause a branch to an error routine if a lower ASCII value character was encountered. If that test, which was most likely a CP 41H instruction, were to be changed to a CP 40H instruction, the Editor-Assembler should no longer choke on an "@" character. So, I called up SUPERZAP, used the DFS command to access sector zero of the Editor-Assembler program, and then (when sector zero was displayed on the screen) typed:

```
F,FE,41 <ENTER>
```

This found the first occurrence of a CP 41H instruction. I used the MOD command to change the 41H to 40H, then used the F command to find the next occurrence of a CP 41H. I kept repeating this sequence until all FE 41 bytes in the program had been changed to FE 40. Suffice it to say that it worked. The Editor-Assembler would now accept labels starting with the "@" character.

At this point you are probably wondering why I don't just give you the locations to zap. Well, in the first place, there are so many versions of Editor-Assemblers around that any locations I might give you could be incorrect in the version you have. And besides that, I haven't fully tested the changes I made yet, though they appear to work. It should go without saying that if you try this, always keep an unpatched backup copy of the original version of your Editor-Assembler program, so that you can go back to it if you discover that your zaps have introduced a bug into the program (or if, heaven forbid, you accidentally zapped one of the disk loader codes and now the program won't even load).

I will tell you that I tested this technique with both Apparat's EDTASM program, and with one version of the old Misosys DiskMod program, and in both cases the zaps appeared to work (strangely, in DiskMod only two locations needed to be zapped, while EDTASM had six places that contained FE 41 bytes, all of which were duly zapped). However, my "testing" consisted of assembling ONLY ONE program that contained the "@SVC" type labels. LET THE ZAPPER BEWARE!!!

LETTER TO SMUG NEWS

by Leigh Sheppard

[The following is reprinted from the official newsletter of the Surrey (British Columbia, Canada) Microcomputer Users Group. The editor of that publication prefaced the following with the comment that "This letter is from Leigh Sheppard and shows the attitude of some rather important individuals (at least they think so)."]

In the September issue of 80 Micro, Hardin Brothers had an article called SUBROUTINE SANDWICHES which contained a list of machine language subroutines called and executed from BASIC for scroll protecting the screen, testing the state of the caps key, etc. I had been experimenting with converting similar routines from BASIC FASTER & BETTER to Model 4, and so thought that I would pass on what I had discovered to Hardin Brothers via the LDOS SIG on Compuserve. Below is my letter which I sent to Hardin, followed by the rather surprising responses that I got from others who obviously enjoy answering other people's mail. I have made a short story rather long but I thought you might be interested in at least some of it for use in your Newsletter. The rest is good for a laugh!

First...my letter to Hardin:

* * * * *

I have very much enjoyed tinkering with some of the routines in your SUBROUTINE SANDWICHES article in 80 Micro. I wanted to pass along a few tips tho' as some of the machine language is really not necessary. If you follow the @FLAGS SVC you will see that all it does is load the IY register with 006AH and return. Well, you don't need to do a SVC to point IY to 006AH, you can do that yourself. Carrying this further, from BASIC then, setting bits in the @FLAGS table should be a matter of POKEing around in the 006AH range...

```
CAPS ON = POKE 116,PEEK(116) OR 32
CAPS OFF= POKE 116,PEEK(116) AND NOT 32
Test CAPS key: IF PEEK(116) AND 32 THEN PRINT"CAPS ON"

BREAK KEY DISABLED = POKE 124,PEEK(124) OR 16
BREAK KEY ENABLED  = POKE 124,PEEK(124) AND NOT 16
Test BREAK key: IF PEEK(124) AND 16 THEN PRINT "BREAK
DISABLED"

CLOCK DISPLAY ON = POKE 127,PEEK(127) OR 16
CLOCK DISPLAY OFF= POKE 127,PEEK(127) AND NOT 16
```

You can also read or alter the system date by PEEK or POKE:

```
PEEK(51)=YEAR
PEEK(52)=DAY OF MONTH
PEEK(53)=MONTH
```

I know that the Technical Reference Manual says "Nothing in the system can be accessed via any absolute address reference", but these have always worked for me! I have never seen them published anywhere so I thought you might like them.

* * * * *

Here are the responses that I got.

* * * * *

```
#:57188 S3/Mod 4/4P TRSDOS 6
  20-Sep-85 03:37:36
Sb: #57177-SUBROUTINES
Fm: Roy Soltoff/MISOSYS,Inc 70140,310
To: LEIGH SHEPPARD 71625,1446
```

It is stupid and very non-portable to insist on the archaic practice of poking into ABSOLUTE locations when the DOS provides a procedure to utilize which is portable across ALL past, present, and future releases of the DOS. It is practices such as your suggested poking which have caused countless numbers of conversion problems for thousands in the past. Why not learn to program in a more professional manner? -Roy

* * * * *

What did I ever do to him??

This one is from the SYSOP for the LDOS SIG. I think it conveys the arrogant attitude that LDOS seems to ooze.

* * * * *

```
#:57198 S3/Mod 4/4P TRSDOS 6
  20-Sep-85 10:43:53
Sb: #57177-#SUBROUTINES
Fm: LDOS Support 76703,437
To: LEIGH SHEPPARD 71625,1446
```

Argggg!!!!! What happens when the pointer returned by @FLAGS moves?? Having *standard* methods of interface is the OS is what makes the TRSDOS 6 environment what it is, portable! Where will @FLAGS point to in 6.2.1? Where did it point on 6.0.0? Where will it point in 7.3.9? If you use the standard calls, you don't care! I think that we should make it a point to make sure that everything like that moves around with every release. That sure will smoke out the "Bad boys" quick.

—jjkd--

* * * * *

Maybe by 7.3.9 they'll get it right!!

Hardin's own reply said much the same thing but is much more civil.

* * * * *

```
#:57209 S3/Mod 4/4P TRSDOS 6
  20-Sep-85 16:08:34
Sb: #57177-SUBROUTINES
Fm: Hardin Brothers 70007,1150
To: LEIGH SHEPPARD 71625,1446
```

The point with TRSDOS, as I imagine the other replies to your message will also tell you, is that future releases can move anything to any location in memory. Your Peeks and Pokes will work for your version of TRSDOS 6, but not necessarily for any others.

Of course, I know the absolute addresses that TRSDOS uses to store many values, but I certainly don't want to write an article that could be made obsolete (and even dangerous to the health of the DOS) if the programs were run on the next version of TRSDOS. My window driver needed absolute addresses to find enough hooks into TRSDOS 6.2 (and I can assure you that I've been criticized, perhaps rightfully, for developing that program in that manner). But the subroutines certainly run well enough and fast enough that they have no excuse for being specific to 6.2.0 (or any other version of TRSDOS).

--Hardin

* * * * *

Realizing that I had been very bad, I did my best to repent. I sent this to LDOS fully expecting to be kicked of the SIG.

* * * * *

Forgive me, Father, for I have sinned! I have PEEKed under the skirts of TRSDOS 6.2.0 and even done a little indiscriminate POKEing into its private parts (i.e. below 3000H). And all because:

```
10 POKE 116,PEEK(116)OR 32
```

looked more attractive to me than:

```
10 DATA 9086,28518,18149,32291,26147,1391,-14907,11790,574,...etc.
20 DIM Q1%(23):RESTORE:FOR Q%=1 TO 23:READ Q1%(Q%):NEXT Q%
30 DATA 25918,32495,470,32509,14346,10253,-13553,15983,10240,...etc.
40 DIM Q4%(17):FOR Q%=1 TO 17:READ Q4%(Q%):NEXT Q%
50 Q%=0:Q1%=VARPTR(Q4%(1)):CALL Q1%(Q%)
```

Its painfully obvious that I have almost committed the unpardonable sin. All I wanted to do was set my CAPS key! Oh, wretched soul that I am! Always looking for the short cut! How many "HAIL ROY"s will I have to do? Please forgive me! Maybe you could re-write BASIC with no PEEK or POKE capabilities, thus removing all temptation!

* * * * *

They actually answered that one with an apology, sort of...

* * * * *

```
#:57356 S3/Mod 4/4P TRSDOS 6
  23-Sep-85 13:19:12
Sb: #57333-SUBROUTINES
Fm: LDOS Support 76703,437
To: LEIGH SHEPPARD 71625,1446(X)
```

How does that old saying go? The laws don't say that you can't do something, only that you shouldn't let yourself get caught... If it's your program, on your diskette, and you know what's going on, the decision, of course, is up to you. We like to discourage this kind of stuff, and especially the publication thereof (exaggerating severely to make a point) 'cause the vast majority of folks latch on to this kind of stuff and start violently shoving the code into everything they can lay their hands on. And who do they call when the DOS revision changes and their programs stop working? I'll give you a hint, it's not 'Ghostbusters', and it's not the author of the code in question.
--jjkd--

* * * * *

Oh well, at least they didn't kick me off, so I decided to do it their way, and actually its not half bad. Some of the routines turned out pretty good.

This is what I ended up sending to Hardin:

* * * * *

Alright, I can understand why you might not like poking absolute addresses in TRSDOS 6.2. However, not everyone is trying to write programs meant for distribution. I just like to play around and see what makes things tick. When a new version of a DOS comes out, it means another new challenge. Anyway, I can see your point for not wanting to publish, or otherwise encourage, absolute calls to DOS routines, so to redeem myself, I thought you might like the routine below, which should be transportable between versions of Model 4 TRSDOS.

```
10000 Q(1)=A             :'A register contents(SVC Number)
10010 Q(3)=CVI(CHR$(L)+CHR$(H)):'HL reg pair(order is important!)
10020 Q(5)=CVI(CHR$(E)+CHR$(D)):'DE reg. pair   "  "  "
10030 Q(7)=CVI(CHR$(C)+CHR$(B)):'BC reg. pair   "  "  "
10040 SVC=VARPTR(Q(0)):CALL SVC:RETURN
20000 'initialization - needs to be called only once.
20010 DIM Q(8):Q(0)=15872:Q(2)=8448:Q(4)=4352:Q(6)=256:Q(8)=-13841
20020 RETURN
```

It sets up a shell routine for calling SVCs. The program should be initialized with 10 DEFINT Q:GOSUB 20000. After that, it's just a matter of setting A=the SVC number, set up the other registers as required, and GOSUB 10000. For instance, to scroll protect 5 lines of the screen, you would set the call up as:

```
A=15:B=7:C=5:GOSUB 10000
```

or to set the cursor character to CHR$(143):

```
A=15:B=8:C=143:GOSUB 10000
```

to save the screen to memory location E800H:

```
A=15:B=6:H=&HE8:L=&H00:GOSUB 10000
```

and to call it back:

```
A=15:B=5:H=&HE8:L=&H00:GOSUB 10000
```

to enter DEBUG from within BASIC

```
A=27:GOSUB 10000
```

Before each call is made, set A,B,C,D,E,H,L to 0.

Unfortunately, the routine as written will not work with @FLAGS because @FLAGS returns a value through the IY register, and this routine does not pass values back to BASIC. I'm sure that with a little work it could be done however. It works like a charm with @VDCTL and although I haven't tried it, I see no reason why the routine shouldn't work with @BANK for doing a little fancy bank switching from within BASIC.

* * * * *

and sure enough a little later, I was able to send this:

* * * * *

Hi. Back again. I never give up! I now have a routine for calling the @FLAGS SVC. All it does is execute @FLAGS and return the address that is in the IY register back to BASIC in the variable IY%. For the current release of TRSDOS 6.2, that is 006AH. However, should the @FLAGS table move in subsequent releases of TRSDOS, this routine will still find the table. After running the routine, the programmer supplies the offset value for IY and which bit to test, set or reset. The routine needs only be called once, as long as the variable IY is left intact. I have also assumed that the program has defined A-Z as integers.

```
30000 DIM Z(5):Z(0)=25918:Z(1)=-529:Z(2)=11803:Z(3)=30587
30010 Z(4)=9082:Z(5)=-13961
30020 IY=0:FLAGS=VARPTR(Z(0)):CALL FLAGS(IY):RETURN
```

Once this routine is executed, the calls would be as follows:

```
CAPS ON:  POKE(IY+10),PEEK(IY+10) OR 2^5
CAPS OFF: POKE(IY+10),PEEK(IY+10) AND NOT 2^5
test CAPS key: IF PEEK(IY+10) AND 2^5 THEN PRINT "CAPS ON"

BREAK ON:  POKE(IY+18),PEEK(IY+18) OR 2^4
BREAK OFF: POKE(IY+18),PEEK(IY+18) AND NOT 2^4
test BREAK key: IF PEEK(IY+18) AND 2^4 THEN PRINT "DISABLED"

TYPE-AHEAD OFF: POKE(IY+3),PEEK(IY+3) OR 2^1
```

...... etc.

If the program had a lot of manipulating of the @FLAGS bits, it would be an idea to have the POKES in a subroutine as follows:

```
1000 POKE(IY+OFFSET),PEEK(IY+OFFSET) OR 2^BIT:RETURN
```

Then, turning on the caps lock would be:

```
OFFSET=10:BIT=5:GOSUB 1000
```

There is a lot you can do from within the @FLAGS table, and with these routines, all the programmer needs to know is the information in the @FLAGS section of the Technical Reference manual and he can test or set any bit on the table. Like this better?

* * * * *

To which Hardin Brothers replied "MUCH".

DOSPLUS 4 RIGID DRIVER NEEDED

Ron Malo is trying to locate a "Rigid Driver" for use with DosPlus 4 and an A&M Hard Drive. The obvious sources have not been able to supply this driver. Failing that, Ron would like to hear from anyone who has the technical expertise to modify another DosPlus 4 Rigid Driver to work with the A&M drive. If you can help, please write to Ron at 1847 North Screenland Drive, Burbank, California 91505, or phone him at (818) 845-5776 (evenings) or (818) 764-4555 (days).

TELECOMMUNICATIONS NEWS

[Before we start - just a reminder that articles appearing in NORTHERN BYTES that do not bear a specific copyright notice, including this article, may be reprinted freely by anyone, anywhere.]

There are two conflicting forces at work in computer communications. One is represented by the computer user, who wants to have access to as much online information and/or entertainment as possible, at the lowest possible cost. The opposite force is probably best represented by the telephone companies and similar interests, who want people to use their services - at an added cost - to access the chosen online services. The conflict is that, ideally, the user would like to pay nothing for the link between his location and the chosen online host computer, while the phone companies would like to charge for every minute of online time. As we have discussed previously, there is absolutely no technical reason that we can't transmit all the computer data that we wish by radio, for no charge whatsoever. But government regulations force most of us to use the telephone monopoly instead of the free airwaves - and the telephone companies, which are nothing if not greedy, now are talking about charging for all local calls. In some areas of the country they are already doing this, with little or no apparent opposition! Where are all of the "consumer protection" agencies? Do they receive funding (read: "hush money") from the phone companies? It makes you wonder!

What is galling about this is that one of the reasons the phone companies say they need to begin charging for local calls is that computer users are overtaxing their equipment! They are, in effect, trying to lay the blame at our doorstep! So, why doesn't the government allow (and even encourage us) to use radio for computer-to-computer communications whenever possible, thus relieving the alleged congestion on the telephone switching equipment. Then the phone companies could concentrate on handling voice traffic, and they wouldn't have to start charging by the minute for local calls (want to bet they'd find another excuse to do it?). But in spite of the supposed congestion, you may have noticed that it is the phone companies that scream the loudest over any proposals to give radio spectrum (for either voice <u>or</u> data) to the general public.

Well, in this column we're going to discuss some ways to avoid spending any more of your money that is absolutely necessary for your modem calls. This is not "pie in the sky" stuff, most of what I'm going to talk about is available right now.

FLAT RATE CALLING

"I am from Milwaukee and my wife is going to kill me if she sees another $200 phone bill. I have friends facing the same predicament. [PC] Pursuit is just what I need!"

- (name withheld to protect the husband...)

The above message was left by a user of PC Pursuit, which is a service of Telenet, the major packet switching network (that, as it happens, is owned by two of the largest non-Bell telephone companies in the U.S.A.). Frequent MODEM users that are fortunate enough to live within the local calling area of a city that has a local Telenet dial-up access number (that includes almost all major- and medium-sized cities in the United States) can sign up for the NEW, expanded PC PURSUIT service, which lets you dial out to computer Bulletin Board Services and other online services in any of 14 major cities (more to be added soon). For a flat fee of $25 per month, PC PURSUIT lets you establish an UNLIMITED number of long-distance connections in the evenings and weekends (the evening/weekend flat-fee usage period extends from 6 p.m - 7 a.m. weekdays and on weekends from 6 p.m. Friday to 7 a.m. Monday [local time]). If you have the need to make calls to online services during business hours, you can use PC PURSUIT during the business day for the rate of $10.50/hr or $14/hr, depending upon your access location. Connections made via Telenet are generally much better than those made using a voice-grade long distance carrier, since the Telenet network is specifically designed to handle data transmissions. Thus, you effectively get the transmission quality of a local telephone call.

Note that you no longer have to be located in one of the 14 cities served by PC PURSUIT to use the service! As long as the online service <u>you are calling</u> is in one of the 14 cities, you can access PC PURSUIT from ANY Telenet node in the United States. This is a change from the former system, where both the caller and the called online service had to be in a PC PURSUIT city. Now you can call from ANY city on the Telenet network TO any of the following 14 major cities: Atlanta, Boston, Chicago, Dallas, Denver, Detroit, Houston, Los Angeles, Newark, New York, Philadelphia, San Francisco, Seattle, and Washington, D.C.

Another change from the former system used by PC PURSUIT is that in the past, users were forced to enter pertinent information about the numbers they wanted to call, then had to hang up. PC PURSUIT would then call back once the connection was established through the network. That method of operation caused problems for many users and potential users, and has been abandoned in the new system. Now you place you call through the nearest Telenet node, and just stay online while your call is completed. You can even retry busy numbers, or try other numbers accessible through PC PURSUIT without having to hang up and call back.

Although PC PURSUIT is touted as a way for computer users to call Bulletin Board Systems in distant communities, you can in fact save money while calling ANY online computer that has a local access number in one of the 14 major cities listed above - your call need not necessarily be to a private BBS. As an example, services such as GENIE and MCI Mail have their own packet networks, but these do not cover many of the smaller cities that Telenet reaches (and in other smaller cities, a per-minute network surcharge may be imposed). You can simply use PC PURSUIT to dial up a no-surcharge access number in any one of the major cities served by PC PURSUIT, and forget about the per-minute access surcharges you've been paying to reach these services. Or, perhaps you have a friend (that owns a computer and a modem) in one of the 14 major cities. Rather than spending tons of money conversing by long distance, you can agree to have his computer answer the phone at a pre-arranged time. You can then place a call to him via PC Pursuit at the specified time, and the two of you can wear you fingers to the bone typing back and forth at each other all night, without worrying about how long you've been online!

PC PURSUIT will be a real boon for the next generation of networking-type BBS's such as FidoNet and SYSLINK, if the BBS software is rewritten to make maximum use of PC PURSUIT. The reason is that a node of the BBS network that is located in any of the major cities served by PC PURSUIT could be designed to receive messages for other BBS's on the network, and hold them (WITHOUT calling out) until the recipient BBS calls in to collect its messages. I'll use the SYSLINK BBS to illustrate how this might work, though the principle would be the same for any group of networking BBS's.

Let's suppose that you are using SYSLINK and that you want to send TeleLink Network messages and files to several SYSLINKs that are NOT in PC PURSUIT cities. A new version of SYSLINK (which is being programmed at this writing) will be designed to send them all to, for example, the Chicago SYSLINK, and then the other SYSLINKs could <u>call into</u> Chicago using PC PURSUIT to receive their messages! In this way a SYSLINK in Chicago (or any of the other 13 PC PURSUIT cities) could become "holding tanks" for messages in transit for all of the SYSLINKs. The only difference from the present mode of operation is that some BBS's on the network would have to call in to receive their messages, rather than the BBS that is holding the messages calling out to deliver them.

Now, there are a couple of unusual things about the above scenario: First, any of the SYSLINKs that happened to be located in one of the the 14 major cities would not even necessarily have to have a PC PURSUIT account, since they would be RECEIVING calls only (however, the SYSOPs of those boards would probably want to have a PC PURSUIT account anyway, in order to communicate with SYSLINKs in other PC PURSUIT cities, and just to have the ability to call other BBS's in other cities). Second, it costs no more to call a distant PC PURSUIT city than a nearby one. While it might be wise to configure a BBS "network" so that no one BBS node carries ALL of the traffic, there is no cost reason that calls for a SYSLINK in Lansing, Michigan (for example) could not be directed first to, say, a Seattle node, since it would cost no more to connect with Seattle than to connect with Chicago. There would probably be practical reasons for choosing an intermediate BBS to route traffic through (such as amount of hard disk storage available, and avoiding congestion on any one node), but distance need not necessarily be part of the calculation.

Of course, there are many smaller cities that do not have a local Telenet node. Even so, BBS operators in those cities may wish to have their messages routed through another BBS in a city that DOES have access to Telenet. In that case, the outlying BBS could make ONE call each evening to the nearest BBS that is part of the same network and is signed up with PC PURSUIT, to send and receive messages. This would result in a three-tiered hierarchy for networking BBS interrelationship (in the U.S.A. only), which would look something like this:

1. PRIMARY BBS - Located in one of the 14 PC PURSUIT Cities.
2. SECONDARY BBS - Located in a city with a Telenet Dial-Up access number that is not one of the 14 PC PURSUIT cities.
3. TERTIARY BBS - Located in a city that has no local access to Telenet, or that does not choose to access PC PURSUIT.

So, a message sent from one tertiary site to another, distant tertiary site would originate at the tertiary BBS, be transmitted from there to a secondary BBS (by regular long distance), from there to a primary BBS (via PC PURSUIT), from there to another primary BBS (if necessary) via PC PURSUIT, from there to a secondary BBS (when it calls into the primary BBS) via PC PURSUIT, and from there to the destination tertiary BBS. Of course, BBS designers should try to design their systems so that as many of these transfers as possible can take place in a single evening, but in the above (worst-case) situation, it could take two or three days to deliver a message. Note, however, that each tertiary BBS would only have to make a single long distance call each night, to interchange messages with its designated secondary BBS (which would hopefully be located in a relatively nearby location).

(NOTE: When doing the type of "routing" mentioned above, keep in mind that interstate calls may be less expensive than intrastate calls of the same distance, depending on the state and the long distance carrier used. In other words, a tertiary BBS may find it less expensive to route traffic through a secondary BBS in another state, to take advantage of interstate rates!).

As I write this column, none of the networking BBS programs are able to make use of PC PURSUIT (including SYSLINK, but a new version that has this capability is in the works). However, with some minor rewriting of the program code by the authors of the networking BBS's, this type of low-cost networking should be a reality within a very short time.

I encourage all modem users in the United States (especially anyone with any connection with a BBS) to dial the PC PURSUIT online BBS at (800) 835-3001 and download all of the online files (there are only a few) for more information. If you sign up for PC PURSUIT, try accessing the Chicago SYSLINK BBS at (312) 622-4442 (if you can get through to it). [A short plug here - The Alternate Source is an authorized dealer for the SYSLINK BBS program.]

What about access to PC PURSUIT from outside of the United States? After all, in many countries you can get an account with the local PTT (Postal, Telephone & Telegraph authority) which permits you to use the local packet network as a "gateway" to U.S. packet networks such as Telenet. Once you've gotten through to Telenet from a distant country, can you then access a U.S. Bulletin Board by dialing out on PC Pursuit (assuming you have an account with PC Pursuit)? As I write this, the answer is, "no, but it's coming, as soon as all the details can be worked out." So, you BBS operators in major cities may soon be getting calls from really far-flung places (does your BBS software need revision to accommodate callers from outside the U.S. during the logon procedure?).

Although I have discussed the use of PC PURSUIT for accessing public BBS's and online services, it's also true that business users may find that a networking type BBS such as SYSLINK (when interconnected via PC PURSUIT) may be the ideal medium for receiving reports, proposals, sales information, etc. from representatives "in the field", since PC PURSUIT can now be accessed from any Telenet node (in almost all major- and medium-size cities).

In the previous issue of NORTHERN BYTES (Volume 7, Number 3, pages 19-22) I proposed a Common Message Base Protocol that would permit the interchange of entire message bases between BBS's. At the time I received several comments to the effect that such a system wasn't practical due to the high long distance charges that would be expended in transferring a message base from one system to another in anything approaching real time. Since the availability of PC PURSUIT effectively negates this argument, I hope this proposal (or something similar; I'm not saying that my proposal can't be improved upon) will now receive a bit more consideration from BBS authors (as a side note, I have heard that the FidoNet BBS's are experimenting with something called Echo Mail, which seems to have many similarities to what I proposed, but I have been unable to obtain much in the way of information on that feature at this writing). In the more distant future, I'm also waiting for the day when someone sets up a multi-user BBS, and then figures out how to link up two (or more) of them via PC PURSUIT for late-night conferences between users in distant cities, in real time!

Before I leave the subject of protocols, here's a comment from Birk Binnard, another PC Pursuit user: "I recently started using the Y-modem protocol and it runs about 95% of the calculated speed. You might suggest to your users that they use Y-modem anytime they can because it is MUCH faster than X-modem. The reason is that Y-modem sends 1024 byte blocks instead of the 128 byte blocks of X-modem. Thus, there is much less need for hand-shaking between the two computers. Y-modem is available on QMODEM and PROCOMM comm. software, and also on many BBS's."

Good idea, but at this writing I know of no TRS-80 based terminal programs that use YMODEM protocol (guess what kind of computers QMODEM and PROCOMM run on. Hint: Not TRS-80's!). C'mon, all you smart terminal program authors, how about giving us some of these newer protocols (what do you want to bet that before this issue gets into print, I'll discover about three such programs...). By the way, I personally feel that even 1024 bytes is not a large enough block size for "clean" connections that are plagued by network delays (especially when high speed modems are used). What's needed is an adaptive protocol that can transmit blocks of up to 4,096 bytes (or possibly even larger blocks) but that will automatically "fall back" to a smaller block size on noisy lines. An adaptive protocol (no block length that is "cast in stone") is the only logical choice to operate over both packet networks (relative "clean" connections but sometimes long network transmission delays) and standard long distance telephone circuits (almost instantaneous transmission, but with much more potential for error-causing line noise).

I hope you are encouraged to think about the capabilities of PC PURSUIT, and to use it for your computer-to-computer communications that have previously been just too expensive. If you sign up, please tell the Telenet representative that you are signing up on the recommendation of NORTHERN BYTES, and that you'd like to see them install a Telenet access node in Sault Ste. Marie, Michigan (so that WE can get online. I figure it can't hurt to badger them a little. Who knows, it may even help!).

FREE CALLING VIA RADIO

There are ways to communicate beyond your local telephone calling area for free - no, I'm not talking about phone phreaking, I'm talking about radio. You are probably aware that amateur radio operators ("hams") send packet radio messages around the world, however, this has a couple of drawbacks. The first is that government regulations say that you can't send messages of a commercial nature over amateur radio, meaning that if one of your BBS users tried to send a message to another node saying that his computer was for sale, and you transmitted it to another BBS via ham radio, you'd be in trouble. This prohibition against commercial traffic is enforced fairly strictly, and too many hams tend to be the sort that tattle to Big Brother whenever they hear another ham even THINKING about breaking a minor regulation.

That's the other drawback - there are all sorts of amateur radio operators, and while many of them are the "salt of the earth", there are also the "snobs" who think of amateur radio the way some people think of a country club - it's "their" fraternity, and let's face it, there are some people who simply do not belong there - such as anyone who happens to believe that learning the antiquated Morse code is a waste of time. I have met a fair number of amateur radio operators and while some of them were VERY nice people, the not-so-nice ones were generally the more vocal. I predict that this latter bunch is going to continue to turn potential newcomers away from ham radio, and their support in Washington is going to dry up as their numbers decrease. When that happens, the public and private interests are going to demand that some of the hams' cherished frequency space be given up to other services, which sorely need more room. Perhaps some of that space might even be allocated for computer communications by we of the "great unwashed" public (we can always hope so, anyway).

Because of the prohibition against anything even resembling commercial traffic, and the licensing requirements for potential operators, amateur radio does not seem to be a viable substitute to telephone lines and packet networks. However, a company called Electronic Systems Technology (1031 North Kellogg Street, Kennewick, Washington 99336 - telephone (509) 735-9092) has developed a device called the "ESTeem Wireless Modem". It operates at 2400 bps and sells for $1,095 retail. This unit does NOT operate on the amateur bands, but instead operates in the

72.040 - 72.960 MHz frequency range. I believe it may be considered a "business band" unit (although I am not certain of this), and you do have to apply to the Federal Communications Commission (on Form 574) for a license to use it (a form is included with the unit that, once filled out, gives you temporary authorization to use the modem on the air while you are waiting for your permanent license).

If, in fact, the unit operates on a frequency that is intended for business use, one might reasonably ask what the F.C.C. considers to be a "business". For example, if a BBS operator charges users a fee for access to the BBS, could that be considered a business operation? One thing you would probably NOT want to say to the F.C.C. is that you are charging to send messages, since that could make you a "common carrier" in the eyes of the government and subject you to all sorts of regulations.

The range of the ESTeem Wireless Modem is claimed to be over 30 miles line of sight, if an external 1/2 wave building mount antenna is used, and a repeater unit is available to extend that range. This could have interesting implications for BBS operators that are "just outside" the local calling area of a major city, as a radio modem could be used to "hop across" telephone exchange barriers.

As an example, let's suppose that we had a BBS operating here in Sault Ste. Marie, Michigan, and another operating just across the river in Sault Ste. Marie, Ontario (for purposes of this example, we won't worry about any special regulations for international communications that might come into effect - although it should be noted that ESTeem Wireless Modems are sold in Canada, and are currently in use between offices of a multinational firm in Nogales, Arizona and Nogales, Mexico. Apparently the Mexican phone lines are so bad that a regular modem wouldn't work). It should be noted that it is a long distance call between the two Sault Ste. Maries, even though the telephone central offices are probably less than five miles apart as the crow flies.

Now, our two Sault Ste. Marie BBS's would be connected to local phone lines in their respective cities in the normal manner, but let's suppose that both are ALSO connected (through a switching arrangement on the RS-232 port, or through a second RS-232 port, either of which would require additional hardware) to a Wireless Modem. Any time one of the pair of BBS's receives a message for the other, it sends it immediately via radio modem.

Let's further suppose that Sault Ste. Marie, Michigan has a local Telenet access number (wishful thinking?!). The Ontario BBS could then send any messages destined for BBS's in the U.S. to the Michigan BBS, where they could then be re-transmitted (to their destinations or to another intermediate BBS) using PC PURSUIT. Similarly, the Michigan BBS could pick up messages for the Ontario BBS when it calls into other systems, and then forward those back to the Ontario BBS via the Wireless Modem. And it would even be possible to have "real time" conferences between users of the two BBS's, if that capability were part of the BBS program.

What if the SYSOP of the Ontario BBS would like to be able to access Telenet directly, without using the Michigan BBS as an intermediary, in order to access CompuServe or some similar service? I tend to suspect that a hardware hacker could probably work out a way to connect a wireless modem "back to back" to a REAL modem in Sault Ste. Marie, Michigan, so that once the connection was made with the Michigan modem, dialing out on a regular Michigan telephone line could take place. This might open up access to many services at a lower cost than would previously have been available.

Can this sort of thing really be done? Is it perfectly legal? I suspect that it is (at least when communicating within the U.S.) but since I don't know of any real, working BBS systems that utilize this type of Wireless Modem, I can't really be sure. But if anyone is sufficiently motivated to do some further checking into this, I'd be interested to know what you find out. I do think there will be a lot more interest in Wireless Modems if and when the price drops some - although it's really not that extravagant when you consider that you're getting the equivalent of a 2400 bps modem and a two-way transceiver, all in one compact package.

By the way, the ESTeem Wireless Modem transmits data in packets, and has a built-in error detection and correction scheme that sounds remarkably similar to that used in amateur radio units. It occurs to me that we should probably discuss the technology used by amateur radio operators, both to understand how data communication via radio operates, and for the benefit of those who may not share my jaded view of amateur radio, and who may even be thinking of joining that fraternity!

AN INTRODUCTION TO PACKET RADIO

[The following information was downloaded from a Data Library of HAMNET on CompuServe, and edited only slightly, mostly to correct spelling errors. Unfortunately, the author's name was not given.]

Radio amateurs in Canada, Sweden, and the United States have been experimenting with packet radio, a system of computer-based communications. This new mode can provide high-speed communication with efficient use of the spectrum, and is resistant to interference due to other stations and to signal degradation due to adverse band conditions. Not only can packet radio be used for informal amateur QSO's [contacts] and traffic handling, but it has additional possibilities for exchange of data between hams with computers, "bulletin boards" and message systems, and remote computer programming.

WHAT IS PACKET RADIO?

Packet radio is a communication system in which information is digitally encoded. In this respect it is similar to RTTY or ASCII, but with important differences. These differences are the key to insuring error-free reception and at the same time allowing maximum use of the spectrum through shared frequency use.

Data integrity is provided by packet radio through a "handshaking" technique and error detection. Along with each transmission, a computed value called a "frame check sequence" (FCS) is sent, which allows the receiving station to check for errors. The receiving station acknowledges an error-free packet with a special acknowledgment (ACK) signal. If the sending station does not receive such a signal within a certain period of time, it automatically retransmits the packet.

A packet also contains identification of the destination station, permitting several QSO's to take place on the same frequency. A packet radio station can automatically ignore any packets which are not addressed to it. Due to the fact that the duration of most packet transmissions is very short, a user does not need the channel most of the time. The time between transmissions is available to other users on frequency. This system is called time-domain multiplexing. On a very busy channel, the user will notice an increased delay time before getting replies to transmissions, but the packet radio equipment will take care of automatic retransmissions and sorting out the replies meant for the station. The user never "hears" the QRM [interference].

WHAT IS A PACKET RADIO STATION?

Packet radio requires the use of a microprocessor-based controller at each station, and it will obviously appeal to the ham who already has a computer in his shack. However, it does not require that the operator be a programmer, or even that the station have a personal computer. All that is really necessary is a terminal, a terminal node controller (TNC), and an amateur radio transceiver.

The terminal can be a simple display (CRT) or typewriter terminal that produces ASCII characters, a personal computer, or even a commercial mainframe computer. What you need is a terminal with a keyboard to allow you to talk and a screen or a printer to allow you to read incoming information. You can even get an inexpensive terminal that uses a TV set for the display.

The way in which most terminals encode ASCII characters is in "asynchronous" format. Since characters are encoded as they are typed, there is a flag consisting of one or more "mark" (binary 1) values to mark the beginning and end of each character. The device decoding the characters expects a specific "baud rate", or number of transitions from "mark" to "space" (binary 0) per second during the character, but no particular time interval between characters themselves.

The terminal node controller is the heart of the packet radio system. It has one port that is connected to the terminal or computer, and communicates through it by asynchronous ASCII format at the baud rate required by the terminal. The TNC converts the data stream from the terminal to a packet by attaching a "header" showing the destination of the packet and control information for the network, a "tail" containing the result of the FCS calculation for error detection, and flags to mark the beginning and end of the packet.

The second port of the TNC connects it to the transceiver microphone and speaker audio lines, and the Push-To-Talk line. Ordinarily, the TNC will produce AFSK [Audio Frequency Shift Keyed] modulation by putting one of two tones into the microphone input, corresponding to a "mark" or "space". In this fashion, the packet is sent out on the air at the packet channel baud rate, which is unrelated to the terminal baud rate at the other port of the TNC.

The receiving TNC reverses this procedure, decoding the audio tones from the speaker audio line of the radio, removing and reading the header and tail information, and passing a successfully received packet to the terminal at the terminal baud rate.

The part of the TNC that does the translation between the sequence of tone levels and the characters is called a "modem", short for MOdulator-DEModulator. This device may or may not be built into the TNC board. Most packet radio modems operate at 1200 baud, which corresponds to about 1200 wpm, although the Federal Communications Commission now authorizes much higher baud rates on some amateur bands. The audio tones used are 1200 hz and 2200 hz. This choice of frequencies is that of the Bell 202 modem, which is available as surplus.

The final component of a packet radio station is an amateur radio transceiver. Most packet radio activity so far has been in the 2-meter band. The only important requirement of the radio is that its audio frequency response at 2200 hz be adequate. In other words, the 2-meter FM rig you already have is probably just fine.

WHAT THE TNC DOES

The TNC consists of a special purpose microcomputer, containing all the necessary software and hardware to communicate with your terminal, assemble a packet, operate your transmitter and receiver to send and receive a packet, and decode a packet. The special functions of the TNC which would be difficult to implement with an ordinary personal computer are the use of protocol to communicate with other TNC's and real-time control.

The encoding and decoding of packets involves a carefully standardized set of procedures called "protocol". The protocol basically determines the exact form of the header and tail parts of the packet. The header allows receiving TNCs to automatically determine the purpose of the packet, e.g., net check-in, part of a QSO, or ACK to a previous transmission. The tail contains the FCS which allows the TNC to automatically determine whether the packet was received correctly, and if so, to automatically acknowledge it. Since the protocol is programmed into the TNC, the operator does not need to know exactly what his packet looks like. In particular, he does not need to know how the destination of his packet is indicated. The operator communicates with other amateurs by call sign, and the TNC translates the call sign into the identification required by the protocol.

The TNC is required to perform a number of tasks simultaneously, including responding to events such as the receipt of a packet or instructions from the operator in "real time", in other words, as they happen. This makes programming in BASIC, the common language of personal computers, undesirable. This is because BASIC uses an "interpreter" which reads each line of the program and translates it into machine-type instructions every time the line is executed. The time required for the translation would prevent a program from responding rapidly enough in a packet radio environment. In order to meet the speed requirement, an assembly-language program or equivalent is required. While BASIC looks pretty much the same on any computer, assembly language is different for every machine. If the TNC were replaced by personal computers, program development would have to be redone for each variety of computer. In addition to maintaining the right pace, the TNC also must be constantly "listening" at both ports simultaneously while putting packets together or taking them apart. The hardware of personal computers may not even be capable of this sort of multi-task application.

Programming of individual TNC's must be as easy as possible, since there will inevitably be unforseen problems in the initial software. In addition, hardware changes may necessitate software changes. For this reason, TNCs are designed around erasable programmable read-only memories (EPROM's), which normally function like the ROM of a personal computer, where the vital software is stored in an indestructible form. However, if the need arises, they can be reprogrammed by "burning in" the new program using special equipment.

WHAT IS A PACKET?

A packet is the basic message unit in packet radio. It ordinarily consists of a text message typed in by the operator, sandwiched between the header and tail information required by the protocol. In a typical QSO, a packet would be encoded and sent out by the TNC when the operator ends a line of typing by hitting the RETURN or ENTER key. In any event, the length of a packet is limited, usually to 128 characters. This helps to prevent a single user from "hogging" the channel, as well as making sure that the sending and receiving TNC's don't get swamped with information.

A packet need not consist of ASCII or Baudot character strings, however. It could contain information in other coding systems, such as BCD or EBCDIC, or even binary data such as a compiled computer program. The TNC, which uses a "bit oriented protocol" based on a standard called High Level Data Link Control (HDLC), can encode any of these equally easily. An advantage to this choice of protocol is that the functions it requires are available on a single large-scale integration (LSI) chip, which simplifies the TNC hardware and software. A second advantage of HDLC protocol is that the beginning and end of the entire message are flagged, making the "start" and "stop" bits for each character unnecessary when the packet is transmitted in "synchronous" format.

The "frame" of an HDLC packet is represented below. Each field of the packet is encoded as a sequence of 1's and 0's (bits) to be transmitted as "mark" and "space" tones. With the exception of the DATA field, all these fields are generated by the TNC as it assembles the packet for transmission. The operator is concerned only with the contents of the DATA field.

Flag	Address	Control	Data	FCS	Flag

The FLAG is a unique bit sequence which identifies the beginning of a packet to the HDLC controller. This pattern corresponds to no sequence which would be encountered in any of the other fields, except possibly in the transmission of binary data. Even in this case, there are provisions for distinguishing data from the flag sequence.

The ADDRESS field contains routing information for the packet. This information may include the destination station, the originating station, and possibly intermediate routing information if the packet will be relayed to the destination. The destination and originating stations might be identified by a network address number or by amateur call sign, depending on the exact form of the protocol being used.

The CONTROL field describes the purpose of the packet to the network. It identifies packets with such functions as network check-in or check-out request, packet acknowledgments, or requests for information from net control. It may also contain a sequence number for a multi-packet message which must be received in the correct order.

The DATA field contains the message being sent, which will ordinarily be the text typed in by the user, converted into an ASCII data string. In the case of a packet identified in the control field as performing a control function, the DATA field may be absent.

The FCS allows the receiving station to verify that the packet has been received correctly. If the FCS calculated by the receiving TNC matches the FCS of the packet, an acknowledgment is sent; otherwise the packet is ignored.

WHAT IS A PACKET NETWORK?

A local area packet radio network (LAN) consists of a net control station and a number of individual operators. The net control station is sometimes referred to as the "station node" and the individual stations as "terminal nodes". The net may also contain a digital repeater or "digipeater", which may be the net control station or a separate repeater station. The repeater station may be a single-frequency simplex repeater which retransmits any correctly received packets, or it can be "normal" split frequency repeater.

As operators sign on to the net, they are recognized by the net control and given net address codes. An operator desiring to start a QSO with another net station will subsequently have his transmissions addressed to that station. Any operator may choose to have his TNC receive all transmissions, rather than just those addressed to his station. Of course, the TNC will only acknowledge those transmissions intended for that station. The operator whose station is functioning as net control participates in exactly the same way as other operators. The net control functions are taken care of automatically by his TNC.

As more packet radio LAN's become active, there will be the possibility of link stations with access to two distinct LAN's. These stations can be members of both nets and serve as communications links through which packets originating in one net can be funneled to an addressee in the other net.

A more sophisticated possibility is that of a "gateway" station, which will be a specialized station having access to some long-distance mode of communications. The gateway station will reformat packets with another layer of protocol containing inter-network linking information and transmit it to another gateway station in a distant LAN. Three possibilities are being explored for long-distance links.

TERRACON will be a high-speed ground-based linking system utilizing UHF and/or microwave relays. It could potentially handle most long-distance packet radio communications in the United States and Canada. It will probably be a few years before TERRACON is implemented as a useful system, and somewhat longer before the continent is linked.

AMICON will be a satellite-based network utilizing one of the special-services channels on the AMSAT Phase III-B satellite. AMICON will allow intercontinental linking and contact with isolated areas not accessible to TERRACON. High data rate experiments are being planned for the 23cm uplink/70cm downlink (mode L) translator. There are also plans for a packet radio digital repeater aboard the AMSAT Phase III-C satellite.

SKIPCON is AMRAD's projected HF network of LAN gateway stations. The nature of HF propagation will require slower data rates (75 to 600 baud) and error correction as well as error detection protocol. SKIPCON experiments have been conducted since the end of 1981.

HOW TO GET IN ON PACKET RADIO

There are currently two TNC designs available. The first packet radio TNC was designed by the Vancouver Amateur Digital Communications Group (VADCG). The Vancouver TNC is available as a bare board, and requires a power supply, and external modem, and parts. It comes with instructions and notes on the power supply. A modem kit is also available from VADCG. The TNC design is based on the Intel 8085 CPU and 8273 HDLC controller and includes 4K bytes of 2114 RAM and 4 K bytes of 2708 EPROM. The TNC requires an 8250 (serial ports) or an 8255 (parallel ports) for interface to the terminal, as well as an interface to the radio.

The Tucson Amateur Packet Radio group (TAPR) is currently testing a second TNC design. This TNC has the modem, radio interface, serial and parallel terminal interfaces, and power supply circuitry (exclusive of the transformer) on a single board. It is based on the Motorola 6809E microprocessor, and can hold a total of 48K bytes of RAM and ROM on the board. the 1933 HDLC chip it uses is compatible with the 8273 chip used on the VADCG board, and the TAPR TNC will be capable of VADCG-compatible protocol.

Additional information on TAPR activities is available from Tucson Amateur Packet Radio, P.O. Box 22888, Tucson, Arizona 85734.

[End of downloaded text]

It should be noted that since the above article was written, several commercial firms have introduced Terminal Node Controllers that are fully compatible with the TAPR unit. These units are advertised in the pages of amateur radio magazines (such as 73 and QST) for well under $200. Thus, a person who has a home computer but no amateur radio equipment at all could get into packet radio for around $500, possibly less. This assumes the purchase of a TNC, a transceiver, and possibly an inexpensive antenna. It also assumes that there are other amateur radio operators using packet radio in the vicinity of the potential packet radio user. This does not include any costs associated in getting a ham license (these costs are more in terms of time than money, since you have to know both Morse code and electronic theory - and your knowledge of digital electronics alone will not be adequate to get you a ham license). For more information on license study courses and/or amateur radio organizations near you, I suggest you contact Fred Maia, W5YI, editor of The W5YI Report, P.O. Box 10101, Dallas, Texas - Telephone (817) 461-6443.

MORE ON THE PUBLIC DIGITAL RADIO SERVICE PROPOSAL

Back in NORTHERN BYTES Volume 7, Number 1 (pages 3-5) we reprinted an article on the "Public Digital Radio Service" proposal by Donald L. Stoner, W6TNS. The September 1, 1986 issue of The W5YI Report mentions an article that appeared in the August 18 issue of InfoWorld, in which Ray Kowalski (Chief of the FCC's Special Services Division) is quoted as saying that a preliminary ruling on the Public Digital Radio Service proposal is scheduled for November. The W5YI Report goes on to say that this "almost assuredly means further consideration via a Notice of Inquiry (requesting further information from the public) or a NPRM [Notice of Proposed Rule Making], actually proposing PDRS implementation." So the issue is not dead yet, and I again encourage all NORTHERN BYTES readers to write to the FCC in support of this proposal!

In addition, this proposal seems to have been warmly received in Australia, as evidenced by the following comments by the editor of the Microcomputer Club of Melbourne (MICOM) newsletter:

"Well, ignoring the 'freedom of speech' outcry of the NORTHERN BYTES editor, and just talking about the proposal for a PDRS, I think its one of the best ideas I've ever seen. I've been involved in the amateur radio packet radio movement virtually from its outset in Australia, and I have a better understanding of the technical aspects that the NORTHERN BYTES editor.

"The service as described WILL NOT be a threat to the postal service, AUSTPAC or any other commercial service. It simply will not provide the level of service that a business would need or accept. Packet transfer would not be instantaneous. Given the short range that 1 watt power output would provide, a packet would have to pass from one node to another until it arrived at its destination. That could take a fair while, even at 1 million bits per second. The other major problem for business use would be privacy. Any radio transmissions are by their very nature, open for all to see. A talented electronics hobbyist could "pluck out" any packets from the airwaves, regardless of whether they were addressed for him or not. I really can't see banks and financial institutions giving up Austpac for PDRS!

"On the technical side, the frequencies in use for TV here are not quite the same as in the U.S.A. Channel 2 is not a problem here, it is channel 0 intrudes into the six metre band (50-54 Mhz). Since channel 0 has now gone forever, amateurs can use the band without interference...

"So what are the chances of a similar service in Australia? Unfortunately, almost none - UNLESS we can get sufficiently large numbers of people to push for an Australian PDRS... Enquiries can be directed to Peter Jetson at Micom, PO Box 60, Canterbury, 3126, Victoria, Australia. I am prepared to act as a focal point in the push for an Australian Public Digital Radio Service. For the record, I am also a licensed radio amateur, my call-sign is VK32MB."

Here in the U.S., several comments on this proposal were filed with the Federal Communications Commission by interested parties, and Mr. Stoner has filed a response to the opposition comments. Because many of our readers may be interested in this response, I am reprinting it here:

Before the FEDERAL COMMUNICATIONS COMMISSION, Washington, DC 20554, in the matter of RM-5241 - Creation of a new radio class and allocation of spectrum for the owners of personal computers.

TO: The members of the Commission

RESPONSE TO OPPOSITION COMMENTS FROM THE PUBLIC, ARRL AND MST

FILED BY Donald L. Stoner, W6TNS, January 20, 1985

OVERALL IMPRESSIONS

The petitioner anticipated significant opposition on the part of radio amateurs. It was a pleasant surprise to find that this was not the case. Responses from the general public (both amateur and non-amateur) were slightly more than 2 to 1 in favor of the concept (or some variation of it). Hopefully, this proportion is reflected in the comments received by the Commission. The ARRL response was the only comment which denied the need for this public service.

A letter from Harold A. Helms, Ph.D. (WA4QLA) is typical. Dr. Helms states "I have no fundamental objection to the concept of a PDRS; in fact, I would probably use such a service myself".

Several amateurs, Dr. Helms included, felt the service should be located in the UHF range.

On the whole, the petitioner was quite pleased with the public comments. The opposition comments can be addressed as part of my response to the submission made by council for The American Radio Relay League.

COMMENTS OF THE AMERICAN RADIO RELAY LEAGUE

The League attempts to make four principal points in their submission. They are:

1. The League denies that the public needs such a service. Their response states "(a) the Amateur Radio Service already provides for communication between computers using packet radio, therefore another radio service is unnecessary;"

2. If the computer public wants access to the airwaves, they must earn an amateur radio license. The League states that "the Amateur Radio Service has the infrastructure and discipline needed for long-term viability of personal computer communications."

3. By means of an amateur radio license, the public will have access to the "orderly development of a data-communications network capable of handling the types and volumes of traffic which may be presented by the computer hobbyists;"

4. The 52-54 segment of the six meter amateur band is adequately populated. The contention of the petitioner, that this band is "underoccupied", is erroneous.

The League does not seem to be totally convinced of the first three points. In fact, they state in their "ARRL Letter" dated 2 January; "It's not the proposal for the service that's the bone of contention, understand - it's just that Amateur Radio intends to go right on keeping that top two megahertz of six exclusively within the Amateur Radio realm".

These points will be examined and this response will show that the League's position is not in the public interest, convenience and necessity.

The Need(1)- The League contends that a PUBLIC DIGITAL RADIO SERVICE is unnecessary and denies that the general public wishes to communicate with their computers.

However, they destroy their credibility by stating: "Amateur packet radio operation is rapidly developing; it is extremely popular". They further state: "The League wishes to point out that there has been a remarkable growth in amateur packet radio over the past few years...". Finally, they provide statistics: "Packet radio in the Amateur Radio Service now stands at about 10,000 units in the field, in contrast to around 4,000 in early 1985".

The statements represent an interesting exercise in convoluted logic in view of their footnote statement: "the entire Petition is no more than an unsupported series of conclusory musings of the Petitioner. There is not one iota of ascertained need for either the proposed service or, therefore, for the allocation of spectrum therefor".

There is something illogical here! The League would have the reader believe there is no desire on the part of the public to have computer-to-computer communication. Then, upon earning an amateur radio license, the exchange of information via computers suddenly becomes viable and desirable!

As pointed out in the petition, there are no conclusive facts and figures available to prove that the public desires computer communications. Inadvertently, the League has proven the basic premise of the petition far more effectively than my "conclusory musings".

The Ham License(2)- The League contends that if an individual desires computer communications, they must have an amateur radio license. According to the League only the Amateur Radio Service provides the "infrastructure and discipline needed for the viability of personal computer communications".

Where is it written that a ham license bestows discipline upon the holder? One need only listen to the 75 meter amateur band to refute this argument. A quick tour of two meters will reveal that good taste, manners and courtesy are not enclosed when one receives the envelope containing an amateur radio license. These things cannot be printed on paper any more than technical expertise can be obtained by studying to pass the Novice/Technician examination.

An amateur license is extremely easy to obtain thanks, in part, to the Commission Rules and Regulations. However, one must face the fact that some people have no desire to enter the wonderful world of amateur radio. It is arrogant to force people to become radio amateurs in order to achieve access to the airwaves. Why not insist that those who fly model airplanes, use Cellular telephones or open their garage doors also become licensed radio amateurs? It is basically dishonest to leverage the ranks of the amateur radio by force and, as a byproduct, increase League membership.

Amateur Packet Radio is Adequate(3)- The PUBLIC DIGITAL RADIO SYSTEM, as proposed, is not simply an adaption of the amateur packet radio system. While it is conceptually similar, the PUBLIC DIGITAL RADIO SERVICE features significant improvements.

It is safe to say that the creators of the amateur packet radio concept never envisioned that it would become so popular so quickly. A growth from 4,000 to 10,000 units in the short space of 12 months is truly incredible.

Unfortunately, this dynamic growth has also revealed a flaw in the system. The data exchange rate is slow, even with a limited number of users. This appears to be a major complaint nationally among "packeteers".

Local systems slow down to an annoying point, when more than 8-10 users are on a given frequency. Some users "spin off" their activities and move to another frequency. The use of many channels makes the automatic delivery of messages extremely difficult. It becomes necessary to propagate the same information on many channels. This is not spectrum efficient.

The use of high-site packet repeaters has severely limited the number of simultaneous users. These elevated "digipeaters" can probably never be eliminated because of the relatively small number of radio amateurs using packet radio in a given area. The powerful high-site digital repeater provides communication over large areas but precludes any more than one user on a given frequency at any given time.

The solution, of course, is a network of very low power stations which directly communicate over very limited distances. Those in one area can communicate simultaneously without interference to those in another area.

There is a second problem which is even more serious. This is the slow rate at which information is exchanged. Presently communications are conducted at 1200 baud (approximately 120 characters per second). As an inducement to become a radio amateur "packeteer", the League offers a digital Valhalla (at some future date) of 19,200 baud.

If the growth of the PUBLIC DIGITAL RADIO SERVICE parallels that of the amateur radio service, the network would soon be overloaded even at 19,200 baud. There is reason to believe that the growth would exceed that of amateur radio since no licensing would be involved. It is also likely that many radio amateurs would utilize the network, as indicated by the public comments. This provides a unique opportunity to introduce the public to other exciting areas of amateur radio.

As pointed out in the petition, it is essential that the popularity of the service be anticipated. Provision must be made for an adequate number of users from the outset. Designing the network for limited distance communication with high data rates will accommodate an unlimited number of users. Clearly, the amateur radio packet network does not qualify for either requirement.

Six Meter Activity(4)- Some confusion resulted from a typographical error in the third paragraph on page -9-. The sentence should read "....less than 1,000 are active on this portion of the six meter band". Even with this correction, the League has shown that activity is probably higher than stated in the petition. The estimates were made by extrapolating sales of commercial equipment for this band. Overlooked were the numerous conversion of obsolete 30-50 mHz low-band FM two-way radios to operate on six meters.

However, a closer look at the six meter repeater list supplied by the League is interesting. Ten of the repeaters listed are in Canada. Further, only 24 of the 50 states have more than one repeater. In 20 states there are no repeaters. Clearly, the band is underutilized in these areas.

The repeater situation in the State of Washington is typical. According to the Western Washington Amateur Repeater Association, there are 60 repeaters on 2 meters, 26 in the 220 mHz band, 51 on the 432 mHz, yet there are only 9 active repeaters on 6 meters.

The repeaters generally consist of modified surplus low-band FM equipment. A minimum of financial hardship would result from moving repeater activities to another amateur band. It would also appear that repeater usage could be accommodated in the 50-52 mHz portion of the six meter band.

While the League has shown there is more activity than previously supposed, they have failed to prove that the 52-54 mHz portion of the six meter band is adequately utilized or why it should not be allocated to the PUBLIC DIGITAL RADIO SERVICE.

COMMENTS OF MST (MAXIMUM SERVICE TELECASTERS)

The response from MST opens with a preconceived assertion. On the first page we learn of "the virtual certainty that Petitioner's proposal would cause significant new interference to television Channel 2". This notion is "proven" by impeccable mathematics, based on dubious assertions. The preordained conclusion is reached that "operation of these radio modems at one watt output power would cause perceptible levels of interference to television receivers within a 2.2 mile radius". This contention is preposterous.

At the outset, MST agrees with the petitioner that amateurs have avoided the 52-54 segment of the six meter band. This is done to support their position that signals in this region cause television interference. MST also mentions other sources of channel 2 interference, such as sporadic "E" reception, which have no bearing on the petition and which are beyond the control of the petitioner.

Concluding their response, MST joins forces with strange bedfellows to contradict themselves. Footnote 1 reads: "Further, as indicated by the comments of the American Radio Relay League, filed this same date, the former belief is likewise unfounded" (my contention that the 52-54 mHz portion of the six meter band is underoccupied).

The ARRL contends there is considerable activity on the six meter band. Transmitters used by amateurs typically vary from 10 watts to 1,000 watts. If one accepts the MST statement that a one watt transmitter on adjacent frequencies will cause interference within a radius of 2.2 miles, then one must logically conclude that radio amateurs cause even greater interference and over greater distances by virtue of their high power equipment. However, we know this is not the case and the assertions of MST are faulty.

It should be mentioned that transmitting equipment used by radio amateurs is not subject to FCC specifications. In particular, "home brew" and modified commercial equipment is seldom measured by the user for spectral purity of emissions.

A Short Course in TVI- Television interference results from two principal sources, i.e., (1) overload of the television receiver r.f. amplifier stage, which causes cross modulation and (2) emissions outside the authorized bandwidth which fall within a television channel.

Commission personnel can prove that a one-watt signal will not cause television interference, within the constraints mentioned in the petition. This can be simply demonstrated by operating a hand-held transceiver in the vicinity of various television receivers. Such a hand-held product may be difficult to find, however, due to the low volume demand for commercial six meter equipment. However, I understand SanTec makes such a transceiver.

The MST submission suggests that a wideband digital transmission will cause more interference than would be generated by the hand-held test described in the previous paragraph. However, the petition states that all spurious emissions outside the 52-54 mHz band must be suppressed by approximately 60 db. Thus, it matters little whether the origin of the spurious is carrier, modulation, data bits or rock and roll music. 60 db of suppression is still 60 db.

Note that video information is also wideband and the radiated spectrum resembles a high speed data transmission. Yet television signals coexist side-by-side without mutual interference as proven by thousands of cable television systems.

If the Commission is not comfortable with 60 db, make the specification 70, or even 80, db. While this makes the design of the equipment more difficult, it can be done. Further, the equipment cost will still be less than for UHF equipment with a more relaxed specification.

MST "clutches at straws" throughout the response. They state "modem operators, particularly in rural area, might find it necessary to boost their operating power to well above one watt in order to reach the nearest transceiver". MST assumes that if a law is passed it will be broken, therefore don't pass the law. They also mention large antennas, man made noise, and so on to support their contention that the service should be at some higher frequency.

Other respondents have made the same suggestion but they do not provide the Commission with any suggestions or guidance as to where this "promised land" might be. The attitude is "they seem like nice folks, I just don't want them living in my neighborhood".

Conclusion - The purpose of my petition was to point out the need for computer-to-computer communications for the public, present certain minimal specifications and identify spectrum space where it could be accommodated.

I had naively hoped that a way could be found to provide for this need within the Amateur Radio Service, as suggested on page 21 of the petition. The League, however, has driven a stake through the heart of this codeless vampire for once and for all.

I believe the Commission recognizes there is a need for the PUBLIC DIGITAL RADIO SERVICE. The implementation of the concept [illegible] far more important than the location of the PUBLIC DIGITAL RADIO SERVICE within the radio spectrum. Technically, it can be located anywhere within the spectrum up to 1 gHz, or so. While equipment costs would be higher at these frequencies, this would not significantly retard the growth of the service in the long term.

If the 52-54 mHz portion of the band is not suitable, where can the service be located? Above the frequencies of concern to MST is the sacred ground of the Cellular Service. Further up is the new 902 mHz amateur band. The PUBLIC DIGITAL RADIO SERVICE could easily be accommodated in the 922-928 portion of this band. This is above the proposed repeater frequencies, does not involve international treaties and television interference would be nonexistent.

Above this band, the satellite industry can be expected to protest another incursion. As we continue to "move to the back of the bus", communication by light beam seems the only answer.

I hope it is not and the Commission can identify a small segment of spectrum where the PUBLIC DIGITAL RADIO SERVICE can be located.

In closing, the writer would like to thank the Commission for the opportunity to respond to the negative comments on RM-5241.

Sincerely, Donald L. Stoner
6014 East Mercer Way, Mercer Island, Washington 98040

GIVING OVERSEAS USERS THE SHAFT

Arne Rohde passed along a copy of a letter he received from CompuServe, in response to an inquiry about the availability of their service in New Zealand. The letter states that "There is a $10.00 CompuServe <u>monthly</u> service fee for subscribers with addresses outside the United States or Canada. International subscribers have the billing option of using an international Visa, Mastercard or American Express." (underline added). Since the credit card companies don't charge anything extra to process card billings outside the U.S., I couldn't understand the basis for the monthly service charge. Well, a call to CompuServe revealed that this additional amount covers the costs of sending the monthly statement and the subscriber's copy of "Online Today" to a foreign address. Just as a point of comparison, we can send a copy of NORTHERN BYTES overseas for only $1.00 to $2.00 more than the cost to send a copy to a U.S. address - and that's air delivery! I suspect that CompuServe's actual additional cost per overseas subscriber is maybe $3.00 per month tops. Overseas users are already heavily surcharged by their local packet networks for their per-minute access to CompuServe, so why this additional dig into subscriber's pocketbooks?

Arne reports that he has used the BIX (Byte Information Exchange) system run by McGraw-Hill a couple of times, and they have no monthly minimum charge (although there is an initial registration fee). I think that CompuServe had better take a good, hard look at their pricing policies or they are going to become a dinosaur. Many TRS-80 users have already fled CompuServe for services such as GEnie and Delphi, and this trend is likely to continue (who wants to pay more money than necessary to stay online?).

SYSOP - BBS USER RELATIONSHIPS

A closing thought - it seems to me that there are people who shouldn't be SYSOPs of a BBS, because they are just a little too intolerant (or maybe too suspicious) of errors made by users. These people assume that EVERYBODY is a "cracker" or a "twit" until proven otherwise. Perhaps in some areas the local community of BBS users (taken as a whole) <u>has</u> done much to earn a poor reputation, but you cannot tar everyone with the same brush.

As just one example - many SYSOPs get REAL upset if you hang up without going through the "proper" logoff procedure. But there are MANY things that can cause the phone connection to break, from the dog knocking an extension phone off the hook (this actually happened to me once in Lansing) to an airplane flying between microwave towers, thus momentarily interrupting transmission. And, if you are calling long distance, does it really make sense to call back just to terminate the call "properly", if you were almost through anyway?

What about BBS users that call and then disconnect after a few seconds? Some SYSOPs immediately assume that they are "phone phreaking" on a cheap long distance service. Well, I have seen ALL of the major long distance carriers (<u>including</u> AT&T) disconnect modem calls for no apparent reason after a few seconds (back in June we found that on modem calls from Lansing, Michigan to Providence, Rhode Island, <u>only</u> MCI would give us a reasonably good connection. AT&T would disconnect after a second or two, and Sprint gave us too many "garbage" characters along with the data).

Then there are the users who send control characters, trying to crash the system. Or are they? Could it just be good old-fashioned LINE NOISE sending random characters? Even AT&T is not immune from this malady anymore. And some terminal programs AUTOMATICALLY send certain control characters (notably XON and XOFF) at various times, without any intervention by the user.

Then there are the users that just go away and let a system "time out" after a download. Well, maybe. But let me tell you about the MODEM80 terminal program. If MODEM80 receives an XOFF character from the host computer (or line noise that looks like an XOFF), it will not send ANY more characters to the host until an XON character is received - NOT EVEN CHARACTERS TYPED FROM THE KEYBOARD!! There is a way to force a restart in MODEM80 (you press BREAK to go to the menu, then T to go back to the terminal mode) but this is not readily apparent, and the novice user of that terminal program will most likely become very frustrated trying to get the BBS to respond to his typing - while at the same time the SYSOP is thinking that they guy just walked off and left the computer running!

Now I realize that some BBS users do in fact seem to have all the manners of a pig at the trough, and that after seeing a few of those, a SYSOP is going to get real suspicious of anything a caller may do. But I must plead for a little mercy, especially for long distance callers. It is a real pain to call a BBS long distance, have the connection break on you for no apparent reason (or for circumstances beyond your control), and then to be told that you can't log back in because you've had your one allowable call for the day or because you didn't terminate your previous call properly!

BBS users can do a lot to help their SYSOP keep a good attitude. Leave a message telling your SYSOP how much you appreciate his board every now and then, and if you download a few files, leave a note saying "thanks!" And, if at all possible, try to PARTICIPATE - leave a few messages on the message base, or upload a few <u>good</u> public domain programs to the board. In short, go out of your way to show a little common courtesy - it will work wonders and might even keep your favorite BBS around a while longer. Finally, operation of a BBS usually requires an expenditure of funds (for telephone line and long distance charges, and to replace worn-out disk drives and upgrade the hardware and software) and this usually comes from the SYSOP's pocket, so if you're having a good year financially and are getting lots of enjoyment and/or benefit from a particular BBS, you might consider sending a contribution to help cover operating expenses (especially if the board is one of the diminishing number that does not charge any kind of subscription fee for use!).

SUPERDOT - A PATCH FOR DOTWRITER 4.0

by Raymond J. Day

9601 Morton Taylor Road, Belleville, Michigan 48111

This patch for Dotwriter 4.0 allows you to get even better print then before. Now, when the double strike command is used, (.DA on) a vertical size reduction of approximately one-half is achieved with the same amount of detail.

The program is designed to separate every other bit going out to the printer so that only four bits in a byte (every other one) is printed out. The printer is then given a line feed of 1/216 and the other four bits are printed out. It works like NLQ, superscript and subscript modes. I think a lot of people would like it. I gave it to the Dearborn TRS-80 Users Group and the editor printed out some of the news letters with it.

I have a couple of printers that are EPSON compatible and can print 1920 dots across 8 inches. I use the following patch to do it: To change SUPERDOT/CMD to print in compressed mode, from MULTIDOS READY type:

```
PATCH SUPERDOT/CMD (REC=97,BYTE=173) 90
```

If you need to remove the above patch, type:

```
PATCH SUPERDOT/CMD (REC=97,BYTE=173) 76
```

It is half as wide but Dotwriter will not let you do the full 1920 dots across (but it comes close).

[Editor's note: TAS Public Domain Library disk # 022 contains the following files:

NEWS/PR	Dotwriter font file created by the author.
NLQ/PR	Dotwriter font file created by the author.
SUPERDOT/ASM	Source code for the SUPERDOT patch.
SUPERDOT/DOC	Documentation file.
SUPERDOT/IDO	MULTIDOS format patch file for DOTWRITER 4.0
SUPERDOT/OBJ	Object code for the SUPERDOT patch.
SUPERDOT/SMP	Demonstration text file for use with SUPERDOT.

The SUPERDOT/IDO file is reprinted below. If you are using MULTIDOS, just have a copy of DOTEPS/CMD on a disk in one of your drives, and DO SUPERDOT/IDO. Otherwise, you can use the printout of SUPERDOT/IDO as the basis for making the patches using a ZAP type program. If you have the PD Library disk mentioned above and a machine language monitor program (such as TASMON) you can load DOTEPS/CMD to memory, then load SUPERDOT/CMD, and then save the combined program back to disk. Be sure to always make your patches on a COPY of the original program, NOT ON YOUR MASTER DISK, in case something goes wrong!]

```
This PATCH is for MULTIDOS. It makes the file DOTEPS/CMD
into a file that will print the lettersets like NLQ print
just give it the command '.DAON' to get it. This is for
DOTWRITER 4.0.  This PATCH was done by Raymond Day
PATCH DOTEPS/CMD (REC=2,BYTE=20) 49
PATCH DOTEPS/CMD (REC=3,BYTE=208) 53.47.49.55.47.56.54
PATCH DOTEPS/CMD (REC=93,BYTE=19) 121
PATCH DOTEPS/CMD (REC=97,BYTE=62) 245.58.232
PATCH DOTEPS/CMD (REC=97,BYTE=69) 55.230.240.254.48.32.247.58
PATCH DOTEPS/CMD (REC=97,BYTE=77) 141.5.254.121.32.5.241.50
PATCH DOTEPS/CMD (REC=97,BYTE=85) 232.55.201.241.211
PATCH DOTEPS/CMD (REC=97,BYTE=99) 30
PATCH DOTEPS/CMD (REC=97,BYTE=113) 42.205.146.204.62.1.205.20
PATCH DOTEPS/CMD (REC=97,BYTE=121) 204.62.13.205.20.204.175
PATCH DOTEPS/CMD (REC=97,BYTE=128) 50.238.210.205.121.204.205
PATCH DOTEPS/CMD (REC=97,BYTE=135) 30.205.62.1.50.238.210.42
PATCH DOTEPS/CMD (REC=97,BYTE=143) 250.188.18.248.188.205.146
PATCH DOTEPS/CMD (REC=97,BYTE=150) 204.62.10.205.20.204.42
PATCH DOTEPS/CMD (REC=97,BYTE=157) 246.188.125.180.192.62.13
PATCH DOTEPS/CMD (REC=97,BYTE=164) 205.20.204.62.27.205.20
PATCH DOTEPS/CMD (REC=97,BYTE=171) 204.62.76.205.20.204.58.6
PATCH DOTEPS/CMD (REC=97,BYTE=179) 189.205.20.204.58.4.189
PATCH DOTEPS/CMD (REC=97,BYTE=186) 205.20.204.195.204.204.62
PATCH DOTEPS/CMD (REC=97,BYTE=193) 27.205.20.204.62.51.195.20
PATCH DOTEPS/CMD (REC=97,BYTE=201) 204.0.0.71.42.40.189.125
PATCH DOTEPS/CMD (REC=97,BYTE=209) 180.120.194.20.204.111.58
PATCH DOTEPS/CMD (REC=97,BYTE=216) 238.210.183.125.32.2.7.111
PATCH DOTEPS/CMD (REC=97,BYTE=224) 230.128.103.125.7.230.64
PATCH DOTEPS/CMD (RFC=97,BYTE=231) 180.103.125.7.7.230.32.180
PATCH DOTEPS/CMD (REC=97,BYTE=239) 103.125.7.7.7.230.16.180
PATCH DOTEPS/CMD (REC=97,BYTE=247) 195.20.204.58.238.210.183
PATCH DOTEPS/CMD (REC=97,BYTE=254) 200.175
PATCH DOTEPS/CMD (REC=98,BYTE=78) 24.232.243.42.58.189.125
PATCH DOTEPS/CMD (REC=98,BYTE=85) 180.40.50.42.230.188.125
PATCH DOTEPS/CMD (REC=98,BYTE=92) 180.40.43.33.0.0.34.230.188
PATCH DOTEPS/CMD (REC=98,BYTE=101) 42.2.189.237.91.250.188
PATCH DOTEPS/CMD (REC=98,BYTE=108) 175.237.82.34.148.188.125
PATCH DOTEPS/CMD (REC=98,BYTE=115) 180.40.20.42.2.189.93.84
PATCH DOTEPS/CMD (REC=98,BYTE=123) 27.237.75.148.188.26.182
PATCH DOTEPS/CMD (REC=98,BYTE=130) 119.43.27.11.121.176.32
PATCH DOTEPS/CMD (REC=98,BYTE=137) 246.42.250.188.34.140.188
PATCH DOTEPS/CMD (REC=98,BYTE=144) 42.2.189.229.42.140.188
PATCH DOTEPS/CMD (REC=98,BYTE=151) 126.205.158.204.42.140.188
PATCH DOTEPS/CMD (REC=98,BYTE=158) 35.34.140.188.193.197.3
PATCH DOTEPS/CMD (REC=98,BYTE=165) 175.237.66.56.234.193.201
PATCH DOTEPS/CMD (REC=104,BYTE=41) 245
PATCH DOTEPS/CMD (REC=104,BYTE=46) 58.24.240
```

[redacted]

by Terry Bibo

[Reprinted from the Canberra Micro-80 Users' Group newsletter (113 Owen Dixon Drive, Canberra, A.C.T. 2617, Australia):]

Some of us have had trouble with the clock on the Model 4 running at 60 hertz on TRSDOS 6.2.x, which gives 72 seconds in every minute. This was solved in TRSDOS 6.1.x by running a HERTZ/JCL file to patch the relevant byte for 50 hertz Australian power. No such JCL file was on TRSDOS 6.2.x and we were confused.

The answer is in a SYSTEM command: SYSTEM (HERTZ 5) solves the problem. To make a permanent patch to the disk, SYSGEN it after the SYSTEM command.

[redacted]

by Bill McQueen

This is the source code for WPFLT/FLT, in MISOSYS EDAS or MRAS listing file format. The documentation for this filter is in the source code comment lines.

```
                00005 ;
                00006 ;
                00007 ;This filter is for use with LDOS 5.1.x and printer listings
                00008 ;of assembly language source code.  During the code portion
                00009 ;of a line, i.e. before the semi-colon, this filter has no
                00010 ;effect on the character stream sent to the printer.
                00011 ;However, between the semi-colon and the carriage return
                00012 ;(i.e. the comment section of the line) the filter switches
                00013 ;into a "word processing" mode.  During this mode, any time
                00014 ;AN "ESCAPE CHARACTER" IS SENT TO THE PRINTER, THE FILTER
                00015 ;causes the first character following the "escape character"
                00016 ;to be interpreted as a control code and toggles certain
                00017 ;printer modes (e.g. underlining) ON/OFF.  This allows
                00018 ;the writer to enhance the comments by highlighting importan
                00019 ;information.
                00020 ;
                00021 ;The "escape symbol" used in this listing is the "@". This
                00022 ;may be changed as desired by changing the defination for
                00023 ;ESC_SYM in the EQUATES list.
                00024 ;
                00025 ;The following word processor functions are supported by
                00026 ;this program as written (Others may be added as desired):
                00027 ;
                00028 ;        @-      Toggles underlining On/Off
                00029 ;        @+      Toggles Bold Face ON/OFF
                00030 ;        @/      Toggles Italics ON/OFF
                00031 ;        @@      Prints the "@" character
                00032 ;
                00033 ;----------------------------------------------------------
                00034 ;
0040            00035 ESC_SYM EQU     '@'             ;Escape Symbol to be used
                00036                                 ;in text to be printed. May
                00037                                 ;be changed as desired.
 00A            00038 LF      EQU     10
 00D            00039 CR      EQU     13
0018            00040 ESC     EQU     27
00F8            00041 PRPORT  EQU     0F8H            ;Model III printer port
4020            00042 @EXIT   EQU     4020H
4030            00043 @ABORT  EQU     4030H
4411            00044 HIGH$   EQU     4411H
4467            00045 @DSPLY  EQU     4467H
428A            00046 @LOGOT  EQU     428AH
000B            00047 @WHERE  EQU     000BH
                00048 ;==========================================================
                00049 ;       The following code is the relocator and is discarded
                00050 ;       after the filter is is place and activated.  The
                00051 ;       program loads at 5200H and the filter proper is
                00052 ;       relocated to high memory just below the current
                00053 ;       HIGH$, then HIGH$ is modified to protect the
                00054 ;       filter. A standard header is put in place with
                00055 ;       the filter so that it is recoginized by high
                00056 ;       memory mapping programs.
                00057 ;
                00058 ;
5200'           00059         ORG     5200H
5200' 1A        00060 ENTRY   LD      A,(DE)          ;get device type
5201' E602      00061         AND     2               ;make sure it is an
5203' 2840      00062         JR      Z,NOGOOD        ;output device.
5205' D5        00063         PUSH    DE              ;save device DCB.
5206' 215B52'   00064         LD      HL,MSG          ;point to initialization
5209' CD6744    00065         CALL    @DSPLY          ;mesage and display it.
520C' DDE1      00066         POP     IX              ;recover device DCB.
520E' 2A1144    00067         LD      HL,(HIGH$)      ;reduce HIGH$ by the
5211' 226853'   00068         LD      (STORE),HL
5214' 012001    00069         LD      BC,LAST-START   ;length of this driver.
5217' AF        00070         XOR     A               ;clear the carry flag.
5218' ED42      00071         SBC     HL,BC           ;calculate new HIGH$.
521A' 221144    00072         LD      (HIGH$),HL      ;driver now protected.
  1D' 23        00073         INC     HL              ;point HL at new start.
 21E' DD7E01    00074         LD      A,(IX+1)        ;xfer orig DCB vector
5221' 328B53'   00075         LD      (PUTBXT+1),A    ;to driver CALL.
5224' 328E53'   00076         LD      (GETBYT+1),A
5227' DD7E02    00077         LD      A,(IX+2)
522A' 328C53'   00078         LD      (PUTBXT+2),A
522D' 328F53'   00079         LD      (GETBYT+2),A
                00080 ;---------------
5230' E5        00081         PUSH    HL
5231' AF        00082         XOR     A               ;clear carry
5232' 116653'   00083         LD      DE,START
5235' ED52      00084         SBC     HL,DE           ;hl = new start - old
5237' AF        00085         XOR     A               ;clear carry
5238' 119A53'   00086         LD      DE,WP_CTL
523B' ED5A      00087         ADC     HL,DE           ;hl = new WP_CTL
523D' 22A153'   00088         LD      (MARK1+2),HL
5240' E1        00089         POP     HL
                00090 ;--------------------------------
5241' F3        00091         DI                      ;not during update.
5242' DD7501    00092         LD      (IX+1),L                ;update DCB Vector
5245' DD7402    00093         LD      (IX+2),H                ;to filter entry.
5248' EB        00094         EX      DE,HL           ;xfer new START to DE.
5249' 216653'   00095         LD      HL,START        ;load address of driver.
524C' EDB0      00096         LDIR                    ;move driver to top.
524E' FB        00097         EI                      ;enable interrupts again.
524F' C32040    00098         JP      @EXIT           ;return to LDOS READY.
                00099 ;
                00100 ;*************************************************
                00101 ;
                00102 ;       Error Handling Routine
                00103 ;
5252' 214353'   00104 NOGOOD  LD      HL,ERR_MSG
5255' CD6744    00105         CALL    @DSPLY
5258' C33040    00106         JP      @ABORT          ;abort to LDOS.
525B' 0A        00107 MSG     DB      LF,LF,'          Installing'
      0A 20 20 20 20 20 20 20 20 20 20 49 6E 73 74 61
      6C 6C 69 6E 67
5271' 0A        00108         DM      LF,'    Word Processing Filter'
      20 20 20 20 57 6F 72 64 20 50 72 6F 63 65 73 73
      69 6E 67 20 46 69 6C 74 65 72
528C' 0A        00109         DM      LF,'             for'
      20 20 20 20 20 20 20 20 20 20 20 20 20 66 6F 72
5290' 0A        00110         DM      LF,'  Assembler Printer Listings'
      20 20 41 73 73 65 6D 62 6C 65 72 20 50 72 69 6E
      74 65 72 20 4C 69 73 74 69 6E 67 73
52BA' 0A        00111         DM      LF,'         LDOS  5.1.x'
      20 20 20 20 20 20 20 20 20 4C 44 4F 53 20 20 35
      2E 31 2E 78
52CF' 0A        00112         DM      LF,'             and'
      20 20 20 20 20 20 20 20 20 20 20 20 20 61 6E 64
52E0' 0A        00113         DM      LF,'         EPSON MX-80',LF
      20 20 20 20 20 20 20 20 20 45 50 53 4F 4E 20 4D
      58 2D 38 30 0A
52F6' 0A        00114         DM      LF,'   written by:  Bill McQueen'
      20 20 20 77 72 69 74 74 65 6E 20 62 79 3A 20 20
      42 69 6C 6C 20 4D 63 51 75 65 65 6E
5313' 0A        00115         DM      LF,'       Date:    06/03/86'
      20 20 20 20 20 20 20 44 61 74 65 3A 20 20 20 20
      30 36 2F 30 33 2F 38 36
532C' 0A        00116         DM      LF,'          Ver:  2.0.1'
      20 20 20 20 20 20 20 20 20 20 56 65 72 3A 20 20
      32 2E 30 2E 31
5342' 0D        00117         DM      CR
5343' 0A        00118 ERR_MSG DM      LF,'***ERROR*** --- For output only.'
      2A 2A 2A 45 52 52 4F 52 2A 2A 2A 20 2D 2D 2D 20
      46 6F 72 20 6F 75 74 70 75 74 20 6F 6E 6C 79 2E
5364' 0A        00119         DM      LF,CR
      0D
                00120 ;
                00121 ;*************************************************
                00122 ;
                00123 ;       The following code is the actual filter
                00124 ;       which is moved to HIGH$.
                00125 ;
5366' 180E      00126 START   JR      TSTART          ;Standard himem HEADER
5368' 0000      00127 STORE   DEFW    0               ;stg for old HIGH$ value
536A' 0B        00128 NAME    DB      TSTART-TEXT
536B' 57        00129 TEXT    DB      'WPFLT 2.0.1'
      50 46 4C 54 20 32 2E 30 2E 31
5376' 3815      00130 TSTART  JR      C,GETBYT        ;don't filter a GET rqst.
                00131 ;
5378' C5        00132         PUSH    BC              ;save registers.
5379' D5        00133         PUSH    DE
537A' E5        00134         PUSH    HL
537B' DDE5      00135         PUSH    IX
```

```
537D' FDE5      00136           PUSH    IY
537F' F5        00137           PUSH    AF
5380' 1810      00138           JR      STRT_WP ;go to word processing
                00139 ;
                00140 ;
5382'           00141 PUTBYT    EQU     $               ;exit to original driver.
5382' F1        00142           POP     AF              ;restore registers.
5383' FDE1      00143           POP     IY
5385' DDE1      00144           POP     IX
5387' E1        00145           POP     HL
5388' D1        00146           POP     DE
5389' C1        00147           POP     BC
538A' C30000    00148 PUTBXT    JP      0
                00149 ;
538D' C30000    00150 GETBYT    JP      0               ;to original driver
                00151 ;
5390'           00152 DMPBYT    EQU     $       ;routine to return without
                00153                           ;sending original character to
                00154                           ;printer.
5390' F1        00155           POP     AF              ;restore registers.
5391' FDE1      00156           POP     IY
5393' DDE1      00157           POP     IX
5395' E1        00158           POP     HL
5396' D1        00159           POP     DE
5397' C1        00160           POP     BC
5398' 79        00161           LD      A,C
5399' C9        00162           RET
                00163 ;
                00164 ;
                00165 ;**************************************************
                00166 ;
539A'           00167 WP_CTL    EQU     $               ;Word Processing
                00168                                   ;Routine Control Block.
                00169 ;########################################################
                00170 ;To make word processing mode effective at all times change
                00171 ;this line to "DB   1", and delete the five lines of code
                00172 ;indicated above.
                00173 ;
539A' 00        00174           DB      0               ;1 = WP mode , 0 = normal
                00175 ;
                00176 ;########################################################
539B' 00        00177           DB      0               ;1= ESC mode, 0= normal
539C' 00        00178           DB      0               ;0FH=Underline on, 0=off
539D' 00        00179           DB      0               ;0FH= Italics on, 0= off
539E' 00        00180           DB      0               ;0Fh= Bold prt on, 0= off
                00181 ;
                00182 ;
                00183 ;______________________________________________
539F'           00184 STRT_WP EQU     $
539F' DD219A53'00185 MARK1      LD      IX,WP_CTL       ;IX-> Routine Cntl Block
53A3' DD7E01    00186           LD      A,(IX+1)        ;Fetch Escape mode flag.
53A6' FE01      00187           CP      1               ;Are we in Excape Mode??
53A8' 2872      00188           JR      Z,ESCAPE        ;If so handle it.
                00189 ;##################################################
                00190 ;
                00191 ;To make wp filter effective at all times for possible use
                00192 ;(with basic print lines etc), delete the following five
                00193 ;lines of code and change the first "DB   0" in the WP_CTL
                00194 ;(Word Processing Control Block) to "DB   1"
                00195 ;
53AA' 79        00196           LD      A,C             ;A = char enroute to prtr.
53AB' FE3B      00197           CP      ';'             ;Start Comments???
53AD' 2816      00198           JR      Z,SET_WP        ;Enter Word Proc mode.
53AF' FE0D      00199           CP      0DH             ;End-of-line???
53B1' 2818      00200           JR      Z,RESET         ;End WP Mode.
                00201 ;
                00202 ;##################################################
53B3' DD7E00    00203           LD      A,(IX)          ;Fetch WP mode
53B6' FE00      00204           CP      0
53B8' 28C8      00205           JR      Z,PUTBYT        ;skip if not WP mode
53BA' 79        00206           LD      A,C             ;A = char enroute to prtr
53BB' FE40      00207           CP      ESC_SYM         ;Escape Symbol???
53BD' 2857      00208           JR      Z,SET_ESC       ;Enter ESCAPE Mode.
53BF' 18C1      00209           JR      PUTBYT
                00210 ;
                00211 ;----------------------------------------------
                00212 ;
53C1' 18BF      00213 TO_PUTBYT2        JR      PUTBYT
53C3' 18CB      00214 TO_DMPBYT2        JR    . DMPBYT
                00215 ;
```

```
                 00216 ;---------------------------------------------------
                 00217 ;
                 00218 ;The following two routines enable or disable the word
                 00219 ;processing facilities as a semi-colon or carriage
                 00220 ;return respectively pass thru the filter and then
                 00221 ;send character on for printing.
                 00222 ;
53C5'            00223 SET_WP   EQU     $
53C5' DD360001   00224          LD      (IX),01H        ;Flag WP Mode ON.
53C9' 18B7       00225          JR      PUTBYT
                 00226 ;
53CB'            00227 RESET    EQU     $
53CB' DD360000   00228          LD      (IX),0          ;Flag WP Mode OFF.
53CF' DD360100   00229          LD      (IX+1),0        ;Reset all Feature flags
53D3' DD360200   00230          LD      (IX+2),0
53D7' DD360300   00231          LD      (IX+3),0
53DB' DD360400   00232          LD      (IX+4),0
53DF' 3E1B       00233          LD      A,ESC           ;Reset all Feature modes
53E1' CD0B00     00234          CALL    @WHERE          ;in printer.
53E4' 1849       00235          JR      BYTOUT
53E6' 3E2D       00236          LD      A,2DH           ;'-'
53E8' CD0B00     00237          CALL    @WHERE
53EB' 1842       00238          JR      BYTOUT
53ED' 3E00       00239          LD      A,0             ;UL off.
53EF' CD0B00     00240          CALL    @WHERE
53F2' 183B       00241          JR      BYTOUT
53F4' 3E1B       00242          LD      A,ESC
53F6' CD0B00     00243          CALL    @WHERE
53F9' 1834       00244          JR      BYTOUT
53FB' 3E35       00245          LD      A,35H           ;Italics off
53FD' CD0B00     00246          CALL    @WHERE
5400' 1820       00247          JR      BYTOUT
5402' 3E1B       00248          LD      A,ESC
5404' CD0B00     00249          CALL    @WHERE
5407' 1826       00250          JR      BYTOUT
5409' 3E48       00251          LD      A,48H           ;Bold off
540B' CD0B00     00252          CALL    @WHERE
540E' 181F       00253          JR      BYTOUT
5410' 1800       00254          JR    . TO_PUTBYT
                 00255 ;
                 00256 ;---------------------------------------------------
                 00257 ;
5412' 18AD       00258 TO_PUTBYT        JR      TO_PUTBYT2
5414' 18AD       00259 TO_DMPBYT        JR      TO_DMPBYT2 .
                 00260 ;
                 00261 ;---------------------------------------------------
                 00262 ;
                 00263 ;The following routine sets the ESCAPE mode and then
                 00264 ;returns to EDAS without printing any character.
                 00265 ;
5416' DD360101   00266 SET_ESC LD       (IX+1),1        ;Escape mode on.
541A' 18F8       00267         JR       TO_DMPBYT       ;return w/o printing.
                 00268 ;
                 00269 ;---------------------------------------------------
                 00270 ;
                 00271 ;ESCAPE DECODER.
                 00272 ;
541C'            00273 ESCAPE   EQU     $
541C' DD360100   00274          LD      (IX+1),0        ;reset ESCAPE mode.
5420' 79         00275          LD      A,C             ;A = character
5421' FE2D       00276          CP      '-'
5423' 281C       00277          JR      Z,UNDERLN               ;Toggle Underlining.
5425' FE2F       00278          CP      '/'
5427' 2836       00279          JR      Z,ITALICS               ;Toggle Italics.
5429' FE2B       00280          CP      '+'
542B' 284C       00281          JR      Z,BOLD          ;Toggle Bold printing.
542D' 18E3       00282          JR    . TO_PUTBYT               ;Otherwise print char.
                 00283 ;
                 00284 ;---------------------------------------------------
                 00285 ;
542F'            00286 BYTOUT   EQU     $
542F' 23         00287          INC     HL              ;set-up for return
5430' 23         00288          INC     HL
5431' E5         00289          PUSH    HL
5432' C5         00290          PUSH    BC
5433' 47         00291          LD      B,A             ;save byte.
5434' DBF8       00292 LOOP1    IN      A,(PRPORT)      ;fetch printer status
5436' E6F0       00293          AND     0F0H            ;mask
5438' FE30       00294          CP      30H
543A' 20F8       00295          JR      NZ,LOOP1        ;wait if not ready
```

```
543C' 78       00296         LD     A,B            ;restore A
543D' C1       00297         POP    BC             ;restore B
543E' D3F8     00298         OUT    (PRPORT),A     ;send byte
5440' C9       00299         RET
               00300 ;
               00301 ;------------------------------------------------
               00302 ;
5441'          00303 UNDERLN EQU    $
5441' DD7E02   00304         LD     A,(IX+2)       ;Fetch UL status.
5444' 2F       00305         CPL                   ;Toggle it.
5445' DD7702   00306         LD     (IX+2),A       ;Store new status.
5448' F5       00307         PUSH   AF             ;save register
5449' 3E1B     00308         LD     A,ESC          ;A = Escape Character.
544B' CD0B00   00309         CALL   @WHERE         ;Send it to printer.
544E' 180F     00310         JR     BYTOUT
5450' 3E2D     00311         LD     A,2DH          ;'-'
5452' CD0B00   00312         CALL   @WHERE
5455' 1808     00313         JR     BYTOUT
5457' F1       00314         POP    AF             ;restore register
5458' CD0B00   00315         CALL   @WHERE         ;Send to printer.
545B' 1802     00316         JR     BYTOUT
545D' 18B5     00317         JR     TO_DMPBYT
               00318 ;
545F'          00319 ITALICS EQU    $
545F' DD7E03   00320         LD     A,(IX+3)       ;Fetch Italics status.
5462' 2F       00321         CPL                   ;Toggle it.
5463' DD7703   00322         LD     (IX+3),A       ;STore new status.
5466' 3E1B     00323         LD     A,ESC          ;A = Escape Char
5468' CD0B00   00324         CALL   @WHERE         ;Send to printer.
546B' 18C2     00325         JR     BYTOUT
546D' 3E35     00326         LD     A,35H          ;Make Italic code.
546F' DD8603   00327         ADD    A,(IX+3)
5472' CD0B00   00328         CALL   @WHERE         ;Send to printer.
5475' 1888     00329         JR     BYTOUT
5477' 1898     00330         JR     TO_DMPBYT
               00331 ;
5479'          00332 BOLD    EQU    $
5479' DD7E04   00333         LD     A,(IX+4)       ;Fetch Bold Status.
547C' 2F       00334         CPL                   ;Toggle it.
547D' DD7704   00335         LD     (IX+4),A       ;Store new status.
5480' 3E1B     00336         LD     A,ESC          ;A = ESCAPE character
5482' CD0B00   00337         CALL   @WHERE         ;Send it to printer
5485' 18A8     00338         JR     BYTOUT
5487' 3E48     00339         LD     A,48H          ;Make printer code.
5489' DD8604   00340         ADD    A,(IX+4)
548C' CD0B00   00341         CALL   @WHERE         ;Send to printer
548F' 189E     00342         JR     BYTOUT
5491' 1881     00343         JR     TO_DMPBYT
               00344 ;
               00345 ;________________________________________________
               00346 ;
               00347 ;################################################
5493'          00348 LAST    EQU    $
5200'          00349         END    ENTRY
```

PATCH SUPERZAP/CMD TO ALLOW MODIFICATIONS TO BE DONE IN ASCII
by Glen Mc Diarmid

Use this version of SUPERZAP as you would the original version, but note the following: While viewing a sector, whether it be a disk, file, or memory sector, press the '@' key to get into ASCII mode. Press the BREAK key to get back into hex mode. While in hex mode, all functions and keys are identical to those of the original version of SUPERZAP.

However, when in ASCII mode, the left side of the screen shows ALL characters in ASCII format. While modifying a sector, pressing a key will result in the ASCII code for that key being written to the buffer - this includes the ENTER key. There are several escape keys - the arrows, for instance, will scroll the cursor. Pressing the '@' key in ASCII mode is the same as pressing ENTER in hex mode. Pressing the CLEAR key in ASCII mode is the same as pressing the 'Q' key in hex mode.

The following patches are to the Model I version of SUPERZAP/CMD:

```
SUPERZAP/CMD,09,AC  change  01        to  00
SUPERZAP/CMD,10,78  change  21 DB 6D  to  CD DB 6F
SUPERZAP/CMD,12,C8  change  EF 61     to  CA 6F
SUPERZAP/CMD,14,E5  change  2B 00     to  5B 6F
SUPERZAP/CMD,14,EB  change  49 00     to  B5 6F
SUPERZAP/CMD,28,71  change  ALL ZEROES  to:
       3A DA6F B728 075F AF32 DA6F 7BC9 CD2B
     00C9 B7C8 FE0A C8FE 5BC8 FE1F 2003 3E51
     C9FE 0828 32FE 4020 033E 0DC9 FE09 2827
     FE18 C8FE 19C8 5FE6 0FC6 30FE 3A38 02C6
     0732 DA6F 7BE6 F0CB 3FCB 3FCB 3FCB 3FC6
     30FE 3AD8 C607 C932 DA6F C9CD 4900 FE40
     2807 FE01 C03E C918 01AF 326B 6F3E 20C9
     F53A 6B6F B728 04F1 C3EF 61F1 7723 23C9
     00F5 AF32 DA6F 21DB 6DF1 C9
```

The following patches are to the Model III version of SUPERZAP/CMD:

```
SUPERZAP/CMD,09,AC  change  01        to  00
SUPERZAP/CMD,10,78  change  21 E4 6D  to  CD DB 6F
SUPERZAP/CMD,12,D1  change  F8 61     to  CA 6F
SUPERZAP/CMD,14,EE  change  2B 00     to  5B 6F
SUPERZAP/CMD,14,F4  change  49 00     to  B5 6F
SUPERZAP/CMD,28,71  change  ALL ZEROES  to:
       3A DA6F B728 075F AF32 DA6F 7BC9 CD2B
     00C9 B7C8 FE0A C8FE 5BC8 FE1F 2003 3E51
     C9FE 0828 32FE 4020 033E 0DC9 FE09 2827
     FE18 C8FE 19C8 5FE6 0FC6 30FE 3A38 02C6
     0732 DA6F 7BE6 F0CB 3FCB 3FCB 3FCB 3FC6
     30FE 3AD8 C607 C932 DA6F C9CD 4900 FE40
     2807 FE01 C03E C918 01AF 326B 6F3E 20C9
     F53A 6B6F B728 04F1 C3F8 61F1 7723 23C9
     00F5 AF32 DA6F 21E4 6DF1 C9
```

THE BASIC DIFFERENCES
by John Reid

[This article (originally titled "The President's Report) is reprinted from LPRINT, the newsletter of the MICRO-80 User Group of Ottawa, Ontario, Canada:]

Converting BASIC programs from Model I and III BASIC to Model 4 BASIC seems to be a simple problem. All one needs, it would appear, is a good word processor or a conversion program, a bit of patience, and nothing to it!

The reality is something different. When the Model 4 came out, we got the latest version of BASIC designed for the IBM and its ilk. This is a different BASIC and as a result the conversion process is a very difficult one.

I wanted to convert a Model I BASIC program that did an evaluation of investments. It used an odd method of taking in the date. Instead of the usual formula of MM/DD/YY, it asked for MMDDYY, in the same order.

After I converted to Model 4 BASIC, I discovered that my dates came out 100 years too early. Instead of 1986, it would return 1886. So I began to tear into the program to see what was causing this problem.

To calculate the first two characters of the year, the program has this line:

KD=(DA(N)-KM*10000)/100.

If DA(N)= 20786 (Feb 7 1986) and KM=2, the answer I get in Model III and 4 modes is 7.86.

However, in Model III mode this translates to a 7, but in Model 4 mode it is 8! The program works as advertised in Model III mode and will return 1986 when further calculations are carried out; but it gives me 1886 in Model 4 mode. I'm out 100 years! Why should this be so?

Now, there is a DEFINT statement in the program which defines the variables used in this example as integers. The important thing to note is that Model III BASIC interprets the integer using the first whole number, while Model 4 uses the first two numbers.

Consequently, the integer in Model III BASIC is "7" while in Model 4 BASIC the integer is "8".

What this all means is that when you convert a BASIC program from Model I or III BASIC to Model 4, make sure you test carefully all of your calculations. You may not like what you get! The differences in BASIC between the two machines are considerable. It may take more work than you think to overcome the differences.

Be prepared to check your program carefully before you commit yourself to it. Otherwise you may have an unwelcome surprise on your hands.

[redacted]

by Frederic R. Watson

[Reprinted (slightly abridged) from the CSRA Computer Society Newsletter, which in turn apparently reprinted it from PC+ (Central Texas PC Users Group).]

If you were lucky enough to have chosen an Epson FX-80 dot-matrix printer a couple of years ago you are in luck once more! Quite recently, IBM competition has forced Epson into offering "NLQ", (near-letter-quality) capability in their top of the FX-line printer, the FX-85. Further, full IBM graphics character capability plus touch control for ten type styles using the three normal OFF LINE, FF and LF pushbuttons. The touch control resembles "Fingerprint", which no longer is available. All this without giving up the Epson character sets of the FX-80.

What makes this good news is that an "Upgrade Kit" is now available at a reasonable price. I got two kits for less than $80 each at discount, and was told that the suggested retail price is now $109 each. Since my Epson dealer had no information on the kit when I placed the order, they first gave me the wrong kit. There are FOUR kits, not just one.

So you will be able to order the correct kit, here are some of Epson's best kept secrets:

Present Printer	Upgrade Kit	Upgraded Printer
FX-80	#84971	FX-85
FX-80+	#849711	FX-85
FX-100	#84981	FX-185
FX-100+	#849811	FX-185

I finally got the right kits, along with some bad news. There were absolutely NO INSTRUCTIONS with the kits! There was a deluxe new manual for the FX-85 along with a new piggy-back board (with complete mounting hardware) and several important looking new chips. There was a good parts list which clearly identified the new parts, but that was no substitute for instructions.

I called the Epson Southwest (Dallas, Texas) Distributor's Customer Support printer expert. He advised that Epson, in its wisdom, had decided to give these instructions ONLY to Official Epson Printer Repair Stations, and NOT to customers. Further, he said that he had those instructions on hand but he could not and would not let me have a copy.

By now, as you might imagine, I was plenty ticked off, and more determined than ever to do the installation myself. Previously I had repaired my printer when it "died" about a year ago. I learned then that chips and sockets do not always work as planned, but can develop bad pin-to-socket electrical continuity. In particular, the largest chip, the main CPU, gets hot and a colored oxide can develop on some of the pins. This can cause an electrical "bad-contact" condition, stopping the printer cold. Cleaning the contacts with the gentle application of #600 (extremely fine grit) wet-or-dry paper did the trick. Removing four machine screws let the cover come off, exposing the "innards". Another confidence builder was my previous experience installing Fingerprint, which was a piece of cake.

Using all the clues available, I was able to install the additional piggy-back board. Because of good design it can only go on one way. All chips are numbered and have location clues, such as 4A, 5A, etc., so I got them all in place. But I still needed the instructions for some important details. For instance, there is always the question of "jumpers". With Fingerprint, I had to cut one end of a cream-colored resistor (which is also called a jumper). What to do about jumpers?

Calling on the branch manager of an Epson Official Printer Repair Station produced INSTANT COOPERATION! He offered to have their repairmen check my installation. Very nice indeed! I was grateful for their help in checking my somewhat intuitive placing of parts and for cheerfully furnishing me with instructions, which I now pass on to you.

Now, looking at the instructions, we checked the question of the cream colored resistor (adjacent to two dark colored, but similar looking resistors). Epson now calls it "Jumper J-1", located just to the left of the (largest chip) Master CPU. The instructions confirmed that IT SHOULD BE CUT (bend it up to prevent accidental reconnection).

Reading the instructions, I found that by using my small screwdriver/prybar (a 1/8" blade with the end half-inch bent to a 45 degree angle) it was NOT necessary to remove the board from the printer (instruction Par 1).

Other than needing the one jumper cut (and locating it correctly in the first place) there really should be no difficulty at all with the kit installation now that you have the "secret" instructions at hand. But let's review a couple of fundamentals first:

1. You must keep in mind that one end of all chip sockets is distinctly marked with a notch or depression in the plastic, likewise chips are marked with some similar marking on one end. MORAL: Match chip and socket markings when installing chips.

2. Since most chips are easily damaged by static electricity, precautions are in order when touching chip pins. If you are a "once-in-a-while" chip installer, a large sheet of aluminum foil connected with a jumper wire to an electrical ground (such as the computer chassis bare metal) makes a good grounding plate for hands and bare arms.

[NORTHERN BYTES Editor's Note - Unless you like the thought of zapping your computer's chassis, which MAY or MAY NOT be properly grounded, with a few hundred thousand volts of static electricity, I suggest you pick another grounding point (such as a metal water pipe).]

Another tip, since the chip pins are manufactured pointing slightly outward, rather than straight up-and-down, it is necessary to start one row of pins first. Then by pushing the chip gently sideways, the second row of pins can be persuaded to enter the socket properly. It is ABSOLUTELY ESSENTIAL to visually inspect afterwards for pins that are collapsed, bent or otherwise damaged. Make corrections as needed.

A deluxe new manual for the FX-85 is included with the kit, along with a new plastic label plate to install over the three normal control buttons and lights. Also an adhesive menu list tells you how to tap in numbers representing compressed, expanded, elite and seven other commands using only the normal "off line", "form-feed" and "line-feed" control buttons. "NLQ" is switched on with the FF button; "DRAFT" quality is controlled with the LF button.

Either the full Epson set of characters or the full IBM set is available by setting a single DIP switch.

INSTALLATION GUIDE FOR FX-85 AND 185 UPGRADE KITS

Use these instructions to upgrade an FX series, or FX+ series printer to an FX-85 or 185. There is a separate kit for each series, so be sure that you have the correct one before starting.

FX-80 and 100 Series

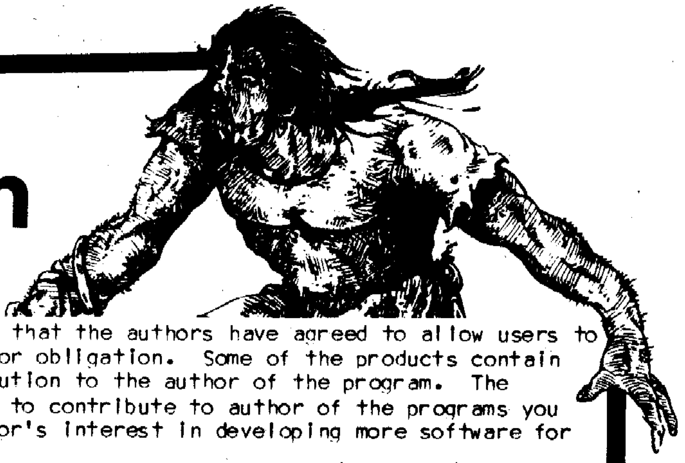
1. Carefully remove the board from the printer. Once out, remove the chips currently installed in locations 4A, 5A, 4B, and 9B.
2. Check to be sure that Jumper J-6 is connected. J-6 is located just above the ROM locations.
3. Carefully install ROM V1-FX4 in location 4A, and ROM V2-FX5 in location 5A.
4. Install the new Fuse Prom in location 4B.
5. Install the new Slave CPU in location 9B. NOTE: the printer may have a Sumi board installed instead of a Slave CPU. The new slave CPU will replace the Sumi board.
6. Check Jumper J-1, located just to the left of the Master CPU, to be sure that it is cut. If not, cut it with a small wire cutter.
7. Install the FXMB in the printer. Do not use the small machine screw that originally came with the printer. Attach the FXEXT board, being very careful that all of the pins are inserted into connector CN-3. Use the long machine screw and spacer that is included in the kit to hold down the board.
8. Check the new owner's manual for the correct DIP switch settings.

FX+ Series

1. Use the same basic procedures that are outlined above. The main difference between the regular FX and FX+ kits is the addition of the Slave CPU in the regular FX series kits. The Slave CPU in the FX+ series is correct, and doesn't need to be changed.
2. The FX+ series has Jumper J-7 connected, and it must be switched back over to J-6.

Fred Watson is one of the most prolific of our Central Texas PC Users Group writers. He is an Engineer and Writer with a penchant for "opening up the box" and seeing what makes it tick. Fred can be contacted at the following address: Route 3 Box 4C Hoffman Road, Bastrop, Texas 78602, or telephone (512) 321-6520.

Public Domain



Each of the following disks are "public domain". This means that the authors have agreed to allow users to copy the programs and share with their friends with no cost or obligation. Some of the products contain solicitations requesting that you send an additional contribution to the author of the program. The contribution is completely voluntary; you will probably wish to contribute to author of the programs you actually decide to use. Contributions help maintain an author's interest in developing more software for your system.

Order Code	Contents Description	Price
D8PD01	Miscellaneous I/III programs/ Newdos/80 programs	$10
D8PD02	Terminal and Communications programs/ Miscellaneous Mod 4, MultiDos, Dosplus	$10
D8PD03	Programs from Northern Bytes (N.B.)/ Games	$10
D8PD04	Programs from Northern Bytes/ Northern Bytes and Bilingual Hangman	$10
D8PD05	Mini-BBS and Programs from N.B./ Programs from Northern Bytes 5-7	$10
D8PD06	Programs by Jerry Vabulas/ A potpourri of miscellaneous programs	$10
D8PD07	Programs from Northern Bytes 6-1, 6-2/ Programs from Northern Bytes 6-3	$10
D8PD08	The Chicago Greene Machine/ Miscellaneous I/III programs	$10
D8PD09	Video Manager and others by Mel Patrick HELP file creator and more by M Patrick	$10
D8PD10	Keyboard Macro and more by Mel Patrick/ More miscellaney by Mel Patrick	$10
D8PD11	Printer Support/File Compare/ Comm and others by Mel Patrick	$10
D8PD12	File Manager/Progs from N.B. 6-4, 6-5/ Mod 4 SETDATE, Northern Bytes 6-4, 6-5	$10

Order Code	Contents Description	Price
D8PD13	Northern Bytes 6-6, 6-7 and 6-8/ Drive cleaner, other miscellaney	$10
D8PD14	Dennis Allen's TRSDOS 6.2 Utilities/ More 6.2 Utilities and FREEWARE	$10
D8PD15	Squeeze/Unsqueeze I, III and 4/ More squeeze utilities	$10
D8PD16	Library Utilities I, III and 4/ More library utilities	$10
D8PD17	I, III and 4 access to MSDOS libraries/ More by David Huelsmann	$10
D8PD18	Roxton Baker's STOPPER and other misc./ STOPPER source and other misc.	$10
D8PD19	Roxton Baker's TRAKCESS/ More TRAKCESS Utilities	$10
D8PD20	Programs from Northern Bytes 7-2/ Printer Utilities and PD font files	$10
D8PD21	Model 4 monitor, other miscellany/ Model 4 Smart Terminal and others	$10
D8PD22	Utilities and Font files/ Programs from Northern Bytes 7-2, 7-3	$10
D3CLAN	The Clan Geneological Data Base System	$10
D8ND1	Newdos/80 Patches and Upgrades	$10
D8BPD1	ISAR Data File Manager	$10
D8GPD1	Public Domain Games Library	$10
D1F	The Alternate Forth (7 diskettes)	$25
D3F	The Alternate Forth (4 diskettes)	$25

Use the coupon below to order. Send your completed order form to:

The Alternate Source Information Outlet
704 North Pennsylvania Avenue
Lansing, MI 48906-5319
(517) 482-8270

VISA MasterCard

Yes, please send me the following public domain diskettes:

Code Price Code Price

Shipping: ________
Total Amount: ________

Charge Card Number: ____________________

Expiration Date: ____________________

Your Name: ____________________

Address: ____________________

Address: ____________________

City/State/Zip: ____________________

Please include $3 with your order for shipping and handling.
Foreign orders, be sure to specify surface or AIR MAIL shipping and include appropriate shipping.
Please do NOT send CASH through the mail!
Michigan Residents please add 4% sales tax.

U.S. orders are shipped UPS unless you include a Box Number

NORTHERN BYTES

Subscription Information

Northern Bytes is 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. Currently there are eight back issues available for Volume 5, and eight back issues for Volume 6, as well as all issues from Volume 7. All back issues prior to Volume 7, Number 3 are $2 each. Back issues starting with Volume 7, Number 3 are $4 each (prices include all shipping charges unless you live overseas and request airmail delivery, in which case we may charge extra to cover postage).

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 credit card expiration date. **PLEASE NOTE** your card expiration date (which will appear on your mailing label) and be sure to send us your new card number and/or expiration date when you receive it to assure uninterrupted delivery of Northern Bytes.

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. The Alternate Source maintains a regular list of readers that have asked **TO BE BILLED** for each issue. You then send a check (made payable to The Alternate Source Information Outlet) for each issue as you receive it. If you don't send a check, we 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 assured 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
EMAIL: CompuServe: 72167,161 / MCI Mail: 109-7407 / Telex: 6501097407 MCI

NORTHERN BYTES

The Alternate Source Information Outlet
704 North Pennsylvania Avenue
Lansing, Michigan 48906-5319
Telephone: (517) 482-8270
CompuServe EasyPlex: 72167,161
MCI Mail: 109-7407
Telex: 6501097407 (Answerback: 6501097407 MCI)

Bulk Mail
U.S. Postage Paid
Permit 815
Lansing, MI

POSTMASTER:
Address Correction Requested
Return Postage Guaranteed

To: