NORTHERN BYTES

Volume 5 Number 1

GREETINGS! Welcome to a "spurious" edition of Northern Bytes. Since Microcomputer Users International (the group that originally published this newsletter) appears to be defunct (no meetings or other activity to speak of since June), I will continue to use the name "Northern Bytes" until someone objects. However, this edition will go to TRS-80 user groups only, in exchange for your newsletters, and to certain others (such as former OPINION-80 and 80-USER DIGEST subscribers). I don't plan to do this on a regular monthly basis, but may issue future editions as time permits.

Anyone that does not receive a copy of this newsletter and that wants one can get a sample copy by sending a long self-addressed stamped envelope with 37 cents postage affixed (you may make this known to your club members if you wish). Newsletter editors are free to reprint anything in this issue, provided proper credit is given to the source of the article.

IN SEARCH OF... 80-USER DIGEST - Sorry, but with only 30 paid subscribers, we just can't do it. So, what about those 30 that have already signed up? Well, here's a deal just for them (please pass this along to those in your club to whom this applies): If anyone that has already entered a subscription to 80-User Digest will drop a line to The Alternate Source and specifically ask for it, they can have a copy of my new book "TRS-80 ROM ROUTINES DOCUMENTED" in place of their subscription. Since the book normally sells for $19.95, that's a deal! If the subscriber decides he doesn't want the book after he sees it, he can sell it at one of your club meetings, probably for more than what he paid for his subscription. Of course, if the subscriber really insists, he can have a refund or apply what he paid on his subscription to any other TAS product. I apologize for getting everyone's hopes up on the new magazine, but after all, you can't publish a magazine for only thirty subscribers. We aren't the only ones - a new magazine called Peripherals, Etc. (your club may have received a promotional package) ceased publication after only one issue - I suspect they didn't get enough subscribers to make it worthwhile, either.

PUBLIC DOMAIN SOFTWARE AND BULLETIN BOARD SYSTEMS (EDITOR'S SOAPBOX REVISITED) - Don't you ever get a little tired of hearing your Apple-owner or Commodore-owner friends talk about the large amount of public domain software available through the International Apple Core or the Toronto Pet Users Group? Don't you ever wonder why there isn't more GOOD public domain software for the TRS-80?

Of course, it's out there, on the many Bulletin Board Systems run for and by TRS-80 owners. Finding it, though, can be worse than looking for the proverbial needle in the haystack, if you can get to it at all!

First of all, many BBS's have a download section that contains maybe 10% good programs (hopefully yours has a higher percentage) and 90% "filler". By "filler" I mean such things as Tic-Tac-Toe games and word processors written in BASIC - the sort of thing that a beginning computer user might find interesting, but hardly a useful program for the experienced user. But even that 10% may be impossible to get at.

Consider this: You hear of a great BBS in some distant city that has a good download section, so you give it a call. After you sign on, you sit through five minutes of "local announcements", and THEN are informed that because you don't have a password, you can't do anything on the system (except apply for a password, which will be MAILED to you in a few days, and on some systems you have to pay to get one). You may well just sign off in disgust at about that point, unless you have stock in Ma Bell.

The fact that some of these SYSOPs originally got their public domain software from an "open" board somewhere (one where THEY were NOT hassled in like manner) doesn't seem to bother them a bit. So, the number of boards that do NOT require a password to access grows smaller day by day, and so does the TRS-80 user's access to public domain software.

There are two things that I'd like to see happen to combat this trend. For one thing, I'd like to start a list of TRS-80 Bulletin Board Systems that allow free access to their download section on the FIRST call, without requiring a password or otherwise frustrating the caller. If you still have such an animal in your area, or know of any, please drop me a line. I'll publish the list in a future issue. If you don't do it now, you'll forget, so please grab a postcard and jot those numbers down!

Second - would you be willing to dump some GOOD public domain software programs onto a disk (including documentation files, if any)? Would you be willing to send that disk to me? All right, assuming that you'd be willing to do that, would you be willing to include enough postage for me to mail the disk back to you? Why should you do that? Because I would then take the best of the programs I receive - the "cream of the crop" - and copy it back onto your disk and send it back to you. Since your disk would then be full of public domain programs, you could use them in your newsletter or put them on your local BBS or just pass out copies at your meeting. In the meantime, I might get enough material to start a good public domain library of TRS-80 programs, which could be distributed through The Alternate Source or through some other means. You'd at least get your disk back with some interesting programs on it!

What do you think? Wouldn't you like to see more GOOD public domain software for the TRS-80? Or do you really enjoy hearing about the vast amounts of easily-accessible public domain software available to Apple and Commodore owners and to CP/M users, while knowing that you can't easily get at most of the available TRS-80 public-domain software?

On a related subject, does anyone know the whereabouts of Eric Greene, author of the GREENE MACHINE, a public domain Bulletin Board System (that's right, as I understand it the BBS software is in the public domain!). I hope to have a copy of the software soon but would like to locate the author, who seems to have dropped out of sight. His original BBS line has been disconnected, and even the SYSOPs of the other GREENE MACHINEs around the country don't seem to know where he is.

You're quite correct Fenwick, your program could replace three out of four of this firm's Vice Presidents.

- Reprinted from THE TORPET

DOCUMENTATION FOR MOD4BASC/BAS by Lee Becker - This program, which I call 'MOD4BASC/BAS', will not guarantee that your program will run properly on your MOD 4, but will at least put it in a syntactically correct form. It will first go through all the lines and insert spaces where they are required by the Model 4 BASIC. It will then research the lines for any improperly formed 'IF THEN' statements. In the Model 4 BASIC you are allowed to write an 'IF THEN' statement with only the 'THEN', but you can't omit the 'THEN' part of the statement and replace it with another BASIC keyword. This program will simply place a 'THEN' in the proper location to allow the statement to work.

The program to be converted must be ASCII format. Use your 'A' command when saving the program from BASIC. If it is not already in ASCII, you will have to reload it into your BASIC and resave it with the 'A' command.

POSSIBLE ERRORS: Two of these errors are unstoppable. However, the last one may not be unstoppable and if anyone has any suggestions, please contact me (Lee Becker) through the CompuNet BBS (517) 339-3367.

1) If the person writing the program has packed the lines to the maximum length, then when this program tries to insert some spaces the line will get too long. Normally you would get a BASIC error message saying that line is 'TOO LONG'. The program will override this error in order to keep processing the rest of the file. A message stating which line # is too long and that it is 'SKIPPING LINE' is shown. It is not totally skipping the line, but once the line has become too long then it ceases to bother with it. However, most of the line may be done. To solve this problem you would have to return to the BASIC used to create the program and break the line. Or, you could break the line in the Model 4 BASIC, but be careful about any line references. They must be correct, and do not do a 'RENUM' of the program or all references to lines will probably be wrong. The Model 4 BASIC does not know how to change them until they are syntactically correct in structure.

2) If there are any BASIC keywords that are not used by the Model 4 BASIC, they will not be dealt with at all. An example is the SET(X,Y) command. The Model 4 does not support the SET command. However, I am contemplating writing a BASIC function that would translate the SET statements to work on the Model 4 as intended on other BASICs.

3) The last problem deals with the 'IF THEN' statements. This problem kind of has two parts to it. First if there is an 'IF THEN' statement in the form of: IF X<0 X=10 no 'THEN' will appear where it should. This is because there is no BASIC keyword for the program to find and use for location. The program should simply leave the line alone and you will get a 'SYNTAX ERROR'. Since it is a syntax error, the program being executed will stop right at that line number and you will have to edit it then. Second, if there is a 'IF THEN' similiar to the one above but beyond it there is a BASIC keyword (example: IF X<0 X=10:GOTO 10) you will find an error causing 'THEN' just before the GOTO (in this case). You will have to delete the 'THEN' and put it in it's proper place. This is the problem that I don't know if something could be done about it. If anyone has a suggestion please let Lee Becker know through a message on CompuNet.

Once the program has converted all the lines it will show you a listing of the program that it has converted. This is simply a listing of the array that the converted program lines are in. Do not try to run this yet. Simply keep pressing the ENTER key until the final prompt for the file's name comes up. Then simply give it the name and disk drive number (if you want) that you want it saved under. The program will save it as an ASCII file, so when you run it under BASIC resave it without the 'A' command and it will then be recompressed. Once the program has finished saving the converted program it will simply end. To reuse, you will have to type 'RUN' again.

Please report any errors or ideas to me (Lee Becker) by leaving a message on CompuNet (517-339-3367). Thank you for using my program.

(Downloaded from CompuNet BBS, Lansing, Michigan (517) 339-3367).

```
1 '              Model 4 BASIC syntax converter
2 '
3 '              This program will convert other BASIC
4 '              programs to be syntactically correct
5 '              for the Model 4 BASIC.
6 '
7 '              Written by:  Lee Becker
8 '              Phone no. :  517-349-7684
9 '
10 CLS:CLEAR 20000:DIM A$(1000):ON ERROR GOTO 900
20 PRINT@ 580,"What is the file's name to convert?";:PRINT@ 5
90,;:LINE INPUT; C$
25 PRINT CHR$(15)
30 OPEN "I",1,C$
40 KOUNT=1
50 IF EOF(1) THEN 80
60 LINE INPUT# 1,A$(KOUNT)
70 KOUNT=KOUNT+1:GOTO 50
80 CLOSE 1
90 '
100 '
110 CLS
120 PRINT@ 420,"There are";KOUNT;"total lines in the file.":
PRINT TAB(20)"It will take approximately";:PRINT USING"##.#"
;KOUNT*3.1/60;:PRINT" minutes.":PRINT @ 670,"STARTING NOW"
130 FOR CNT=1 TO KOUNT:Y=1:Z=1:W=1
135 V=INSTR(1,A$(CNT),"DATA"):IF V>0 AND V<8 THEN A$(CNT)=MI
D$(A$(CNT),1,V+3)+" "+MID$(A$(CNT),V+4):GOTO 300
140 Z=INSTR(Z,A$(CNT),"'"):IF Z>0 AND Z<8 THEN 300
150 IF Z=0 THEN Z=500 ELSE T=INSTR(Z,A$(CNT),CHR$(34)):IF T<
>0 THEN Z=T:GOTO 140
160 W=INSTR(W,A$(CNT),"REM"):IF W>0 AND W<8 THEN 300
170 IF W=0 THEN W=500 ELSE T=INSTR(W,A$(CNT),CHR$(34)):IF T<
>0 THEN W=T:GOTO 160
180 FOR COT=1 TO 47:Q=1:READ B$
190 IF Q<>0 THEN Q=INSTR(Q,A$(CNT),CHR$(34))
200 X=INSTR(Y,A$(CNT),B$)
210 IF X>Z OR Z>W THEN 290
220 IF Q<>0 AND X>Q THEN 540
230 IF X<>0 THEN IF B$<>"THEN" AND B$<>"ELSE" AND B$<>"TO" A
ND B$<>"OR" AND B$<>"AND" AND B$<>"STEP" THEN 250
240 IF X<>0 THEN E$=MID$(A$(CNT),X-2,2):F$=MID$(A$(CNT),X-3,
3):IF E$<>":F" AND E$<>" F" AND E$<>"F R" AND E$<>":R" AND E$
<>"GO" AND F$<>"RES" AND F$<>"ERR" AND MID$(A$(CNT),X-4,4)<>
"REST" THEN GOSUB 640:GOTO 190
250 IF X=0 THEN 290 ELSE IF  B$="AND" OR B$="OR" OR B$="TO"
OR B$="STEP" THEN Y=X+LEN(B$):GOTO 280
260 IF MID$(A$(CNT),X-1,1)=" " OR MID$(A$(CNT),X-1,1)=":" TH
EN GOSUB 610:GOTO 190
270 IF MID$(A$(CNT),X-1,1)="L" AND B$="PRINT" THEN Y=X+LEN(B
$):GOTO 280
280 GOSUB 640:GOTO 190
290 Y=1:NEXT COT
300 PRINT@ 827,"Line";CNT;"is now done.":RESTORE:NEXT CNT
310 '
320 '
330 CLS:PRINT@ 418,"I am now changing all IF THEN statements
":PRINT@ 670,"STARTING NOW"
340 FOR CNT=1 TO KOUNT:X=1:F=1
350 X=INSTR(X,A$(CNT),"IF"):IF X=0 THEN 430 ELSE IF F<>0 THE
N 670
360 Y=INSTR(X,A$(CNT),"THEN"):IF Y=0 THEN 390
370 Z=INSTR(X+2,A$(CNT),"IF"):IF Y<Z THEN X=Z:GOTO 360
380 IF Z=0 THEN 430
390 W=999:RESTORE 800:FOR S=1 TO 35:READ B$
400 V=INSTR(X+2,A$(CNT),B$):IF V<W AND V>X THEN W=V
410 NEXT S:IF W=999 THEN 430
420 A$(CNT)=MID$(A$(CNT),1,W-1)+"THEN "+MID$(A$(CNT),W):X=X+
2:GOTO 350
430 PRINT@ 824,"Line";CNT;"is finished now.":NEXT CNT
440 SOUND 0,0:SOUND 7,1:SOUND 0,0:CLS:E=1:FOR X=1 TO KOUNT S
TEP 10:FOR W=1 TO 10:PRINT A$(E):E=E+1:NEXT W:PRINT:INPUT"Hi
t ENTER for rest of list";U:PRINT:NEXT X
450 '
460 '
470 CLS:PRINT CHR$(14):PRINT@ 410,"Input file's name and d
ve to save converted file under":PRINT@ 507,;:LINE INPUT C$
480 OPEN "O",1,C$
490 FOR X=1 TO KOUNT:PRINT# 1,A$(X):NEXT X
500 CLOSE 1
```

```
510 END
520 '
530 '
540 Q=INSTR(Q+1,A$(CNT),CHR$(34)):IF Q=0 THEN GOTO 290
550 IF X<Q THEN Y=Q:Q=INSTR(Q+1,A$(CNT),CHR$(34)):GOTO 190
560 Q=INSTR(Q+1,A$(CNT),CHR$(34)):IF Q=0 THEN 230
570 IF X>Q THEN 540
580 GOTO 230
590 '
600 '
610 A$(CNT)=MID$(A$(CNT),1,X-1+LEN(B$))+" "+MID$(A$(CNT),X+L
EN(B$)):Y=X+LEN(B$):RETURN
620 '
630 '
640 A$(CNT)=MID$(A$(CNT),1,X-1)+" "+MID$(A$(CNT),X,LEN(B$))+
" "+MID$(A$(CNT),X+LEN(B$)):Y=X+LEN(B$)+1:RETURN
650 '
660 '
670 F=INSTR(F,A$(CNT),CHR$(34)):IF F>X OR F=0 THEN 360
680 G=INSTR(F+1,A$(CNT),CHR$(34)):IF G=0 OR G=LEN(A$(CNT)) T
HEN 430
690 IF G<X THEN F=G+1:GOTO 670
700 X=G:GOTO 350
710 '
720 '
730 '
740 '
750 END
760 DATA THEN,ELSE,TO,FOR,OR,AND,PRINT,?,#,GOTO,GOSUB,NEXT,S
TEP,IF,DIM
770 DATA INPUT,LET,ON,READ,USING,CLEAR,DATA,DEFDBL,DEFINT,DE
FSNG,RETURN
780 DATA ERROR,GET,LSET,RSET,MERGE,FIELD,OPEN,POKE,RESTORE,S
WAP,VARPTR
790 DATA WHILE,CALL,CHAIN,COMMON,LINE,LPRINT,MERGE,OUT,ERASE
800 DATA ELSE,PRINT,GOTO,GOSUB,FOR,IF,DIM,INPUT,LET,ON,READ,
CLEAR,DATA,ERASE
810 DATA NEXT,ERROR,GET,LSET,RSET,MERGE,FIELD,OPEN,POKE,REST
ORE,SWAP,VARPTR
820 DATA RETURN,CLS,CALL,CHAIN,COMMON,LINE,LPRINT,MERGE,OUT
900 IF ERR=15 THEN PRINT CHR$(16);:PRINT @ 1221," Line";CNT;
"is too long.  SKIPPING IT ";:PRINT CHR$(17);:RESUME NEXT
905 IF ERR=5 THEN RESUME 290
910 ERROR ERR
```

PRINTER PROBLEM SOLVER - This is for anyone that has ever wanted to feed single sheets of paper through a printer, especially one that has a tractor feed mechanism only (no fricton feed). It's called the Paper Tractor and sells for $11.95, and carries your single sheets through your printer. It can be used with any adjustable tractor-feed printer (or even one with non-adjustable tractors if the printer handles standard 9-1/2" wide forms). It is made of heavy gauge plastic and holds letter or legal size paper, or any 8-1/2" wide form (some checks and invoice forms, etc.). Even if your printer has friction feed, this product would be useful in assuring that the form is straight within the printer. Note that it's mainly intended for use with one form at a time, so it wouldn't be useful for production runs, but for occasional printouts on your letterhead, individual checkwriting, etc., this may be just what you need. For info write The Paper Tractor, One South Fairview, Goleta, California 93117.

MODEL 4 "KBMOD/ASM" PROGRAM by Jack Decker - The comments in this program pretty well explain its purpose, so I won't bore you with further details here. I will say that the program could be adapted to work on the Model I or III, but since those models do not have the control key, it would be more difficult to make this program work properly without lousing up the function of any of the other keys. Anyway, here 'tis:

```
00100 ;Stand-alone control-arrow key keyboard enhancement
00110 ;for the Model 4 (for use in the Model III mode)
00120
00130 ;Program written by:
00140
00150 ;                Jack Decker
00160 ;                1804 West 18th Street  Lot # 155
00170 ;                Sault Ste. Marie, Michigan  49783
00180 ;                (906) 632-3248
00190
00200 ;Copyright 1983 by Jack Decker
00210
00220 ;Permission is granted to freely use and distribute this
00230 ;program for non-commercial purposes only, provided that
00240 ;these credit lines are left intact.  If you like this
00250 ;program, you may want to consider obtaining the full
00260 ;24 X 80 video driver package for use with the Model III
00270 ;mode of the TRS-80 Model 4.  Now you can run many of
00280 ;your Model III programs without conversion, and still
00290 ;enjoy the benefits of the 24 X 80 screen display. The
00300 ;program also copies the BASIC ROM into RAM and then
00310 ;patches it so that functions such as PRINT @, TABs,
00320 ;SET/RESET/POINT, the screen print routine, etc. will
00330 ;work properly with the 24 X 80 display.  Fully
00340 ;commented Editor-Assembler source code is included on
00350 ;the disk!  For further information regarding the 24 X 80
00360 ;video driver package, please contact:
00370
00380 ;                The Alternate Source
00390 ;                704 North Pennsylvania Avenue
00400 ;                Lansing, Michigan  48906
00410 ;                (517) 482-8270
00420
00430 ;USE OF THIS PROGRAM:  This program is fully
00440 ;self-relocating.  To initialize, from DOS READY type:
00450 ;       KBMOD            or            KBMOD F
00460 ;The second form (with the "F" argument) kicks in the
00470 ;fast clock speed of the Mod 4.  The program will
00480 ;relocate and initialize itself.  Thereafter, you may
00490 ;use the CTRL key along with any of the four arrow keys
00500 ;to move the cursor anywhere on the video display,
00510 ;without erasing the characters passed over by the
00520 ;cursor.  If you hold down the CTRL, SHIFT, and right
00530 ;arrow keys at the same time, the character at the
00540 ;current cursor position will be returned to the calling
00550 ;routine, as if it had been entered from the keyboard.
00560 ;This allows you to enter without re-typing anything
00570 ;on the video display.  For example, after you have
00580 ;displayed a disk directory, you can run the cursor over
00590 ;any /CMD filename using the CTRL/SHIFT/right arrow, and
00600 ;the DOS will think you typed in that filename and will
00610 ;proceed to run the program!  This routine has MANY uses,
00620 ;so experiment with it.  You may find it convenient to
00630 ;use an AUTO command to automatically run this program
00640 ;each time you boot your system.
00650
00660 ;WARNINGS:  There are two things that can happen with
00670 ;this routine in use that might cause you some concern.
00680 ;The first is that if you use the CTRL-arrow function to
00690 ;pass the cursor over a zero byte on the display, it will
00700 ;turn off the cursor!  The first time this happens, you
00710 ;may think that something terrible has happened, but
00720 ;pressing BREAK or ENTER will usually bring back your
00730 ;cursor.  The problem is with the video driver, but you
00740 ;will find that under normal conditions, there will NEVER
00750 ;be a zero byte on the screen, since the video driver
00760 ;won't put one there.  However, a zero byte can be POKEd
00770 ;to the screen, but that isn't done very often.  Another
00780 ;warning is that if you use the CTRL/SHIFT/right arrow
00790 ;keys to input characters from the screen, and you pass
00800 ;over a graphics character in the range 0C0H - 0FFH, the
00810 ;space compression/special characters flag will
00820 ;automatically be set to special characters, to avoid
00830 ;printing unwanted tabs.  Similarly, the CTRL/SHIFT/right
00840 ;arrow routine will not pass over or input a character in
00850 ;the 00H - 1FH range, since it would be interpreted as
00860 ;a control character by the calling routine, with
00870 ;unpredictable results.
00880
00890 ;This routine depends on the calling routine to echo the
00900 ;character.  Therefore, it cannot be used with INKEY type
00910 ;routines that do not echo characters to the video
00920 ;display.
00930
00940 ;If you write regarding this program, please be sure to
00950 ;enclose a self-addressed, stamped envelope (U.S. or
00960 ;Canadian postage O.K.) if you wish a reply.  No replies
00970 ;will be sent within Canada or the U.S.A. unless the SASE
00980 ;is included with your correspondence.
00990
```

```
                00
                01010 ;*****  INITIALIZATION & RELOCATION SEGMENT BEGINS  *****
                01020
7000            01030         ORG     7000H
7000 /E         01040 INTLZE  LD      A,(HL)          ;Get argument if any
7001 E6DF       01050         AND     0DFH            ;Mask out lowercase
7003 FE46       01060         CP      'F'             ;Set fast speed?
7005 200A       01070         JR      NZ,SLOW         ;If fast not specified
7007 3A1042     01080         LD      A,(4210H)       ;Bit mask for port ECH
700A F640       01090         OR      40H             ;Set bit 6
700C D3EC       01100         OUT     (0ECH),A        ;Output to port ECH
700E 321042     01110         LD      (4210H),A       ;Re-save bit mask
7011 2A1640     01120 SLOW    LD      HL,(4016H)      ;Get current KB dvr addr
7014 226670     01130         LD      (START+1),HL    ;Patch into program
7017 AF         01140         XOR     A               ;Clear Carry flag
7018 112D40     01150         LD      DE,402DH        ;RET address if DOS
701B E1         01160         POP     HL              ;Get actual RET address
701C E5         01170         PUSH    HL              ;Re-save it as well
701D ED52       01180         SBC     HL,DE           ;Result zero if under DOS
701F 2011       01190         JR      NZ,NOTDOS       ;If not under DOS
7021 211144     01200         LD      HL,4411H        ;Mod 3 DOS top-of-memory
7024 223770     01210         LD      (MEMSIZ+2),HL   ;Self-modify program
7027 224070     01220         LD      (STRMEM+1),HL   ;  memory pointers & put
702A 225970     01230         LD      (GETMEM+1),HL   ;  DOS addr in exit of
702D ED536370   01240         LD      (IEXIT+1),DE    ;  initialization routine
7031 3D         01250         DEC     A               ;Flag as under DOS
7032 21A970     01260 NOTDOS  LD      HL,END          ;End of unrelocated pgrm
7035 ED5BB140   01270 MEMSIZ  LD      DE,(40B1H)      ;End of unprotected mem
7039 014500     01280         LD      BC,END-OFFST+1  ;Length of main program
703C EDB8       01290         LDDR                    ;Move the program
703E EB         01300         EX      DE,HL           ;New memory size in HL
703F 22B140     01310 STRMEM  LD      (40B1H),HL      ;Save new memory size
7042 3C         01320         INC     A               ;A=0 if under DOS
7043 2816       01330         JR      Z,SKPCLR        ;Skip CLEAR if under DOS
7045 310670     01340         LD      SP,INTLZE+6     ;Don't crash moved code
7048 21BB41     01350         LD      HL,41BBH        ;HL=1st byte DOS vector
704B 7E         01360         LD      A,(HL)          ;Save DOS vector in
704C 08         01370         EX      AF,AF'          ; alternate A register
704D 3EC9       01380         LD      A,0C9H          ;Plug DOS vector with
704F 77         01390         LD      (HL),A          ;  a RET instruction
7050 113200     01400         LD      DE,32H          ;"CLEAR 50"
7053 CD831E     01410         CALL    1E83H           ;& reset other pointers
7056 08         01420         EX      AF,AF'          ;Get original DOS vector
7057 77         01430         LD      (HL),A          ;Restore original vector
7058 2AB140     01440 GETMEM  LD      HL,(40B1H)      ;Get new prgrm location
705B 23         01450 SKPCLR  INC     HL              ;HL=Entry point address
705C 221640     01460         LD      (4016H),HL      ;Put into keyboard DCB
705F 01181A     01470         LD      BC,1A18H        ;For tape BASIC re-entry
7062 C3AE19     01480 IEXIT   JP      19AEH           ;Re-enter BASIC (or DOS)
                01490
                01500 ;*****  MAIN PROGRAM (WILL BE RELOCATED) BEGINS  *****
                01510
7065            01520 OFFST   EQU     $               ;Used by relocator
                01530
7065 CD0000     01540 START   CALL    0               ;Modified to call KB dvr
7068 C8         01550         RET     Z               ;Return if no character
7069 47         01560         LD      B,A             ;Save char in A
706A 3A8038     01570         LD      A,(3880H)       ;Get CTRL key row
706D E604       01580         AND     4               ;Mask out other keys
706F 2002       01590         JR      NZ,CTRL         ;Go if control char
7071 78         01600 KBEXIT  LD      A,B             ;Get original character
7072 C9         01610         RET                     ;  & return
7073 3A3C40     01620 CTRL    LD      A,(403CH)       ;Arrow key row storage
7076 E678       01630         AND     78H             ;Arrow key pressed?
7078 28F7       01640         JR      Z,KBEXIT        ;Go if not CTRL+arrow
707A 78         01650         LD      A,B             ;Get original character
707B FE0B       01660         CP      0BH             ;Greater than 0AH?
707D 300A       01670         JR      NC,HICTRL       ;Go if greater than 0AH
707F FE08       01680         CP      8               ;Less than 08H?
7081 D8         01690         RET     C               ;Return it so
7082 F610       01700         OR      10H             ;Make cursor ctrl char
7084 CD3300     01710 MVCRSR  CALL    0033H           ;Output char to video
7087 AF         01720 NULRET  XOR     A               ;Make null (zero) byte
7088 C9         01730         RET                     ;Return with no char
7089 E6BF       01740 HICTRL  AND     0BFH            ;Make 5BH same as 1BH
708B FE1B       01750         CP      1BH             ;Was char CTRL+uparrow?
708D 28F5       01760         JR      Z,MVCRSR        ;Send to video if so
708F FE19       01770 GETCHR  CP      19H             ;CTRL+SHIFT+rt arrow?
7091 20F4       01780         JR      NZ,NULRET       ;Return null chr if not
7093 3A2240     01790         LD      A,(4022H)       ;Get char under cursor
7096 B7         01800         OR      A               ;Check for valid char
7097 2004       01810         JR      NZ,GOTCHR       ;Go if valid character
7099 2A2040     01820         LD      HL,(4020H)      ;Get current cursor pos
709C 7E         01830         LD      A,(HL)          ;Get char at cursor
709D FE20       01840 GOTCHR  CP      20H             ;Control character?
709F 38E6       01850         JR      C,NULRET        ;Return null if control
70A1 FEC0       01860         CP      0C0H            ;"Special" character?
70A3 D8         01870         RET     C               ;Output if regular chr
70A4 17         01880         RLA                     ;Set bit 1 of A register
70A5 322440     01890         LD      (4024H),A       ;Set spec. chars flag
70A8 1F         01900         RRA                     ;Restore original char
70A9 C9         01910         RET
                01920
70A9            01930 END     EQU     $-1             ;Used by relocator
                01940
7000            01950         END     INTLZE
00000 TOTAL ERRORS

CTRL    7073    END     70A9    GETCHR  708F    GETMEM  7058    GOTCHR  709D
HICTRL  7089    IEXIT   7062    INTLZE  7000    KBEXIT  7071    MEMSIZ  7035
MVCRSR  7084    NOTDOS  7032    NULRET  7087    OFFST   7065    SKPCLR  705B
SLOW    7011    START   7065    STRMEM  703F
```

REROUTE/ASM - A PROGRAM FOR NEWDOS/80 V.2 USERS by Jack Decker - Have you ever wanted to reroute the text that normally would go to your printer to a disk file instead, so that you can read it into your word processing program or otherwise process it further? If so, you may have thought you'd have to switch to DOSPLUS or LDOS to get that capability. Not so, just assemble the program below. Then use it as follows:

When you begin a session where you will need to be able to route printer output to a disk file, simply type REROUTE from DOS READY (this assumes you named the object file REROUTE/CMD). Then go about your business. When you are ready to send printer output to a disk file, simply press the DFG keys to go into MINI-NEWDOS, then type:

*PD filespec

You must type the asterisk! PD is a mnemonic I have chosen that stands for "Printer to Disk." "filespec" is of course any valid filename, with or without an extension. Then enter the MDRET command to return to your program. From BASIC you can also use the statement CMD "*PD filespec" to accomplish the same thing without going into MINI-NEWDOS. After that, any printer output that goes

through the standard printer Device Control Block will instead be routed to the disk file. Once you've sent everything you want to the disk file, press DF6 again and type:

*PR

This is very important, since it closes the disk file and restores normal output to the printer. If you forget to do this, your file will NOT be properly closed and you will be unable to recover it (without a great deal of effort, anyway). Once again, you can use the equivalent BASIC statement CMD "*PR".

REROUTE is self-relocating, and automatically resets the high memory pointer and patches the "*PD" and "*PR" commands into the DOS command chain. This ability to define "asterisk" commands and place them into the DOS command library is a rather unique feature of NEWDOS/80, so even if you don't need this program, you may want to study it to see how these commands are added.

Hope you find this program useful. If you make any improvements to it, I'd appreciate receiving a copy of your efforts. This program can be considered public domain, so feel free to use it as you wish!

```
                00100 ;*****  INITIALIZATION & RELOCATION SEGMENT BEGINS  *****
                00110
7000            00120          ORG     7000H
7000 3A5400     00130 INTLZE   LD      A,(54H)         ;Check Model (I or III)
7003 3D         00140          DEC     A               ;Z if Mod I
7004 2809       00150          JR      Z,RELOC         ;Go if Mod I
7006 211144     00160          LD      HL,4411H        ;Mod III HIMEM pointer
7009 221470     00170          LD      (GETMEM+2),HL   ;Replace 4049H in
700C 223A70     00180          LD      (STRMEM+1),HL   ;  initialization routine
700F 210B72     00190 RELOC    LD      HL,END          ;End of unrelocated pgrm
7012 ED5B4940   00200 GETMEM   LD      DE,(4049H)      ;End of unprotected mem
7016 01A101     00210          LD      BC,END-OFFSET+1 ;Length of main program
7019 EDB8       00220          LDDR                    ;Move the program
701B D5         00230          PUSH    DE              ;Save new memory size
701C 13         00240          INC     DE              ;DE=Start relocated prgrm
701D 215370     00250          LD      HL,TABLE        ;Get start of reloc table
7020 7E         00260 NXTLOC   LD      A,(HL)          ;Get first address byte
7021 47         00270          LD      B,A             ;Save first byte in B
7022 23         00280          INC     HL              ;Point @ 2nd address byte
7023 B6         00290          OR      (HL)            ;Do both bytes equal 0?
7024 2019       00300          JR      NZ,REPLCE       ;No - replace address
7026 D5         00310          PUSH    DE              ;Save entry address
7027 EB         00320          EX      DE,HL           ;Put *PR entry addr in HL
7028 CD6144     00330          CALL    4461H           ;NEWDOS/80 *NAME routine
702B 2009       00340          JR      NZ,ERR          ;Display error if any
702D D1         00350          POP     DE              ;Restore entry address
702E 212100     00360          LD      HL,PD-OFFSET    ;Offset for PD routine
7031 19         00370          ADD     HL,DE           ;HL = *PD entry address
7032 CD6144     00380          CALL    4461H           ;NEWDOS/80 *NAME routine
7035 E1         00390          POP     HL              ;Restore new memory size
7036 C20944     00400 ERR      JP      NZ,4409H        ;Display error if any
7039 224940     00410 STRMEM   LD      (4049H),HL      ;Save new memory size
703C C32D40     00420          JP      402DH           ;Back to DOS
703F E5         00430 REPLCE   PUSH    HL              ;Save reloc table pointer
7040 66         00440          LD      H,(HL)          ;H=MSB of addr from table
7041 68         00450          LD      L,B             ;L=LSB of addr from table
7042 19         00460          ADD     HL,DE           ;Get address of label
7043 4E         00470          LD      C,(HL)          ;Get byte displacement
7044 23         00480          INC     HL              ;  from program and
7045 46         00490          LD      B,(HL)          ;  put in BC
7046 2B         00500          DEC     HL              ;HL=address of label
7047 D5         00510          PUSH    DE              ;Save new START address
7048 EB         00520          EX      DE,HL           ;HL=new START address
7049 09         00530          ADD     HL,BC           ;HL=new address for label
704A EB         00540          EX      DE,HL           ;  now put in DE
704B 73         00550          LD      (HL),E          ;New address calculated -
704C 23         00560          INC     HL              ;  now write it into
704D 72         00570          LD      (HL),D          ;  label (address field)
704E D1         00580          POP     DE              ;Restore new START addr
704F E1         00590          POP     HL              ;Restore reloc table pntr
7050 23         00600          INC     HL              ;Bump pointer to nxt addr
7051 18CD       00610          JR      NXTLOC          ;Process next table entry
                00620
                00630 ;*****  RELOCATION TABLE  *****
                00640 ;Must be located prior to "START" label of main program!
                00650
7053 0D00       00660 TABLE    DEFW    RELO1+1-OFFSET
7055 1500       00670          DEFW    RELO2+1-OFFSET
7057 1D00       00680          DEFW    RELO3+1-OFFSET
7059 2F00       00690          DEFW    RELO4+1-OFFSET
705B 3200       00700          DEFW    RELO5+1-OFFSET
705D 3A00       00710          DEFW    RELO6+1-OFFSET
705F 4200       00720          DEFW    RELO7+1-OFFSET
7061 4F00       00730          DEFW    RELO8+1-OFFSET
7063 5200       00740          DEFW    RELO9+1-OFFSET
7065 5800       00750          DEFW    RELO10+1-OFFSET
7067 5E00       00760          DEFW    OUTBYT+1-OFFSET
7069 0000       00770          DEFW    0
                00780
                00790
                00800 ;*****  MAIN PROGRAM (WILL BE RELOCATED) BEGINS  *****
                00810
706B            00820 OFFSET   EQU     $               ;Used by relocator
                00830
                00840 ;*PR (Restore printer output to printer)
706B 0000       00850          DEFW    0               ;Four bytes required
706D 0000       00860          DEFW    0               ;  for use by DOS
706F 50         00870          DEFM    'PR     '       ;8-byte routine name
7070 5220202020202020
7077 3A7E00     00880 RELO1    LD      A,(FLAG-OFFSET) ;Check if file open
707A B7         00890          OR      A               ;NZ if file is open
707B 3E26       00900          LD      A,26H           ;"File Not Open" error
707D 2851       00910          JR      Z,ERROR         ;If file not open
707F 118100     00920 RELO2    LD      DE,FCB-OFFSET   ;Put DE @ DCB
7082 CD2844     00930          CALL    4428H           ;Close file
7085 2049       00940          JR      NZ,ERROR        ;Display error if any
7087 2A7F00     00950 RELO3    LD      HL,(PDVR-OFFSET);Get original DCB addr
708A 1833       00960          JR      CONT            ;Store DCB & flag
                00970
                00980 ;*PD filespec (Route printer output to disk file)
708C 0000       00990 PD       DEFW    0               ;Four bytes required
708E 0000       01000          DEFW    0               ;  for use by DOS
7090 50         01010          DEFM    'PD     '       ;8-byte routine name
7091 4420202020202020
7098 EB         01020          EX      DE,HL           ;Save filename pointer
7099 216800     01030 RELO4    LD      HL,MSG-OFFSET   ;Point to message
709C 3A7E00     01040 RELO5    LD      A,(FLAG-OFFSET) ;Check if file open
709F B7         01050          OR      A               ;NZ if file already open
70A0 C26744     01060          JP      NZ,4467H        ;Display message & return
70A3 EB         01070          EX      DE,HL           ;Restore filename pointer
70A4 118100     01080 RELO6    LD      DE,FCB-OFFSET   ;Put DE @ DCB
70A7 CD1C44     01090          CALL    441CH           ;Extract filespec
70AA 2024       01100          JR      NZ,ERROR        ;Display error if any
```

```
70AC 21A100   01110 RELO7   LD      HL,BUFFER-OFFSET;Point HL to buffer
70AF 47       01120         LD      B,A             ;LRL=256
70B0 CD2044   01130         CALL    4420H           ;Open file
70B3 201B     01140         JR      NZ,ERROR        ;Display error if any
70B5 3D       01150         DEC     A               ;A = 0FFH
70B6 2A2640   01160         LD      HL,(4026H)      ;Get current DCB value
70B9 227F00   01170 RELO8   LD      (PDVR-OFFSET),HL;Store it for close
70BC 215D00   01180 RELO9   LD      HL,OUTBYT-OFFSET;Get output routine addr
70BF 222640   01190 CONT    LD      (4026H),HL      ;Put in printer driver
70C2 327E00   01200 RELO10  LD      (FLAG-OFFSET),A ;Flag file is open
70C5 C32D40   01210         JP      402DH           ;Back to DOS READY
70C8 118100   01220 OUTBYT  LD      DE,FCB-OFFSET   ;Point DE to DCB
70CB 79       01230         LD      A,C             ;Get byte to output
70CC CD1B00   01240         CALL    001BH           ;Output byte to disk
70CF C8       01250         RET     Z               ;If no error
70D0 C30944   01260 ERROR   JP      4409H           ;Display error if any
70D3 2A       01270 MSG     DEFM    '*PD File already open'
70D4 504420466696C6520616C726561647920F70656E
70E8 0D       01280         DEFB    0DH
              01290
              01300 ;Following locations used by both *PD and *PR routines
70E9 00       01310 FLAG    DEFB    0               ;File opened/closed flag
70EA          01320 PDVR    DEFS    2               ;Printer dvr addr storage
70EC          01330 FCB     DEFS    20H             ;File control block
710C          01340 BUFFER  DEFS    100H            ;Output buffer
              01350
720B          01360 END     EQU     $-1             ;Used by relocator
              01370
7000          01380         END     INTLZE
00000 TOTAL ERRORS
```

```
BUFFER  710C    CONT    70BF    END     720B    ERR     7036    ERROR   70D0
FCB     70EC    FLAG    70E9    GETMEM  7012    INTLZE  7000    MSG     70D3
NXTLOC  7020    OFFSET  706B    OUTBYT  70C8    PD      708C    PDVR    70EA
RELO1   7077    RELO10  70C2    RELO2   707F    RELO3   7087    RELO4   7099
RELO5   709C    RELO6   70A4    RELO7   70AC    RELO8   70B9    RELO9   70BC
RELOC   700F    REPLCE  703F    STRMEM  7039    TABLE   7053
```

* * * * * * Feature * * * * * *

*

* The Trials, Horrors, and *

* Somewhat Guarded Feelings *

* of Glee Gotten From Trying *

* to Back Up Super Utility Plus *

*

* by Mark Gladstone, 320-5383 *

*

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

(Reprinted from the TCUG Newsletter)

A large brouhaha started when some fellow had the audacity to write in to 80 Micro complaining that the proliferation of protected diskettes had reached its apex - or nadir - with Super Utility Plus. Here we have a program that can - among many other important utilities - back up many "protected" programs. This program was PROTECTED! This fellow felt that such was hypocrisy, and offered to provide a method to back up SU+ in the event that Kim Watt met his somewhat-deserved demise. That 80 Micro refused to print his name really frosted my liver, and thus began my tilting at windmills. However, I have experienced some success - with a great deal of help from my friends - and a measure of frustration that I would like to share with you.

There has been much written concerning the backup of SU+, and few panaceas. There is even a software house that will sell you a program to back up SU+, but there are a few (not so many that you should notice) catches. What I write here is the result of a union of many ideas from many unsung sources, and several levels of refinement. It is not a panacea in itself, and is presented mainly to deal with what I consider a great hypocrisy in "protecting" SU+ from dissemination to other than honest owners, and not to violate the legitimate rights of authors to their due return.

The problems involved in backing up SU+ fall into several categories. The first problem is finding the most appropriate format. The most logical and useful one might be a command file of a loadable program, tailored to your DOS, that RESIDES on your DOS. Then you can copy it, move it, or destroy it to your heart's content. The second problem results from the program being DOS-independent, having its own use of lower RAM in the 4000H-5500H range that is in conflict with every DOS. It must be moved somewhere before dumping to the DOS and restored after load time. But where? In addition, once it is restored, what is the program's real entry point? To where is program control transferred in order to begin execution of SU+? And what about the idiosyncracies of the different DOSes?

The March/April 1983 issue of the TCUG Newsletter contained a reprint of an article by Roger Whitehead of the Central Alabama Microcomputer Society on how to make SU+ into a CMD file. His technique uses NEWDOS/80 2.0 to dump from 5200H. What about DOSPLUS people, who can't dump below 5700H? Or TRSDOS people, who can't dump below 6000H? Or Gobbledy-DOS people who can't dump whatever? Unfortunately, there is no common technique that will satisfy the requirements of all DOSes, for those and other reasons. In addition, you will notice that his procedure transfers control to the memory mode, rather than to the master menu. To further complicate the matter, versions for the Model I and Model III have different entry points and different buffer areas.

The problem of the entry point is easiest to deal with. Your manual tells you how to obtain routine addresses for your version and model. The address for the master menu is referenced symbolically by MENU. This entry point is documented at 4905H for the Model I and 4963H for the Model III in Version 2.2z. Entering a jump to this routine before the master menu is presented IMMEDIATELY BEFORE DUMPING will transfer control to the master menu when the program is called from your DOS. I have not found the entry point for the flashing screen immediately before the master menu on boot-up, but it can be found by the entry point list. Powersoft was not helpful in that area, but they did divulge that Version 3.0 uses the same MENU symbol for the master menu.

The problem of where to move lower memory is a complicated one. On the surface, there is free area above the buffer areas GATBUFF, BUFFER, and FBUFF. The example given in the TCUG Newsletter uses E400H as a buffer to hold the low-RAM information, and the small program to move the low-RAM information back at execution time. That location is well above those three buffer areas mentioned above, so we are golden when we move the lower-RAM information to E400H, right? WRONG! If you have a 2.2(P) version of SU+, it will probably bomb after loading when using E400H as a holding buffer. The (P) versions of SU+ contain patches, and my guess is that the E400H area interferes with some of the patches. If you are using Version 2.2z Release 21, you should not have any trouble using E400H. I was able to get the patched version I was testing to run using E500H as the buffer area on the Model III, but I could not get that same buffer area of E500H to work on the Model I with the 2.2 (P) version I was testing. In any event, E500H will work as a buffer area for both 2.2z and 2.2z (P) on the Model III; if you have a Model I and 2.2z Release 21, E500H should work for you. Don't forget to <SHIFT-BREAK> to the master menu immediately before booting in your DOS.

The next issue is how much to move, now that we have decided (?) where to move it. How much you need to move is a function of the lowest address to which your DOS will allow a dump. Here lieth the rub! For example, TRSDOS 1.3 will not allow a dump below 6000H. Simple arithmetic will show that there is not enough buffer to hold 2000H bytes. A partial solution is to patch the following to a copy of TRSDOS 1.3 to be used for dumping only, and copy the resulting command file to another copy:

```
PATCH *6 (ADD=5798,FIND=FF5F,
  CHG=0000)
```

I don't know how far down you can dump with this patch - probably not enough - but you TRSDOS 1.3 people can give it a whirl. If it's not low enough, you can try two or more dumps, breaking the moved area into several pieces, and use the SU+ sectors-to-memory and memory-to-sectors to merge the results. You need to change the load addresses on the disk, which is beyond the scope of this discussion. Frankly, this is a hard way to do it, but it can be done. If you have NEWDOS/80 Version 2.0, you should have success with Roger Whitehead's technique, remembering to change the last two bytes of his patch to the appropriate MENU entry point and to <SHIFT-BREAK> to the master menu immediately before booting NEWDOS. I have successfully used E400H as a buffer with DOSPLUS 3.4 and SU+ 2.2z Release 21 on the Model III, moving 4000H-56FFH to the buffer and dumping from 5700H to FB11H (includes the patch to move the 1700H bytes back to lower RAM), using FB00H as the transfer address.

In the interest of uniformity, I will present the following LDOS modifications to the prodecure in the TCUG Newsletter that I have used to dump both 2.2z (P) and 2.2z Release 21 to a command file using the Model III. If you have a Model I and Release 21 this should work, although I haven't tried it. However, if you have access to LDOS and a Model III, the procedure is proven, and you can use the resulting SU+ to transfer the command file to your DOS afterward. As a modification to the procedure in the Newsletter, you do not need to hard-configure SU+ according to your desires; a soft-configure will work equally as well. That is how you can dump to LDOS, configured to (for example) DOSPLUS, and end up with a DOSPLUS-configured version on DOSPLUS itself. Warning: the NEWDOS procedure in the Newsletter as modified by my suggestions may cause a problem when transferred to DOSPLUS due to the low-RAM dump, but the LDOS procedure will not. Also, please note that Model I goes to Model I and Model III goes to Model III, and never the twain (speech impediment?) shall meet.

First, your LDOS must be free of upper memory filters, etc., and have sufficient space. You will never get into trouble using a first-generation backup of a pristine copy. Then, boot in your SU+ and soft-configure it to your desires on your target DOS. Enter the MEMORY MOVE option and enter the following:

```
4000H,56FFH,E500H
```

In the DISPLAY MEMORY option, enter the Modify mode and enter the following starting at FC00H:

```
F3 21 00 E5 11 00 40 01 00
17 ED B0 31 20 FC C3 yy xx
```

where xx yy represents the appropriate MENU entry point for your model and version. Note the minor addition to the Newsletter procedure to set the stack pointer to FC20H, well above the patch and buffers.

Once you have entered the above hex code, immediately <SHIFT-BREAK> to the master menu and, without disturbing anything, boot LDOS, enter a date, and dump memory to the LDOS disk with the following:

```
DUMP SUPL22/CMD (START=X'5700',
  END=X'FC11',TRA=X'FC00')
```

If you then need to transfer the program to another DOS, enter SUPL22, soft-configure to the target DOS on the target disk drive, insert your target diskette, and copy away!

There is another option to the above madness. Kim Watt, responding to the brouhaha with the generosity of his own little black heart, has offered a procedure himself to back up your very own SU+ to a command file. All you needed to do was to send in a self-addressed business envelope stamped with two stamps and a note containing the serial number of your SU+ to Powersoft, and you would receive a Kim Watt Special. I tried this, just for chuckles.

Kim Watt's procedure involves an extensive zap to the master diskette. After the zap is applied and the diskette re-booted, the program will stop with the message "Boot Your DOS." At that point, you boot your DOS and dump as instructed. However, the dump instruction given is for LDOS; it will NOT work with DOSPLUS or TRSDOS! I was able to apply this procedure to LDOS. After then calling in the SU+ program from the LDOS disk, I tried it out. The resulting command file seemed to work until I tried a backup. I was trying to do a standard backup of a data diskette. When SU+ responded with the "Disk contains data - use anyway" prompt, I replied "Y". The program then went absolutely bananas, made a mess of the screen, and then caused a re-boot. So CAVEAT PROGRAMMER!

(This is the article from the March/April TCUG Newsletter that was referenced in the above article):

```
* * * * * *  Feature  * * * * * *
*                                 *
*        Super Utility +          *
*      As a Command File          *
*                                 *
*       by Roger Whitehead        *
*  (Reprinted from READY, the     *
*   newsletter of the Central     *
* Alabama Microcomputer Society)  *
*                                 *
* * * * * * * * * * * * * * * * * *
```

The following will show you how to get Super Utility +, Version 2.2, on a disk as a CMD file. First, hard-configure SU+ according to your desires. Then, with SU+ active, go to the SELECT MEMORY UTILITY and choose the MEMORY MOVE option. In response to the prompt, enter "4000H,51FFH,E400H." Press the BREAK key and select the DISPLAY MEMORY option, responding with F600H to the prompt. Press "M" to enter the Modify mode, and enter the following hex codes if you have a Model I:

```
F3 21 00 E4 11 00 40  01 00 12 ED B0
C3 0F 4C
```

If you have a Model III, the last three hex codes should be C3 6C 4C.

Without disturbing anything, remove the SU+ disk and insert a NEWDOS/80 Version 2.0 system disk and push Reset. Then dump memory to a data disk with at least 34 grans of free space using the command:

```
DUMP,SUPL22/CMD,5200H,F610H,F600H
```

This will save SU+ to disk as the command file SUPL22/CMD.

When activated, SU+ will come up in the memory mode, but pressing the BREAK key will get you back to the main menu.

JUMP TO 0000H TO REBOOT? Ah, but if you have a Model I with a double density board, you'll find that you don't get reset to single density. Try using a HALT instruction instead - in the TRS-80, this causes a NMI (Non Maskable Interrupt), which is the same thing that happens when you push the RESET button. So, in the TRS-80 a HALT instruction is the software equivalent of pushing RESET! This info is excerpted from a letter by Glenn L. Bennett of the Cabrillo Computer Club, which appeared in the Voice of the '80 Newsletter.

COLOR COMPUTER AND MODEL I/III COMPATIBILITY by Clayton Tavernier (NYBBLERS SOFTWARE CHAIRMAN) is reprinted from NYBBLER:

While the "Coconuts", the new Color Computer Committee, were meeting for the first time, there were a few of us who were playing with the Model III that someone had thoughtfully brought. Well, one thing led to another and someone suggested that we try and read one of the CoCo disks in the Model III. After a little piddleing with the PDRIVE (we were using NEWDOS/80), the following setup was arrived at:

```
PDRIVE 0,1,TI=AM,TD=E,TC=40,SPT=18,TSR=3,GPL=6,DDSL=17,DDGA=2
```

To read a CoCo disk in a Model I change TI=AM to TI=C. Now if you want to copy one CoCo disk to another (still using NEWDOS/80), try:

```
COPY,0=35,1=35,,NFMT,BDU,SPDN=x,DPDN=x
```

with x being the PDRIVE number of the above setup. Now this copy only works if both disks are formatted. I believe we tried to format a CoCo disk but weren't able to make it readable to the Color Computer (heavy sigh!). Anyway, experiment and have fun and our thanks to Maury Goff (the Coconuts chairman) for tolerating us in such a nice manner.

(EDITOR'S NOTE: For those of you that don't have NEWDOS/80, an item in the GEMS newsletter states that Computer Shack, 1681 Eason, Pontiac, Michigan 48054 has a program called CIII that permits you to copy on your Model I or III to or from a Color Computer disk. The price is $24.85)

A SEVEN-SIDED TRS-80 MODEL I - The circuit below is from the TBUG (TRS-80 Baltimore Users Group) newsletter. It provides the ability to use four drives, up to three of which may be double sided, on the Model I. Radio Shack cut a corner and used the side select line for drive 3 originally, thus, software that supports double sided drives can only access three drives total (drives 0, 1, and 2). The circuit below allows a single sided drive to be used as drive three, even though double sided drives are used elsewhere in the system.

The logic behind this is that if any of drives 0, 1, or 2 are selected, then the drive 3 line is considered to be the side select line. However, if the drive 3 line is selected while none of the other drives are also selected, then it is assumed that we really do want drive three.

As is the case in any Model I system using double sided drives, the stock Radio Shack drive cable cannot be used, since it has contacts within the drive connectors removed (in a pinch, you can use a Shack cable by simply turning ALL of the connectors over, including the one at the Expansion Interface and all of the ones at the drives). However, on the connector used for drive three, substitute the output of pin 4 of the 74LS02 (see the diagram) for the wire that normally connects to pin 10 of the connector. All of the other connectors are wired normally.

No warranties are given on this project - the principle is what is being expressed, this is not a construction article. However, it is presently running on Jerry Slacks' BBS, according to the TBUG article. Now you can put one of your old single-sided drives back to work!

NOTE: IN is from the Expansion Interface (disk cable connector pin numbers shown). OUT is to pin 10 of the drive 3 disk cable connector. Do NOT extend pin 10 from the Expansion Interface through to drive 3, use the output from this circuit instead. Also, I believe that drive three will have to be internally configured as drive zero with the circuit shown below, although I could be wrong about that.

UPGRADE YOUR COLOR COMPUTER - This information is from the Silicon Valley Color Computer Club Newsletter:

For those of you with TRS-80 Color Computers, the following parts, available from the National Parts Division (817) 870-5662 comprise the RS disk interface and will allow you to add your own drives:

AX-9060	interface board	$107.77
ART-3878	shield	1.00
AZ-6839	plastic cover	2.43
AW-2769	two drive cable	18.34

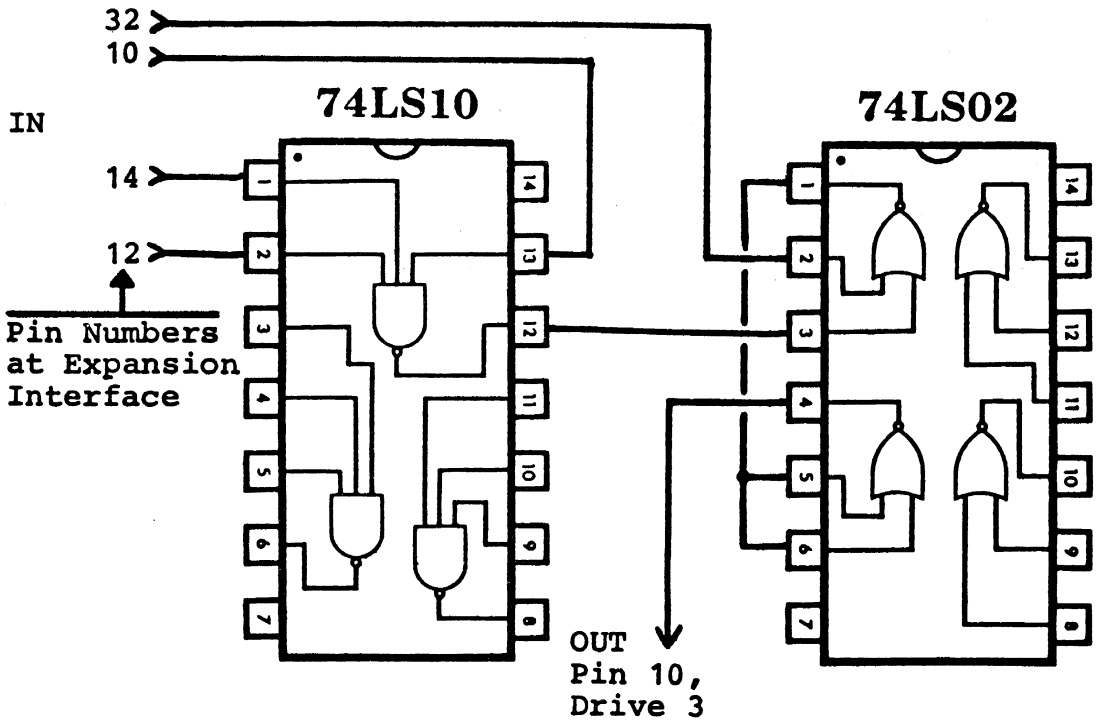
Manuals can be ordered as well:

MS-260-3022	Disk drive service	$7.35
MT-260-3022	Disk interface service	1.49

And you can even order a TDP-100 case if you like white better than the Shack Grey:

AZ-6858	top case (fits all)	$11.77
AZ-6859	bottom case	8.58

(bottom case fits only Rev. NC - alais A, F, or ET)



A Quick Z80 Table Search Routine

by
Roxton Baker Box 8272, APO San Francisco 96555

(Reprinted from the OCTUG Newsletter)

I was recently involved in the design of a Z80-based communications interface board. One of its functions was to translate the standard ASCII codes for alphanumeric characters into the nonstandard code used by another computer. It also had to perform the same translation in reverse, and it had to service several such computers simultaneously via timesharing. Because of the random relationship between the two codes to be translated, the most reasonable approach was to use look-up tables, in which the given character (one byte) would be searched for and from which the equivalent decode character (also one byte) would be read out.

Considering the necessary data rate and the number of code translations that would on the average be required, it was clear that I needed some method faster than a sequential search of the translation tables. I decided to use a binary search technique, because it would be both relatively simple and sufficiently fast. All that it required was to arrange in ascending order the input codes of each table, and to come up with a fast, reliable algorithm for the actual search.

The binary search technique is very familiar to us: you start looking for your code in the middle of the table. If the code you find there is greater than the one you want, you go back halfway towards the beginning and try again. Otherwise you go half again further towards the end of the table and try there. Based on the relative value of the next code found compared to the one you seek, you either jump ahead or back by one-half the amount of your previous jump. You keep this up until you either find the code you wanted or determine that it is not in the table. In the case where you do find the code sought, you will then according to some indexing scheme determine its decode. The simplest thing is to just have the decode character for each code be stored adjacent to that code.

An example of this will make it clear. Suppose we have the following eight input codes:

3E 40 43 44 4B 4D 50 51

and the equivalent decodes into which they must translate:

42 59 41 2A 36 35 58 47

Note that the input codes have been arranged in ascending order. We can put these input codes and their output decodes in a table of pairs as follows:

3E42 4059 4341 442A 4B36 4D35 5058 5147

If we are given the code 43 to translate, we could just start at the beginning of the table and examine the first byte, and every other one thereafter, until we either find a match (which in this case would occur on the third test) or we reach the end of the table. That would be a sequential search for the input code, and for a short table like this it would be the best way to go. At worst we would have to make eight actual character comparisons. When the match is found, of course, we would obtain as the decode character the very next byte in the table - in this case 41.

For a larger table, the worst-case sequential search may take too long. Then we could use a binary search. With the same input code of 43, we would now first go to the middle of the table and check there. In this case there is no exact middle pair in the table; we might equally well choose either the fourth (44) or the fifth (4B) entry to examine. In such a situation lets say we will always take the lower of the two choices.

Thus we compare against the fourth entry. The 44 is greater than our 43, so we know we have gone too far into the table. Therefore we jump back half the distance to the beginning of the table. Again there is no exact middle entry, so we take the lower of the two choices (40) and compare to it. Our input code is greater than this, so we must reverse direction again and go halfway further into the table towards where we just came from (towards the 44 code, that is). This lands us exactly on the 43 code. We detect a match and read off 41 as the decode. The advantage of this binary search technique is that although each comparison takes longer because of the jump calculation that must be made, the worst-case number of comparisons that can be required is (for a large table) much less than for a sequential search. A table of 63 pairs can always be searched in six comparisons, and a table of 64 pairs in seven. In general, a table with anything less than 2**N (meaning two to the Nth power) pairs can be searched in a maximum of N comparisons. If all table input codes are equally likely to be sought, then the average number of comparisons needed for a table of 2**N pairs will be about N-1. A 64-pair table would on the average require only five comparisons.

So it is not hard to decide when to use a binary search, or to understand how it works. All that's left is to code it up. In doing this we would like to achieve both speed and reliability. That is, we want an efficient routine that will never hang up in a loop or miss an entry. In Vol. 3 of Donald Knuth's monumental work "The Art of Computer Programming" there are several such algorithms, very rigorously stated. I took one of them (Algorithm U, pg. 411) and spent some time trying to implement it in concise Z80 code. The result is the search subroutine shown in the assembler listing below. There are some genuine subtleties to the coding, just as there are in the algorithm. These arise in the way that the jump distance is calculated and applied each time. The program comments relate directly to the terminology used in Knuth's book, where the procedure is analyzed. It is not hard to follow what is happening, once you know that ",N/2," means "the largest integer less than or equal to N/2" and that "'N/2'" means "the smallest integer greater than or equal to N/2". Even so, if you have access to this or any of Knuth's books, you should take the chance to read them. They are fascinating, awe-inspiring, and just plain wonderful.

I did make a few assumptions in this code that you should be aware of. In order to keep things at the byte (rather than the word) level, the maximum number of pairs allowed in the table is 127. And a decode of 00 is not allowed, although this could easily be relaxed by finding a different way to set NZ in the code at LOOKF. Also, the table must be headed with a byte that states the number of entries in the table. In the example table below, I just stated this value explicitly. However, you may be able to have your assembler automatically calculate this value. The technique for doing this would vary with each assembler.

To use this LOOKUP subroutine, load HL with the address of the first byte in the table (the one that gives the number of pairs), and load A with the code that you want to translate. On return, you will either have the decode in A and NZ set, or 00 in A and Z set. The program comments explain this. Specifically, if you wanted to use the example table to attempt to translate the code 4D, you could write:

```
        LD      A,4DH
        LD      HL,EXMTBL
        CALL    LOOKUP
        JP      Z,NOBOOD
        JP      FOUND
```

Finally, although I think this routine is nicely done and worth recounting, I DON'T claim that it cannot be shortened, sped-up, or otherwise bettered. For one thing, it could use JRs instead of JPs. If you have an improvement, write it up!

```
00010 ; :::::::::::::::::::::::::::::::::::::::::::::::::::::::
00020 ; Uniform Binary Search algorithm from Knuth V3, pg. 411.
00030 ; Z80 code version by Roxton Baker  4/81
00040 ;
00050 ; Searches a translation table of pairs of byte entries.
00060 ; First byte of table must be number of pair entries in
00070 ; it.  Each entry consists of one byte of input code
00080 ; followed by the byte of output code into which it is
00090 ; to be translated.  The table must have the bytes of
00100 ; input code in ascending order.
00110 ;
00120 ; Enter with the address of the table in HL, and the
00130 ; byte (character) to be translated in A.  If this
00140 ; byte is found as an input byte in the table, its
00150 ; decode byte (character) will be returned in A, with
00160 ; NZ set.  Otherwise 00 is returned in A, with Z set.
00170 ;
00180 ; Temporary storage locations required are one byte at
00190 ; LKVAL and two bytes at TABLM1, as shown.
00200 ;
00210 LOOKUP  EQU     $
00220         LD      DE,00
00230         LD      (LKVAL),A       ;Save searched key K
00240         LD      B,(HL)          ;N=# pairs in table
00250         DEC     HL              ;Form and save Start of
00260         LD      (TABLM1),HL     ;  Table minus 1.
00270         SRL     B               ;Form ,N/2,
00280         LD      C,B             ;m gets ,N/2,
00290         JP      NC,LOOKA        ;i gets 'N/2'
00300         INC     B
00310 ;
00320 LOOKA   LD      HL,(TABLM1)
00330         LD      E,B             ;i
00340         SLA     E               ;DE gets 2i for i<128
00350         ADD     HL,DE           ;Point to trial key Ki
00360         LD      A,(LKVAL)       ;Compare K to Ki
00370         CP      (HL)
00380         JP      Z,LOOKF         ;Go if K = Ki (found)
00390         LD      A,C             ;A gets m
00400         JP      P,LOOKC         ;Go if K > Ki
00410         OR      A               ;Here when K < Ki
00420         JP      Z,LOOKE         ;Quit if m=0
00430         SRL     A               ;Form  ,m/2,
00440         LD      C,A             ;New m gets ,m/2,
00450         CPL
00460         JP      C,LOOKB         ;Form -'m/2' in A
00470         INC     A
00480 ;
00490 LOOKB   ADD     A,B             ;Form i-'m/2' in A
00500         LD      B,A             ;i gets i-'m/2'
00510         JP      LOOKA
00520 ;
00530 LOOKC   OR      A               ;Here if K > Ki
00540         JP      Z,LOOKE         ;Quit if m=0
00550         SRL     A               ;Form ,m/2,
00560         LD      C,A             ;New m gets ,m/2,
00570         JP      NC,LOOKD        ;Form 'm/2' in A
00580         INC     A
00590 ;
00600 LOOKD   ADD     A,B
00610         LD      B,A             ;i gets i+'m/2'
00620         JP      LOOKA
00630 ;
00640 LOOKE   XOR     A               ;Can't find.  Set Z
00650         LD      A,(LKVAL)       ;Retrieve orig value
00660         RET
00670 ;
00680 LOOKF   INC     HL              ;Get decode of K
00690         LD      A,(HL)          ;Return it in A
00700         OR      A               ;Set NZ
00710         RET
00720 ;
00730 LKVAL   DEFB    0
00740 TABLM1  DEFW    00
00750 ;
00760 ; :::::::::::::::::::::::::::::::::::::::::::::::::::::::
00770 ; An example translation table to be searched by the
00780 ; Uniform Binary Search routine.  Code 3E should be
00790 ; translated into 42, etc.  Note bytes must be reversed
00800 ; to state them as DEFWs in this way.
00810 ;
00820 EXMTBL  EQU     $
00830         DEFB    8               ;# pairs in table.
00840         DEFW    423EH
00850         DEFW    5940H
00860         DEFW    4143H
00870         DEFW    2A44H
00880         DEFW    364BH
00890         DEFW    354DH
00900         DEFW    5850H
00910         DEFW    4751H
00920 ; :::::::::::::::::::::::::::::::::::::::::::::::::::::::
```


MISCELLANEOUS HINTS FROM ALL OVER:

Need a free-standing printer stand? I have found that the larger sized milk crates (11"x13"x19") make an excellent little stand by putting it up on end. The crate itself has the paper inside, with the printer sitting on top. Very handy height, and it can be easily moved from one place to another. I have also found that the smaller sized milk crates are ideal for the storage of most computer magazines. (From GEMS, newsletter of the Greensboro Eighty Micro Society).

The latest Model 4's are now coming with the new Tandy designed and built Video power board, instead of the old Mod II/III RCA video board. This new board is a much cleaner, state of the art design using a number of IC's instead of all transistors, has more controls including a working focus control and a fantastic resolution. You can actually see all the dots in a horizontal line with this board. Also, Radio Shack has recently announced to their Service Departments through an I/O Tech Bulletin, that they can now install Model 12/16 green CRT's in Model II, III, 4 and DT-1 units. Cost of the CRT to the customer is $77.00 + $30 installation and you get your old tube back. (From the Marin County TRS-80 Users Group Newsletter).

If you want a 128K Model 4, call Radio Shack National Parts (817) 870-5600 and order a MX-5725 chip (part of 26-1122 128K PLA) and get 8 200 nsec 4164 RAM chips. Install the RAM in the 8 empty sockets and the PLA chip in the socket in the center lower portion of the CPU board that has a DIP jumper with 4 shunts in it. Insure you get the dots for pin 1 lined up properly. The PLA costs $29.95, the RAM goes for $50 to $70, check the current ads. R/S wants $150 plus labor for the same upgrade. You can then set up a 64K RAM disk or a 32K RAM disk and a 32K printer buffer with the additional 64K. Sorry, you can't use it in the Model III mode as yet, but software shouldn't be too long in coming. (From Data Important to Members Everywhere - DIME newsletter).

THE ALTERNATE SOURCE

Presents:

More Fine Products for your TRS-80!

ALE -- a Z80 assembler and full screen editor. A most unique assembler for the TRS-80! *ALE* includes support for most popular EDTASM filetypes you are likely to encounter. Dozens of unique capabilities not found in other assemblers, plus most "standard" commands. *ALE* supports source files longer than memory. You can also generate editable files containing both source and object code, perfect for magazine articles and documented source listings. *ALE* is only $49.95 complete with a 150 page manual.

VIDEO4 is a special screen driver for the Model 4 that enables you to use existing Model III programs (and disk operating systems) while taking advantage of the superior features of the Model 4. VIDEO4 tomtes the Model III ROM into RAM and includes patches to several "buas" that have slipped into the ROM. An optional feature enables the Model 4 high-speed modification. Complete source code is provided on the master disk to assist you with writing your own custom screen driver. Note: VIDEO4 will not work with programs that do not honor the standard video device control block, such as Scripsit and VisiCalc. Limited compatibility with LDOS and Newdos/80; most compatible with TRSDOS 1.3, DosPlus and MultiDos. VIDEO4 is $24.95. Included FREE with VIDEO4 is a keyboard modification that adds screen editing similar to certain 6502 machines!

TRS-80 ROM Routines Documented. Jack Decker has spent years collecting and studying various versions of the TRS-80 ROMs that have appeared in various machines, including the TRZ-80, the System80, the PMC-80 and the SEVEN known versions of the Radio Shack ROM. This book is NOT a disassembly. Jack goes beyond one-line comments and gives you the "big picture". He shows you the background and the whys and wherefores of the ROM subroutines. He offers tips and suggestions, and he warns you of the pitfalls that can drive you bonkers when you use a subroutine knowing only enough to be dangerous. TRS-80 ROM Routines Documented is $19.95.

EDX -- a full screen editor for your TRS-80. EDX provides full-screen editing capabilities previously unavailable on the TRS-80. Get to any portion of your text quickly. Support is provided for BIG files. Especially helpful for preparing files for downloading and uploading when using your modem (many word processors use long lines that cause host systems to hang). EDX can learn your editing keystrokes to make many boring jobs easy. Dozens of excellent features! Best of all, the price: Only $29.95. Model I and III: $39.95 for Model 4.

The Alternate Forth -- A superior and full implementation of Fig Forth for the TRS-80 that includes all standard words through Forth '79. TASForth includes editors, an assembler, a decompiler, a PDRIVE utility (Mod III only), virtual memory and real time clock support, a FORTHDOS operating system that enables you to use familar DOS commands for manipulating your FORTH diskettes and much more. A fantastic buy at $79.95!

Basic Disk I/O Faster and Better - from IJG. Volume II (originally intended to be two separate volumes!) from Lewis Rosenfelder, author of BASIC FASTER and BETTER. This volume includes over 400 pages of important information for both beginning and professional programmers, including file searching, indexing and accessing. Random file techinques have never been made simpler. Many sample demonstration programs are included. $29.95.

How To Do It On The TRS-80. This new book from IJG and Bill Barden (the TRS-80 author that was providing us valuable information when hardly anyone knew anything!) is the ideal reference guide for miscellaneous TRS-80 information. How To Do It On The TRS-80 presents a unique approach to organizing TRS-80 information. First the book has no page numbers. Second, it does have a 32-page index and cross-references to that index printed as thumb-tabs on appropriate pages. This system allows you to quickly find the information you need. Nothing is worse than a programming project where you can't find the information you need. Some of the hundreds of subjects (chosen at random):

/JCL, what is it?
ASCII files, what they are
Assembly language printer driver (I and III)
BASIC, error trapping
Changing attributes of a disk file
Control codes
DEF FN, BASIC command
Delimiters, sequential disk files

There is also information for part-time hardware hackers such as "How to remove IC chips", "How to solder", etc. Describing every routine in How To Do It On The TRS-80 would be either several pages or a real strain on your eyes! How To Do It On The TRS-80 should be on every beginner's and every reference book shelf. Special information is also included for the COCO and Model 100.

These fine products are only a sample of the many programs we provide especially for your Model I, III and 4. Unless you request otherwise, we will take the liberty of adding your name to our mailing list so you will learn of new products as they are available. We accept Visa, Master Card, COD and written P.O.'s from most major schools and government organizations. We also provide service for your TRS-80. Call for details. If coupon is missing, mail to: TAS, 704 N. Pennsylvania Avenue, Lansing, MI 48906. Phone: (517) 482-8270.

Please send the following items:

To:
Address:

Total enclosed: $

Send coupon to: The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan, 48906. Phone orders: (517) 482-8270. Please add $3 to your total order for shipping and handling.

A RARE OPPORTUNITY!

Many folks have requested back issues of The Alternate Source Programmer's Journal, but we have been sold out of many of the back issues for quite some time - almost. What we do have left is twenty-five complete sets of The Alternate Source. That's right, complete sets - all eighteen issues of TAS! We will sell one set of issues one through eighteen for $49.95 plus $3.00 shipping and handling. This is once in a lifetime chance to own what are already collectors items. Read the early articles of such notables as Dennis Kitsz, Roger Fuller, and Jesse Bob Overholt. If you belong to a TRS-80 user group, be sure to let your club librarian know about this deal. And hurry, because these twenty-five sets are all we have. Honest! To reserve your set, telephone TAS at (517) 482-8270 today!

NEW PRODUCTS FROM MODULAR SOFTWARE ASSOCIATES

It had to happen - someone had to come out with programs that serve the same purpose as Prosoft's "Faster" and "Trashman" programs, but at a lower price. Modular Software Associates (the company that brought you NEWBASIC 2.0) has met the challenge with two new products now available at TAS.

THE ANALYST "analyzes" your BASIC program as it executes. It tells you how many times each variable was referenced. Once you have that information, you can add a simple DIM statement so that the most-frequently-used variables will be located at the beginning of BASIC's variable lookup table. The instruction manual tells you how to do this, and just adding this one statement will generally speed up program execution by anywhere from fifteen to fifty percent.

In addition, THE ANALYST can tell you which lines in your program are executed most frequently, and you can then concentate your efforts to improve program execution speed on those lines. The manual offers several hints for speeding up the execution of frequently-used lines.

The features of THE ANALYST can be controlled from the keyboard or from within your program, so that you can selectively analyze a portion of the program. There is even a function to display all line numbers that have CLEAR, DEFxxx, or DIM statements, to aid you in placing the new DIM statement within a complex program. You use THE ANALYST to speed up almost any BASIC program, even if you're not the author of the program.

The other new Modular Software product is called THE COLLECTOR. It is an improved "garbage collector", replacing the string compression routine of BASIC. In case you're not aware of it, any program that uses string variables leaves "garbage" strings behind each time a string variable is assigned a new value. Periodically, these garbage strings must be "cleaned up" to make room in memory for more strings. When the ROM does it, it may take up to several MINUTES, during which the computer will appear to be "dead" (even the BREAK key won't work). The problem is especially acute with programs that use string search or string sorting routines that are written in BASIC.

Use of THE COLLECTOR can result in performance improvements in excess of 95% in programs that use many strings. For example, in a program that used 2,000 strings, the ROM garbage collection took 713.3 seconds to do its job. THE COLLECTOR reduced the time required to 7.8 seconds - that's 91.4 times faster!

THE COLLECTOR uses only 498 bytes of memory (compared to 578 for "Trashman"), plus an additional two bytes for each active string. If THE COLLECTOR runs out of memory, it will automatically deactivate itself and let the ROM routine take over, so that your program won't crash.

Both THE ANALYST and THE COLLECTOR are supplied on standard 35-track, single density (UNprotected) data diskettes, and will run on the TRS-80 Models I, III, and 4 (in the Model III mode). Both programs are normally self-relocating, but the user has the ability to specify where in memory the programs are to load, so they can be used right along with almost any other machine-language program. The price is the same for both programs - only $24.95 each (compare with "Faster" at $29.95 and "Trashman" at $39.95).

Use the coupon on the reverse side of this page to order, or write or phone The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906 (telephone (517) 482-8270).

Address Correspondence and Newsletter Exchanges To:

NORTHERN BYTES
c/o Jack Decker
1804 West 18th Street
Lot #155
Sault Ste. Marie, Michigan 49783

Postmaster: If undeliverable, please return to:
The Alternate Source, 704 North Pennsylvania Avenue, Lansing, MI. 48906

TO: