NORTHERN BYTES

Volume 5 Number 7

GREETINGS! Welcome to a "Where did this issue come from?" issue of Northern Bytes! I didn't expect to accumulate enough material to put together another issue this soon, but thanks to YOU, our loyal readers, we've slipped in another issue. With any luck at all, we should have one more issue before the holiday season. That would make eight issues in 1984! Incredible!

Because this issue is coming out rather soon following Volume 5 Number 6, Paul Snively's FORTH-WISE column will not appear in this issue. Paul's back in school and hitting the books pretty hard, but we'll see if we can coax a few more columns out of him. If you enjoyed that feature, why not drop Paul an MCI Mail letter and let him know?

You may wonder if I'm allowed to say anything good about competing products since The Alternate Source took over publication of Northern Bytes. Sure I can, and here's proof! I've started playing around with AllWrite!, the new Word Processor by Prosoft, and I think it must be the state of the art in word processing programs right now, at least on the TRS-80. Now, The Alternate Source markets EDM and EDX, but there is a difference. EDM and EDX are super text editors, and they will do a very fine job if you don't have a fancy printer with proportional spacing (or if you don't care to use the proportional spacing capabilities of your printer). But, while AllWrite's editor is pretty good (much better than the editor in NewScript, the forerunner of AllWrite!), where it really shines is in its text formatting capabilities. Not only can it handle right justification on proportional printers (which EDM and EDX cannot), but it can handle things like footnotes (in your choice of formats), legal numbering of pages, automatic insertion of text from disk or keyboard into "form" letters, and even automatically building an index and/or table of contents from your text. In fact, if you need to do something and AllWrite! can't handle it, it's either because your printer can't do it, your printer is not supported by AllWrite (not too likely unless it's a real orphan), or you have some very specialized requirements. One warning - with added capabilities comes added program complexity, and AllWrite! is no exception. If all you need a Word Processor for is to write an occasional letter to Aunt Tillie, then use EDX - it's a LOT less expensive! But, if you'd like to make use of the proportional printing capabilities of your printer or you need any of the other special features, AllWrite! is for you.

Of course, you could get a REALLY SUPER Word Processing system by using EDM or EDX to edit your text, and then using AllWrite's text formatter to print it out. This would give you the best of both worlds! This idea is not as strange as it sounds - most Word Processors actually consist of TWO programs in one. First, you have to have a text editor (usually a full screen editor) that allows you to enter the text into memory, and eventually save it to disk. Then, you have to have a text formatter program that reads your text file, interprets any embedded printer control codes that you've put into the text, and prints the file as you intended, using the printer features you've specified. There's no reason that both functions have to be done by the same program!

In the same way, I know that some folks use the ALE Editor-Assembler program (sold by TAS) to create assembly language source code, because it has the BEST full-screen editor for creating source code (the editor is basically the same one used in EDM and EDX, but with a few modifications to facilitate entering source code, and reading or writing some of the popular source code formats to/from disk). But then these folks use some other Editor-Assembler to actually assemble the program, generally because they are already used to using some feature offered by their former Editor-Assembler and want to continue to use it. Another reason for using another assembler program might be that you want to use a super-fast assembler (such as ZEUS by Cosmopolitan Electronics, which assembles a source code file faster than greased lightning but doesn't offer the full-screen editor of ALE).

The nice thing about EDM (and ALE) is that they are "programmable" through the use of a special macro language (this has nothing to do with "macros" in Editor-Assembler source code). Very few folks have bothered to try to learn and/or use the macro language (I have to admit I'm one of those that hasn't), but those that have learned it have made these programs do some pretty amazing things.

Turning to other things, the rumor mill has been pretty active lately. I can't pass on any specifics of rumors I've heard here, but let's talk about computer magazines a moment. There's quite a turnover in this field. New ones start and then quietly fade away later. And then sometimes, a large publisher will purchase the publications of a smaller one, intending to use them as a tax writeoff. Of course, for that to happen, the publications thus purchased have to lose money (and possibly go bankrupt). You can sometimes tell that this is happening when the top brass of the purchased publications are fired and replaced by people that seem to be totally incompetent. Another sign might be that the advertising department will start passing out cut rates on ad space (the ad department people are hoping against hope that they can keep their jobs, so they are pretty desperate to sell advertising!). All the while, of course, the page count of the publications involved will keep shrinking, and the quality of the articles may drop to new lows (remember, the idea is to lose money, and for that to happen you have to discourage subscription renewals). Any resemblance between the above fictitious scenario and actual events in the computer publishing field are purely coincidental(?), but don't be surprised if you discover something like this happening!

And that's all I'm going to say for this month, except for this: When you go to vote in November, here's something to think about. Some labor unions are getting pretty upset about the possibility of folks working on computers in their homes, and would even like to have a federal law banning this practice (if you saw "60 Minutes" on September 23, 1984, you saw an official of the AFL-CIO grudgingly admit this). If such a law were ever actually passed, it would mean the end of NORTHERN BYTES, and most of our readers would probably be adversely affected by such a law as well (it can be argued, of course, that such a law would in fact be unenforceable, but even if that's so, what the world does NOT need now is another unenforceable law. Anyway, I wouldn't want to have to hide in a closet while I keyboard text for NORTHERN BYTES). Now, we all know which political party almost always gets the endorsement of the labor unions, and we know which party supports free enterprise. Think about it. This election year should be very interesting, because never in our recent history has the difference in the philosophies of the two parties been so clearly apparent. Editors of other publications get to make political endorsements, so I will too - if my opinion counts for anything, I'm solidly behind President Reagan for four more years. Unfortunately, there won't be another issue of NORTHERN BYTES before the election, so don't bother to take pen in hand (or word-processing disk in drive) to agree or disagree, because I won't be able to publish your letter in time to do any good.

THE EXTERMINATOR

We thought we'd killed all of the BUGS in this program, but lo and behold, it appears we missed one. The bug I'm about to describe appears in some editions of "TRS-80 ROM ROUTINES DOCUMENTED", and was also published back in NORTHERN BYTES Volume 5 Number 3, as part of an earlier fix to "TRS-80 ROM ROUTINES DOCUMENTED"!

The problem is in the "Improved Ampersand Function" routine, found in Appendix VI of the book and on pages 4-5 of the NORTHERN BYTES issue mentioned above. The program (as corrected in NORTHERN BYTES Volume 5 Number 3) works as advertised in all but one situation. The lone exception is that if you go into BASIC with no memory protected (HIMEM or TOPMEM = FFFFH) and enter the following one-liner:

```
10 INPUT A$: ?&H(A$): GOTO 10
```

the function will return a value of zero on at least the first use, and possibly on subsequent uses following a "garbage collection" in BASIC. Note that the same effect would be observed if the &B or &O functions were used instead of &H. The problem is that the "END OF STRING + 1" address used by the program will in this case be 0000H, which will always be "lower" than the start-of-string address.

Fortunately, the fix couldn't be much simpler. Make the following change in line 1030 of the published listing:

```
From:
7FE0 D0     01030     RET    NC          ;RETURN IF END OF STRING
To:
7FE0 C8     01030     RET    Z           ;RETURN IF END OF STRING
```

This should solve the problem. If you own a copy of "TRS-80 ROM ROUTINES DOCUMENTED" ($19.95 plus shipping from The Alternate Source), be sure to note the above correction on page 108 of your copy (NOTE: If line 1030 in your copy doesn't match the "old" line 1030 shown above, you have an early edition of the book, and need to use the corrected listing in NORTHERN BYTES Volume 5 Number 3 AND the correction shown above).

One more BUG to mention, this time in Dave McGlumphy's telephone dialer program that appeared on page 20 of NORTHERN BYTES Volume 5, Number 5. To fix the bug, insert a new line 265 in the published listing:

```
00265       LD    SP,6FFFH          ;SET STACK PTR.
```

That concludes the bug report for this time around!

LETTERS DEPARTMENT

Here's the latest batch of correspondence received from our readers:

Dear Jack,

I think I have a solution to the problem of going back to the beginning of a file and rewriting some bytes at the beginning. I have used it and tested it on all the DOSses I have access to. I have only used it together with the single-byte get and put routines at locations 0013H and 001BH, and have not tested it with the sector read and write routines.

The procedure is to position to record zero with a call to 4442H, reset bit 7 of FCB+1 (single-byte read flag), and zero byte FCB+9 (record length). The new information can then be written at the beginning of the file, and when closed the end of file should be set to the highest location written to the file. If this does not work in all cases then I would like to know about it since I have a program on the market which uses it.

Also does anyone out there have a set of ZAPs for the Model I version of M-ZAL release 3 which will allow lower case letters to be used, at least in text strings. A letter to Computer Applications Unlimited has been met with complete silence. If not, does anyone want to buy an unused, apparently unsupported, copy of M-ZAL?

I do not know anyone with a Color Computer, so I have no idea of the disk format, but perhaps you have some documentation on the format and a sample disk with some data on it? I still haven't given up on the conversion program [mentioned in Arne's letter in the previous issue of NORTHERN BYTES -editor] but it has been reduced in priority due to other commitments.

Sincerely, Arne Rohde

[Editor's note: I do not have access to a CoCo, but perhaps one of our readers would be willing to help Arne by sending him a CoCo disk with some data, or any information you might have about the CoCo disk format and/or directory structure. If you'd be willing to help, you can send the disk and/or information to us here at NORTHERN BYTES, and we'll pass it on to Arne.]

Dear Jack,

I have a four drive Model 4 and a 4P, but know nothing about taking a computer apart.

The 4P came with TRSDOS 6.1.1, which will not recognize more than two floppy drives because it is basically a hard disk system (a person at Radio Shack told me this was the reason). I wanted to update the Model 4 system from 6.0, but 6.1.0 had disk timing problems and 6.1.2 will not operate my BASIC programs. TRSDOS 6.1.1 can be made to recognize four drives if you give it 6.1.2's BOOT/SYS:

```
BACKUP :2 :1 (q=y)      <answer no to all>
                        6.1.2 in drive 2
                        blank formatted disk in drive 1
BACKUP :2 :1 (S,I)      6.1.1 move to drive 2
                        The (S,I) forces a backup by class that
                        doesn't touch tha BOOT/SYS.
```

If doing it on the 4P, put the system on the MEMDISK. Don't know if this is any help, but it works fine on my Model 4.

Sincerely, Hope Dunham

[Editor's note: What this letter goes to prove, I think, is that with a little thought there's a way around just about any situation, with the possible exception of misinformed Radio Shack personnel!

Have you ever dealt with the telephone company business office? Frustrating experience, isn't it? There's a reason for it: Supervisors monitor calls at random, and job performance is based on how fast the business office operator (or regular operator, for that matter) can complete each call. So, the unspoken message from management is, tell the customer anything, but just get them off the line - fast! If you don't know the answer to a question, make up something that sounds good, but complete that call quickly (and whatever you do, don't bother the Big Bosses with foolish questions). This scenario may be more or less true in your phone company, but for a while I lived in an area serviced by the nation's second largest telephone company and I'm just about certain this is how it was. I could never get a straight answer from the phone company, until I learned two things: 1) NEVER deal with the business office - always call the company President's office and work down from there, and 2) The management personnel feared the Michigan Public Service Commission more than they feared God (or at least it seemed that way), so if the going got tough, a threat to call the MPSC worked wonders (actually CALLING the MPSC would put the entire phone company on code red - they would actually work overtime to fix the problem!).

Well, from what I've been hearing about some Radio Shack salesmen, they seem to operate under a similar philosophy. If a customer asks a question and they don't know the answer, they just make up something plausible, and for goodness sake don't bother Fort Worth. Unfortunately, they don't fear the Public Service Commission, either, but may I suggest another route. If you have problems with TRSDOS 6, try calling Logical Systems, Inc. at (414) 355-5454 and ask for Bill Schroeder. You probably won't get him (he's the company President), but at least you might get to talk to someone in upper management. Now, the point of this is that first of all, they are probably in a better position to help you with your TRSDOS 6 problems than Tandy is, since they wrote TRSDOS 6. But more to the point, they are also in a position to make their voice heard at Tandy Towers. If they get hundreds of calls complaining about incompetent Radio Shack salesmen, you can bet that Tandy's gonna hear about it! And maybe, just maybe, they'll actually do something about it.

On the other hand, I wouldn't hold my breath...]

```
Date:     Mon Aug 20, 1984 4:26 am EDT      **RECEIPT
From:     Jim Davis / MCI ID: 195-9314

TO:     * Jack Decker / MCI ID: 102-7413
Subject:  Super Utility 3.2 CMD file
```

Dear Mr. Decker,

First, I want to thank you for the procedure to create a CMD file for Super Utility Plus. Unfortunately, after I updated my Model III to a 4, it would no longer run TASMON, even in the Model III mode. I had to devise my own methods of finding the value in the interrupt register. My procedure is contained in the following copy of the file I wrote to remind me if I needed to again.

Incidentally, the process does work on version 3.2. However my procedure for jumping to memory from Super Utility on 3.1 is more difficult because of a "bug" in version 3.1.

BACKING UP SUPER UTILITY PLUS 3.2

The following is the procedure to create a CMD file for Super Utility Plus, version 3.2.

Step 1.

Boot up Super Utility. Go into the MEMORY UTILITIES section and enter the following code at F002.

F5,E5,21,00,F0,ED,57,77,F1,E1,C9

Press <BREAK> and use the jump feature of Super Utility to Jump to F002. You will end up back at the main menu. Go back into the MEMORY UTILITIES section and go into "DISPLAY MEMORY" function. Look at the value at F000 and write it down. You'll need it later. It is the value stored in the interrupt register, and is the check-sum of your serial number.

Step 2.

Return to the MEMORY UTILITIES section and move memory from 4000H - 67FFH to D500H.

Step 3.

Now make the following entries in memory at FD00:

F3,21,00,D5,11,00,40,01,00,28,ED,B0,3E,XX,ED,47,31,00,FE,C3,15,40

XX, is of course the number you wrote down.

Step 4.

Boot up your favorite DOS by pressing reset; DON'T USE SUPER UTILITY'S EXIT FUNCTION.

Use your DOS's DUMP function to make your CMD file. With TRSDOS 1.3 it would be as follows:

DUMP SUPLUS/CMD:0 (START=6800,END=0FD15,TRA=0FD00)

You now have SU Plus as a CMD file.

Sincerely
Jim Davis, 4440 Lancaster Drive NE, Salem, Oregon 97305

[Editor's note: There is an upgraded version of TASMON that contains its own keyboard driver, and adds many new features to TASMON. Details were in NORTHERN BYTES Volume 5 Number 6 and in TAS UPDATE #6. Registered owners of the Model I and III versions of TASMON can upgrade for $10 (that includes a new disk), and there is a NEW version of TASMON that runs under TRSDOS 6.1.x that sells for $39.95 ($29.95 if you order before November 1, 1984).]

MCI MAIL FILE TRANSFER PROGRAMS
by Jack Decker

As most NORTHERN BYTES readers know, I have been advocating the use of MCI Mail as a cost-effective method of transferring files between computer users. However, there is one problem with MCI Mail - it cannot be easily used to transfer certain types of files. These programs remove that limitation.

The problem stems from the fact that MCI Mail is basically a text transfer medium. Plain vanilla ASCII text files are easily transferred, but there is no automatic error checking. BASIC programs can be transferred if saved in ASCII format, but once again there is the lack of error checking and in addition, lines cannot be greater than 132 characters long for reliable transfer.

MCI Mail cannot handle ASCII characters outside the range of normal text - that is, under ASCII 32 ("control" characters) or greater than ASCII 127 ("graphics" or "special" characters). But, virtually any file that is not one of the two types listed above will contain such characters. What do you do if you wish to transfer, for example, a short machine language object or source code file to someone else?

The programs below will take ANY regular TRS-80 Model I/III type files and convert them to a format that can be transmitted over MCI Mail. Even Model 4 (TRSDOS 6) format files can be sent using these programs, but the programs themselves will only run in the Model III mode on a Model 4 or 4P. You can send ANY type of file, be it text, compressed BASIC, ASCII BASIC (with any line length), word processing, Editor-Assembler source, machine language, or whatever. TWO checksums are maintained to inform you of any errors in the transfer.

The programs work by converting a file containing bytes in the range 0-255 ASCII to a file containing bytes in the range 32-126 ASCII. Text bytes in the range 32 through 123 ASCII are not converted, but are sent as-is. Other values are converted to a two-byte format. As an example, bytes in the range 0-32 ASCII have a value of 32 added to them, and then are sent with a "prefix byte" of 124. The chart below may make this a bit clearer

ORIGINAL ASCII VALUE	PREFIX BYTE VALUE	VALUE ADDED TO/ SUBTRACTED FROM BYTE
0 - 31	124	+ 32
32 - 123	none	none
124 - 175	124	- 60
176 - 255	125	- 144

ASCII 126 is used as an "end-of-file" marker, and ASCII 127 is not used since it sometimes has a special meaning to certain terminal programs.

The individual bytes of each file are transmitted as shown above. After 64 characters on a line have been transmitted, a "line checksum" byte is transmitted followed by a carriage return. The "line checksum" will only detect gross errors (where an entire byte has been garbled, for example) because it is obtained by adding each transmitted byte on the line to the preceding byte, then at the end of the line it is ANDed with 63 to mask off the upper two bits and then a value of 32 is added to make it a printable character. Note that the upper two bits are not checked at all! Thus, if only two consecutive bits were garbled and they happened to be the upper two bits of a byte, the error would not be detected by the "line checksum" (but would still be detected by the main checksum). The main purpose of the line checksum is to help point out the line in which an error occurs, so that the recipient of the file has an oppertunity to attempt manual correction (possible only in some types of files).

Note that the above formula used to derive the "line checksum" can yield a character in the range 32-95 ASCII, so there is no chance of it being an "unprintable" character.

When the end of the file is reached, an "end-of-text" (ASCII 126) byte is transmitted, followed by the "line checksum" for the line (which may be a "short" line of less than the normal 65 characters). A carriage return is then sent. The master checksum is then sent on a line by itself, in plain ASCII format (in other words, using the numerals 0-9). The master checksum may be in the range -32768 to 32767 (keeping it within integer limits), and is the sum of all file bytes (not including "line checksums" or carriage returns). Should the checksum exceed the 32767 limit, a value of 65536 is subtracted and the count continues. Those of you into assembly language will recognize why these values have been used. I'm sort of hoping that someone will convert these programs to machine language someday (I might, if I can ever find the time), so I set up the master checksum so that it could be accumulated in a register pair or two-byte memory location. Note, however, that the checksum is printed on the final line in decimal rather than hexadecimal format, with a leading minus sign if the number is negative.

The program FILTOMCI/BAS converts a file to the format shown above. You must give it an input filename (the file you want to convert) and an output filename (the file you will upload to MCI Mail). It will then make the conversion according to the specifications given above. If you're converting a long file, this may take awhile (BASIC is nothing if not slow!).

MCITOFIL/BAS does the reverse, taking the received file and converting back to a mirror image of the original file. One caution must be observed by the person receiving such a file - before you download it, use the ACCOUNT command to set your terminal line length to a value greater than 65 (either 80 or 132 is suggested). If you don't do this you'll get extra carriage returns and MCITOFIL/BAS won't be able to convert the program. Also, you PRINT rather than READ your received mail to avoid getting prompt strings mixed into the program. If you've already read a file and realize you didn't do it right, remember that you have up to 24 hours to go back and re-read any mail on your "desk". Just use ACCOUNT to set the line length, then have MCI Mail PRINT the contents of you DESK (or your INBOX, if you've just started reading your mail for the first time). Open your terminal program's text buffer and save the received mail to disk (MODEM80 from TAS works quite well for this purpose).

When you're ready to use MCITOFIL/BAS, you have the option to skip lines at the start of file, which you should do if you saved the header information to disk, or if the person sending the file sent some text for you to read prior to sending the program. Just keep pressing the <S> key until you see the first line of the transmitted file, then press <ENTER> and MCITOFIL/BAS will do the rest.

Note that these programs could also be used with other electronic mail services (such as Western Union's EASYLINK),

although I haven't tried it with any of them (by the way, did you know that MCI MAIL and EASYLINK users can communicate with each other by using the TELEX facilities of each service? It's probably pretty expensive to do it this way, but it can be done!). Similarly, these programs could probably be converted for use on other versions of BASIC without too much difficulty. The one portion of MCITOFIL/BAS that might confuse even some of you "experts" is in lines 30 and 90. What this does is to set up a string storage area (referenced by BASIC as V$) on the upper right-hand corner of the video display, so that a string containing the line count can be "stored" (and thus displayed) there. I have to give credit to Alan Abrahamson, editor of Voice of the '80 (newsletter of the Fairfield County Computer Users Group) for giving me the idea for this one. He used this technique in a BASIC program to move data from one place to another on the video display. Alan explains:

"The technique of using the dummy variables to block move the DATA using "LSET" [I used RSET to pad the left side of the string with spaces] is about ten times the speed of a similar PRINT statement. The LSET [and RSET!] command appears to do a Z80 block move (LDIR,LDDR) between the VARPTR's (pointers) of the variables involved. This technique can be used in any screen data manipulation routine to move data from point A to point B. Remember that the LSET routine physically moves the data from one place to another, it is NOT just a pointer swap.

"The key ingredient in using this method is to establish the dummy or real variables prior to using LSET [or RSET]. It is not necessary to OPEN any files in using LSET, but you cannot use LSET on a variable that has not yet been initialized..." [I added the comments in brackets.]

The above two paragraphs were included for the benefit of the advanced programmers in the crowd. If they didn't make any sense to you, then all you need to know is that line 30 (the part after the CLS) and line 90 will probably have to be deleted if you attempt to convert this program for any other system. The purpose of those lines in this program was to let me print a running line count at a fixed location on the video display, without changing the current cursor location (so that if any "checksum error" type messages were being printed, they wouldn't keep overprinting the same line).

Two warnings about these programs: One, they were thrown together in some haste and do not contain any great amount of error trapping. There's no telling what they'll do if you use them incorrectly. Two, note the version number (1.0). We can always hope they are bug free, but that's not the way to bet. Besides that, I'm really hoping some of you readers will see fit to improve upon these programs and then share the results with the rest of us. In any case, watch future editions of NORTHERN BYTES for corrections and/or enhancements.

And one final note: There is a program called HEX/CMD that is included in the MODEM80 package that does something similar to what I'm doing here, but the only problem with it is that it converts ALL bytes within a file to two hexadecimal characters. When you're paying by the character, that may be just a bit too much code expansion, especially when the file you're converting is mostly text but with an occasional control code or bit of graphics or a machine language segment thrown in. The truth is that unless you deliberately go out of your way to set up a situation where this is not the case, ANY converted file will be shorter, in many cases MUCH shorter, when converted using the programs below as opposed to conversion using HEX/CMD. However, HEX/CMD is in machine language and does the job much faster, so if you're only sending a short file anyway, it may pay to use that program instead.

```
10 REM FILTOMCI/BAS VERSION 1.0 - CREATION DATE 9/26/84
20 CLEAR 10000: DEFINT A-Y: LINE INPUT "INPUT FILENAME?
";A$: OPEN "R",1,A$,1: LINE INPUT "OUTPUT FILENAME? ";A$:
OPEN "O",2,A$: FIELD 1, 1 AS A$
30 IF NOT EOF(1) THEN GET 1: GOSUB 60: GOTO 30
40 N=63: A=126: GOSUB 80: A$=STR$(Z): IF LEFT$(A$,1)=" " THEN
A$=MID$(A$,2)
50 PRINT A$: PRINT#2,A$: END
60 A=ASC(A$): IF A>175 THEN A=A-144: B=125 ELSE B=124: IF A>
123 THEN A=A-60 ELSE IF A<32 THEN A=A+32 ELSE 80
70 T=A: A=B: GOSUB 80: A=T
80 PRINT CHR$(A);: PRINT#2,CHR$(A);: C=C+A: Z=Z+A: N=N+1: IF
N<64 THEN RETURN ELSE N=0: IF Z>32767 THEN Z=Z-65536
90 C=(C AND 63)+32: PRINT CHR$(C): PRINT#2,CHR$(C): C=0: RET
URN
```

```
10 REM MCITOFIL/BAS VERSION 1.0 - CREATION DATE 9/26/84
20 CLEAR 10000: DEFINT A-Y: LINE INPUT "INPUT FILENAME?
";A$: OPEN "I",1,A$: LINE INPUT "OUTPUT FILENAME? ";A$: ON
ERROR GOTO 180: KILL A$: ON ERROR GOTO 0: OPEN "R",2,A$,
1: FIELD 2, 1 AS A$
30 CLS: V$="LINE 00000": POKE VARPTR(V$)+1,54: POKE VARPT
R(V$)+2,60
40 LINE INPUT "DO YOU WISH TO SKIP LINES AT START OF FIL
E (Y OR N)? ";B$
50 B$=LEFT$(B$,1): IF B$="N" OR B$="n" THEN CLS: GOTO 80 EL
SE IF B$<>"Y" AND B$<>"y" THEN 40 ELSE PRINT "PRESS <S> TO
 SKIP LINES, <ENTER> TO START USING LINES...":PRINT
60 IF EOF(1) THEN 80 ELSE B$="": LINE INPUT #1,B$: PRINT B$:
 PRINT CHR$(14);: I$=INKEY$
70 I$=INKEY$: IF I$="S" OR I$="s" THEN 60 ELSE IF I$=CHR$(13)
 THEN PRINT CHR$(15);: CLS: GOTO 90 ELSE 70
80 IF EOF(1) THEN PRINT"***** UNEXPECTED END OF FILE **
***": CLOSE: END ELSE B$="": LINE INPUT #1, B$
90 N=N+1: RSET V$="LINE"+STR$(N)
100 IF LEN(B$)<>65 AND INSTR(B$,CHR$(126))=0 THEN PRINT "**
*** FATAL ERROR - LINE WRONG LENGTH *****": PRINT B$: C
LOSE: END ELSE C=0: FOR L=1 TO LEN(B$): GOSUB 130: NEXT: IF
 F=0 THEN 80
110 LINE INPUT #1,B$: IF VAL(B$)<>Z THEN PRINT "***** MAST
ER CHECKSUM ERROR *****"
120 CLOSE: END
130 A=ASC(MID$(B$,L,1)): IF L<LEN(B$) THEN 150 ELSE IF (A-32)
<>(C AND 63) THEN PRINT"***** CHECKSUM ERROR *****": PR
INT B$
140 RETURN
150 C=C+A: Z=Z+A: IF Z>32767 THEN Z=Z-65536
160 IF P=125 THEN A=A+144 ELSE IF P=124 THEN IF A<64 THEN
A=A-32 ELSE A=A+60 ELSE IF A=126 THEN F=1: RETURN ELSE IF
 A>123 THEN P=A: RETURN
170 P=0: LSET A$=CHR$(A): PUT 2: RETURN
180 RESUME NEXT
```

DON'T GIVE ME THAT LINE

by David R. McGlumphy
4429 Paula Lane, Chattanooga, Tennessee 37415
MCI Mail ID: 181-7759

Once upon a time, and once upon a place, I saw the question "How do you draw a line on a TRS-80?" Too easy for piddling with, thought I. Then for grins, I tried it. If you too think it's easy, try it before looking over my program. You may also find it harder than it sounds.

My thinking went like this: Suppose I want to draw a line from (0,47) to (127,0) which is from the lower left corner of the screen to the upper right corner. For every X position that I move to the right, I'd have to go up a fraction of a Y position, and that fraction would be the number of Y positions divided by the number of X positions. That sort of thinking is fine as long as the fraction is less than one. If it were greater than one, I'd get a gap (space) in the line. In that case, I'd just go along the Y axis instead of the X axis, and the fraction would be computed by dividing the number of X positions by the number of Y positions. I'd also have to take into account the sign of the change in positions. In this first case, the change in the X axis is positive (from left to right) while the change in the Y axis is negative (from bottom to top). If the line were to go from the lower right corner (47,127) to the upper left corner, (0,0), the change in both the X and Y axes would be negative. With that, let's proceed to the program.

Lines 40 and 50 pick two random points on the screen for using the SET and RESET commands to turn on or turn off a point. Remember that the minimum for both X and Y coordinates is zero, the maximum for X is 127, and the maximum for Y is 47.

As written, my program will draw a line, erase the same line, then draw another line starting where the first line ended. If you want to see the lines continued as though drawing without lifting a pencil from paper, make line 90 a REMark by inserting an apostrophe (') at the beginning of the line. In order to erase the line, I needed to save the two sets of coordinates, so that's what line 60 does.

Line 70 puts a 1 in variable SR. If SR has a 1, that means to SET (turn on) a point. If SR has a 0, it means to RESET (turn off) a point. Line 70 then calls the routine at 120 which determines which of two "draw routines" to use.

Line 80 restores the original two sets of coordinates because they were corrupted in the drawing subroutines. Line 90 then goes over the same graphics points as was done by the GOSUB in line 70, but this time, the points are RESET (erased).

I decided always to draw from (X1,Y1) to (X2,Y2), so after I've done so, I let line 100 move the last set of coordinates to the first set so that I'll start drawing from that point on the next go-around via line 110 which sends me back to the beginning of the mainline routine, lines 50 thru 110.

Lines 120 thru 140 are the subroutine called from the mainline section. They compute the distance between the sets of coordinates and call one of two routines which actually do the drawing. Why two subroutines instead of just one? Remembering what I'd though about in the second paragraph, I wanted a solid line from point to point rather than one filled with gaps and spaces. Notice according to the remarks on lines 150 and 240 that one routine is used when the difference between the X coordinates is greater than the difference between the Y coordinates and another subroutine is used when the difference between the Y coordinates is greater than the difference between the X coordinates. Notice also that in the 150 subroutine, I vary the X coordinate one full integer at a time while in the subroutine at 240, it's the Y axis which gets varied at the rate of an integer per loop. The remaining axis in either subroutine varies at a decimal fraction. To put line 130 into English, "If the distance along the X axis varies faster than the distance along the Y axis, then use the drawing subroutine at line 150. Otherwise use the subroutine at line 240." Clear as mud, right? Anyway, it works.

You should wonder what the fudge factor is in lines 160 and 250. The best way to answer that is tell you to REMark out those two lines, REMark out line 90, install line 95 to say:

95 IF INKEY$="" THEN 95

which will cause a pause after each line is drawn. Then you can compare the difference. Suffice it to say I like the line image that I get with the fudge factor better than the image without the fudge factor.

So what's with the SGN stuff in lines 180 and 270? SGN(n) returns either a -1 or a +1 depending on the value of n when n is not 0. Therefore, in line 180, if the change in to X coordinate as I proceed from X1 to X2 is positive, I use "STEP +1", but if the change is negative, as when the second point is to the left of the first point, then I use "STEP -1".

So here it is. It works. Even if you don't understand it, you can use it to draw straight lines from one point on the screen to another.

How do you do it in assembly? I have absolutely no idea, but I'd sure like to see it done. How about showing me!

```
10 '"LINE" DRAWS FROM ONE POINT (X1,Y1) TO ANOTHER.
20 'DAVE MCGLUMPHY 4429 PAULA LN CHATTANOOGA TN 3741
5
21 '09/03/84   MCI# 181-7759
30 CLS
40 X1=RND(128)-1 : Y1=RND(48)-1 'End of initializations.
41 '
50 X2=RND(128)-1 : Y2=RND(48)-1
60 X3=X1 : X4=X2 : Y3=Y1 : Y4=Y2 'SAVE COORDINATES
70 SR=1 : GOSUB 120 '(SR MEANS SET(1) OR RESET(0))
80 X1=X3 : X2=X4 : Y1=Y3 : Y2=Y4 'RESTORE COORDINATES
90 SR=0 : GOSUB 120 'ERASE LINE
100 X1=X2 : Y1=Y2 'START NEXT LINE @ END OF PREV. LINE
110 GOTO 50 ' DO IT AGAIN.
111 '
120 DX=X2-X1 : DY=Y2-Y1
130 IF ABS(DX) > ABS(DY) THEN GOSUB 150 ELSE GOSUB 240
140 RETURN
141 '
150 'SUBROUTINE FOR WHEN ABS(DX) > ABS(DY)
160 Y1=Y1+.5
170 IF DX=0 THEN IY=1 ELSE IY=DY/DX
180 FOR J=0 TO DX STEP SGN(DX)
190 X=J+X1
200 Y=Y1+J*IY
210 IF SR=1 THEN SET(X,Y) ELSE RESET(X,Y)
220 NEXT J
230 RETURN
231 '
240 'SUBROUTINE FOR WHEN ABS(DX) < ABS(DY)
250 X1=X1+.5
260 IF DY=0 THEN IX=1 ELSE IX=DX/DY
270 FOR J=0 TO DY STEP SGN(DY)
280 Y=J+Y1
290 X=X1+J*IX
300 IF SR=1 THEN SET(X,Y) ELSE RESET(X,Y)
310 NEXT J
320 RETURN
```

LINE REVERSAL PROGRAM
by David R. McGlumphy

[If you suffer from personality reversals, you can make your computer as crazy as you are. Somehow, it seems appropriate that Dave McGlumphy wrote the text and program that follows!]

After leaving the 'puter alone for quite a while to enjoy sailing, my withdrawal pains drove me crazy and I decided to write a subroutine to reverse a string. Will it EVER do anyone any good? Who cares. I had fun. Example: "This is a line." becomes ".enil a si sihT". I wrote my routine in assembly to satisfy my pseudo-masochistic (sic or sick) desires, and then figured out the DATA statement's numbers to allow me to pack the subroutine in a string. It'll handle BASIC strings of normal length, 240 or 255 or whatever it is. Voila!

```
10 'REVSTR 09/03/84 reverses a BASIC string.
20 'Dave McGlumphy 4429 Paula Ln Chattanooga Tn 37415
30 CLS
40 CLEAR 1000
50 DEFINT I-N
60 RV$="12345678911234567892123456789312 3"
70 R2$="12345678911234567892123456789312 3"
80 K=VARPTR(RV$) : AD=PEEK(K+1) + (256*PEEK(K+2))
90 DEFUSR=AD
100 DATA 205,127,10,229,221,225,78,175,71,221,110,1,221,102
110 DATA 2,229,209,9,43,235,65,203,56,78,26,119,121,18,35
120 DATA 27,16,247,201
130 FOR J=0 TO 32 : READ JJ : POKE AD+J,JJ : NEXT J
140 CLS
150 LINEINPUT"What do you want reversed? ";A$
160 CLS
170 PRINT A$
180 X=USR(VARPTR(A$))
190 PRINT A$
200 GOTO 150
```

```
                00100 ;CALLED FROM BASIC TO REVERSE A STRING.
                00110 ;DAVID MCGLUMPHY    09/03/84
                00120 ;4429 PAULA LN.
                00130 ;CHATTANOOGA, TN.   37415
                00140 ;
FFD6            00150          ORG    0FFD6H
FFD6 CD7F0A     00160          CALL   0A7FH      ;GET VARPTR(STRING)
FFD9 E5         00170          PUSH   HL         ;PUT IT
FFDA DDE1       00180          POP    IX         ; INTO IX
FFDC 4E         00190          LD     C,(HL)     ;LENGTH TO C
FFDD AF         00200          XOR    A          ;ZERO OUT A
FFDE 47         00210          LD     B,A        ;PUT IT IN B.
FFDF DD6E01     00220          LD     L,(IX+1)   ;ADDRESS
FFE2 DD6602     00230          LD     H,(IX+2)   ; TO HL
FFE5 E5         00240          PUSH   HL         ;PUT HL
FFE6 D1         00250          POP    DE         ;  INTO DE
FFE7 09         00260          ADD    HL,BC      ;PT HL PAST LAST CHAR
FFE8 2B         00270          DEC    HL         ;PT HL TO LAST CHAR
FFE9 EB         00280          EX     DE,HL ;PT DE TO END, HL TO STRT
FFEA 41         00290          LD     B,C        ;PUT LENGTH IN B
FFEB CB38       00300          SRL    B          ;DIVIDE IT BY TWO
FFED            00310 REVLP    EQU    $
FFED 4E         00320          LD     C,(HL)     ;LEFT STRING PART
FFEE 1A         00330          LD     A,(DE)     ;RIGHT STRING PART
FFEF 77         00340          LD     (HL),A     ;SWITCH
FFF0 79         00350          LD     A,C
FFF1 12         00360          LD     (DE),A     ; THEM
FFF2 23         00370          INC    HL         ;POINT TO NEXT
FFF3 1B         00380          DEC    DE         ; CHAR PAIR
FFF4 10F7       00390          DJNZ   REVLP
FFF6 C9         00400          RET               ;TO BASIC
0000            00410          END
00000 TOTAL ERRORS

REVLP   FFED
```

WHERE

by Michael Brotherton

[This article is reprinted from the Voice of the '80, the fine newsletter of the Fairfield County (Connecticut) Computer Users Group edited by Alan Abrahamson.]

It has been 4 years since my programming debut in the Voice of the '80. The program, written as a joke and given to John Krause, appeared in the newsletter, much to my surprise. I feel it is time to redeem myself.

First, I would like to make a small statement. At my first FCUG meeting, the club's 1st anniversary meeting, the vast wealth of knowledge displayed by the group was overwhelming. I thought I knew my stuff, but I quickly learned the contrary. This club has been an excellent source of information and I'm glad to be a part of it. I noticed, however, some members stereotyped the club's youngsters. We were classified, not surprisingly, as game-playing video game addicts. This is quite understandable since most youngsters use computers for game purposes only. However, please try to keep in mind that games are not the only things which interest the younger computer group. I have worked hard to raise my status above the "gamester" level and it seems to have paid off. I thoroughly enjoy talking with members of the club and hope to meet many more. Enough of this drivel - I believe I have made my point.

This is the first of two articles I intend to write in the immediate future. Most of us have slaved for hours writing programs to do all sorts of stuff. We all know the headaches caused by errors and bugs, and have frequently said "$#%&#", "#%&#&" or when we are really upset, "#$%&%!!!!" Well, some help is on the way. If you pack your lines like I (and of course Sid Gross & his BLINKEY program) do, then you know the problems which arise when the computer says "ERROR" and all you can do is stare and say Where? It seems logical that the computer knows where the error is, but it just won't say. Well, now it can. Although this program does have some limitations, it will give you aid in finding those rotten errors. The program is called, WHERE.

Here is a general breakdown of how the program functions. In the Level II ROM (19ECH), there is a "DOS EXIT" or call to 41A6H. Normally this simply returns when Level II BASIC is active, but Disk Basic uses this call to display the elongated error messages. Here is where I too intercept the error processing. When you execute the program from DOS, it simply sends the message DOSCM (line 1910) to the operating system and returns. This string is interpreted as a command, and then executed. BASIC is entered, memory size is set to 64512 and a USR function is defined at FC13H (the program initialize routine) and the program initializes. Then a USR function is pointed to the DATA TRACE routine (FD7AH). I put the data trace as a USR call since this routine is not always needed.

INITIALIZATION:

140-210 take the code at 41A6H-41A8H and stores it for later. Then, it places a jump to my routine in that area. We need the code replaced by my jump statement since it is part of the regular error processing routine.

220-340 puts my banner on the screen and moves the cursor down below the banner. It then returns to BASIC

PROGRAM:

350-440 clears the screen, stores the error code, and stores the current variable type for later. Storing the variable type is important since the call to 28A7H changes the variable type and produces a TM ERROR when you return to BASIC. [NORTHERN BYTES editor's note: This problem could also be eliminated by calling 2B75H rather than 28A7H to display the message - see page 11 of TRS-80 ROM ROUTINES DOCUMENTED for further information on these two routines.]

450-570 finds the error line, if it exists, and displays the line number. If no line exists, it jumps to the exit routine.

580-660 gets the address of the last byte executed. The next byte, presumably, is the error. A test is made to see if the error is at the beginning of the line. The pointers are moved over the line # and next line pointer stored in the BASIC program.

670-1010 is a list routine. This displays the line and searches for the error position. When the error is found, the marker is displayed, and listing continues.

1020-1140 prints the error marker. Model 3 users might want to change this to display the hand.

1150-1220 prints two carriage returns, removes the excess garbage on the stack, and then restores the variable type.

1230-1290 returns to BASIC from this program. If it was called from BASIC, a RET is executed. If it was called "automatically" (by the jump statement placed by the initializing routine), then the E register is loaded with the error code and execution continues.....

1300,1310 through these statements (which were taken from the code at 41A6H).

1320-1330 enables interrupts and calls 41A9H, the first byte after our jump statement. It is assumed that the current operating system will POP the return address off the stack.

1340-1400 backs up the cursor a 5 spaces. Looking at it now, I don't remember why there is a SPACE imbedded in the backspaces, but, going along with Murphy's Laws, we'll leave it in there since it works!

1450-1500 is the return routine when no error is found.

1510-1580 are assorted messages.

DATA TRACE ROUTINE

1590-1770 is the DATA search routine. All it does is display it's message, change the "last byte executed" to the byte after the last usable data was read in, change the "error line" to the current data line, and (if data exists) call the error find routine. If no data exists, lines 1730-1760 will exit the routine.

1780-1850 displays the no data found message

1860-1870 is the command string executed when the program is executed from DOS.

1880-1900 is the banner (I went in with SUPERZAP after assembling the program and changed the dashes to 8CH's)

1910-1920 execute the DOS command upon entry of the program via DOS.

1930 I'll let you figure this one out for yourselves.

The origin for the program is set fairly low to allow for my density recognition routine. If you do not need any routines at the top of RAM, then by all means move the program up to the tippy top of memory.

There are some changes which may have to be made to allow this program to work with different DOSes. One spot which I see needs changing is the DOS command line. This will be different for each DOS. You can find the correct syntax for your DOS in the manuals or ASK ALAN! These lines are set up for use with NEWDOS/80 Version 2. Some DOSes may require you to type these commands individually. I suggest, in this case, using either a DO file or one of those programs which makes chain files. The rest of the program seems to be compatible, but I have not tried it on every system.

Enjoy this new "game" folks, and please feel free to ask me any question. I shall do my best to answer them.

WHERE/ASM

```
FC00            00100           ORG     0FC00H
FC00 45         00110 MESG0     DEFM    'Error Diagnosis :'
     72 72 6F 72 20 44 69 61 67 6E 6F 73 69 73 20 3A
FC11 0D         00120           DEFB    0DH
FC12 00         00130           DEFB    00H
FC13 2AA641     00140 START2    LD      HL,(41A6H)      ; DOS ERROR ROUTINE
FC16 22FFFC     00150           LD      (DZAP),HL       ; STORE IT
FC19 3AA841     00160           LD      A,(41A8H)
FC1C 32D1FD     00170           LD      (DZAP1),A
FC1F 2143FC     00180           LD      HL,START        ; OUR PROG. BEGINNING
FC22 22A741     00190           LD      (41A7H),HL      ; RIG FOR AUTO INVOKE
FC25 3EC3       00200           LD      A,0C3H          ; JUMP STATEMENT
FC27 32A641     00210           LD      (41A6H),A       ; STORE IT
FC2A 2A2040     00220           LD      HL,(4020H)      ; GET CURSOR POS
FC2D E5         00230           PUSH    HL              ; COPY IT INTO
FC2E D1         00240           POP     DE              ; DE FOR USAGE
FC2F 2102FE     00250           LD      HL,BANNER
FC32 EDA0       00260 LOOP8     LDI                     ; MOVE A BYTE
FC34 7E         00270           LD      A,(HL)          ; GET BYTE
FC35 B7         00280           OR      A               ; SET FLAGS
FC36 20FA       00290           JR      NZ,LOOP8        ; IF NOT 0, CONTINUE
FC38 2A2040     00300           LD      HL,(4020H)      ; GET START AGAIN
FC3B 110001     00310           LD      DE,256
FC3E 19         00320           ADD     HL,DE
FC3F 222040     00330           LD      (4020H),HL      ; MOVE CURSOR DOWN A FEW
FC42 C9         00340           RET                     ; LINES
FC43 F3         00350 START     DI                      ; DISABLE INTERRUPTS
FC44 79         00360           LD      A,C             ; GET ERROR TYPE
FC45 321FFD     00370           LD      (K1K1+1),A      ; STORE IT FOR ERROR EXIT
FC48 AF         00380           XOR     A               ; ZERO A
FC49 32CDFD     00390           LD      (NONE),A        ; ZERO OUT CALLER ROUTINE
```

```
FC4C 3AAF40   01400          LD    A,(40AFH)     ; TYPE OF VARIABLE USED
FC4F 3274FD   01410          LD    (TOVU),A      ; STORE IT FOR RETURN
FC52 CDC901   01420          CALL  01C9H         ; CLEAR SCREEN
FC55 2100FC   01430          LD    HL,MESG0      ; ERROR DIAGNOSIS
FC58 CDA728   01440          CALL  28A7H         ; PRINT IT
FC5B 2AEA40   01450 LTFIND   LD    HL,(40EAH)    ; ERROR LINE
FC5E 2272FD   01460          LD    (TSTR),HL     ; STORE FOR PRINTING
FC61 EB       01470          EX    DE,HL         ; PUT IT IN DE
FC62 CD2C1B   01480          CALL  1B2CH         ; FIND LINE IN MEMORY
FC65 D218FD   01490          JP    NC,NOERR      ; NO ERROR/ERL TOO BIG
FC68 C5       01500          PUSH  BC            ; PUT START OF LINE IN HL
FC69 E1       01510          POP   HL
FC6A E5       01520          PUSH  HL            ; STORE POS
FC6B 2A72FD   01530          LD    HL,(TSTR)     ; GET LINE NUMBER
FC6E CDAF0F   01540          CALL  0FAFH         ; PRINT LINE NUMBER
FC71 3E20     01550          LD    A,20H         ; BLANK AFTER LINE #
FC73 CD3300   01560          CALL  33H           ; PRINT BLANK
FC76 E1       01570          POP   HL            ; GET LINE POS.
FC77 ED5BF740 01580 BTFIND   LD    DE,(40F7H)    ; GET ADDR. OF ERROR -1
FC7B 1A       01590          LD    A,(DE)        ; GET POINT OF ERROR
FC7C B7       01600          OR    A             ; IF A=0, ERROR AT START
FC7D CCC6FC   01610          CALL  Z,HERE        ; OF LINE, SO, MARK IT
FC80 010400   01620          LD    BC,4          ; SKIP LINE #
FC83 09       01630          ADD   HL,BC         ; AND NEXT LINE POINTER
FC84 E5       01640          PUSH  HL            ; STORE HL
FC85 13       01650          INC   DE            ; ADDR. OF ERROR NOW
FC86 D5       01660          PUSH  DE            ; STORE IT FOR LOOPB
FC87 D1       01670 LOOPB    POP   DE            ; GET ADDR. OF ERROR
FC88 E1       01680          POP   HL            ; GET POS. IN TEXT LINE
FC89 CD901C   01690          CALL  1C90H         ; ARE WE AT THE ERROR?
FC8C CCC6FC   01700          CALL  Z,HERE        ; IF SO, MARK IT.
FC8F 7E       01710          LD    A,(HL)        ; GET CHARACTER
FC90 23       01720          INC   HL            ; BUMP POS IN TEXT LINE
FC91 E5       01730          PUSH  HL            ; STORE IT FOR LATER
FC92 D5       01740          PUSH  DE            ; STORE ADDR. OF ERROR
FC93 B7       01750          OR    A             ; SET FLAGS
FC94 CAE2FC   01760          JP    Z,EOL         ; IF A=0, END OF LINE
FC97 FA9FFC   01770          JP    M,TOKEN       ; IF BIT 7 SET, A TOKEN
FC9A CD3300   01780          CALL  33H           ; ELSE PRINT CHARACTER
FC9D 18E8     01790          JR    LOOPB         ; CONTINUE TILL EOL
FC9F FEFB     01800 TOKEN    CP    0FBH          ; TEST FOR " ' " REM
FCA1 CC06FD   01810          CALL  Z,BACKUP      ; BACK UP OVER GARBAGE
FCA4 D67F     01820          SUB   127           ; FIND POS IN RW TABLE
FCA6 47       01830          LD    B,A           ; PUT POS IN B
FCA7 215016   01840          LD    HL,1650H      ; START OF RW TABLE
FCAA 7E       01850 LOOPC    LD    A,(HL)        ; GET A BYTE
FCAB B7       01860          OR    A             ; SET FLAGS
FCAC 23       01870          INC   HL            ; BUMP POS. IN TABLE
FCAD F2AAFC   01880          JP    P,LOOPC       ; JUMP IF NOT START OF RW
FCB0 10F8     01890          DJNZ  LOOPC         ; KEEP GOING TILL OUR RW
FCB2 E67F     01900          AND   127           ; CLEAR BIT 7
FCB4 E5       01910          PUSH  HL            ; STORE HL
FCB5 CD3300   01920          CALL  33H           ; PRINT CHARACTER
FCB8 E1       01930          POP   HL            ; RESTORE HL
FCB9 7E       01940 LOOPD    LD    A,(HL)        ; GET NEXT BYTE OF RW
FCBA 23       01950          INC   HL            ; BUMP POINTER
FCBB B7       01960          OR    A             ; SET FLAGS
FCBC FA87FC   01970          JP    M,LOOPB       ; DONE WITH TOKEN
FCBF E5       01980          PUSH  HL            ; STORE HL
FCC0 CD3300   01990          CALL  33H           ; PRINT CHARACTER
FCC3 E1       02000          POP   HL            ; RESTORE HL
FCC4 18F3     02010          JR    LOOPD         ; CONTINUE TILL DONE
FCC6 F5       02020 HERE     PUSH  AF            ; STORE BYTE HERE
FCC7 3E20     02030          LD    A,32          ; SPACE
FCC9 CD3300   02040          CALL  33H           ; PRINT SPACE
FCCC 3E99     02050          LD    A,153         ; GRAPHIC MARKER
FCCE CD3300   02060          CALL  33H           ; PRINT IT
FCD1 3E99     02070          LD    A,153         ; MARKER AGAIN
FCD3 CD3300   02080          CALL  33H           ; PRINT IT
FCD6 3E99     02090          LD    A,153         ; YOU KNOW BY NOW
FCD8 CD3300   02100          CALL  33H           ; PRINT IT
FCDB 3E20     02110          LD    A,20H         ; SPACE
FCDD CD3300   02120          CALL  33H           ; PRINT IT
FCE0 F1       02130          POP   AF            ; RESTORE ORIGINAL BYTE
FCE1 C9       02140          RET                 ; DONE
FCE2 3E0D     02150 EOL      LD    A,13          ; CARRIAGE RETURN
FCE4 CD3300   02160          CALL  33H           ; PRINT LINE
FCE7 3E0D     02170          LD    A,13          ; ANOTHER CR
FCE9 CD3300   02180          CALL  33H           ; PRINT IT
FCEC E1       02190          POP   HL            ; GET RID OF
FCED E1       02200          POP   HL            ; GARBAGE ON STACK
FCEE 3A74FD   02210          LD    A,(TOVU)
FCF1 32AF40   02220          LD    (40AFH),A     ; RESTORE VARIABLE TYPE
FCF4 3ACDFD   02230 DEXIT    LD    A,(NONE)      ; WHO CALLED THIS?
FCF7 B7       02240          OR    A             ; SET FLAGS
FCF8 C0       02250          RET   NZ            ; BACK TO CALLER ROUTINE
FCF9 E1       02260          POP   HL            ; PULL RETURN OFF STACK
FCFA 3A9A40   02270          LD    A,(409AH)     ; GET ERROR CODE
FCFD 5F       02280          LD    E,A           ; PUT IT IN E
FCFE FB       02290          EI
FCFF 0000     02300 DZAP     DEFW  0000
FD01 00       02310 DZAP1    DEFB  00
FD02 FB       02320          EI
FD03 CDA941   02330          CALL  41A9H         ; DOS WILL TAKE OVER NOW
FD06 E5       02340 BACKUP   PUSH  HL            ; STORE IT
FD07 F5       02350          PUSH  AF            ; STORE IT TOO
FD08 2111FD   02360          LD    HL,BUMESG     ; BACKUP STRING
FD0B CDA728   02370          CALL  28A7H         ; PRINT IT
FD0E F1       02380          POP   AF
FD0F E1       02390          POP   HL
FD10 C9       02400          RET
FD11 0808     02410 BUMESG   DEFW  0808H         ; BACK SPACES
FD13 0808     02420          DEFW  0808H
FD15 0820     02430          DEFW  2008H
FD17 00       02440          DEFB  00H           ; MESG. TERMINATOR
FD18 3A74FD   02450 NOERR    LD    A,(TOVU)      ; RESTORE VARIABLE TYPE
FD1B 32AF40   02460          LD    (40AFH),A     ; DESTROYED BY 27A8H CALL
FD1E 3E00     02470 K1K1     LD    A,00H         ; STORED FOR LATER
FD20 329A40   02480          LD    (409AH),A     ; USE ON ERROR EXIT
FD23 FB       02490          EI
FD24 C3F4FC   02500          JP    DEXIT         ; DOS EXIT
FD27 4E       02510 MESGN    DEFM  'No Error Found!'
     6F 20 45 72 72 6F 72 20 46 6F 75 6E 64 21
FD36 0D       02520          DEFB  0DH
FD37 00       02530          DEFB  00H
FD38 44       02540 MESGD    DEFM  'Data examination reveals last set of data
     61 74 61 20 65 78 61 6D 69 6E 61 74 69 6F 6E 20        accessed was :'
     72 65 76 65 61 6C 73 20 6C 61 73 74 20 73 65 74
     20 6F 66 20 64 61 74 61 20 61 63 63 65 73 73 65
     64 20 77 61 73 20 3A
FD7B 0D       02550          DEFB  0DH
FD7C 00       02560          DEFB  00H
FD7D 0000     02570 TSTR     DEFW  0000H         ; TEMP. LINE # STORAGE
FD7F 00       02580 TOVU     DEFB  00H           ; TYPE OF VARIABLE USED
FD80 F3       02590 STARTD   DI
FD81 2138FD   02600          LD    HL,MESGD
FD84 3AAF40   02610          LD    A,(40AFH)
FD87 3274FD   02620          LD    (TOVU),A
FD8A CDA728   02630          CALL  28A7H
FD8D 21FF40   02640          LD    HL,40FFH      ; BYTE AFTER LAST USABLE
FD90 2279FC   02650          LD    (BTFIND+2),HL ; DATA READ IN
FD93 21DA40   02660          LD    HL,40DAH      ; LINE # FOR DATA STMT
FD96 225CFC   02670          LD    (LTFIND+1),HL ; STORE IT
FD99 2AFF40   02680          LD    HL,(40FFH)    ; GET POS
FD9C 7E       02690          LD    A,(HL)        ; WHAT'S THERE
FD9D B7       02700          OR    A             ; SET FLAGS
FD9E 2810     02710          JR    Z,NODATA      ; IF NO DATA, SKIP
FDA0 CD5BFC   02720          CALL  LTFIND
FDA3 21EA40   02730 K11      LD    HL,40EAH      ; RESTORE OTHER PART
FDA6 225CFC   02740          LD    (LTFIND+1),HL ; (ERROR FINDER)
FDA9 21F740   02750          LD    HL,40F7H
FDAC 2279FC   02760          LD    (BTFIND+2),HL
FDAF C9       02770          RET
FDB0 21B3FD   02780 NODATA   LD    HL,MESGND     ;NO DATA FOUND
FDB3 CDA728   02790          CALL  28A7H         ; PRINT MESSAGE
FDB6 3A74FD   02800          LD    A,(TOVU)      ; RESET VARIABLE TYPE
FDB9 32AF40   02810          LD    (40AFH),A     ; TO FIX TM ERROR
FDBC 18E5     02820          JR    K11           ; RESTORE THINGS CHANGED
FDBE 4E       02830 MESGND   DEFM  'No data statement found!'
     6F 20 64 61 74 61 20 73 74 61 74 65 6D 65 6E 74
     20 66 6F 75 6E 64 21
FDCE 0D00     02840          DEFW  000DH
FDD0 00       02850 NONE     DEFB  00H
FDD1 42       02860 DOSCM    DEFM  'BASIC,64512,DEFUSR0=&HFC13:X=USR(0)'
     41 53 49 43 2C 36 34 35 31 32 2C 44 45 46 55 53       DEFUSR0=&HFD7A'
     52 30 3D 26 48 46 43 31 33 3A 58 3D 55 53 52 28
     30 29 3A 44 45 46 55 53 52 30 3D 26 48 46 44 37
     41
FDFF 0D00     02870          DEFW  000DH
FE01 20       02880 BANNER   DEFM  '———— Error Finder  Ver. 2.1  By
Mike Brotherton ———————————————— X = USR(0) to trace data '
     20 20 20 20 20 20 20 20 20 45 72 72 6F 72 20 46
     69 6E 64 65 72 20 20 20 56 65 72 2E 20 32 2E 31
     20 20 20 42 79 20 4D 69 6B 65 20 42 72 6F 74 68
     65 72 74 6F 6E 20 2D 2D 2D 2D 2D 2D 2D 2D 2D 20
```

```
    20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
    20 20 20 20 20 20 20 28 58 28 30 28 55 53 52 28 30
    29 28 74 6F 28 74 72 61 63 65 28 64 61 74 61 28
FE73 20          01890          DEFM    '-------------'
    20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
FE82 00          01900          DEFB    00
FE83 21CEFD      01910 START1   LD      HL,DOSCM
FE86 C30544      01920          JP      4405H          ; DOS COMMAND/DEAD END
FE83             01930          END     START1         ; BASIC AUTOMATICALLY!
00000 TOTAL ERRORS

BACKUP  FD06   BANNER  FE02   BTFIND  FC77   BUMESG  FD11   DEXIT  FCF4
DOSCM   FDCE   DZAP    FCFF   DZAP1   FD01   EOL     FCE2   HERE   FCC6
K11     FD98   K1K1    FD1E   LOOP8   FC32   LOOPB   FC87   LOOPC  FCAA
LOOP0   FCB9   LTFIND  FC5B   MESG0   FC00   MESGD   FD38   MESGN  FD27
MESGND  FD83   NODATA  FDA5   NOERR   FD18   START   FC43   START1 FE83
START2  FC13   STARTD  FD75   TOKEN   FC9F   TOVU    FD74   TSTR   FD72
WONE    FDCD
```

CONVERTING SUPER UTILITY PLUS VERSION 3 TO A /CMD FILE

by Arne Rohde

The Super Utility Plus program, version 3.0, provided on a self-booting disk can be converted relatively easily to a /CMD file which can be called directly from the DOS READY prompt. It is Super Utility itself, with the memory modify feature which makes this possible.

Before I continue, I would like to make one thing perfectly clear. I am not a fan of Super Utility, and I do not own a copy of it. The conversion was done for a friend who has it and uses it. I admit that I am impressed by the programming which Kim Watt has done, and the program can certainly perform many things, but I am sticking to my trusty old Superzap from NEWDOS/80, and a home-grown zap utility which can be modified and extended and converted as much as I please. In other words, I am not in the Super Utility backup industry, and have no intention of getting into it. Also the usual warning - do NOT use this procedure for making backup copies for others. If you need SU+ then buy a copy and make backup copies for your own personal use.

The version that I have seen will load and use up to about address D500H (all values from now on will be in hexadecimal). Unless you perform some disk I/O the area from address D500 should be free for inserting any desired code and saving the code which will be overlaid by the DOS. Before you start making /CMD files from SU+, you should make all the necessary configuration changes. You CANNOT permanently configure the /CMD version using the write to disk option from the program configuration, you will have to produce a new /CMD file from scratch if you want it reconfigured permanently.

Start the process by loading SU+ and configuring as desired. Then use the memory display or string search to find a reference to the keyboard strobe at 3840. On the Model I the instruction sequence 3A4038 should be found at or near location 4301, on the Model III at address 42FB. This loads the keyboard strobe containing the Break key into A. The location of this instruction should be noted, as it is required later.

Now use the memory modify to enter the following instructions from address E700:

Location	Value		Instruction	
E700	F3		DI	
E701	313141		LD	SP,4131H
E704	010012	**	LD	BC,1200H
E707	110040		LD	DE,4000H
E70A	2100D5		LD	HL,D500H
E70D	EDB0		LDIR	
E70F	F1		POP	AF
E710	ED47		LD	I,A
E712	F1		POP	AF
E713	E1		POP	HL
E714	D1		POP	DE
E715	C1		POP	BC
E716	D9		EXX	
E717	08		EX	AF,AF'
E718	FDE1		POP	IY
E71A	DDE1		POP	IX
E71C	F1		POP	AF
E71D	E1		POP	HL
E71E	D1		POP	DE
E71F	C1		POP	BC
E720	C30143	*	JP	4301H

The address in the last line should be set to the address found earlier. On the Model III the new value will probably be 42FB making the instruction C3FB42.

This is the code for restoring the status and jumping to the address where SU+ can be restarted. I have not taken the time to find out whether it is necessary to save and restore all the registers, but it is very important that the I register (used for interrupt base addressing) is restored. This register is normally not used in TRS-80 programs, which is probably why Kim Watt chose to check it and catch some of the would-be back up copies.

The next piece of code is the code which will be executed when the /CMD file is to be produced. We will place this code at location E730, out of the way of the previous code.

Location	Value		Instruction	
E730	F3		DI	
E731	C5		PUSH	BC
E732	D5		PUSH	DE
E733	E5		PUSH	HL
E734	F5		PUSH	AF
E735	DDE5		PUSH	BC
E737	FDE5		PUSH	BC
E739	08		EX	AF,AF'
E73A	D9		EXX	
E73B	C5		PUSH	BC
E73C	D5		PUSH	DE
E73D	E5		PUSH	HL
E73E	F5		PUSH	AF
E73F	ED57		LD	A,I
E741	F5		PUSH	AF
E742	ED7302E7	**	LD	(E702H),SP
E746	3E3A		LD	A,3AH
E748	320143	*	LD	(4301H),A
E74B	214038		LD	HL,3840H
E74E	220243	*	LD	(4302H),HL
E751	010012	**	LD	BC,1200H
E754	1100D5		LD	DE,D500H
E757	210040		LD	HL,4000H
E75A	EDB0		LDIR	
E75C	76 or C30000		HALT or JP 0000H	

The lines containing single asterisks will have to modified depending on where the original load instruction was found. They will probably be 42FB and 42FC for the Model III. The first of the two instructions converts the original instruction code back to the original value, and the second instruction restores the address from the instruction. The lines marked with double asterisks will have to be modified if the instructions are moved to an address other than E700.

The last line of the status save can be a HALT instruction on the Model I, or a jump to location 0000, which will not reset the hardware and disk density. On Model III the jump instruction should probably be used.

After making these changes, a DOS disk should be mounted in drive zero, and the instruction at location 4301 (or 42FB or wherever) modified. Change the original 3A4038 value to the following value:

C330E7

which is a jump to the code which will save the status and reboot the DOS system. Now press Shift Break in SU+ to make it return to the original menu. Instead of seeing the menu, your DOS system should reboot.

After the DOS has been loaded, perform the following DUMP instruction as the first thing you do.

DUMP SU/CMD,5200H,E77FH,E700H

This is the format for NEWDOS/80 which allows code to be dumped from address 5200H. If you are using a DOS which does not allow you to dump from address 5200H then either get NEWDOS/80, or modify the instructions above so that the code is moved higher up in memory. If you can only dump from location 7000H and above you're out of luck unless you are sure you can use addresses above about CFB0H in SU+. I haven't tried it, and I haven't found the lowest address unused by SU+.

One final word of caution. Take the SU+ disk out of the drive before you start this procedure, and use a backup copy of your DOS system for your first attempt. And if you can't get the procedure to work then either find someone conversant in TRS assembler code, or give up the project. Please do NOT write and ask for advice. By the time you read this I will be moved to another as yet unknown address, and I cannot guarantee that I will receive the letters or answer the correspondence.

DOCUMENTATION FOR MENU/CMD
By Jim Doffing

MENU/CMD is a program that was written because of laziness on my part, and because I thought that the computer should be able to do most of the redundant things that I had been doing, such as typing DIR after returning to DOS. There were other things that I did repeatedly, such as going to DEBUG, going to BASIC, and typing in the names of the programs that I wanted to run.

When I started this project, there weren't any such programs on the market. Since then there have been two introduced. When you are not the first, you are a copycat so I decided to release this program into public domain. It is free for the taking and it has served me well for 2 years now.

This program is rather lengthy, but that is because I wanted it to work from another TRS-80 using modems or directly connected by RS-232 cables.

So much for the introduction.

To install this program on a TRSDOS 1.3 Diskette you MUST follow directions to the letter. What you are going to do is install this program as a system file. When you get through you will not be able to get this program to show up on a directory. It will be hidden just like the FORMAT & the BACKUP files are.

Remember that the following instructions must be just as I have them outlined. If you downloaded these programs from a BBS you will probably have to convert them to /CMD format.

Instructions:

1. Make a backup of your original TRSDOS 1.3 Diskette (I will refer to the new disk as the installation diskette).

2. Erase all three visible files. (LPC/CMD, MEMTEST/CMD & HERZ50/BLD) from the installation diskette.

3. Insert the installation diskette into Drive zero.

4. Type "CREATE DUMMY:0 (REC=183)", then press the <ENTER> key.

At this point, If you type "FREE:0" and press the <ENTER> key, you should have displayed on the screen:

```
                    Free Space Map
Trk #   TRSDOS  ------------------------  Drive: 0
00-04   XXXXXX : XXXXXX : XXXXXX : XXXXXX : XXXXXX
05-09   XXXXXX : XXXXXX : XXXXXX : XXXXXX : XXXXXX
10-14   XXXXXX : XXXXXX : XXXXXX : XXXXXX : XXXXXX
15-19   XXXXXX : XXXXXX : Direct : XXXXXX : XXXXXX
20-24   XXXXXX : XXXXMM : ...... : ...... : ......
25-29   ...... : ...... : ...... : ...... : ......
30-34   ...... : ...... : ...... : ...... : ......
35-39   ...... : ...... : ...... : ...... : ......
TRSDOS Ready
.....................................................
```

The "MM" at the end of track 21 will show up as periods on your video display. This is where MENU/CMD will be located. All the available spaces up that point should be "X". If there are any periods on the Free Space Map between track 00 and track 21 then you will have to increase the number of "REC=183" by three for every period and go back to step 4. If there are X's in the positions of the M's or beyond then you will have to decrease the number of "REC=183" by three for each "X" and go back to step 4. As a final check Type "DIR :0 (INV,SYS)" and press the <ENTER> key. You should get four files displayed:

BASIC/CMD
CONVERT/CMD
XFERSYS/CMD
DUMMY

5. At this point you should insert the installation diskette into Drive #1.

6. Insert the Diskette containing MENU/CMD into Drive #0 and copy MENU/CMD to the installation diskette.

7. With the installation Diskette in Drive #1 type "MENINSTL" and press the <ENTER> key.

8. At TRSDOS Ready you can type "MENU" and press the <ENTER> key. When the MENU is displayed you should see only one file, "DUMMY". If you now press the "K" key and then the "Y" key you will have killed "DUMMY". If you now type DIR and press the <ENTER> key, there shouldn't be any file names displayed. If MENU/CMD or any other file name is displayed, you didn't follow the instructions and you will have to start over.

9. If you have followed the above instructions you will now be in possession of a diskette that will save you much time.

10. MENU will now show up on the library of commands by typing "LIB" and pressing the <ENTER> key.

11. If the program does not work (heaven forbid) try again and remember to follow the sequence exactly.

Features:

I find that if you type "AUTO MENU" and press the <ENTER> key, when ever you use that diskette in drive #0 and press the reset key it will automatically give a MENU.

After the MENU program is active:

You will notice that there are the "<" and the ">" symbols surrounding the first file name on the diskette.

Pressing the "K" key will result in the message: Are you sure you want to KILL (file):0? (Y/N)

If you press the "Y" key the program will be killed. If you press the <BREAK> or the "N" keys the kill will be aborted and the "<" and the ">" will return to the first file name.

Pressing the "C" key will result in the message: Copy to which drive?

If you press the "1" key, that file will be copied to drive one and upon completion you will be returned to the display of the menu.

Pressing the <ENTER> key will run any file with an extension of "/CMD" or will DO any file with the extension of "/BLD" All others will be treated as BASIC files and will be auto loaded with BASIC. If the file is a data file you will receive an error from BASIC. I have found that it is best to use the "/BAS" extension so you can tell a BASIC file. The reason that most people leave the "/BAS" extension off is so that they won't have to type it every time that they want to run the program. Now you don't have to type a BASIC file name again. If you end your basic programs with (CMD"S") instead of (END) then you will automatically be back in the MENU, ready for selection of your next program.

Pressing the "B" key will put you in BASIC.

Pressing the "D" key will put you in DEBUG.

Pressing one of the arrow keys will move the "<" and the ">" to the next file name in the direction of that arrow key.

Pressing a number key will get a MENU of that drive. It will not violate the maximum number of drives established at powerup.

Drawbacks:

Some programs do not use the returns provided for returning to TRSDOS and may not function correctly. I have found that most programs will work just fine. The programs that execute DOS commands from within the program are usually the ones that I have problems with. When you run across one like that you will have to use regular TRSDOS 1.3 with it. The other programs that I have had trouble with are the ones that leave an active stack in the area of the MENU program. One of the first things that MENU does when it loads is to reestablish the stack at the TRSDOS location to avoid troubles.

When DOing a file the program will not return to the MENU so you will have to type "MENU" again from TRSDOS Ready. Hopefully you will not hurt your fingers when you do so.

My Name is Jim Doffing I live at 5602 N. 49th, Tacoma, Washington 98407. If you make any changes to the MENU program I would be interested. If you cannot manage the transfer for what ever reason, I will install the program for you on a copy of TRSDOS 1.3 that you provide for $10.00 to cover my time and return postage. Good Luck and Happy Computing.

[The source code listing for MENINSTL/CMD follows the listing for the MENU/CMD program below]

```
00010 ;MENU/CMD by Jim Doffing of Tacoma, Washington  11/26/83
00020 ;This program will make it easier for the novice
00030 ;computer user to make the most of (his,her) Mod III, IV
00040 ;using TRSDOS 1.3.  It will recognize programs with an
00050 ;extension of /CMD or /BLD, treating all others as basic
00060 ;programs.  By using the arrow keys you can move the cur-
00070 ;sors to enclose the desired program.  Pressing the ENTER
00080 ;key, will run the program or DO the /BLD File.  Pressing
00090 ;a number between 0 and 3 will get a directory of that
00100 ;drive, pressing BREAK will return to TRSDOS 1.3., and
00110 ;pressing 'D' will load and run DEBUG.  MENU/CMD will
00120 ;modify the return to DOS vector so the system will load
00130 ;and run MENU/CMD instead of return to DOS.
4E00    00140           ORG     4E00H           ;PROGRAM LOCATION & START
```

```
4E00 319F40   00150 ST      LD      SP,409FH        ;RESTORE STACK POINTER
4E03 F5       00160         PUSH    AF              ;SAVE REGISTERS 'AF'
4E04 3EFF     00170 ST1     LD      A,0FFH          ;PUT OVERLAY 15 NUMBER
4E06 329742   00180         LD      (4297H),A       ;  INTO RETDOS OVERLAY
4E09 F1       00190         POP     AF              ;  AND 'AF'
4E0A C31D4E   00200         JP      START           ;GO TO PROG START
              00210 ;SYSTEM CALLS AND STANDARD ADDRESSES
4B0D          00220 READ    EQU     4B0DH   ;SYSTEM "READ DIR SECTOR" ROUTINE
021B          00230 PRTLN   EQU     021BH   ;SYSTEM "PRINT LINE OF TEXT" RTN
4020          00240 CURPOS  EQU     4020H   ;SYSTEM STOR FOR CURSOR POSITION
0049          00250 INKEY   EQU     0049H   ;SYSTEM "GET KEY PRESS" ROUTINE
402D          00260 DOS     EQU     402DH   ;SYSTEM "RETURN TO DOS" CALL
0033          00270 VDCHR   EQU     0033H   ;SYSTEM "CHARACTER TO VIDEO" RTN
429C          00280 CONDOS  EQU     429CH   ;EXECUTE DOS COMMAND ROUTINE
4225          00290 DOSBUF  EQU     4225H   ;DOS COMMAND BUFFER ADDRESS
4411          00300 TOPMEM  EQU     4411H   ;TOP OF USER MEMORY STORAGE
4D00          00310 SECBUF  EQU     4D00H   ;PROGRAM "SECTOR STORAGE" ADDRESS
FFFF          00320 DRNO    EQU     0FFFFH  ;STORAGE OF LAST DRIVE #
004E          00330 SECEND  EQU     4EH     ;PROGRAM "END OF SECTOR" CHECK
0020          00340 SPACE   EQU     20H     ;ASCII ' ' (SPACE)
001F          00350 CLS     EQU     1FH     ;CLEAR TO BOTTOM OF SCREEN
000D          00360 CR      EQU     0DH     ;CARRIAGE RETURN & LINE FEED
001A          00370 DWLN    EQU     1AH     ;DOWN ONE LINE ON VIDEO
001B          00380 UPLN    EQU     1BH     ;UP ONE LINE ON VIDEO
0018          00390 BKSP    EQU     18H     ;BACKSPACE WITHOUT ERASE
0008          00400 BKSPER  EQU     08H     ;BACKSPACE WITH ERASE
              00410 ;PROGRAM STORAGE OF VARIABLES
4E0D          00420 TABLE   EQU     $       ;ADDRESS FOR LOADING TRK AND SEC
4E0D 03       00430 SEC     DEFB    03H     ;FIRST SECTOR OF DIRECTORY
4E0E 11       00440 TRK     DEFB    11H     ;TRACK NUMBER OF DIRECTORY
4E0F 00       00450 DR      DEFB    00H     ;BINARY DRIVE NUMBER
4E10 00       00460 TEMSEC  DEFB    00H     ;TEMP STOR OF ACTIVE SECTOR #
4E11 3000     00470 FILOFF  DEFW    0030H   ;OFFSET FROM ONE FILE TO NEXT
4E13 0500     00480 FILNAM  DEFW    0005H   ;OFFSET FILE START TO FILE NAME
0002          00490 HLSAV   DEFS    2       ;IN MEM NEXT FILE LOCAT.
0002          00500 FILE1   DEFS    2       ;PRESENT FILE LOCATION
0002          00510 HL2     DEFS    2       ;TEMP STOR FOR 'HL' REGISTER
0002          00520 LAST    DEFS    2       ;LAST FILE NAME DISPLAYED
              00530 ;
              00540 ;START OF PROGRAM
              00550 ;
4E1D E6F0     00560 START   AND     0F0H            ;MASK OFF OVERLAY
4E1F FEF0     00570         CP      0F0H            ;CALL FROM MODIFIED DOS
              00580                                 ;ENTRY?
4E21 200F     00590         JR      NZ,STRT1        ;IF NOT GOTO START1
4E23 3E0F     00600         LD      A,0FH           ;TURN OFF
4E25 CD0952   00610         CALL    PRTCHR          ;  CURSOR
4E28 3E80     00620         LD      A,080H          ;DEFAULT CURSOR CHAR
4E2A 322340   00630         LD      (4023H),A       ;STORE IN CURSOR CHAR
4E2D 3AFFFF   00640         LD      A,(DRNO)        ;SEE IF MAYBE STILL GOOD
4E30 1808     00650         JR      START1          ;GET DRIVE #0 MENU
4E32 7E       00660 STRT1   LD      A,(HL)          ;CHECK TO FIND
4E33 FE0D     00670         CP      0DH             ;EITHER A CARRIAGE RETURN
4E35 2819     00680         JR      Z,PUTMIN        ;if CR then Drive #0
4E37 FE3A     00690         CP      ':'             ;OR A COLON
4E39 2002     00700         JR      NZ,START1       ;if not assume drive #
4E3B 23       00710         INC     HL              ;IF SO THEN NEXT CHAR
4E3C 7E       00720         LD      A,(HL)          ;INTO THE 'A' REGISTER
4E3D 32FFFF   00730 START1  LD      (DRNO),A        ;SAVE FOR PROG ENDS
4E40 D630     00740         SUB     '0'             ;MAKE IT BINARY
4E42 DA504E   00750         JP      C,PUTMIN        ;IF < 0 THEN DRIVE #0
4E45 4F       00760         LD      C,A             ;PUT VALUE INTO 'C'
4E46 3A1344   00770         LD      A,(4413H)       ;# OF DRIVES IN SYS TO'A'
4E49 B9       00780         CP      C               ;IS IT TOO HIGH
4E4A 3804     00790         JR      C,PUTMIN        ;IF SO DRIVE #0
4E4C 79       00800         LD      A,C             ;PUT VALUE BACK IN 'A'
4E4D C3574E   00810         JP      DISDIR          ;ELSE CONTINUE
4E50 3E30     00820 PUTMIN  LD      A,'0'           ;IF NOT THEN
4E52 32FFFF   00830         LD      (DRNO),A        ;SAVE FOR PROG ENDS
4E55 D630     00840         SUB     '0'             ;DRIVE #0
4E57 320F4E   00850 DISDIR  LD      (DR),A          ;INSERT DRIVE NUM IN PROG
4E5A CD084F   00860         CALL    DISMES          ;DISPLAY SIGN-ON MESSAGE
4E5D ED5B0D4E 00870         LD      DE,(TABLE)      ;DE=> TRK17 AND SEC3
4E61 7B       00880 DISD0   LD      A,E             ;SECTOR NUM INTO 'A'
4E62 32104E   00890         LD      (TEMSEC),A      ;STORE FOR LATER
4E65 21004D   00900         LD      HL,SECBUF       ;HL=> SECTOR STORAGE LOC
4E68 3A0F4E   00910         LD      A,(DR)          ;GET BINARY DRIVE# AND
4E6B 4F       00920         LD      C,A             ;  PUT IT INTO 'C'
4E6C 0600     00930         LD      B,00H           ;B=> 256 BYTE SECTOR READ
4E6E CD0D4B   00940         CALL    READ            ;PUT SECTOR INTO BUFFER
4E71 22154E   00950 DISD1   LD      (HLSAV),HL      ;STORE 'HL' FOR LATER
4E74 7E       00960         LD      A,(HL)          ;FIRST FILE CHAR INTO 'A'
4E75 CB5F     00970         BIT     3,A             ;IS IT AN INVISIBLE FILE
4E77 2004     00980         JR      NZ,NXTFIL       ;IF SO NEXT FILE
4E79 FE00     00990         CP      00H             ;IS IT A BLANK POSITION
4E7B 2025     01000         JR      NZ,DISFIL       ;IF SO, DISPLAY NAME
4E7D 2A154E   01010 NXTFIL  LD      HL,(HLSAV)      ;GET LAST FILE LOCATION
4E80 ED5B114E 01020         LD      DE,(FILOFF)     ;DE=> OFFSET TO NEXT FILE
4E84 19       01030         ADD     HL,DE           ;ADD WITH SUM IN 'HL'
4E85 7C       01040         LD      A,H             ;'H' INTO 'A' FOR TEST
4E86 FE4E     01050         CP      SECEND          ;IS IT PAST END OF BUFFER
4E88 3002     01060         JR      NC,NXTSEC       ;IF => THEN NEXT SECTOR
4E8A 18E5     01070         JR      DISD1           ;IF NOT GET NEXT FILE
4E8C 3A104E   01080 NXTSEC  LD      A,(TEMSEC)      ;A=> PRESENT SECTOR NUM
4E8F ED5B0D4E 01090         LD      DE,(TABLE)      ;DE=> TRK & SEC
4E93 3C       01100         INC     A               ;'A'='A'+1
4E94 5F       01110         LD      E,A             ;NEW SECTOR INTO 'E'
4E95 FE13     01120         CP      19D             ;ARE WE PAST SECTOR 18
4E97 38C8     01130         JR      C,DISD0         ;IF < 19 THEN NEXT SECTOR
4E99 2A2040   01140         LD      HL,(CURPOS)     ;ELSE GET CURSOR POSITION
4E9C 221B4E   01150         LD      (LAST),HL       ;STORE FOR COMPARE
4E9F C3D14F   01160         JP      SEL1            ;GOTO SELECTION ROUTINE
              01170 ;
              01180 ;NOW WE DISPLAY THE FILE NAME OF THE USER FILE
              01190 ;
4EA2 ED5B134E 01200 DISFIL  LD      DE,(FILNAM)     ;DE=> OFFSET TO FILE NAME
4EA6 19       01210         ADD     HL,DE           ;ADD TO 'HL' INTO 'HL'
4EA7 0608     01220         LD      B,8             ;B=> MAX CHARS IN NAME
4EA9 0E00     01230         LD      C,00H           ;C=> ZERO FOR COUNTER
4EAB 7E       01240 DISF0   LD      A,(HL)          ;NEXT CHAR INTO 'A'
4EAC FE20     01250         CP      SPACE           ;IS IT A SPACE
4EAE 2807     01260         JR      Z,EXT           ;IF SO PUT ON EXTENSION
4EB0 CD0952   01270         CALL    PRTCHR          ;ELSE PRINT CHR TO VIDEO
4EB3 0C       01280         INC     C               ;KEEP TRACK OF VIDEO CHRS
4EB4 23       01290         INC     HL              ;SET UP FOR NEXT CHAR
4EB5 10F4     01300         DJNZ    DISF0           ;DISPLAY NAME UNTIL DONE
4EB7 CD034F   01310 EXT     CALL    LOOP            ;MOVE 'HL' TO EXT. CHRS
4EBA 0603     01320         LD      B,3             ;MAX NUM EXT. CHARS
4EBC 7E       01330         LD      A,(HL)          ;FIRST CHR INTO 'A'
4EBD FE20     01340         CP      SPACE           ;IS IT A SPACE (NO EXT)
4EBF 281Z     01350         JR      Z,DRNUM         ;IF SO ADD DRIVE NUMBER
4EC1 3E2F     01360         LD      A,'/'           ;ELSE PUT SLASH INTO 'A'
4EC3 CD0952   01370         CALL    PRTCHR          ;PUT IT ON VIDEO
4EC6 0C       01380         INC     C               ;KEEP TRACK OF VIDEO CHRS
4EC7 7E       01390 EXT1    LD      A,(HL)          ;PUT NEXT CHR INTO 'A'
4EC8 FE20     01400         CP      SPACE           ;IS IT A SPACE
4ECA 2807     01410         JR      Z,DRNUM         ;IF SO ADD DRIVE NUMBER
4ECC CD0952   01420         CALL    PRTCHR          ;ELSE PUT ON VIDEO
4ECF 0C       01430         INC     C               ;KEEP TRACK OF VIDEO CHRS
4ED0 23       01440         INC     HL              ;POINT TO NEXT CHAR
4ED1 10F4     01450         DJNZ    EXT1            ;DO UNTIL DONE WITH EXT
4ED3 3E3A     01460 DRNUM   LD      A,':'           ;A=> COLON
4ED5 CD0952   01470         CALL    PRTCHR          ;PUT ON VIDEO
4ED8 0C       01480         INC     C               ;KEEP TRACK
4ED9 3AFFFF   01490         LD      A,(DRNO)        ;A=> ASCII DRIVE NUMBER
4EDC CD0952   01500         CALL    PRTCHR          ;PUT ON VIDEO
4EDF 0C       01510         INC     C               ;KEEP TRACK
4EE0 3A2040   01520         LD      A,(CURPOS)      ;A=> LSB CURSOR POSITION
4EE3 E63F     01530         AND     3FH             ;MASK BITS 6 & 7
4EE5 FE32     01540         CP      50D             ;ARE WE PAST TAB(49)
4EE7 380C     01550         JR      C,DRNUM1        ;IF NOT GOTO NEXT VID POS
4EE9 3E0D     01560         LD      A,CR            ;ELSE ISSUE CAR RET
4EEB CD0952   01570         CALL    PRTCHR          ;PUT ON SCREEN
4EEE 3E20     01580         LD      A,SPACE         ;  AND A SPACE
4EF0 CD0952   01590         CALL    PRTCHR          ;  ALSO ON SCREEN
4EF3 1888     01600         JR      NXTFIL          ;    THEN NEXT FILE
4EF5 3E10     01610 DRNUM1  LD      A,16D           ;CHRS TO NEXT VIDEO POS.
4EF7 91       01620         SUB     C               ;SUBTRACT CHARS TO VIDEO
4EF8 47       01630         LD      B,A             ;PUT INTO 'B' FOR LOOP
4EF9 3E20     01640 DRNUM2  LD      A,SPACE         ;A=> SPACE TO BE PRINTED
4EFB CD0952   01650         CALL    PRTCHR          ;PUT ON VIDEO
4EFE 10F9     01660         DJNZ    DRNUM2          ;DO UNTIL DONE
4F00 C37D4E   01670         JP      NXTFIL          ;GO GET NEXT FILE
4F03 78       01680 LOOP    LD      A,B             ;CHRS LEFT IN 'B' FOR LOP
4F04 FE00     01690         CP      00H             ;COMPARE TO ZERO (DONE?)
4F06 C8       01700         RET     Z               ;IF SO RETURN
4F07 23       01710 LOOP1   INC     HL              ;SPACE OVER 1 CHR AT TIME
4F08 10FD     01720         DJNZ    LOOP1           ;  UNTIL AT EXTENSION
4F0A C9       01730         RET                     ;THEN RETURN
              01740 ;
              01750 ;THIS ROUTINE DISPLAYES SIGN ON MESSAGE AND RETURNS.
              01760 ;
4F0B 217A4F   01770 DISMES  LD      HL,MESS1        ;HL=> MESS1 LOCATION
4F0E CD0152   01780         CALL    VIDEO           ;DISPLAY LINE ON VIDEO
4F11 3A0F4E   01790 GETDRN  LD      A,(DR)          ;GET BINARY DRIVE NUMBER
4F14 4F       01800         LD      C,A             ;  INTO 'C'
```

```
4F15 CD934A  01810          CALL   4A93H          ;GET GRAN SEC
4F18 2ED0    01820          LD     L,0D0H         ;LOCATE NAME & DATE
             01830 ;
             01840 ;PUT NAME ON VIDEO
             01850 ;
4F1A 0608    01860          LD     B,8            ;SET UP FOR LOOP
4F1C CD6D4F  01870          CALL   GETG1          ;8 CHARS TO VIDEO (NAME)
             01880 ;
             01890 ;PUT DATE ON VIDEO
             01900 ;
4F1F 0608    01910          LD     B,8            ;SET UP FOR LOOP
4F21 CD6D4F  01920          CALL   GETG1          ;8 CHARS TO VIDEO (DATE)
             01930 ;
             01940 ;SET UP FOR GRANS TO VIDEO
             01950 ;
4F24 2E00    01960          LD     L,00H          ;POSITION TO TOP BUFFER
4F26 E5      01970          PUSH   HL             ;PUT 'HL' ON STACK
4F27 DDE1    01980          POP    IX             ;BACK INTO 'IX'
4F29 210000  01990          LD     HL,0000H       ;ZERO HL
4F2C E5      02000          PUSH   HL             ;  AND ALSO
4F2D D1      02010          POP    DE             ;   'DE'
4F2E 0E28    02020          LD     C,28H          ;SEARCH 40 TRACKS
4F30 DD3460  02030 GETG2    INC    (IX+60H)       ;IS IT A FLAWED TRACK
4F33 2004    02040          JR     NZ,GETG3       ;SKIP IF NOT
4F35 DD3680CF 02050         LD     (IX+80H),3FH   ;MARK THIS GRAN USED
4F39 DD7E00  02060 GETG3    LD     A,(IX+00H)     ;TRACK ALLOC INTO 'A'
4F3C 0606    02070          LD     B,06H          ;6 GRANS PER TRACK
4F3E 1F      02080 GETG4    RRA                   ;CHECK GRAN
4F3F 3F      02090          CCF                   ;FIX STATUS
4F40 ED5A    02100          ADC    HL,DE          ;ADD TO COUNT
4F42 10FA    02110          DJNZ   GETG4          ;LOOP OVER TRACK
4F44 DD23    02120          INC    IX             ;NEXT TRACK
4F46 0D      02130          DEC    C              ;ONE LESS TO DO
4F47 20E7    02140          JR     NZ,GETG2       ;DO FOR ALL 40 TRACKS
             02150 ;
             02160 ;PUT GRANS ON VIDEO
             02170 ;
4F49 CD9A0A  02180          CALL   0A9AH          ;CONVERT HL TO INT ACCUM
4F4C AF      02190          XOR    A              ;NO FORMATTING
4F4D CD3410  02200          CALL   1034H          ;CONVERT
4F50 B6      02210          OR     (HL)           ;  "ACCUM"
4F51 CD09BF  02220          CALL   0FD9H          ;    TO ASCII.
4F54 213141  02230          LD     HL,4131H       ;HL=> CONVERTED NUMBER
4F57 7E      02240 GETG5    LD     A,(HL)         ;GET CHAR
4F58 B7      02250          OR     A              ;TEST FOR END
4F59 2806    02260          JR     Z,ENDMES       ;IF SO DO END OF MESSAGE
4F5B CD0952  02270          CALL   PRTCHR         ;CHAR TO VIDEO
4F5E 23      02280          INC    HL             ;NEXT CHARATER
4F5F 18F6    02290          JR     GETG5          ;DO UNTIL DONE
4F61 21964F  02300 ENDMES   LD     HL,GRNMES      ;HL=> 'Free Granules'
4F64 CD0152  02310          CALL   VIDEO          ;PUT IT ON THE VIDEO
4F67 3E20    02320          LD     A,SPACE        ;ONE SPACE BEFORE FILE
4F69 CD0952  02330          CALL   PRTCHR         ;  NAME OF FIRST FILE
4F6C C9      02340          RET                   ;BACK TO CALLER
4F6D 7E      02350 GETG1    LD     A,(HL)         ;PUT NEXT CHAR INTO 'A'
4F6E CD0952  02360          CALL   PRTCHR         ;PUT ON VIDEO
4F71 23      02370          INC    HL             ;NEXT POSITION
4F72 10F9    02380          DJNZ   GETG1          ;ALL 8 CHARCTERS
4F74 3E20    02390          LD     A,SPACE        ;PUT SPACE ON THE
4F76 CD0952  02400          CALL   PRTCHR         ;  VIDEO
4F79 C9      02410          RET                   ;GO BACK TO CALLER
4F7A 1C      02420 MESS1    DEFB   1CH            ;HOME CURSOR
4F7B 1F      02430          DEFB   1FH            ;CLEAR TO BOT OF SCREEN
4F7C 4D      02440          DEFM   'MENU/CMD'     ;START OF ACTUAL MESSAGE
    45 4E 55 2F 43 4D 44
4F84 20      02450          DEFM   ' by Jim Doffing  '
    62 79 20 4A 69 6D 20 44 6F 66 66 69 6E 67 20 20
4F95 03      02460          DEFB   03H            ;END OF MESSAGE H/O LF
4F96 20      02470 GRNMES   DEFM   ' Free Granules'
    46 72 65 65 20 47 72 61 6E 75 6C 65 73
4FA4 0D      02480          DEFB   0DH            ;END OF MESSAGE H/LF
4FA5 CDCB50  02490 RETDOS   CALL   RSTCUR         ;FIX SCREEN AND ERASE CUR
4FA8 3E91    02500          LD     A,91H          ;RESET RETURN TO DOS
4FAA 329742  02510          LD     (4297H),A      ;OVERLAY NUMBER
4FAD C32D40  02520          JP     DOS            ;AND GOTO DOS READY
4FB0 CDCB50  02530 GOBAS    CALL   RSTCUR         ;GET READY TO PRINT
4FB3 21C64F  02540          LD     HL,GBMES       ;HL=> MESSAGE TERMINATED
4FB6 112542  02550          LD     DE,DOSBUF      ;DE=> DOS COMMAND BUFFER
4FB9 D5      02560          PUSH   DE             ;SAVE FOR LATER
4FBA 010800  02570          LD     BC,0008H       ;SET UP FOR TRANSFER
4FBD EDB0    02580          LDIR                  ;TRANSFER
4FBF E1      02590          POP    HL             ;GET BUFFER ADDRESS
4FC0 CD0152  02600          CALL   VIDEO          ;PUT ON VIDEO
4FC3 C39942  02610          JP     4299H          ;EXECUTE COMMAND & RETDOS
4FC6 42      02620 GBMES    DEFM   'BASIC -F:3'
    41 53 49 43 20 2D 46 3A 33
4FD0 0D      02630          DEFB   0DH            ;TERMINATED BY CR
4FD1 CD9A50  02640 SEL1     CALL   TOP            ;GET TOP OF SCREEN
4FD4 2A2040  02650          LD     HL,(CURPOS)    ;HL=> CURSOR POSITION
4FD7 22174E  02660          LD     (FILE1),HL     ;SAVE FOR LATER
4FDA 22194E  02670 SELECT   LD     (HL2),HL       ;SAVE FOR PRESENT FILE #
4FDD 3E3C    02680          LD     A,'<'          ;PUT FIRST MARKER BEFORE
4FDF CD0952  02690          CALL   PRTCHR         ;  FILE NAME
4FE2 23      02700 SEL2     INC    HL             ;NEXT CHAR POSIT.
4FE3 7E      02710          LD     A,(HL)         ;GET NEXT CHAR INTO 'A'
4FE4 FE20    02720          CP     SPACE          ;IS IT A SPACE
4FE6 2805    02730          JR     Z,SEL3         ;IF SO GOTO SEL3
4FE8 CD0952  02740          CALL   PRTCHR         ;ELSE PUT ON VIDEO
4FEB 18F5    02750          JR     SEL2           ;DO AGAIN
4FED 3E3E    02760 SEL3     LD     A,'>'          ;PUT SECOND MARK AFTER
4FEF CD0952  02770          CALL   PRTCHR         ;  THE FILE NAME
             02780 ;
             02790 ;THE NEXT ROUTINE GETS THE COMMANDS AND ACTS APPROPIATELY
             02800 ;
4FF2 CD4900  02810 GETCOM   CALL   INKEY   ;GET SINGLE KEY FROM KEYBOARD
4FF5 010E00  02820          LD     BC,LENGTH ;LENGTH OF TABLE ARGUMENT
4FF8 211750  02830          LD     HL,COMTBL ;LAST ENTRY IN COMMAND TABLE
4FFB 50      02840          LD     D,B     ;COPY LENGTH FROM BC (BYTE COUNT)
4FFC 59      02850          LD     E,C     ;...INTO DE (TO SAVE FOR LATER)
4FFD EDB9    02860          CPDR           ;SEARCH UP ARGUMENT ENTRIES LIST
4FFF C2F24F  02870          JP     NZ,GETCOM ;NO COMPARE FOUND TRY AGAIN
5002 23      02880          INC    HL      ;COMPENSATE FOR CPDR OVERSHOT
5003 09      02890          ADD    HL,BC   ;ADD REMNANT OF BYTE COUNT
5004 19      02900          ADD    HL,DE   ;ADD ORIGINAL LENGTH
5005 5E      02910          LD     E,(HL)  ;PUT LOW ORDER BYTE INTO 'E'
5006 23      02920          INC    HL      ;NEXT BYTE
5007 56      02930          LD     D,(HL)  ;PUT HIGH ORDER BYTE INTO 'D'
5008 EB      02940          EX     DE,HL   ;PUT IN HL
5009 E9      02950          JP     (HL)    ;GOTO ADDRESS OF ROUTINE
000E         02960 LENGTH   EQU    000EH   ;FOURTEEN (14) COMMANDS IN TBL
500A 30      02970 TBL      DEFB   '0'     ;FIRST COMMAND
500B 31      02980          DEFB   '1'
500C 32      02990          DEFB   '2'
500D 33      03000          DEFB   '3'
500E 44      03010          DEFB   'D'
500F 4B      03020          DEFB   'K'
5010 43      03030          DEFB   'C'
5011 42      03040          DEFB   'B'
5012 01      03050          DEFB   01H
5013 0D      03060          DEFB   0DH
5014 5B      03070          DEFB   5BH     ;UP ARROW
5015 0A      03080          DEFB   0AH     ;DOWN ARROW
5016 08      03090          DEFB   08H     ;LEFT ARROW
5017 09      03100 COMTBL   DEFB   09H     ;RIGHT ARROW ALSO LAST COM ENTRY
5018 5050    03110          DEFW   DRIVE
501A 5050    03120          DEFW   DRIVE
501C 5050    03130          DEFW   DRIVE
501E 5050    03140          DEFW   DRIVE
5020 5350    03150          DEFW   DEBUG
5022 D050    03160          DEFW   KILL
5024 7252    03170          DEFW   COPY
5026 B04F    03180          DEFW   GOBAS
5028 A54F    03190          DEFW   RETDOS
502A 5750    03200          DEFW   RUN
502C 1852    03210          DEFW   UPAR
502E DD51    03220          DEFW   DNAR
5030 2F52    03230          DEFW   LTAR
5032 F451    03240          DEFW   RTAR
             03250                         ;PROGRAM
             03260 ;
             03270 ;THIS ROUTINE ERASES CURSOR1 AND CURSOR2 AND RETURNS
             03280 ;TO THE CALLING ROUTINE
             03290 ;
5034 3E08    03300 ERASE    LD     A,BKSPER       ;BACKSPACE AND ERASE
5036 CD0952  03310          CALL   PRTCHR         ;  CURSOR2
5039 2A2040  03320          LD     HL,(CURPOS)    ;HL=> CURSOR POSITION
503C 2B      03330 ERA1     DEC    HL             ;GET PREVIOUS CHAR
503D 7E      03340          LD     A,(HL)         ;PUT CHAR INTO 'A'
503E FE3C    03350          CP     '<'            ;IS IT CURSOR CHAR
5040 2807    03360          JR     Z,ERA2         ;IF SO CONTINUE
5042 3E18    03370          LD     A,BKSP         ;IF NOT BACK SPACE
5044 CD0952  03380          CALL   PRTCHR         ;  TO PREV CHAR
5047 18F3    03390          JR     ERA1           ;DO UNTIL DONE
5049 3E08    03400 ERA2     LD     A,BKSPER       ;BACK SPACE AND ERASE
504B CD0952  03410          CALL   PRTCHR         ;  CURSOR1
```

```
504E 2B       03420          DEC   HL              ;POINT TO CURSOR POSIT
504F C9       03430          RET                   ;RETURN TO CALLER
5050 C33D4E   03440 DRIVE    JP    START1          ;GET DRIVE DIRECTORY
5053 AF       03450 DEBUG    XOR   A               ;NO STATUS
5054 C30D44   03460          JP    4400H           ;LOAD AND JUMP TO DEBUG
5057 2A194E   03470 RUN      LD    HL,(HL2)        ;GET PRESENT FILE LOCAT.
505A 23       03480          INC   HL              ;NEXT POSITION AFTER '<'
505B 010A00   03490          LD    BC,10D          ;SET UP FOR SEARCH LOOP
505E 3E2F     03500          LD    A,'/'           ;SEARCH CHAR INTO 'A'
5060 EDB1     03510          CPIR                  ;AUTOMATIC SEARCH
5062 C2A051   03520          JP    NZ,BASIC        ;IF NOT MUST BE BASIC
5065 110500   03530 RUN11    LD    DE,0005H        ;BYTES BETWEEN COMMANDS
5068 DD21B050 03540          LD    IX,EXTTBL       ;IX=> EXTENT TABLE
506C E5       03550 RUN2     PUSH  HL              ;SAVE 'HL' FOR LATER
506D DD7E00   03560          LD    A,(IX+00H)      ;PUT FIRST CHAR INTO 'A'
5070 FE00     03570          CP    00H             ;IS IT END OF TABLE
5072 CA9650   03580          JP    Z,BASICR        ;IF ZERO TABLE END, BASIC
5075 7E       03590          LD    A,(HL)          ;1ST FILE CHAR INTO 'A'
5076 DD8E00   03600          CP    (IX+00H)        ;IS IT SAME AS TABLE
5079 2016     03610          JR    NZ,NXTEXT       ;IF NOT, NEXT EXT. LOC.
507B 23       03620          INC   HL              ;POINT TO NEXT CHAR
507C 7E       03630          LD    A,(HL)          ;NEXT CHAR INTO 'A'
507D DD8E01   03640          CP    (IX+01H)        ;IS IT SAME AS 2ND CHAR
5080 200F     03650          JR    NZ,NXTEXT       ;IF NOT, NEXT EXT. LOC.
5082 23       03660          INC   HL              ;POINT TO THIRD CHAR
5083 7E       03670          LD    A,(HL)          ;PUT IT INTO 'A'
5084 DD8E02   03680          CP    (IX+02H)        ;IS IT SAME AS 3RD CHAR
5087 2008     03690          JR    NZ,NXTEXT       ;IF NOT, NEXT EXT LOC
5089 E1       03700          POP   HL              ;FIX STACK
508A DD6E03   03710          LD    L,(IX+03H)      ;PUT ROUTINE ADDRESS
508D DD6604   03720          LD    H,(IX+04H)      ;  INTO 'HL'
5090 E9       03730          JP    (HL)            ; GO TO IT
5091 DD19     03740 NXTEXT   ADD   IX,DE           ;INC TO NEXT EXT
5093 E1       03750          POP   HL              ;GET ORIGINAL FILE LOC
5094 18D6     03760          JR    RUN2            ;CHECK ANOTHER FILE EXT.
5096 E1       03770 BASICR   POP   HL              ;CORRECT STACK
5097 C3A051   03780          JP    BASIC
509A 3A2040   03790 TOP      LD    A,(CURPOS)      ;GET LSB SCREEN ADDR
509D E63F     03800          AND   3FH             ;MASK OFF BITS 6&7
509F FE00     03810          CP    0               ;IS IT POSISION 0 OF LINE
50A1 2807     03820          JR    Z,TO1           ;IF SO CONTINUE
50A3 3E18     03830          LD    A,BKSP          ;ELSE BACKSPACE
50A5 CD0952   03840          CALL  PRTCHR          ;  ONE SPACE
50A8 18F0     03850          JR    TOP             ;AND CHECK AGAIN
50AA 2A2040   03860 TO1      LD    HL,(CURPOS)     ;PUT ADDRESS INTO 'HL'
50AD 114000   03870          LD    DE,0040H        ;SET UP 'DE' FOR
50B0 ED52     03880          SBC   HL,DE           ;  SUBTRACTION
50B2 7E       03890          LD    A,(HL)          ;PUT CHAR INTO 'A'
50B3 FE20     03900          CP    SPACE           ;IS IT A SPACE
50B5 C0       03910          RET   NZ              ;IF NOT RETURN TO CALLER
50B6 3E1B     03920          LD    A,UPLN          ;ELSE PRINT A
50B8 CD0952   03930          CALL  PRTCHR          ;UP LINE FEED
50BB 18ED     03940          JR    TO1             ;  AND CONTINUE
50BD 43       03950 EXTTBL   DEFM  'CMD'           ;COMMAND FILE EXT
    4D 44
50C0 BE51     03960          DEFW  COMAND          ;COMMAND FILE ROUTINE
50C2 42       03970          DEFM  'BLD'           ;BUILD FILE EXT
    4C 44
50C5 7351     03980          DEFW  DO              ;DO BUILD ROUTINE
50C7 00       03990          DEFB  00H             ;END OF EXTTBL MARKER
50C8 CD3450   04000 RSTCUR   CALL  ERASE           ;GET RID OF CURSORS
50CB CD9A50   04010          CALL  TOP             ;GO TO TOP OF VIDEO
50CE 2A2040   04020 RST1     LD    HL,(CURPOS)     ;GET CURRENT POSITION
50D1 23       04030          INC   HL              ;POINT AT NEXT CHAR
50D2 7E       04040          LD    A,(HL)          ;PUT INTO 'A'
50D3 FE20     04050          CP    SPACE           ;IS IT A SPACE
50D5 C8       04060          RET   Z               ;IF SO RETURN TO CALLER
50D6 3E1A     04070          LD    A,DNLN          ;ELSE MOVE DOWN ONE
50D8 CD0952   04080          CALL  PRTCHR          ;  LINE ON VIDEO
50DB 18F1     04090          JR    RST1            ;    AND TRY AGAIN
50DD CDC850   04100 KILL     CALL  RSTCUR          ;GET CURSOR TO CLEAR LINE
50E0 115551   04110          LD    DE,KMES2        ;DE=> KMES2 STOR AREA
50E3 3E20     04120          LD    A,SPACE         ;'A'= CHAR AT END OF FILE
50E5 CD6352   04130          CALL  MOVFIL          ;MOVE FILE TO =>'DE'
50E8 3E03     04140 KIL2     LD    A,03H           ;EOF MARKER INTO 'A'
50EA 12       04150          LD    (DE),A          ;PUT INTO MESSAGE
50EB 213751   04160          LD    HL,KILMES       ;HL=> KILL MESSAGE?
50EE CD0152   04170          CALL  VIDEO           ;PUT LINE ON VIDEO
50F1 216551   04180          LD    HL,KMES3        ;HL=> SECOND HALF KILMES
50F4 CD0152   04190          CALL  VIDEO           ;PUT LINE ON VIDEO
50F7 CD4900   04200 KIL3     CALL  INKEY           ;GET KEY PRESSED
50FA FE01     04210          CP    01H             ;WAS BREAK KEY PRESSED
50FC 2804     04220          JR    Z,QTBK          ;IF SO CORRECT SCREEN RTN
50FE FE4E     04230          CP    'N'             ;IS IT AN "N"
5100 200F     04240          JR    NZ,KIL4         ;IF NOT MAYBE A "Y"
5102 216E51   04250 QTBK     LD    HL,ERAKIL       ;HL=> ERASE KILL MESSAGE
5105 CD0152   04260          CALL  VIDEO           ;PUT IT ON THE VIDEO
5108 CD9A50   04270          CALL  TOP             ;GO TO TOP SCREEN
510B 2A2040   04280          LD    HL,(CURPOS)     ;HL=> CURSOR POSITION
510E C3DA4F   04290          JP    SELECT          ;START OVER WITH SELECT
5111 FE59     04300 KIL4     CP    'Y'             ;IS IT A "Y"
5113 20E2     04310          JR    NZ,KIL3         ;IF NOT GET ANOTHER COMM
5115 CD0952   04320          CALL  PRTCHR          ;PUT IT ON VIDEO
5118 3E0D     04330          LD    A,CR            ;MOVE TO NEXT LINE
511A CD0952   04340          CALL  PRTCHR          ;DO IT NOW
511D 215051   04350          LD    HL,KMES1        ;HL=> KMES1 LOCATION
5120 112542   04360          LD    DE,DOSBUF       ;DE=> 4225H COMMAND BUFF
5123 D5       04370          PUSH  DE              ;SAVE FOR LATER
5124 2B       04380          DEC   HL              ;BACKUP ONE SPACE
5125 3E03     04390          LD    A,03H           ;END OF FILE INTO 'A'
5127 326C52   04400          LD    (MOV2+01H),A    ;PUT INTO RTN FOR COMPAR
512A CD6952   04410          CALL  MOV1            ;MOVE IT
512D 3E0D     04420 KIL6     LD    A,CR            ;PUT CAR RET INTO 'A'
512F 12       04430          LD    (DE),A          ;STORE IT
5130 E1       04440          POP   HL              ;NOW WE CAN USE IT
5131 CD0152   04450          CALL  VIDEO           ;PUT ON SCREEN
5134 C3F0FF   04460          JP    DOSCMD          ;USE ROUTINE IN DOS
5137 41       04470 KILMES   DEFM  'Are you sure you want to '
    72 65 20 79 6F 75 20 73 75 72 65 20 79 6F 75 20
    77 61 6E 74 20 74 6F 20
5150 4B       04480 KMES1    DEFM  'KILL '
    49 4C 4C 20
0010          04490 KMES2    DEFS  16D             ;RESERVE 16 SPACES
5165 3F       04500 KMES3    DEFM  '? (Y/N)'
    20 28 59 2F 4E 29
516C 0E       04510          DEFB  0EH             ;TURN ON CURSOR
516D 03       04520          DEFB  03H             ;EOF MARKER
516E 0F       04530 ERAKIL   DEFB  0FH             ;TURN OFF CURSOR
516F 0A       04540          DEFB  0AH             ;ISSUE LINE FEED & CR
5170 1B       04550          DEFB  UPLN            ;GO UP ONE LINE
5171 1F       04560          DEFB  CLS             ;CLEAR TO BOTT OF SCREEN
5172 03       04570          DEFB  03H             ;TERMINATE WITH EOF MARK
5173 CDC850   04580 DO       CALL  RSTCUR          ;GOTO FIRST BLANK LINE
5176 219651   04590          LD    HL,DOMES        ;HL=> BUILD MESSAGE
5179 112542   04600          LD    DE,DOSBUF       ;DE=> DOS INPUT BUFFER
517C D5       04610          PUSH  DE              ;SAVE BUF ADDR FOR LATE
517D 010300   04620          LD    BC,0003H        ;NUMBER OF CHARS TO TRANS
5180 EDB0     04630          LDIR                  ;MOVE'EM
5182 3E2F     04640          LD    A,'/'           ;'A'= CHAR AT END OF FILE
5184 CD6352   04650          CALL  MOVFIL          ;MOVE NAME TO BUFFER
5187 3E0D     04660 DO2      LD    A,CR            ;TERMINATE COMMAND
5189 12       04670          LD    (DE),A          ;IN COMMAND BUFFER
518A E1       04680          POP   HL              ;HL=> DOS BUFFER
518B CD0152   04690          CALL  VIDEO           ;PUT COMMAND ON VIDEO
518E 3E91     04700          LD    A,91H           ;RE DO OVERLAY01 INFO
5190 329742   04710          LD    (4297H),A       ;STORE IT
5193 C39942   04720          JP    4299H           ;EXECUTE COMMAND
5196 44       04730 DOMES    DEFM  'DO '           ;DO COMMAND MESSAGE
    4F 20
5199 03       04740          DEFB  03H             ;EOF MARKER
519A 42       04750 BASMES   DEFM  'BASIC,'
    41 53 49 43 2C
51A0 CDC850   04760 BASIC    CALL  RSTCUR          ;FIX VIDEO
51A3 219A51   04770          LD    HL,BASMES       ;HL=> BASIC RTN MESSAGE
51A6 112542   04780          LD    DE,DOSBUF       ;DE=> DOS INPUT BUFFER
51A9 010600   04790          LD    BC,0006H        ;BC = LENGTH OF MESSAGE
51AC D5       04800          PUSH  DE              ;SAVE 'DE' FOR LATER
51AD EDB0     04810          LDIR                  ;MOVE MES INTO DOSBUF
51AF 3E20     04820          LD    A,SPACE         ;'A'= END OF FILE MARK
51B1 CD6352   04830          CALL  MOVFIL          ;MOVE IT TO =>'DE'
51B4 3E0D     04840 BAS2     LD    A,CR            ;PUT A CARRIAGE RETURN
51B6 12       04850          LD    (DE),A          ;AT END OF COMMAND
51B7 E1       04860          POP   HL              ;NOW WE NEED THAT INFO
51B8 CD0152   04870          CALL  VIDEO           ;PUT IT ON THE SCREEN
51BB C39942   04880          JP    4299H           ;TURN IT OVER TO DOS
51BE CDC850   04890 COMAND   CALL  RSTCUR          ;FIX SCREEN
51C1 112542   04900          LD    DE,DOSBUF       ;DE=> DOS COMMAND BUFFER
51C4 D5       04910          PUSH  DE              ;SAVE FOR LATER
51C5 3E2F     04920          LD    A,'/'           ;'A'=END OF FILE MARK
51C7 CD6352   04930          CALL  MOVFIL          ;PUT IT INTO BUF=>'D'
51CA 3E3A     04940 COM2     LD    A,':'           ;PUT COLON INTO 'A'
51CC 12       04950          LD    (DE),A          ;PUT INTO BUFFER
51CD 13       04960          INC   DE              ;POINT TO NEXT POSITION
51CE 3AFFFF   04970          LD    A,(DRNO)        ;PUT ASCII DR# INTO'A'
51D1 12       04980          LD    (DE),A          ;PUT INTO BUFFER
51D2 13       04990          INC   DE              ;NEXT POSITION
```

```
51D3 3E0D    05000         LD    A,CR          ;TERMINATE COMMAND IN THE
51D5 12      05010         LD    (DE),A        ;COMMAND BUFFER
51D6 E1      05020         POP   HL            ;HL=> COMMAND BUFFER
51D7 CD0152  05030         CALL  VIDEO         ;PUT LINE ON VIDEO
51DA C39942  05040         JP    4299H         ;TURN IT OVER TO DOS
51DD 114000  05050 DNAR    LD    DE,0040H      ;DE=> NUM CHAR TO NEXT
51E0 CD0F52  05060         CALL  DNRTCK        ;CHECK FOR FILE THERE
51E3 CAF24F  05070         JP    Z,GETCOM      ;IF SO THEN NO MORE FILES
51E6 CD3450  05080         CALL  ERASE         ;ELSE GET RID OF CURSORS
51E9 3E1A    05090         LD    A,DNLN        ;DUMMY VALUE
51EB CD0952  05100         CALL  PRTCHR        ;PUT IT ON THE SCREEN
51EE 2A2040  05110         LD    HL,(CURPOS)   ;HL=> CURSOR POSITION
51F1 C3DA4F  05120         JP    SELECT        ;GOTO SELECT ROUTINE
51F4 111000  05130 RTAR    LD    DE,0010H      ;DE = NUM CHARS TO NEXT
51F7 CD0F52  05140         CALL  DNRTCK        ;CHECK FOR FILE THERE
51FA CAF24F  05150         JP    Z,GETCOM      ;IF SO THEN NO MORE FILES
51FD CD3450  05160         CALL  ERASE         ;ELSE GET RID OF CURSORS
5200 0610    05170         LD    B,16D         ;MOVE TO NEXT FILE
5202 23      05180 RTAR1   INC   HL            ;POINT TO NEXT CHAR
5203 7E      05190         LD    A,(HL)        ;PUT NEXT CHAR INTO 'A'
5204 CD0952  05200         CALL  PRTCHR        ;PUT IT ON VIDEO
5207 10F9    05210         DJNZ  RTAR1         ;DO UNTIL DONE
5209 2A2040  05220         LD    HL,(CURPOS)   ;HL=> CURSOR LOCATION
520C C3DA4F  05230         JP    SELECT        ;TURN OVER TO SELECT
520F 2A194E  05240 DNRTCK  LD    HL,(HL2)      ;HL=> CURR FILE NAME
5212 19      05250         ADD   HL,DE         ;ADD, RESULTS IN 'HL'
5213 23      05260         INC   HL            ;MOVE TO NEXT CHAR
5214 7E      05270         LD    A,(HL)        ;PUT CHAR INTO 'A'
5215 FE20    05280         CP    SPACE         ;IS IT A SPACE
5217 C9      05290         RET                 ;RETUR NO MATTER
5218 114000  05300 UPAR    LD    DE,0040H      ;DE = CHARS TO NEXT FILE
521B CD5852  05310         CALL  UPLTCK        ;IS TOO FAR UP OR LEFT
521E DAF24F  05320         JP    C,GETCOM      ;IF SO GET NEXT COMMAND
5221 CD3450  05330         CALL  ERASE         ;GET RID OF CURSORS
5224 3E1B    05340         LD    A,UPLN        ;ELSE LOAD UP LINE IN 'A'
5226 CD0952  05350         CALL  PRTCHR        ;PUT TO VIDEO
5229 2A2040  05360         LD    HL,(CURPOS)   ;HL=> NEW FILE NAME
522C C3DA4F  05370         JP    SELECT        ;GET ANOTHER CHOICE
522F 111000  05380 LTAR    LD    DE,0010H      ;DE = CHARS TO NEXT FILE
5232 CD5852  05390         CALL  UPLTCK        ;IS TOO FAR UP OR LEFT
5235 DAF24F  05400         JP    C,GETCOM      ;IF SO GET NEXT COMMAND
5238 CD3450  05410         CALL  ERASE         ;GET RID OF CURSORS
523B 0610    05420         LD    B,10H         ;MOVE TO NEXT FILE
523D 3E18    05430 LTAR1   LD    A,BKSP        ;BACK SPACE W/O ERASE 'A'
523F CD0952  05440         CALL  PRTCHR        ;PUT ON SCREEN
5242 3A2040  05450         LD    A,(CURPOS)    ;GET LSB OF CURSOR POSIT
5245 E63F    05460         AND   3FH           ;MASK BITS 6 & 7
5247 FE3F    05470         CP    3FH           ;IS IT END OF LINE
5249 2005    05480         JR    NZ,LTAR2      ;IF NOT FIRST CHAR CONT
524B 3E1B    05490         LD    A,UPLN        ;MOVE UP ONE LINE
524D CD0952  05500         CALL  PRTCHR        ;PUT IT ON VIDEO
5250 10EB    05510 LTAR2   DJNZ  LTAR1         ;DO UNTIL DONE
5252 2A2040  05520         LD    HL,(CURPOS)   ;GET CURRENT FILE
5255 C3DA4F  05530         JP    SELECT        ;GET NEXT SELECTION
5258 2A194E  05540 UPLTCK  LD    HL,(HL2)      ;HL=> FILE NAME
525B ED52    05550         SBC   HL,DE         ;SUBTRACT TO SEE IF WE
525D 01403C  05560         LD    BC,3C40H      ;  ARE TRYING TO SELECT
5260 ED42    05570         SBC   HL,BC         ;  THE FIRST LINE
5262 C9      05580         RET                 ;GO BACK EITHER WAY
5263 326C52  05590 MOVFIL  LD    (MOV2+01H),A  ;PUT COMP VALUE IN RTN
5266 2A194E  05600         LD    HL,(HL2)      ;HL=> FILE NAME ADDRESS
5269 23      05610 MOV1    INC   HL            ;NEXT CHAR
526A 7E      05620         LD    A,(HL)        ;PUT CHAR INTO 'A'
526B FE00    05630 MOV2    CP    00H           ;DUMMY CHECK SET BY RTN
526D C8      05640         RET   Z             ;BACK TO CALLER IF END
526E 12      05650         LD    (DE),A        ;ELSE STORE AT =>'DE'
526F 13      05660         INC   DE            ;NEXT POSITION
5270 18F7    05670         JR    MOV1          ;CONTINUE UNTIL DONE
5272 CDC850  05680 COPY    CALL  RSTCUR        ;MOVE CURSOR TO EMTY LINE
5275 21B752  05690         LD    HL,COPMES     ;HL=> Which Drive?
5278 CD0152  05700         CALL  VIDEO         ;PUT ON SCREEN
527B CD4900  05710 COP1    CALL  INKEY         ;GET CHARACTER
527E FE01    05720         CP    01H           ;WAS BREAK PRESSED
5280 CA0251  05730         JP    Z,QTBK        ;IF SO CORRECT SCREEN RTN
5283 32CF52  05740         LD    (COPNUM),A    ;STORE FOR LATER
5286 D630    05750         SUB   30H           ;MAKE IT BINARY
5288 38F1    05760         JR    C,COP1        ;IF LESS THAN 0 TRY AGAIN
528A 4F      05770         LD    C,A           ;PUT IN 'C' FOR COMPARE
528B 3A1344  05780         LD    A,(4413H)     ;GET HIGH SYS DR #
528E B9      05790         CP    C             ;IS IT TOO BIG
528F 38EA    05800         JR    C,COP1        ;IF SO TRY AGAIN
5291 21B752  05810         LD    HL,COPMES     ;HL=> "COPY "
5294 112542  05820         LD    DE,DOSBUF     ;DE=> DOS INPUT BUFFER
5297 D5      05830         PUSH  DE            ;SAVE BUFFER ADDRESS
5298 010500  05840         LD    BC,0005H      ;LENGTH OF COPY MESSAGE
529B ED80    05850         LDIR                ;MOVE IT
529D 3E20    05860         LD    A,SPACE       ;'A'= CHAR AT END OF FILE
529F CD6352  05870         CALL  MOVFIL        ;MOVE IT TO =>'DE'
52A2 21CD52  05880 COP3    LD    HL,COPSPC     ;HL=> COPY SPACE MESSAGE
52A5 010400  05890         LD    BC,0004H      ;MOVE THREE CHARS TO
52A8 ED80    05900         LDIR                ;DOSBUF
52AA 21CF52  05910         LD    HL,COPNUM     ;HL=> DRIVE NUMBER AND CR
52AD CD0152  05920         CALL  VIDEO
52B0 E1      05930         POP   HL            ;RESTORE BUFFER ADDRESS
52B1 CD0152  05940         CALL  VIDEO         ;PUT COMMAND ON SCREEN
52B4 C3F0FF  05950         JP    DOSCMD        ;TURN IT OVER TO DOS
52B7 43      05960 COPMES  DEFM  'COPY to which Drive?'
     4F 50 59 20 74 6F 20 77 68 69 63 68 20 44 72 69
     76 65 3F
52C8 0E      05970         DEFB  0EH           ;TURN ON CURSOR
52CC 03      05980         DEFB  03H
52CD 20      05990 COPSPC  DEFB  SPACE
52CE 3A      06000         DEFB  ':'
52CF 00      06010 COPNUM  DEFB  00H
52D0 0D      06020         DEFB  0DH
52D1 C5      06030 VIDEO   PUSH  BC            ;SAVE POINTER
52D2 E5      06040         PUSH  HL            ;SAVE 'HL'
52D3 CD1802  06050         CALL  PRTLN         ;PUT IT ON THE VIDEO
52D6 E1      06060         POP   HL            ;GET 'HL'
52D7 C1      06070         POP   BC            ;PUT BACK
52D8 C9      06080         RET                 ;BACK TO CALLER
52D9 C5      06090 PRTCHR  PUSH  BC            ;SAVE POINTER
52DA CD3300  06100         CALL  VDCHR         ;PUT IT ON THE VIDEO
52DD C1      06110         POP   BC            ;PUT IT BACK
52DE C9      06120         RET
FFF0         06130         ORG   0FFF0H        ;RST ROUTINE LOCATION
FFF0 E5      06140 DOSCMD  PUSH  HL            ;SAVE 'HL' FOR NOW
FFF1 21FBFF  06150         LD    HL,TEMTOP     ;PROTECT RETURN FROM DOS
FFF4 221144  06160         LD    (TOPMEM),HL   ;  STORE IT
FFF7 E1      06170         POP   HL            ;GET 'HL' BACK
FFF8 CD9C42  06180         CALL  COMDOS        ;EXECUTE COMMAND & RETURN
FFFB 3EFF    06190 TEMTOP  LD    A,0FFH        ;LOAD OVERLAY 15 &
FFFD EF      06200         RST   28H           ;AND EXECUTE IT
4E00         06210         END   ST            ;END STATEMENT
00000 TOTAL ERRORS

BAS2  51B4   BASIC 51A0   BASICR 5096   BASMES 519A   BKSP  0018
BKSPER 0008  CLS   001F   COM2   51CA   COMAND 51BE   COMDOS 429C
COMTBL 5017  COP1  527B   COP3   52A2   COPMES 52B7   COPNUM 52CF
COPSPC 52CD  COPY  5272   CR     000D   CURPOS 4020   DEBUG  5053
DISD0  4E61  DISD1 4E71   DISDIR 4E57   DISF0  4EAB   DISFIL 4EA2
DISMES 4F0B  DNAR  51DD   DNLN   001A   DNRTCK 520F   DO     5173
DO2    5187  DOMES 5196   DOS    402D   DOSBUF 4225   DOSCMD FFF0
DR     4E0F  DRIVE 5050   DRNO   FFFF   DRNUM  4ED3   DRNUM1 4EF5
DRNUM2 4EF9  ENDMES 4F61  ERA1   503C   ERA2   5049   ERAKIL 516E
ERASE  5034  EXT   4EB7   EXT1   4EC7   EXTTBL 508D   FILE1  4E17
FILNAM 4E13  FILOFF 4E11  GBMES  4FC6   GETCOM 4FF2   GETG1  4F6D
GETG2  4F30  GETG3 4F39   GETG4  4F3E   GETG5  4F57   GETGRN 4F11
GOBAS  4FB0  GRNMES 4F96  HL2    4E19   HLSAV  4E15   INKEY  0049
KIL2   50E8  KIL3  50F7   KIL4   5111   KIL6   512D   KILL   5000
KILMES 5137  KMES1 5150   KMES2  5155   KMES3  5165   LAST   4E1B
LENGTH 000E  LOOP  4F03   LOOP1  4F07   LTAR   522F   LTAR1  523D
LTAR2  5250  MESS1 4F7A   MOV1   5269   MOV2   526B   MOVFIL 5263
NXTEXT 5091  NXTFIL 4E7D  NXTSEC 4E8C   PRTCHR 52D9   PRTLN  021B
PUTHIN 4E50  QTBK  5102   READ   4B0D   RETDOS 4FA5   RST1   50CE
RSTCUR 50C8  RTAR  51F4   RTAR1  5202   RUN    5057   RUN11  5065
RUN2   506C  SEC   4E0D   SECBUF 4D00   SECEND 004E   SEL1   4FD1
SEL2   4FE2  SEL3  4FED   SELECT 4FDA   SPACE  0020   ST     4E00
ST1    4E04  START 4E1D   START1 4E3D   STRT1  4E32   TABLE  4E0D
TBL    500A  TEMSEC 4E10  TEMTOP FFFB   TO1    50AA   TOP    509A
TOPMEM 4411  TRK   4E0E   UPAR   5218   UPLN   001B   UPLTCK 5258
VDCHR  0033  VIDEO 52D1
```

[Source code for MENINSTL/CMD:]

```
              00100 ;MENINSTL/CMD  a program to install the MENU program
              00110 ;TRSDOS CALLS
4675          00120 READ    EQU   4675H
4600          00130 WRITE   EQU   4600H
021B          00140 VIDEO   EQU   021BH
0049          00150 INKEY   EQU   0049H
402D          00160 DOS     EQU   402DH
5400          00170 BUFFER  EQU   5400H
              00180 ;
```

```
5200            00190         ORG     5200H
                00200 ;  THE BELOW IS AN EXAMPLE OF HOW TO READ AND WRITE
                00210 ;SPECIFIC TRACKS AND SECTORS USING TRSDOS 1.3
                00220 ;DISK   LD      HL,5300H        ;BUFFER LOCATION
                00230 ;DRIVE  LD      BC,0000H        ;00=256BYTE, 00=DRIVE0
                00240 ;TKSC   LD      DE,1004H        ;10=TRACK, 04=SECTOR
                00250 ;OPER   CALL    READ            ;DO READ OR WRITE
                00260 ;       RET                     ;BACK TO CALLER
1102            00270 FIRST   EQU     1102H           ;FIRST RECORD TK11 SEC2
1103            00280 SECOND  EQU     1103H           ;SECOND RECORD TK11 SEC3
1005            00290 THIRD   EQU     1005H           ;THIRD RECORD TK10 SEC5
5200 49         00300 PROMPT  DEFM    'Insert Disk to be upgraded into Drive #1,
     6E 73 65 72 74 20 44 69 73 6B 20 74 6F 20 62 65  then press any key.'
     20 75 70 67 72 61 64 65 64 20 69 6E 74 6F 20 44
     72 69 76 65 20 23 31 2C 20 74 68 65 6E 20 70 72
     65 73 73 20 61 6E 79 20 6B 65 79 2E
5230 0A         00310         DEFB    0AH
5231 0D         00320         DEFB    0DH
523F 01         00330 NEWCOM  DEFB    01H     ;MENU to be included in LIB
5240 0C         00340         DEFB    0CH
5241 F0         00350         DEFB    0F0H
5242 51         00360         DEFB    51H
5243 42         00370         DEFB    42H
5244 4D         00380         DEFM    'MENU  '
     45 4E 55 20 20
524A 93         00390         DEFB    93H
524B 42         00400         DEFB    42H
524C 00         00410         DEFB    00H
524D 0202       00420         DEFW    0202H
524F 00         00430         DEFB    00H
5250 4E         00440         DEFB    4EH
5251 E5         00450         DEFB    0E5H
5252 210052     00460 START   LD      HL,PROMPT
5255 CD1B02     00470         CALL    VIDEO           ;PRINT START MESSAGE
5258 CD4900     00480         CALL    INKEY           ;GET KEYPRESS
525B 210054     00490         LD      HL,BUFFER       ;FIRST SECTOR
525E 110211     00500         LD      DE,FIRST        ;TO BE READ
5261 010100     00510         LD      BC,0001H        ;256BYTE RECORDS DRIVE1
5264 C5         00520         PUSH    BC
5265 CD7546     00530         CALL    READ
5268 210054     00540         LD      HL,BUFFER
526B 110211     00550         LD      DE,FIRST
526E C1         00560         POP     BC
526F C5         00570         PUSH    BC
5270 CD7546     00580         CALL    READ            ;DO IT TWICE FOR SURE
5273 3E00       00590         LD      A,00H
5275 210454     00600         LD      HL,5404H        ;CLEAN UP "HIT"
5278 77         00610         LD      (HL),A
5279 3E82       00620         LD      A,82H
527B 21FE54     00630         LD      HL,54FEH        ;INSTAL MENU INTO SYSTBL
527E 77         00640         LD      (HL),A
527F 23         00650         INC     HL
5280 3E15       00660         LD      A,15H
5282 77         00670         LD      (HL),A
5283 210054     00680         LD      HL,BUFFER
5286 110211     00690         LD      DE,FIRST
5289 C1         00700         POP     BC
528A C5         00710         PUSH    BC
528B CD0046     00720         CALL    WRITE           ;PUT IT BACK AFTER CHANGE
528E 210054     00730         LD      HL,BUFFER       ;SECOND SECTOR
5291 110311     00740         LD      DE,SECOND       ;TO BE READ
5294 C1         00750         POP     BC
5295 C5         00760         PUSH    BC
5296 CD7546     00770         CALL    READ
5299 21C054     00780         LD      HL,54C0H        ;START OF FILE INFO
529C 11C154     00790         LD      DE,54C1H
529F 01FF00     00800         LD      BC,FFH
52A2 77         00810         LD      (HL),A
52A3 EDB0       00820         LDIR                    ;CLEAR OUT INFO FROM DIR
52A5 210054     00830         LD      HL,BUFFER
52A8 110311     00840         LD      DE,SECOND
52AB C1         00850         POP     BC
52AC C5         00860         PUSH    BC
52AD CD0046     00870         CALL    WRITE           ;PUT IT BACK CORRECTED
52B0 210054     00880         LD      HL,BUFFER       ;THIRD SECTOR
52B3 110510     00890         LD      DE,THIRD        ;TO BE READ
52B6 C1         00900         POP     BC
52B7 C5         00910         PUSH    BC
52B8 CD7546     00920         CALL    READ
52BB 210054     00930         LD      HL,BUFFER
52BE 110510     00940         LD      DE,THIRD
52C1 C1         00950         POP     BC
52C2 C5         00960         PUSH    BC
52C3 CD7546     00970         CALL    READ            ;TWICE TO BE SURE
52C6 213F52     00980         LD      HL,NEWCOM
52C9 110054     00990         LD      DE,BUFFER
52CC 011200     01000         LD      BC,12H
52CF EDB0       01010         LDIR
52D1 210054     01020         LD      HL,BUFFER
52D4 110510     01030         LD      DE,THIRD
52D7 C1         01040         POP     BC
52D8 CD0046     01050         CALL    WRITE
52DB C32D40     01060         JP      402DH           ;RETURN TO DOS READY
5252            01070         END     START
00000 TOTAL ERRORS

BUFFER  5400    DOS     402D    FIRST   1102    INKEY   0049    NEWCOM  523F
PROMPT  5200    READ    4675    SECOND  1103    START   5252    THIRD   1005
VIDEO   021B    WRITE   4600
```

Making an Auto-Boot-Loading Newdos80 V2.0 System Disk for the Model 4P

by Tony Domigan

MODELA/III can be copied to LDOS 5.x and TRSDOS 1.3B diskettes so that they can Cold Start on the Model 4P. If, however, you wish to use NEWDOS/80 version 2.0 the only option is to load MODELA/III with 'pause' and then enter your NEWDOS disk - two operations; not very convenient. Furthermore, the Model 4P hardware seems to check the boot sector (sector 1) in a way which excludes any DOS other than TRSDOS or LDOS.

The method I have used relocates the NEWDOS directory to to lump 36 i.e. track 20; the same track on which the LDOS directory resides.

The FPDE for MODELA/III is created and the directory entry modified so that when the file is copied from the TRSDOS MODELA/III disk it will be copied to a true track. I have used track 30 as it is not allocated in a standard NEWDOS diskette.

The ROM file is then modified on the NEWDOS diskette so that the bootstrap loader will select sector 0 to boot the diskette.

Using Superzap the MODELA/III FPDE is again modified to read Track 1EH or 30 decimal and the total granules changed to 09H. The normal boot sector (sector 1) is modified with 3 bytes located by the hardware in checking for a TRSDOS or LDOS disk.

The diskette is now configured such that the hardware assumes it is an LDOS diskette with the ROM starting on track 30 decimal. Once loaded the ROM bootstrap boots NEWDOS from sector 0.

=====O=====

1. Construct a new system diskette with a directory on lump 36 - e.g.

```
PDRIVE,0,1,DDSL=36,A
COPY,0,1,00/00/00,CBF,FMT,NDP=NEWDOS4P,DPDN=1
```

2. Boot the new system diskette.
3. Create a file called MODELA/III - e.g.

```
CREATE MODELA/III:0
```

4. Using Superzap, page through the directory, from relative sector 360, till the FPDE for MODELA/III is located. Modify Bytes 21-24 in FPDE i.e.

```
0000 FFFF  to  3900 360B
```

5. From "NEWDOS READY" copy MODELA/III from the TRSDOS 1.3B boot disk to the new NEWDOS disk - e.g.

```
COPY MODELA/III:1 :0 SPDN=4
```

6. Using Superzap locate the MODELA/III FPDE and modify bytes 20-24 i.e.

```
Find  3900 360B  Change to  3900 1E09
```

Using Superzap 'DFS' modify MODELA/III at FRS 53 Relative Byte 39

```
Find  3E01 D3F2  Change to  3E00 D3F2
```

Using Superzap modify the first sector (360) of the directory (GAT). Change relative bytes 36H to 3BH inclusive to FF (i.e. lump(s) allocated).

Modify Disk Relative Sector 1 at relative byte 0: Do a 'ZTFF'<enter> to zero the sector and modify:

```
Relative Byte 02 ==> 14
Relative Byte 14 ==> 28
Relative Byte 8A ==> CD
```

7. Now the easy part, RESET the 4P holding the 'P' and 'L' keys. If the ROM fails to load successfully then recheck the FPDE entry and the Sector 1 bytes. If the ROM loads successfully but NEWDOS fails, either partly or completely, then recheck the patch to MODELA/III (or perhaps you have accidentally written to the wrong part of the system!).

- Tony Domigan, PO Box 150, Thomastown, Victoria, 3074, AUSTRALIA.

-July 17th 1984-

ALDATE

by Jack Decker

This program is specifically designed for use with ALLWRITE!, the new word processing system by PROSOFT, but it could be modified for use with other word processors. One thing I have noticed about most word processing programs is that they have no way to automatically bring the system date into a letter. It has always seemed kind of silly to me to have the current date stored in the computer's memory, but no way for the word processing program to do anything with it. This program is a partial solution to this situation.

The program below is designed to be patched into ALK/CMD, the keyboard driver program provided with ALLWRITE!. I keep the combination program (this program merged with ALK/CMD) on my ALLWRITE! disk under the name A/CMD for easy entry into ALLWRITE!. The code segment shown below gets the system date, translates it into words and numbers, and then writes it to a file called DATE/TXT which can then be imbedded into any ALLWRITE! text file using ALLWRITE's ";IM" control word. In this manner, "boilerplate" letters and/or logos can be developed which can call in the current date at any time.

Further instructions for installation and use of the program may be found in the assembly language source code comments. If you would like to try and modify this program for use with another word processor or as a stand alone program, please note the following:

1) You may change the ORG address in line 780, but the new address MUST end with 00H or the program will not work. In a stand-alone application (where this program might be part of a /JCL file, for example) you could ORG it at a lower memory location, such as 5600H or 7000H.

2) The PUSH HL instruction in line 1250 saves HL for ALK/CMD (PUSH HL is the instruction found at the ALK/CMD entry point, and the program below jumps into ALK/CMD just <u>past</u> this instruction). In most other applications you will have to eliminate this instruction and move the START label to line 1260.

3) Line 1730 is the program exit and in this case jumps to 7444H, which is (as mentioned above) one instruction past the normal ALK/CMD entry point of 7443H (in the version of ALK/CMD that I have). In most other applications this will have to be changed. You could return to DOS READY by making this a jump to 402DH.

If you <u>do</u> own ALLWRITE! and would like to combine this program with ALK/CMD as originally intended, you will have to use either a monitor program such as TASMON or a program such as CMDFILE/CMD (originally sold as a stand-alone program by Misosys, now included with LDOS). I can't give you specific instructions for this project because it will differ depending on the monitor program you are using. There are other ways to go about it, of course - you could disassemble ALK/CMD and combine it with the assembly language code below and then re-assemble, or you could even use a disk zap type program and manipulate bytes in the disk file to combine the programs (not recommended for the inexperienced!). The point is, there are many ways you can go about it, but the best method for you will depend on what types of utility programs you may have available.

```
00100 ;xxxxx ALDATE/ASM  -  by Jack Decker
00110 ;Version 1.0 - creation date 9/18/84.
00120
00130 ;This program is designed to work with the ALLWRITE! word
00140 ;processing system by Prosoft.  It is NOT a stand-alone
00150 ;program, but is designed to be patched into ALK/CMD (the
00160 ;keyboard driver program provided by Prosoft).  Whenever
00170 ;ALK/CMD is executed (after being patched with this
00180 ;program), it will first read the system date (the date
00190 ;entered at power-up or whenever the DOS DATE command is
00200 ;executed), convert it to a text string of the format:
00210 ;month, day, year (example:  December 31, 1985) and then
00220 ;output it to a file named DATE/TXT on drive zero.  This
00230 ;file can then be used by "boilerplate" letters and/or
00240 ;letterheads by using the ALLWRITE! command:
00250 ;                ;IM DATE/TXT
00260 ;This command will "imbed" the text of DATE/TXT within
00270 ;body of the document being created.  Note that the ;IM
00280 ;command does not cause a "control break", but must be
00290 ;the last command on a multi-command screen line.
00300
00310 ;If the system date stored in memory appears to be
00320 ;invalid, this program will terminate with an error
00330 ;message and the DATE/TXT file will be killed (if it
00340 ;existed previously).
00350
00360 ;There are several ways to combine this program with the
00370 ;ALK/CMD program.  This version of this program assumes
00380 ;that ALK/CMD loads from 7000H through 752EH and that the
00390 ;normal entry point is 7443H (this program uses an entry
00400 ;point of 7444H, skipping a PUSH HL instruction which has
00410 ;already been done at the start of the code below).
00420 ;One method of combining the programs is to use TASMON to
00430 ;load ALK/CMD (make sure the entry point address is 7443H
00440 ;in the version you have), load the object code produced
00450 ;by this program, and then write to disk using the
00460 ;ALK/CMD start address and this program's ending and
00470 ;entry point addresses.  The CMDFILE/CMD program
00480 ;(supplied with LDOS) could also be used, or as a last
00490 ;resort, you could use the DOS LOAD command to bring
00500 ;ALK/CMD into memory, and then execute this program.
00510
00520 ;I suggest using the SETDATE/CMD program (available on
00530 ;TAS Public Domain Library Disk # 001) to set the date
00540 ;easily when booting your DOS, unless you have a hardware
00550 ;clock module and appropriate software to do this for
00560 ;you.
00570
00580 ;This program will only work correctly during the 20th
00590 ;century (change line 1640 after the year 1999).
00600
00610 ;It would be easy to modify this program to output the
00620 ;date in a different format (such as date, month, year)
00630 ;but that will be left up to the individual.  Should you
00640 ;need a such a specialized version of this program and
00650 ;not feel capable of doing it yourself, I will consider
00660 ;producing customized versions of this program for a
00670 ;nominal fee.
00680
00690 ;Questions or comments MUST be accompanied by a self-
00700 ;addressed stamped envelope if you live in the U.S.A. or
00710 ;Canada (Canadian postage is O.K.) and wish a reply.
00720
00730 ;               Jack Decker
00740 ;               1804 West 18th Street   Lot # 155
00750 ;               Sault Ste. Marie, Michigan  49783
00760
00770
7600          00780          ORG     7600H           ;MUST end with 00H
00790
00800 ;String and date storage area used by program
00810
7600 1876     00820 TABLE    DEFW    JAN             ;Table of string pointers
7602 1F76     00830          DEFW    FEB             ;  point to strings
7604 2776     00840          DEFW    MAR             ;  containing months of
7606 2C76     00850          DEFW    APR             ;  the year
7608 3176     00860          DEFW    MAY
760A 3476     00870          DEFW    JUN
760C 3876     00880          DEFW    JUL
760E 3C76     00890          DEFW    AUG
7610 4276     00900          DEFW    SEP
7612 4B76     00910          DEFW    OCT
7614 5276     00920          DEFW    NOV
7616 5A76     00930          DEFW    DEC
7618 4A       00940 JAN      DEFM    'Januar'        ;Strings containing
     61 6E 75 61 72
761E F9       00950          DEFB    'y'+80H         ;  months of year
761F 46       00960 FEB      DEFM    'Februar'
     65 62 72 75 61 72
7626 F9       00970          DEFB    'y'+80H
7627 4D       00980 MAR      DEFM    'Marc'
     61 72 63
762B E8       00990          DEFB    'h'+80H
762C 41       01000 APR      DEFM    'Apri'
     70 72 69
7630 EC       01010          DEFB    'l'+80H
7631 4D       01020 MAY      DEFM    'Ma'
     61
7633 F9       01030          DEFB    'y'+80H
7634 4A       01040 JUN      DEFM    'Jun'
     75 6E
7637 E5       01050          DEFB    'e'+80H
7638 4A       01060 JUL      DEFM    'Jul'
     75 6C
763B F9       01070          DEFB    'y'+80H
763C 41       01080 AUG      DEFM    'Augus'
     75 67 75 73
7641 F4       01090          DEFB    't'+80H
```

```
7642 53         01100 SEP     DEFM    'Septembe'
     65 70 74 65 6D 62 65
764A F2         01110         DEFB    'r'+80H
764B 4F         01120 OCT     DEFM    'Octobe'
     63 74 6F 62 65
7651 F2         01130         DEFB    'r'+80H
7652 4E         01140 NOV     DEFM    'Novembe'
     6F 76 65 6D 62 65
7659 F2         01150         DEFB    'r'+80H
765A 44         01160 DEC     DEFM    'Decembe'
     65 63 65 6D 62 65
7661 F2         01170         DEFB    'r'+80H
7662 45         01180 ERRMSG  DEFM    'ERROR WHILE CREATING/UPDATING FILE: DATE/TXT:0'
     52 52 4F 52 20 57 48 49 4C 45 20 43 52 45 41 54
     49 4E 47 2F 55 50 44 41 54 49 4E 47 20 46 49 4C
     45 3A 20 44 41 54 45 2F 54 58 54 3A 30
7690 0D         01190         DEFB    0DH
7691 49         01200 INVALD  DEFM    'INVALID DATE IN MEMORY - KILLING DATE/TXT:0'
     4E 56 41 4C 49 44 20 44 41 54 45 20 49 4E 20 4D
     45 4D 4F 52 59 20 2D 20 4B 49 4C 4C 49 4E 47 20
     44 41 54 45 2F 54 58 54 3A 30
76BC 0D         01210         DEFB    0DH
                01220
                01230 ;Start of actual machine-language program
                01240
76BD E5         01250 START   PUSH    HL              ;Save input buffer pointr
76BE 0600       01260         LD      B,0             ;Logical Record Lngth=256
76C0 115777     01270         LD      DE,FCB          ;File Control Block pntr
76C3 217777     01280         LD      HL,FILBUF       ;File I/O Buffer pntr
76C6 CD2D44     01290         CALL    442DH           ;DOS INIT routine
76C9 2047       01300         JR      NZ,ERREXT       ;Go if error
76CB CD4B77     01310         CALL    GETBFR          ;Get loc memory date stor
76CE 0A         01320         LD      A,(BC)          ;Get year from memory
76CF FE64       01330         CP      100D            ;Is it in range 00 - 99?
76D1 300E       01340         JR      NC,NODATE       ;Go if invalid year
76D3 03         01350         INC     BC              ;Point to day in memory
76D4 0A         01360         LD      A,(BC)          ;Get day from memory
76D5 3D         01370         DEC     A               ;Adjust valid to 0 - 30
76D6 FE1F       01380         CP      31D             ;Is day 1 - 31 in memory?
76D8 3007       01390         JR      NC,NODATE       ;Go if invalid month
76DA 03         01400         INC     BC              ;Point to month in memory
76DB 0A         01410         LD      A,(BC)          ;Get month from memory
76DC 3D         01420         DEC     A               ;Adjust valid to 0 - 11
76DD FE0C       01430         CP      12D             ;Is month 1 - 12 in mem?
76DF 380D       01440         JR      C,CONT          ;Go if memory date valid
76E1 219176     01450 NODATE  LD      HL,INVALD       ;Point to invalid dt msg
76E4 CD6744     01460         CALL    4467H           ;Display it on video
76E7 CD2C44     01470         CALL    442CH           ;Kill DATE/TXT:0
76EA 2831       01480         JR      Z,EXIT          ;Go if no error
76EC 1824       01490         JR      ERREXT          ;Use error exit
76EE CD3577     01500 CONT    CALL    OUTSTR          ;Output month string
76F1 CD2577     01510         CALL    PRTSPC          ;Print space character
76F4 0B         01520         DEC     BC              ;Point to day byte
76F5 0A         01530         LD      A,(BC)          ;Get day (1 - 31)
76F6 6F         01540         LD      L,A             ;Put day in L
76F7 2600       01550         LD      H,0             ;HL = day
76F9 C5         01560         PUSH    BC              ;Save date storage pntr
76FA CD2A77     01570         CALL    PRTNUM          ;Print day
76FD C1         01580         POP     BC              ;Restore date storage ptr
76FE CD2077     01590         CALL    PRTCOM          ;Print comma and space
7701 0B         01600         DEC     BC              ;Point to year byte
7702 0A         01610         LD      A,(BC)          ;Get year (0 - 99)
7703 4F         01620         LD      C,A             ;Put year in C
7704 0600       01630         LD      B,0             ;BC = last 2 digits year
7706 216C07     01640         LD      HL,1900D        ;Get century offset
7709 09         01650         ADD     HL,BC           ;HL = Year (all 4 digits)
770A CD2A77     01660         CALL    PRTNUM          ;Print year
770D CD2944     01670         CALL    4429H           ;CLOSE file
7710 2808       01680         JR      Z,EXIT          ;Go if no error
7712 F680       01690 ERREXT  OR      80H             ;Set bit 7 to ret to pgm
7714 CD0944     01700         CALL    4409H           ;Display DOS error msg
7717 216276     01710         LD      HL,ERRMSG       ;Point to additonal msg
771A CD6744     01720         CALL    4467H           ;Display it on video
771D C34474     01730 EXIT    JP      7444H           ;Go to ALK initialization
7720 3E2C       01740 PRTCOM  LD      A,','           ;Comma character in A
7722 CD1B00     01750         CALL    1BH             ;Output it to file
7725 3E20       01760 PRTSPC  LD      A,' '           ;Space character in A
7727 C31B00     01770         JP      1BH             ;Output to file & RET
772A D5         01780 PRTNUM  PUSH    DE              ;Save FCB pointer
772B CD9A0A     01790         CALL    0A9AH           ;Number in HL to ACCUM
772E CDBD0F     01800         CALL    0FBDH           ;Convert # to string
7731 D1         01810         POP     DE              ;Restore FCB pointer
7732 23         01820         INC     HL              ;Skip leading space char
7733 1808       01830         JR      OUTTXT          ;Display converted number
7735 07         01840 OUTSTR  RLCA                    ;A=A*2 (2 byte pointers)
7736 6F         01850         LD      L,A             ;L=LSB string table addr
7737 2676       01860         LD      H,TABLE<-8      ;H=MSB string table addr
7739 7E         01870         LD      A,(HL)          ;A=LSB actual string addr
773A 23         01880         INC     HL              ;Point to MSB string addr
773B 66         01890         LD      H,(HL)          ;H=MSB actual string addr
773C 6F         01900         LD      L,A             ;HL=string location addr
773D 7E         01910 OUTTXT  LD      A,(HL)          ;Get byte to output
773E B7         01920         OR      A               ;See if zero terminator
773F C8         01930         RET     Z               ;Finished if zero byte
7740 F5         01940         PUSH    AF              ;Save sign flag status
7741 E67F       01950         AND     7FH             ;Mask off bit 7
7743 CD1B00     01960         CALL    1BH             ;Output it to file
7746 F1         01970         POP     AF              ;Restore sign flag
7747 F8         01980         RET     M               ;Finished if bit 7 set
7748 23         01990         INC     HL              ;Advance string pointer
7749 18F2       02000         JR      OUTTXT          ;Go print next byte
774B 014440     02010 GETBFR  LD      BC,4044H        ;BC=Mod I date storage
774E 3A5400     02020         LD      A,(54H)         ;Check which model TRS-80
7751 3D         02030         DEC     A               ;A will be 0 on Model I
7752 C8         02040         RET     Z               ;Return if Model I
7753 011A42     02050         LD      BC,421AH        ;BC=Mod III date storage
7756 C9         02060         RET                     ;Pointing to Mod III date
7757 44         02070 FCB     DEFM    'DATE/TXT:0'    ;File Control Block area
     41 54 45 2F 54 58 54 3A 30
7761 03         02080         DEFB    3               ;  with program filename
0015            02090         DEFS    21D             ;  (total 32 bytes)
0100            02100 FILBUF  DEFS    100H            ;File I/O buffer area
76BD            02110         END     START
00000 TOTAL ERRORS
```

APR	762C	AUG	763C	CONT	76EE	DEC	765A	ERREXT	7712
ERRMSG	7662	EXIT	771D	FCB	7757	FEB	761F	FILBUF	7777
GETBFR	774B	INVALD	7691	JAN	7618	JUL	7638	JUN	7634
MAR	7627	MAY	7631	NODATE	76E1	NOV	7652	OCT	764B
OUTSTR	7735	OUTTXT	773D	PRTCOM	7720	PRTNUM	772A	PRTSPC	7725
SEP	7642	START	76BD	TABLE	7600				

NEWS FLASH
MCI MAIL NOW ACCESSIBLE IN SEVERAL COUNTRIES WORLDWIDE

We don't have full details of this at press time, but apparently the procedure for foreign users is to contact your local PTT (postal/telephone/telegraph) authority for information on connecting with MCI Mail. As far as MCI Mail is concerned, the charges are the same (that is, no charge for connect time or to read mail, only $1.00 U.S. to send an "instant" letter to another user, etc.) BUT you may incur connect-time charges from your local data network.

As an example, in Canada a potential MCI Mail user would contact Bell Datapac for information on accessing MCI Mail, and would presumably have to pay a connect-time charge to Bell Datapac for time spent in contact with MCI Mail, in addition to the charge for any mail sent over MCI Mail. I am hoping to get some better information on this in the near future, and will pass it on when I receive it. In the meantime, if any of our foreign readers manage to get onto MCI Mail through a local access number, I'd appreciate it if you'd drop me a line and let me know how you're being charged.

ARCHBOLD SPEEDUP MOD WANTED

If anyone has an Archbold speedup modification for the Model I that you'd like to sell, or if you have a schematic diagram for this unit, Steve Winokur would like to hear from you. Apparently the Archbold mod is out of production, and Steve wants that particular modification. If you can help, contact Steve at 435 Norristown Road, Horsham, Pennsylvania 19044 or call (215) 675-7708 after 4 P.M. Eastern time.

MODEL I VIDEO MONITOR FOR SALE

Includes hard plastic green screen. The plastic cabinet is very slightly blemished (could be easily fixed with a little Tandy silver-grey paint) but otherwise it seems to work fine. Asking $75 (price is negotiable). Also for sale: TC-8 high speed tape storage unit made by JPC Products Company (see ads in early issues of 80-Microcomputing), also a "Memory Box" 2K memory expansion unit that provides RAM memory at the (normally vacant) memory area from 3000H-37DFH in the Model I. Make an offer on the latter two items. Contact NORTHERN BYTES at our Sault Ste. Marie address (see return address on back page).

Information Brokers:
The Indispensable Service of the Information Age

By John H. Everett and Elizabeth Powell Crowe

[Editor's note: This article has been provided to NORTHERN BYTES and other publications by J. Norman Goode, publisher of Micro Moonlighter Newsletter, and is Copyright 1984 by Ferret Press.]

The Information Age has spawned its share of cliches; one of the most pervasive is "Information is Power." As John Naisbitt, author of Megatrends, has said "We are drowning in information but starved for knowledge." That starvation for knowledge means that INFORMATION IS PROFIT.

We all have too much information. What we need is not more information, but the ability to access precisely the information we need precisely when we need it and not have to worry too much about it otherwise. Our individual ability to accomplish this will be crucial to our success, whether employees of a major corporation, entrepreneurs, students or even among the unemployed (whether by choice or circumstance).

A new kind of information specialist is emerging to help us manage our information needs. This specialist has the talents and abilities to take advantage of the latest technology while utilizing traditional information management skill. This specialist can not only help us identify what information we really do need, but can help us get it, interpret it and use it. This specialist is going to be an integral part of the Information Age and our ability to function in it. This specialist is the INFORMATION BROKER.

In truth, Information Brokers have been around for many years; they are not a by-product of computer databases. The need to manage printed information has long demanded specialists to catalog, retrieve and use the information. It is the growth in the amount of information, its availability through computer communications and its importance to our decision-making in today's rapidly changing world that has elevated Information Brokering to its present importance.

While public libraries are debating their responsibility to make information available as a public service in light of the business nature and high cost of much of the information online, the private sector is staking out a major role in information retrieval using these same expensive, business oriented databases. No one business, or type of business, has so penetrated the information market that the directions in which this market will grow have become unalterable. But it is likely that a small number of small businesses and individual entrepreneurs will capture the significant share of that market that remains.

In case you're wondering what good an Information Broker could be to someone, consider this example:

Two competitors need information on the use of computers in special education classes.

One of them goes to the library, for three days in a row. First he looks in the magazine index; there he finds that education magazines, general magazines, medical magazines and computer magazines have had something on the subject. The he looks in Books in Print; finally he checks the library's monograph file. He reads each article that seems to fit, scans the books that seem pertinent, reads the monographs. He finds that some will not do. After three days, he has copious notes and a vague idea of what's going on. He will take it all back to the office and study it to winnow out what he needs.

The other person knows about Information Brokering. She calls her favorite database searcher and asks for bibliography and short summary of the most recent articles that have the words SPECIAL EDUCATION and COMPUTER, MENTAL RETARDATION and COMPUTER, and COMPUTER AIDED EDUCATION and SPECIAL STUDENTS. The next morning, she has a list of the books and articles that fit her subject. She orders hard copies of several that are right on target. They are delivered less than 48 hours after she made the first call. She never left her office, and she had time to devote to her other duties while the search was made.

No matter what the information need, it is highly likely that a computer database exists on that subject. Somewhere out there are facts about chickens who wear contact lenses, how many college degrees Art Garfunkel holds, and the latest patents. Wandering around in the same "out there" are people who need that information and don't know where to find it.

Traditional methods of research for finding these facts are still useful. This is especially true for information of a historical nature, since many databases weren't computerized before the late 70's. But for the most current facts, figures and analyses, one must go to the online databases. No one can master each and every byte of them; most will not try. To be able to get to those parts that affect you when you need the data is important; but it is not easy.

The information service industry came about because the amount of data has grown unimaginably large, retrieval has become more complicated and expensive, and most people do have time to manage their information. This does not decrease their need for information or the urgency of their need.

When computers were mostly confined to businesses and colleges, they were used as number crunchers and formula blenders. Then someone invented a game for these no-nonsense machines called "Adventure". It was filled with mazes, treasures, pitfalls and monsters. But as this was before graphics were as available as they are today, you had to keep a map of the "geography" in your head in order to gather the treasures and survive.

Online databases are not unlike the adventure games. Many treasures of great value are hidden in realms where "magic" words can retrieve them; but perils await the unwary. Within minutes or seconds, the magic word, if known, can conjure up data that would take weeks, even months, of manual research.

But gaining the knowledge of the magic words, and to which realms they apply, is costly. These dynamic, ever-growing kingdoms of statistics, facts, and formulae are not easy to navigate unless you have training, lots of experience, and the right computer hardware and software.

Steven K. Roberts, in an article in ONLINE TODAY calls the Information Broker the free-lancer of the Information Age. Kelly Warnken in her book The Information Brokers calls it, "a fee-based information service." Bob Sherman of COMPUTER ASSISTED RESEARCH ON LINE (C.A.R.O.L.) defines his business as, "A service that does database research for clients and, if needed, tracks and obtains the documents." Norm Goode, publisher of MICRO MOONLIGHTER NEWSLETTER, compares Information Brokers to retailers: "The product is information, in all its variant forms; the 'value added' is the ability to gather all available material about a particular subject that is of use to the end user..."

Some of the things that can be done using computers to research are:

- Abstracting
- Analyzing information
- Bibliography collecting
- Computer software design
- Consulting
- Directory compilation
- Document collection and delivery
- Identifying experts
- Assistance in grant preparation
- Indexing
- Industry overviews
- Instant education
- Literature searching
- Purchasing reports
- Specific subject updates
- Systems design
- Thesaurus construction
- Verifying facts

You can probably add a few more, using your imagination. What an Information Broker does that is so valuable to the client is she delivers these things fast and precisely on target for the client's needs. After narrowly defining the client's question, she knows what database service to search, which services within that service, what commands are necessary to call up that specific information. She can deliver a bibliography, the actual documents, an abstract, a report, or a combination, depending on how how she defines her business. Most often, Information Brokers deliver a report, or synopsis with the pertinent facts, the source and the further reading necessary.

Information Brokers are not restricted by geographical location. Telephones, modems, electronic mail and overnight delivery add up to a continent-wide potential market.

Can just anyone pick up a modem and begin to solve the world's need for pertinent information? NO! One needs skills in research. One needs training and experience in databases in general, and each of them specifically. One needs skills in compiling and communicating the information found into a useable form. One needs skills in interviewing the client to define the "what" he really needs. One needs marketing skills, business skills, and the ability to see when he needs help from the outside.

Is anyone out there willing to pay enough to make your investment in all that time and money not only worth it but also profitable? YES! Bob Sherman of C.A.R.O.L. lists among his clients: doctors, politicians, television producers, writers, companies that want corporate snooping, researchers, and lawyer. RESPONSE TIME in Irving, Texas has had clients representing small business, real estate and the arts. In short, anyone or any company, large or small, that uses information could at some time use an Information Broker.

Even large companies with existing research and development departments come to Information Brokers. Perhaps the need is outside the department's repertoire. Or perhaps the data is on a database that will rarely be used and has a high monthly minimum charge. Or maybe there's a crunch on: the data is needed right now, but the staff is already handling all it can.

An attorney who needs a file on a specific medical condition, a summary of judgements in cases decided concerning that medical condition, and statistics on how many people currently suffer from it could have his legal aide flounder among the databases for two weeks, or call an Information Broker and ask for it tomorrow. Which do you think he'll do?

A small business wants to grow, but is it possible? The owner needs facts on her product, how it is doing in the region, and nationally. Does advertising help in her industry? How about hiring the handicapped?

How many miles is it from New York to Zanzibar, and who crossed it on a bicycle? Some television producer in Hollywood HAS to know, and no one but his Information Broker can help. And a business-to-be might be more successful with a pre-opening data search on the competition, especially if the search is computerized, up-to-the-minute and comprehensive.

All these people know they have a need for information. Anyone can use an Information Broker. And pretty soon, anyone will.

* * * * * * * * * * * * * * * *

This is an excerpt from the new book: The Information Broker's Handbook: How to Profit From The Information Age by John H. Everett and Elizabeth Powell Crowe. The book is available for $24.95 plus $2.00 Postage & Handling from: Ferret Press, 4121 Buckthorn Court, Lewisville, Texas 75028

BASIC TELEPHONE DIALER PROGRAM
by David R. McGlumphy / MCI ID: 181-7759

[Dave McGlumphy gave us an assembly language telephone dialer program in NORTHERN BYTES Volume 5, Number 5. It had a small bug in it, so Dave sent the fix (which appears in "THE EXTERMINATOR" column) and at the same time, sent a dialer program written in BASIC for those who feel more comfortable with BASIC than with assembly language. He sent this to me via MCI Mail at 2:32 A.M. on a Wednesday morning, and without knowing he had done this, I signed on at approximately 2:40 A.M. and got it! Talk about fast delivery! (Talk about people who don't have sense enough to go to bed...) The speedy delivery made a difference, too, since I was just finishing up this issue (in fact I had just finished running the files through a spelling checker in preparation for printing them out)! Anyway, without further ado, here's Dave's program!]

```
10 GOTO 80 'Dave McGlumphy  4429 Paula Ln  Chattanooga Tn 374
15 02/19/84
20 CLS : PRINT"save 'dialer'" : SAVE"dialer"
30 'THIS PROGRAM IS FOR A TRS-80 MODEL I TO DIAL A PHON
E.
40 'I THINK MODEL III USERS CAN USE IT IF THEY CHANGE VA
RIABLE
50 'MN TO 236.  IT TOGGLES THE CASSETTE REMOTE PLUG ON/
OFF.
60 'MAKE THAT PLUG CONTROL A NORMALLY CLOSED RELAY
WHICH
70 'IS IN SERIES WITH THE RED OR GREEN PHONE LINE.
80 POKE14308,1 'SELECTS CASSETTE CABLE #2 SINCE I USE #1
90 '          FOR SOUND GENERATION AND TAPES.
100 MN=255 'CASSETTE REMOTE PLUG ON/OFF PORT
110 CLS
120 PRINT"A. ATC"
130 PRINT"B. BASIC"
140 PRINT"C. M80SLOW/CMD"
150 PRINT"D. DOS"
160 PRINT"E. 68MICRO JOURNAL.... 842-6809"
170 PRINT"H. WOMACK BB.....891-0136 (6:00 - 23:00)"
180 PRINT"I. CRABAPPLE.....875-6035 (W21IL)"
190 PRINT"J. MCI,(Jack=102-7413),1-800-323-7751"
200 PRINT"K. CHRIS SMITH.....899-5377"
210 PRINT"M. Manually dial a number."
220 PRINT"N. BUTCH          ....16146953056"
230 PRINT"O. Pete Chiello.....870-1324"
240 PRINT"R. boot      "
250 PRINT"X. hang up.  "
260 PRINT"Y. pick up.  ";
280 I$=INKEY$
290 I$=INKEY$ : IF I$="" THEN 290 ELSE PRINT I$;"  ";
300 ON INSTR("AaBbCcDdEeHhIiJjKkMmNnOoRrXxYy",I$) GOTO 3
40,340,360,360,370,370,380,380,390,390,400,400,410,410,420, 420,4
30,430,350,350,440,440,450,450,460,460,320,320,330,330
310 GOTO 290
320 OUT MN,4 : GOTO 280 'hang up.
330 OUT MN,0 : GOTO 280 'pick up.
340 CMD"s=atc"
350 INPUT"WHAT NUMBER";PH$ : GOTO 470
360 CLS : END
370 CMD"S=MODEM80"
380 CLS : CMD"S"
390 PH$="8426809" : GOTO 470
400 PH$="8910136" : GOTO 470
410 PH$="8756035" : GOTO 470
420 PH$="18003237751" : GOTO 470
430 PH$="8995377" : GOTO 470
440 PH$="16146953056":GOTO470
450 PH$="8701324" : GOTO 470
460 CMD"boot"
470 PRINT : GOSUB 640 'TO GET DIAL TONE
480 FOR I = 1 TO LEN(PH$)
490 Z$=MID$(PH$,I,1)
500 IF Z$<"0" OR Z$>"9" THEN 620
510 Z=VAL(Z$)
520 PRINT Z;
530 IF Z=0 THEN Z=10
540 ' TIMER BETWEEN DIGITS
550 GOSUB 700
560 FOR J=1 TO Z
570 OUT MN,4
580 GOSUB 720
590 OUT MN,0
600 GOSUB 720
610 NEXT J
620 NEXT I
630 GOTO 110
640 ' HANG UP, THEN GET DIAL TONE.
650 OUT MN,4
660 FOR I=1 TO 400 : NEXT
670 OUT MN,0
680 FOR I=1 TO 200 : NEXT I
690 RETURN
700 ' TIMER BETWEEN DIGITS
710 FOR K=1 TO 80 : NEXT K : RETURN
720 ' TIMER WITHIN A DIGIT
730 FOR L=1 TO 1 'INCREASE THIS IF YOU NEED 10 PULSES/SEC
.
740 NEXT L '     IT GOES ABOUT 20/SEC NOW.
750 RETURN
```

PATCHES FOR TAS PROGRAMS TO WORK ON THE MODEL 4P
by Tony Domigan

Patch for KBE package to work on the Model 4P

Using Superzap Modify DVRIII64/CMD:

FRS 04 Rel Byte 61 find C3 FC 33 change to C3 74 31

FRS 05 Rel Byte 08 find CD 40 04 change to CD DC 01

Patch for TASMON [not the new upgrade!] to work on the Model 4P

Load TASMON/CMD from NEWDOS Ready.

Execute Debug '123' and:

modify 799F find 21 C5 30 change to 21 CE 30

modify 79B9 find 21 05 31 or 21 25 31 change to 21 8E 30

Now jump to 6000H 'G6000' and test that Tasmon works OK including the arithmetic operations. Then write a new copy of tasmon to disk.... E.G.

```
W D 6000 71FF 6000
TAS4P/CMD:0
```

- Tony Domigan, PO Box 150, Thomastown, Victoria, 3074, AUSTRALIA.

-July 18th 1984-

GETPROG .
by Tony Domigan

GETPROG/ASM is a simple demonstration use of a Model 4 32K memory bank in the Model III mode.

The Model 4 technical reference manual tells us how to decode port 84H (page 18) and the code to move each memory bank is given in figure 3-2.

Summary of Control Bytes To Move the Memory Banks.

Upper bank 1 (32k)	moved low	- address 0000-7FFFH = 60H
	moved high	- address 8000-7FFFH = 20H
Upper bank 2 (32k)	moved low	- address 0000-7FFFH = 70H
	moved high	- address 8000-7FFFH = 30H
12K RAM in lieu of ROM	Model 4 only	- address 0000-2FFFH = 01H
Open ROM to 'write'	Model 4P only	- address 0000-2FFFH = 01H

In this routine I have used the upper 32k bank and moved it to overlay the bottom 32K memory of the 4/4P. Superzap, Edtasm and Tasmon were loaded to the 32K bank and the keyboard driver intercepted so that the programs could be invoked at any time by the 3 function keys.

I used the following simple steps to store each of the 3 programs.

1. The program was loaded to its normal load addresses. (<8000H).

2. The program was relocated to 8000H and above.

3. The first upper 32K bank was switched low (0000H-7FFFH).

4. The program was relocated into the lower addressed memory.

5. The ROM area bank was reinstated.

Once the three programs were stored in the extra bank memory the keyboard DCB was patched with this program's intercept address and HIMEM was modified to protect the last part of the GETPROG routine.

The intercept address of the routine checks for the F1, F2 or F3 keys and if found then the extra memory bank is selected low, the program relocated to 8000H+, the default ROM state selected and the program relocated to its proper load address and executed.

Simple stuff!. The only thing you must remember is to make no ROM calls when the ROM is deselected. We could also have used the second upper bank switched low or either bank switched high with the driver relocated below 8000H. Incidentally, Model 4 users can also use the 12k portion of RAM that shadows ROM, any Model 4P users like myself can eat your hearts out - it sure would be handy to have ROM resident on-board.

(P.S. Anyone worked out how to attach external drives to the 4P?)

- Tony Domigan, PO Box 150, Thomastown, Victoria, 3074, AUSTRALIA.
-August 9,1984-

[EDITOR'S NOTE: This program as printed below will NOT work with the new, upgraded version of TASMON because the upgraded version loads from 6000H through 8100H. In order to use the program below with the NEW version of TASMON, the following steps should work:

1. Enter TASMON and type X 6000 80FF C000 to relocate TASMON to C000H. Then type G C000 to jump to the relocated version.

2. Now type X C000 E0FF 5E00 to relocate TASMON to 5E00H. The goal of these two steps is to put the entire TASMON program below 7FFFH. Save the relocated TASMON program back to disk using W D 5E00 7EFF 5E00 TASMON/CMD (you might want to rename your original copy of TASMON first so you don't overwrite it).

3. In the program below, change the address 6000H as found in lines 540, 590, 1190, 1240, and 1270 to 5E00H. Also, change the value 1FFFH as found in lines 560, 600, 1210, and 1250 to 20FFH. Finally, change "[6000-7FFFH]" to "[5E00-7EFFH]" in line 820.

Keep in mind that when the new TASMON is loaded from a memory bank after the above procedure, it will reside from 5E00H through 7EFFH rather than the usual 6000H - 71FFH.

As for Tony's question about external drives on the 4P, I've heard a rumor to the effect that the 4P disk controller is simply the regular Model 4 disk controller with the edge card connector sawed off. If that's true, then it should be possible to either attach a connector (perhaps using something similar to a "gold plug") to the existing disk controller board, or else simply replace the existing controller board with a standard Model 4 disk controller board (from Tandy or one of the aftermarket suppliers). The additional drives would then be simply plugged into the connector as on a standard Model 4. If anyone tries this and gets it to work, let us know about it! Here's Tony's program:]

```
                00100 ;
                00110 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00120 ;x              -- GETPROG/ASM --                      x
                00130 ;x                -Demo Routine-                       x
                00140 ;x          Moves progs to upper 32K bank 2            x
                00150 ;x         [requires NEWDOS80/v2,48KP-128K]            x
                00160 ;x           (EDTASM-SUPERZAP-TASMON)                  x
                00170 ;x           Invoke with F1, F2 and F3                 x
                00180 ;x          BY TONY DOMIGAN - August'84                x
                00190 ;x PO Box 150 Thomastown,Victoria,3074,Australia       x
                00200 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00210 ;
                00220 ;
                00230 ;
FDC0            00240          xORG    0FDC0H          ;Demo assembly pt
FDC0 2174FE     00250 START    LD      HL,MSG1         ;'Load edtasm' msg
FDC3 CD6744     00260          CALL    4467H           ;Print it
FDC6 2142FE     00270          LD      HL,CMD1         ;LOAD command
FDC9 CD1944     00280          CALL    4419H           ;Execute & return
FDCC 210052     00290          LD      HL,5200H        ;Reloc pt for edtasm
FDCF 118080     00300          LD      DE,8080H        ;Start of bank 2
FDD2 01FF24     00310          LD      BC,24FFH        ;Length of edtasm
FDD5 EDB0       00320          LDIR                    ;Move it
FDD7 218080     00330          LD      HL,8080H        ;Reloc start -edtasm
FDDA 110000     00340          LD      DE,0000H        ;Start of upper bank 2
FDDD 01FF24     00350          LD      BC,24FFH        ;Lenght of edtasm
FDE0 CDC3FF     00360          CALL    SWAP            ;Switch up.bank 2 low &
                00370                                  ;mv edtasm to 0000-24FFH
FDE3 21ADFE     00380          LD      HL,MSG2         ;'Load superzap' msg
FDE6 CD6744     00390          CALL    4467H           ;
FDE9 2152FE     00400          LD      HL,CMD2         ;
FDEC CD1944     00410          CALL    4419H           ;
FDEF 210052     00420          LD      HL,5200H        ;
FDF2 118080     00430          LD      DE,8080H        ;
FDF5 01FF1D     00440          LD      BC,1DFFH        ;
FDF8 EDB0       00450          LDIR                    ;
FDFA 218080     00460          LD      HL,8080H        ;Start of reloc. s'zap
FDFD 110040     00470          LD      DE,4080H        ;Dest. in upper bank 2
FE00 01FF1D     00480          LD      BC,1DFFH        ;Lenght of superzap
FE03 CDC3FF     00490          CALL    SWAP            ;Mv s'zap to 4080-5DFFH
FE06 21E5FE     00500          LD      HL,MSG3         ;'Load tasmon' msg
FE09 CD6744     00510          CALL    4467H           ;
FE0C 2164FE     00520          LD      HL,CMD3         ;
FE0F CD1944     00530          CALL    4419H           ;
FE12 218060     00540          LD      HL,6080H        ;
FE15 118080     00550          LD      DE,8080H        ;
FE18 01FF1F     00560          LD      BC,1FFFH        ;
FE1B EDB0       00570          LDIR                    ;
FE1D 218080     00580          LD      HL,8080H        ;Start of reloc tasmon
FE20 116060     00590          LD      DE,6080H        ;Dest. in up. bank 2
FE23 01FF1F     00600          LD      BC,1FFFH        ;Length of tasmon
FE26 CDC3FF     00610          CALL    SWAP            ;Mv tasmon to 6080-7FFFH
FE29 211DFF     00620          LD      HL,MSG4         ;Completion message
FE2C CD6744     00630          CALL    4467H           ;
FE2F 2A1040     00640          LD      HL,(4010H)      ;Get driver address
FE32 225EFF     00650          LD      (KICEPT+1),HL   ;store address
FE35 215DFF     00660          LD      HL,KICEPT       ;keybd intercept addr
FE38 221640     00670          LD      (4016H),HL      ;Store it
FE3B 2B         00680          DEC     HL              ;
FE3C 221144     00690          LD      (4411H),HL      ;Store HIMEM
FE3F C32D40     00700          JP      402DH           ;EXIT TO NEWDOS
FE42 4C         00710 CMD1     DEFM    'LOAD EDTASM/CMD'
    4F 41 44 20 45 44 54 41 53 4D 2F 43 4D 44
FE51 0D         00720          DEFB    0DH
FE52 4C         00730 CMD2     DEFM    'LOAD SUPERZAP/CMD'
    4F 41 44 20 53 55 50 45 52 5A 41 50 2F 43 4D 44
FE63 0D         00740          DEFB    0DH
FE64 4C         00750 CMD3     DEFM    'LOAD TASMON/CMD'
    4F 41 44 20 54 41 53 4D 4F 4E 2F 43 4D 44
FE73 0D         00760          DEFB    0DH
FE74 0A         00770 MSG1     DEFB    0AH
FE75 4C         00780          DEFM    'Loading EDTASM   to upper 32k memory bank
    6F 61 64 69 6E 67 20 45 44 54 41 53 4D 20 20 20                [0080-24FFH]'
    74 6F 20 75 70 70 65 72 20 33 32 6B 20 6D 65 6D
    6F 72 79 20 62 61 6E 6B 20 20 5B 30 30 30 30 2D
    32 34 46 46 48 5D
FEAC 0D         00790          DEFB    0DH
FEAD 4C         00800 MSG2     DEFM    'Loading SUPERZAP to upper 32k memory bank
                                                   [4080-5100H]'
```

```
    6F 61 64 69 6E 67 20 53 55 50 45 52 5A 41 50 20
    74 6F 20 75 70 70 65 72 20 33 32 6B 20 6D 65 6D
    6F 72 79 20 62 61 6E 6B 20 20 5B 34 30 30 30 2D
    35 31 46 46 48 5D
FEE4 0D         00810           DEFB    0DH
FEE5 4C         00820 MSG3      DEFM    'Loading TASMON   to upper 32k memory bank
    6F 61 64 69 6E 67 20 54 41 53 4D 4F 4E 20 20 20                 [6000-7FFFH]'
    74 6F 20 75 70 70 65 72 20 33 32 6B 20 6D 65 6D
    6F 72 79 20 62 61 6E 6B 20 20 5B 36 30 30 30 2D
    37 31 46 46 48 5D
FF1C 0D         00830           DEFB    0DH
FF1D 0A         00840 MSG4      DEFB    0AH
FF1E 45         00850           DEFM    'EDTASM,SUPERZAP,TASMON now installed -
    44 54 41 53 4D 2C 53 55 50 45 52 5A 41 50 2C 54   Invoke with F1,F2 or F3'
    41 53 4D 4F 4E 20 6E 6F 77 20 69 6E 73 74 61 6C
    6C 65 64 20 2D 20 49 6E 76 6F 6B 65 20 77 69 74
    68 20 46 31 2C 46 32 20 6F 72 20 46 33
FF5C 0D         00860           DEFB    0DH
FF5D CD0000     00870 KICEPT    CALL    0000H           ;DOS keyboard driver
FF60 F5         00880           PUSH    AF              ;
FF61 3AF441     00890           LD      A,(41F4H)       ;
FF64 CB67       00900           BIT     4,A             ;Function key F1?
FF66 200A       00910           JR      NZ,GETEDT       ;Yes, get EDTASM
FF68 CB6F       00920           BIT     5,A             ;Function key F2?
FF6A 2021       00930           JR      NZ,GETZAP       ;Yes, get SUPERZAP
FF6C CB77       00940           BIT     6,A             ;Function key F3?
FF6E 2038       00950           JR      NZ,GETTAS       ;Yes, get TASMON
FF70 F1         00960           POP     AF              ;
FF71 C9         00970           RET                     ;Continue to DOS
FF72 F1         00980 GETEDT    POP     AF              ;Clear stack
FF73 210000     00990           LD      HL,0000H        ;EDTASM in bank 2
FF76 110080     01000           LD      DE,8000H        ;Reloc address
FF79 01FF24     01010           LD      BC,24FFH        ;Length
FF7C CDC3FF     01020           CALL    SWAP            ;Move it to 8000H
FF7F 210080     01030           LD      HL,8000H        ;Move it..
FF82 110052     01040           LD      DE,5200H        ;   back to ..
FF85 01FF24     01050           LD      BC,24FFH        ;     real address
FF88 EDB0       01060           LDIR                    ;
FF8A C3006F     01070           JP      6F00H           ;Execute EDTASM
FF8D F1         01080 GETZAP    POP     AF              ;
FF8E 214040     01090           LD      HL,4040H        ;
FF91 110080     01100           LD      DE,8000H        ;
FF94 01FF1D     01110           LD      BC,1DFFH        ;
FF97 CDC3FF     01120           CALL    SWAP            ;
FF9A 210080     01130           LD      HL,8000H        ;
FF9D 110052     01140           LD      DE,5200H        ;
FFA0 01FF1D     01150           LD      BC,1DFFH        ;
FFA3 EDB0       01160           LDIR                    ;
FFA5 C3C554     01170           JP      54C5H           ;Execute SUPERZAP
FFA8 F1         01180 GETTAS    POP     AF              ;
FFA9 216060     01190           LD      HL,6060H        ;
FFAC 110080     01200           LD      DE,8000H        ;
FFAF 01FF1F     01210           LD      BC,1FFFH        ;
FFB2 CDC3FF     01220           CALL    SWAP            ;
FFB5 210080     01230           LD      HL,8000H        ;
FFB8 116060     01240           LD      DE,6060H        ;
FFBB 01FF1F     01250           LD      BC,1FFFH        ;
FFBE EDB0       01260           LDIR                    ;
FFC0 C30060     01270           JP      6000H           ;Execute TASMON
FFC3 F3         01280 SWAP      DI                      ;Must be done!
FFC4 3E61       01290           LD      A,61H           ;Move 2nd upper bank to
                01300                                   ;overlay RAM/ROM area
FFC6 D384       01310           OUT     (84H),A         ;Switch banks
FFC8 EDB0       01320           LDIR                    ;Move program
FFCA AF         01330           XOR     A               ;A=0
FFCB D384       01340           OUT     (84H),A         ;Default state
FFCD FB         01350           EI                      ;
FFCE C9         01360           RET                     ;
FDC0            01370           END     START
00000 TOTAL ERRORS

CMD1    FE42    CMD2    FE52    CMD3    FE64    GETEDT  FF72    GETTAS  FFA8
GETZAP  FF8D    KICEPT  FF5D    MSG1    FE74    MSG2    FEAD    MSG3    FEE5
MSG4    FF1D    START   FDC0    SWAP    FFC3
```

Video4 Patch for the Model 4P
Requires Model 4P 128K & NEWDOS/80 v2.0
by Tony Domigan

After successfully using Video4 on my Model 4 for some time I was devastated to find that it bombed out after disk I/O on my 4P. I rechecked the required patches to NEWDOS and they were correct, or at least as correct as I had first found them (i.e. patch 4 to sys19 ..the bytes located were 21 00 CD in place of 11 00 CD.)

This patch has not been tested exhaustively so I cannot guarantee that it will work perfectly every time.

I believe that the problem lies with the exit condition within the program. With the 4P there is no extra bank to be displaced by ROM and hence when you exit with select bit zero set, then the ROM is open to a direct write(?). During disk I/O this situation appears to cause a problem - just how I do not know [Editor's note: See "THE EXTERMINATOR" on page 2 of NORTHERN BYTES, Volume 5 Number 5 for an explanation of the problem and the four patches that are normally required when using Video4 with NEWDOS/80]. This small patch, however, does seem to work; thankfully, as I enjoy using Video4kb. My solution was to exit with both select bits off and 80 column mode selected. The change in Video4kb/cmd is as follows:

ADDRESS	ORIGINAL	CHANGED TO
53A5	LD A,5	LD A,4
53A7	OUT (84H),A	OUT (84H),A

Another problem occurs when you wish to reboot NEWDOS while Video4(kb) is still active. Pressing the reset key or using the NEWDOS BOOT command results in the 4P 'hanging-up'. I assumed that this occurred because of a changed state in the ROM as modified by Video4, although I guess it could also be due to an unstable Bank-switched state(?).

My solution was therefore to save the standard ROM to the second upper 32K spare bank and to retrieve it later prior to rebooting.

GETVID4/ASM moves the ROM to the second upper spare bank, loads VIDEO4KB/CMD, patches address 53A5H and executes Video4KB.

ROMBOOT/ASM is called when the user wishes to reboot NEWDOS. The program reloads the ROM and reboots the system (RST00).

- Tony Domigan, PO Box 150, Thomastown, Victoria, 3074, AUSTRALIA.
-August 21st,1984-

```
                00100 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00110 ;x            GETVID4P/ASM - Model 4P                 x
                00120 ;x  1. Saves ROM to 32K Bank for later retrieval       x
                00130 ;x   when REBOOT is required after using VIDEO4       x
                00140 ;x   VIDEO4/CMD package [by JACK DECKER (TAS)].        x
                00141 ;x  2. Loads, Patches and Executes VIDEO4(KB) for 4P.  x
                00150 ;x       BY Tony Domigan (August 1984).               x
                00160 ;x   PO Box 150,Thomastown,Victoria,3074,Australia.   x
                00170 ;x     See ROMBOOT/ASM for reverse procedure on        x
                00180 ;x     EXIT and REBOOT from VIDEO4 and VIDEO4KB        x
                00190 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
6000            00200           ORG     6000H           ;Must be below 8000H
                00210                                   ;and above Video4(kb)
6000 210000     00220 START     LD      HL,0000H        ;Start of ROM
6003 110080     00230           LD      DE,8000H        ;Start bank 3 (high)
6006 01FF37     00240           LD      BC,37FFH        ;Bytes to Transfer
6009 CD5E60     00250           CALL    SWAP            ;Transfer ROM to Bank3
600C 212960     00260           LD      HL,MSG1         ;Pt to info message
600F CD6744     00270           CALL    4467H           ;Print message
6012 CD6000     00280           CALL    60H             ;delay
6015 CD6000     00290           CALL    60H             ;delay
6018 CD6000     00300           CALL    60H             ;delay
601B 214C60     00310           LD      HL,CMD          ;Pt to Dos Cmd
601E CD1944     00320           CALL    4419H           ;Load 'Video4kb F'
6021 3E04       00330           LD      A,04H           ;patch
6023 32A653     00340           LD      (53A6H),A       ;patch it
6026 C30752     00350           JP      5207H           ;exec video4kb F
                00360 ;         JP      5208H           ;exec video4kb
6029 52         00370 MSG1      DEFM    'ROM has been saved to Upper Bank 2'
    4F 4D 20 68 61 73 20 62 65 65 6E 20 73 61 76 65
    64 20 74 6F 20 55 70 70 65 72 20 42 61 6E 6B 20
    32
604B 0D         00380           DEFB    0DH
604C 4C         00390 CMD       DEFM    'LOAD VIDEO4KB/CMD'     ;Dos Command
    4F 41 44 20 56 49 44 45 4F 34 4B 42 2F 43 4D 44
605D 0D         00400           DEFB    0DH
                00410 ;
605E F3         00420 SWAP      DI                      ;must do this
605F 3E30       00430           LD      A,30H           ;upper bank switched high
6061 D384       00440           OUT     (84H),A         ;switch bank to 8000-FFFF
6063 EDB0       00450           LDIR                    ;Move ROM to 8000+
6065 AF         00460           XOR     A               ;A=0
6066 D384       00470           OUT     (84H),A         ;Reinstate banks
6068 FB         00480           EI
6069 C9         00490           RET
6000            00500           END     START
00000 TOTAL ERRORS
```

ARTICLE CONTINUED NEXT PAGE >>>>>>>>>

```
CMD     604C    MSG1    6029    START   6000    SWAP    605E

                00100 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00110 ;x              ROMBOOT/ASM - Model 4P                  x
                00120 ;x    For use with Video4 package written by            x
                00130 ;x    Jack Decker (TAS) when System Boot is             x
                00140 ;x  required whilst Video4/Video4kb is active           x
                00150 ;x    ROMBOOT reloads the standard ROM image            x
                00160 ;x  saved under GETVID4P/CMD and reboots the            x
                00170 ;x          Newdos 80 v2.0 system.                      x
                00180 ;x       by Tony Domigan (August - 1984)                x
                00190 ;x PO Box 150,Thomastown,Victoria,3074,Australia        x
                00200 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00210 ;
5200            00220           ORG     5200H           ;Must be below 8000H
5200 CDC901     00230 START     CALL    01C9H           ;cls
5203 211F52     00240           LD      HL,MSG1         ;pt to info msg
5206 CD6744     00250           CALL    4467H           ;print msg
5209 110000     00260           LD      DE,0000H        ;start of Ram
520C 210080     00270           LD      HL,8000H        ;start of bank2
520F 01FF37     00280           LD      BC,37FFH        ;bytes to move
5212 CD5152     00290           CALL    SWAP            ;xfer ROM back
5215 CD6000     00300           CALL    60H             ;delay
5218 CD6000     00310           CALL    60H             ;delay
521B CD6000     00320           CALL    60H             ;delay
521E C7         00330           RST     00H             ;reboot Newdos
521F 52         00340 MSG1      DEFM    'Restoring Saved ROM image for Reboot of Newdos 80'
     65 73 74 6F 72 69 6E 67 20 53 61 76 65 64 20 52
     4F 4D 20 69 6D 61 67 65 20 66 6F 72 20 52 65 62
     6F 6F 74 20 6F 66 20 4E 65 77 64 6F 73 20 38 30
5250 0D         00350           DEFB    0DH
5251 F3         00360 SWAP      DI                      ;must be done!
5252 3E31       00370           LD      A,31H           ;move bank 2 & open ROM to write
5254 D384       00380           OUT     (84H),A         ;do it now
5256 EDB0       00390           LDIR                    ;move Bank ram to ROM area
5258 3E04       00400           LD      A,04H           ;ROM default mode & 80x24 screen
525A D384       00410           OUT     (84H),A         ;do it now
525C FB         00420           EI
525D C9         00430           RET
5200            00440           END     START
00000 TOTAL ERRORS

MSG1    521F    START   5200    SWAP    5251
```

DATE/ASM

by Tony Domigan

This program will probably not be of much interest to folks living in the U.S. or Canada, but those in other parts of the world may find it of great interest. The program listing is actually a series of patches to the Model III version of NEWDOS/80 version 2, that changes the format of the date (as reported by the system DATE command) to DD/MM/YY. See the remark statements in the program for further information.

```
                00010 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                00020 ;x Patch to allow DD/MM/YY format with Newdos80 v2 x
                00030 ;x for date cmd and julian date conversion         x
                00040 ;x          by Tony Domigan                       x
                00050 ;x  Po Box 150, Thomastown, Vic., 3074, Australia x
                00060 ;x           ph 466-1738                          x
                00070 ;x       (model 3 Version Only)                   x
                00080 ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
4CAA            00090           ORG     4CAAH
                00100 ;
                00110 ; Apply the following bytes to Sys0/sys,9,AD
                00120 ; Skip over the Loader bytes at 93 - 96
                00130 ; Modify other Modules as indicated in Rmks
                00140 ;
4CAA F5         00150 STANDO    PUSH    AF
4CAB C5         00160           PUSH    BC
4CAC D5         00170           PUSH    DE
4CAD E5         00180           PUSH    HL
4CAE E5         00190 CHECK     PUSH    HL
4CAF 23         00200           INC     HL
4CB0 23         00210           INC     HL
4CB1 23         00220           INC     HL
4CB2 E5         00230           PUSH    HL
4CB3 D1         00240           POP     DE
4CB4 0602       00250 SWAP      LD      B,02H
4CB6 E1         00260           POP     HL
4CB7 7E         00270 SLOOP     LD      A,(HL)
4CB8 F5         00280           PUSH    AF
4CB9 1A         00290           LD      A,(DE)
4CBA 77         00300           LD      (HL),A
4CBB F1         00310           POP     AF
4CBC 12         00320           LD      (DE),A
4CBD 13         00330           INC     DE
4CBE 23         00340           INC     HL
4CBF 10F6       00350           DJNZ    SLOOP
4CC1 E1         00360           POP     HL
4CC2 D1         00370           POP     DE
4CC3 C1         00380           POP     BC
4CC4 F1         00390           POP     AF
4CC5 C9         00400           RET
4CC6 CDAA4C     00410 RTNE0     CALL    STANDO  ;SYS0/SYS,12,28 =CD C6 4C
4CC9 C35D4F     00420           JP      4F5DH
4CCC CDAA4C     00430 RTNE1     CALL    STANDO  ;SYS0,11,CF = CD CC 4C
4CCF CD6744     00440 PEXIT     CALL    4467H
4CD2 C9         00450           RET
4CD3 3A4F4D     00460 RTNE2     LD      A,(4D4FH)
4CD6 FE33       00470           CP      33H
4CD8 CCAA4C     00480           CALL    Z,STANDO
4CDB C9         00490           RET
4CDC CD034C     00500 RTNE3     CALL    RTNE2   ;SYS7,0,94 = C3 DC 4C
4CDF 18EE       00510           JR      PEXIT
4CE1 CD034C     00520 RTNE4     CALL    RTNE2   ;SYS7,0,82 = C3 E1 4C
4CE4 F3         00530           DI
4CE5 0603       00540           LD      B,03H
4CE7 C3814D     00550           JP      4D81H
4CEA E5         00560 RTNE5     PUSH    HL      ;BASIC,3,7E = CD EA 4C
4CEB 21875B     00570           LD      HL,5B87H
4CEE CDAA4C     00580           CALL    STANDO
4CF1 E1         00590           POP     HL
4CF2 3E08       00600           LD      A,08H
4CF4 C3C95A     00610           JP      5AC9H
4CF7 CDAA4C     00620 RTNE6     CALL    STANDO  ;BASIC,4,33 = C3 F7 4C 00
4CFA CD475B     00630           CALL    5B47H
4CFD C9         00640           RET
0000            00650           END
00000 TOTAL ERRORS

CHECK   4CAE    PEXIT   4CCF    RTNE0   4CC6    RTNE1   4CCC    RTNE2   4CD3
RTNE3   4CDC    RTNE4   4CE1    RTNE5   4CEA    RTNE6   4CF7    SLOOP   4CB7
STANDO  4CAA    SWAP    4CB4
```

BOOT/BAS

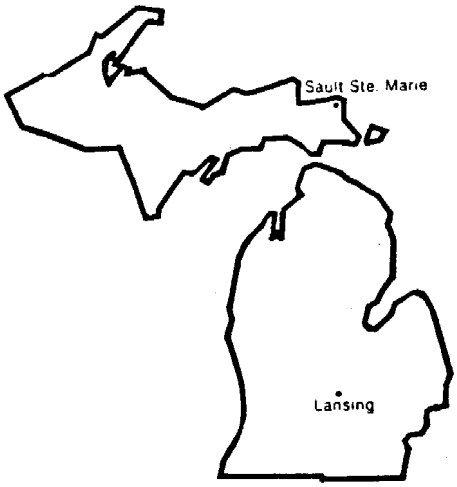
This program, like the disk alarm program in the last issue of NORTHERN BYTES, was sent to us by Paul Fransen who is the secretary of a TRS-80 users group in The Netherlands. This one lets you change the banner on your NEWDOS/80 version 2 system disk (probably on the Model I version only, but I'm not certain about that). Just to be on the safe side, make a backup of any disk you want to try using this program on. Have fun!

```
10 REM ***********************************************
***
20 REM ***  Idea: P Keus, Program: P Fransen (C) 1984   ***
30 REM ***                                              ***
40 REM *** This program let you create your own Boot-head ***
50 REM *** Every diskette its own Boot-head, so you will  ***
60 REM *** immediatly know which diskette your are using  ***
70 REM *** The program is simple, so there is no need for ***
80 REM *** further instructions. Much fun.              ***
90 REM ***********************************************
***
100 REM
110 CLEAR1000:CLS:DEFINTA-Z:DIMX,Y,C1,C2,C3,C4,Z,Z1,Q,R,A$,
B$,C$,A(55,3),Q(13):FORI=0TO13:READQ(I):NEXT:C$=CHR$(34):DA
TA8448,15360,256,1023,6014,2096,14359,12037,14111,30495,2851,-
20104,-4576,201
120 PRINT@384,CHR$(31);"-=: CREATION OF YOUR OWN BOOT-H
EAD FOR NEWDOS80 2.0 :=-

<ARROWS> = move cursor          <F> = frame on/off
<S>      = SET-mode             <R> = RESET-mode
<@>      = Text-mode on/off     <K> = reverse head"
130 PRINT"<P>      = to Printer          <B> = Basic-prog. to disk
<#>      = write BOOT-head      <*> = load BOOT-head
<O>      = restart              <E> = end (boot)"
```

```
140 X=2:Y=3:S=1:C1=128:B$=CHR$(10):GOSUB410
150 A$=INKEY$:IFA$=""THENSET(X,Y):RESET(X,Y):GOTO150ELSE
ONINSTR(CHR$(91)+CHR$(10)+CHR$(8)+CHR$(9)+"SsRrOoFfBbPp@
Kk**Ee",A$)GOTO180,190,200,210,160,160,170,170,140,140,220,22
0,230,230,310,310,320,350,350,370,390,400,400:GOTO150
160 S=1:GOTO150
170 S=0:GOTO150
180 IFY<4THEN150ELSEGOSUB450:Y=Y-1:GOTO150
190 IFY>10THEN150ELSEGOSUB450:Y=Y+1:GOTO150
200 IFX<3THEN150ELSEGOSUB450:X=X-1:GOTO150
210 IFX>110THEN150ELSEGOSUB450:X=X+1:GOTO150
220 GOSUB420:GOTO150
230 GOSUB440:OPEN"O",1,"HEAD/BAS":PRINT#1,10;"CLS:CLEAR
1000:B$=CHR$(10):PRINT"+C$+"<1> screen only
<2> change SYS
<3> end

Make your choice..."+C$:PRINT#1,20;"Y$=INKEY$:IFY$="+C$+C$+
"THEN20ELSEY=VAL(Y$):IFY<1ORY>3THEN20ELSEIFY=3THEN12
0ELSEIFY1THEN30"
240 PRINT#1,25;"OPEN"+C$+"R"+C$+",1,"+C$+"SYS0/SYS"+C$+":F
IELD1,1ASZ1$,4ASDUMMY$,57ASZ2$,1ASY2$,55ASZ3$,1ASY3$,77A
SDUMMY$,59ASZ4$,1ASY4$:GET1,13:A$="+C$+C$:PRINT#1,30;"CL
S:FORZ=1TO55:READA:POKE15360+Z,A:IFY=2THENA$=A$+CHR$(A
)"
250 PRINT#1,40;"NEXT:IFY=2THENA$=A$+STRING$(4,32):LSETZ4
$=A$:LSETY4$=B$:PUT1,13:GET1,14":PRINT#1,50;"READA:POKE1
5425,A:IFY=2THENA$=CHR$(A):LSETZ1$=A$:A$="+C$+C$:PRINT#1
,60;"FORZ=2TO55:READA:POKE15424+Z,A:IFY=2THENA$=A$+CHR
$(A)"
260 PRINT#1,70;"NEXT:IFY=2THENA$=A$+STRING$(2,32):LSETZ2
$=A$:LSETY2$=B$:A$="+C$+C$:PRINT#1,80;"FORZ=1TO55:READA:
POKE15488+Z,A:IFY=2THENA$=A$+CHR$(A)":PRINT#1,90;"NEXT:
IFY=2THENLSETZ3$=A$:LSETY$=B$:PUT1,14:CLOSE"
270 PRINT#1,100;"PRINT@896,"+C$+"Press a key for menu..."+C$:
PRINT#1,110;"A$=INKEY$:IFA$="+C$+C$+"THEN110ELSECLS:GO
TO10":PRINT#1,120;"CLS:CMD"+C$+"BOOT"+C$:R=120
280 FORZ=1TO3:PRINT#1,R+Z*10;"DATA";:FORZ1=1TO55:PRINT#1
,A(Z1,Z-1);:IFZ1=30THENR=R+10:PRINT#1,:PRINT#1,R+Z*10;"DAT
A";ELSEIFZ1<55THENPRINT#1,CHR$(44);
290 NEXTZ1:PRINT#1,:NEXTZ:CLOSE:PRINT@896,"*** <1>=run Ba
sicprogram, <2>=menu ***":A$=""
300 A$=INKEY$:IFA$="1"THENCLS:RUN"HEAD/BAS"ELSEIFA$="
2"THENPRINT@896,CHR$(30);:GOTO150ELSE300
310 GOSUB440:FORZ=0TO2:FORZ1=1TO55:LPRINTA(Z1,Z);:NEXTZ1:
NEXTZ:LPRINT:LPRINT:FORZ=0TO2:FORZ1=1TO55:POKE14312,A(Z
1,Z)+32:NEXTZ1:LPRINT:NEXTZ:GOTO150
320 A$="":A$=INKEY$:IFA$=""THEN320ELSEONINSTR(CHR$(9)+C
HR$(10)+CHR$(91)+CHR$(13)+CHR$(8)+"@",A$)GOTO320,320,320,32
0,340,150
330 PRINT@X/2+INT(Y/3)*64,A$;:X=X+2:GOTO320
340 X=X-2:PRINT@X/2+INT(Y/3)*64," ";:GOTO320
350 IFC1<>32THENGOSUB420
360 Q=0:DEFUSR0=VARPTR(Q(0)):Q=USR0(0):GOTO150
370 GOSUB440:GOSUB470:A$="":FORZ=1TO55:A$=A$+CHR$(A(Z,0))
:NEXT:A$=A$+STRING$(4,32):LSETZ4$=A$:LSETY4$=B$:PUT1,13:G
ET1,14:A$=CHR$(A(1,1)):LSETZ1$=A$:A$="":FORZ=2TO55:A$=A$+C
HR$(A(Z,1)):NEXT:A$=A$+STRING$(2,32):LSETZ2$=A$:LSETY2$=B$
380 A$="":FORZ=1TO55:A$=A$+CHR$(A(Z,2)):NEXT:LSETZ3$=A$:LS
ETY3$=B$:PUT1,14:CLOSE:GOTO150
390 GOSUB470:FORZ=1TO55:A(Z,0)=ASC(MID$(Z4$,Z,1)):POKE15424
+Z,A(Z,0):NEXT:GET1,14:A(1,1)=ASC(Z1$):POKE15489,A(1,1):FORZ=2
TO55:A(Z,1)=ASC(MID$(Z2$,Z-1,1)):POKE15488+Z,A(Z,1):NEXT:FORZ
=1TO55:A(Z,2)=ASC(MID$(Z3$,Z,1)):POKE15552+Z,A(Z,2):NEXT:CLOS
E:GOTO150
400 CLS:CMD"s=boot"
410 FORZ=0TO2:PRINT@65+Z*64,STRING$(55,128);:NEXT
420 IFC1<>176THENC1=176:C2=131:C3=170:C4=149ELSEC1=32:C2=
32:C3=32:C4=32
430 FORZ=15361TO15415:POKEZ,C1:POKEZ+256,C2:NEXTZ:FORZ=1
5424TO15552STEP64:POKEZ,C3:POKEZ+56,C4:NEXT:RETURN
440 FORZ=0TO2:FORZ1=1TO55:A(Z1,Z)=PEEK(15424+Z1+Z*64):PRIN
T@990,A(Z1,Z);:NEXTZ1,Z:PRINT@990,CHR$(30);:RETURN
450 IFSTHENSET(X,Y)ELSERESET(X,Y)
460 RETURN
470 OPEN"R",1,"SYS0/SYS":FIELD1,1ASZ1$,4ASDUMMY$,57ASZ2$,
1ASY2$,55ASZ3$,1ASY3$,77ASDUMMY$,59ASZ4$,1ASY4$:GET1,13:
RETURN
```



THE MAP ABOVE (showing the only two really important cities in the state of Michigan) is being printed to make a point. Lansing (home of THE ALTERNATE SOURCE) and Sault Ste. Marie (home of NORTHERN BYTES) are not near each other! When you send letters, articles, orders, etc. to The Alternate Source or to Northern Bytes, you will experience needless delay if you send them to the wrong address. Here's the quick and simple guide to what goes where:

ALL EDITORIAL MATERIAL, including article submissions and programs donated to Northern Bytes and/or the TAS PUBLIC DOMAIN LIBRARY, "letters to the editor", and similar material should be sent to Northern Bytes in SAULT STE. MARIE. The only exception to this if you are requesting some sort of compensation for your article and/or program, in which case you must communicate directly with Charley Butler at The Alternate Source in LANSING. ANY and ALL material sent t Northern Bytes in SAULT STE. MARIE will t assumed to have been contributed to the public domain!

ALL ORDERS FOR MERCHANDISE must be sent directly to The Alternate Source in LANSING. If you fail to do this, it may delay processing of your order by up to a week. In addition, all requests for information about products (including the TAS PUBLIC DOMAIN LIBRARY) must be sent to the LANSING address.

PROBLEMS WITH SUBSCRIPTIONS (if you are a former Opinion-80 or 80-User Digest subscriber, or have received a FREE six issue subscription for contributing an article to Northern Bytes) should be sent to Northern Bytes at SAULT STE. MARIE, because that's where the mailing list is maintained.

ORDERS FOR BACK ISSUES of Northern Bytes should be sent to The Alternate Source in LANSING, unless you are a TRS-80 User Group and wish to trade copies of back issues of your newsletter for copies of Northern Bytes. The current price for back issues (of Volume 5 only) is $2.00 each.

EXCHANGE NEWSLETTERS and other exchange publicaions absolutely MUST be sent to Northern Bytes in SAULT STE. MARIE!! Sending them to The Alternate Source does not count, although you may send copies to both Northern Bytes and TAS if you like.

One more note: If you send an order to TAS in Lansing, but put in a note to "tell Jack such-and-such", there is a good possibility that the message will never get passed on. Therefore, if you have comment about Northern Bytes, please spring for th extra postage and send them directly to Northern Bytes at SAULT STE. MARIE.

We hope this clears up any confusion you may have had in regard to when to send something to Northern Bytes, and when to send something to TAS.

NEW SOFTWARE!

Bridge Academy

BRIDGE ACADEMY (TM) is a series of three instructional packages teaching the card game Bridge. Books I, II, and III introduce the bidding and playing of the average hand. More than 90 percent of all opening bids are covered in these books.

Bridge is played by four people. This puts the beginner in an awkward position: First, it may be embarassing to play with three seasoned players when one makes so many elementary mistakes; second, the novice makes a mistake, he is corrected and taught a new rule - unfortunately, a similar situation may not come up again for weeks and by that time the rule has been forgotten.

BRIDGE ACADEMY (TM) overcomes these problems. Every new rule comes with many exercises. You practice the new rule until you feel that you have mastered it.

BRIDGE ACADEMY (TM) was designed for the beginner who wants to learn the elements of the game and for the seasoned player who wants to brush up on some rules.

From the very beginning you can play bridge with the most patient partner you can imagine: YOUR COMPUTER. You can bid and play the same hands over and over again, ask for the same type of hand, or ask for new hands.

BOOK I teaches you the fundamental rules of bridge, and one notrump bidding and playing. BOOK II introduces the bidding and playing of one suit. Two responses to such an opening bid (the two-over-one and the jump shift) are practiced in BOOK III, along with some more pointers on playing.

Each BOOK consists of three programs, called Chapter 1, Chapter 2, and Chapter 3 (these are all hybrid programs: a basic program and a machine language program); a Cue Card, and Instructions for the User.

The programs are designed to run on the TRS-80 Model I, Model III(4) (32K disk).

Those who order BEFORE December 15th can receive all three books for just $39.95. After this date, they are available for only $19.95 each.

Macro Typing

MACRO TYPING is a unique program that will help you become a better typist! Currently available for the Models I, III, and 4 (in the native 4 mode), MACRO TYPING can accurately track your typing speed to over 100 words per minute. A special feature allows you to generate "random sentences" for practice drills. An editor is included that will allow you to construct specific drills and exercises. The Model 4 provides both audible and visual feedback when you type a character wrong. This feature requires an auxillarily amplifier on the Models I and III. A special "MACRO MODE" prints the letters to be typed in large graphic letters, perfect for persons with impaired vision. After some practice, you can take 1, 3 and 5 minute timed-writings, just like those in formal typing classes, and find out how fast you REALLY are!

All three versions of MACRO TYPING are included with the package. Support is also included for Model I systems without lower case. MACRO TYPING is only $29.95 complete with instructions.

Please send me the following:

- ☐ BRIDGE ACADEMY BOOK I $19.95
- ☐ BRIDGE ACADEMY BOOK II $19.95
- ☐ BRIDGE ACADEMY BOOK III $19.95

- ☐ ALL THREE FOR JUST $39.95 (Before December 15th)

- ☐ MACRO TYPING $29.95

NAME:
ADDRESS:
ADDRESS:
CITY: STATE: ZIP:
COUNTRY:
Visa/MasterCard #:
Expiration Date:
Michigan residents, please add 4% sales tax.
North America: Please add $3 for shipping and handling.
Foreign: Surface, add 10%; Air, add 20%.
COD add $2.00.

STRUCTURED PROGRAMMING!

TRSDOS and MSDOS users, you don't need a Macintosh and a $250 BASIC package to do structured programming! The Alternate BASIC (ABASIC) eliminates line numbers, but that's only the beginning. ABASIC is an enhancement to your current Microsoft BASIC.

Use your word processor to create libraries of relocatable BASIC code that can be used and reused by many different programs. These routines can be merged with your programs, simply by using a "COPY" command. All ABASIC subroutines will have a label. You simply use one line in your BASIC COPY LABLE from "filename" and the code will be included in the translation to Microsoft BASIC. The library of subroutines with ABASIC include a complete HELP system.

ABASIC code can be written with any word processor. With ABASIC, you can use "structured constructs". What this actually means is that we're adding new BASIC commands to your existing BASIC. Some examples are:

IF ENDIF -- With this command, you can start an IF statement, and follow it with as many lines as is necessary for this condition, until an ENDIF is encountered. Your IF statements are no longer limited to one 255 character line.

DO WHILE and DO UNTIL -- The difference between these two structures is that if a condition doesn't exist, it will never be executed with a DO WHILE loop: it will be executed until it does exist with a DO UNTIL loop. Examples are provided in the 100 page ABASIC manual.

SELECT/CASE -- With this powerful command, ABASIC can selective use and bypass portions of BASIC code. Programming a large set of conditional situations has never been easier.

There's lots more. ABASIC includes a set of utility programs written in ABASIC and their Microsoft translation that you can use immediately. These include CREF, LVAR, XVAR and XLATE. Full descriptions are provided in the documentation.

The Model 4 and MSDOS versions of ABASIC are compiled. The Model I III versions are in BASIC. ABASIC is "written in itself" and can translate itself. The Alternate BASIC, complete with manual, lots of sample code, help files, utilities and extensive documentation is only $69.95. There are lower priced packages, but we urge you to compare, feature for feature. To our knowledge, you won't find more comprehensive support for structured BASIC programming for microcomputers at a better price. Please specify which machine you are running, whether it is MSDOS or TRSDOS and the operating system you generally use. Orders for The Alternate BASIC can be placed with The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan, 48906 or by phoning (517) 482-8270.

NORTHERN BYTES

c o Jack Decker
1804 West 18th Street
Lot # 155
Sault Ste. Marie, Michigan 49783
MCI Mail Address: 102-7413
Telex: 6501027413
(Answerback: 6501027413 MCI)

POSTMASTER: If undeliverable return to:

The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906

To: