NORTHERN BYTES

Volume 7, Number 5

Greetings! Perhaps you're concerned that I have abandoned Northern Bytes, but that is not so. Unfortunately, Northern Bytes is not a profit making venture, and I have had to seek other recompense just to pay the heat bill (a very important consideration in Michigan).

This means that if you need technical questions answered, phones will only be answered between seven and ten, Michigan time, during most evenings. At other times expect an answering machine, or worse.

These actions are necessary in order to keep Northern Bytes going. The thought of discontinuing publication has crossed my mind, but the abundance of material should be organized. Northern Bytes is the best way I can think of to do that.

We had been receiving a half dozen newsletters at 704 North Pennsylvania. Since Jack officially directed everything to this address, the stack of prospective material for Northern Bytes has grown to about three feet. Of course, a substantial amount of material in that stack is either material reprinted from Northern Bytes or MSDOS stuff, but every now and then a TRS-80 gem is uncovered!

I would prefer to measure a volume by the amount of material published, thus, the volume number will stay at seven for this issue. However, this will be the last issue in Volume Seven, since the amount of material is the same as the last two volumes. The next issue will be the first of Volume Eight; that and successive volumes will consist of four forty-eight page issues.

There are a ton of TRS-80 products that are no longer available. The manufacturer has quit providing them or went out of business. At least a couple calls per week are asking about products such as Vivace or MuMath. Should you be sitting on either of those products, you can probably recover most of your investment immediately just by dropping me a note. For people looking for old TRS-80 product, or others interested in the TRS-80 market, we will immediately start accepting classified advertising at the rate of fifty cents per word, minimum ten words. Send ad copy, along with payment to 704 N. Pennsylvania.

It sort of bothers me that many people spend more on phone calls than most of our products cost. Unfortunately, trying to please everybody has let to the demise of more than one software company. The cost of software support is expensive. I feel sad for many of the callers that own TRS-80 computers. They are now in a situation that TI-99 owners were in a couple of years ago.

The cycle is occurring once again. Rumor has it that the new PageMaker software will not run in 640K and there's more to come. Northern Bytes has concentrated on assisting you with getting maximum value from your 64K machine. There are still many applications that could be documented, should you prove interested enough.

I would like to shift the focus of Northern Bytes from low level programming to higher level programming. During the next couple of issues we will be running a few articles to give you a sample. The reasons for doing this are numerous.

Most important, it will insure continued support for the TRS-80. Folks, it is possible to write one set of program code that will run on ALL versions of TRSDOS, CP/M, MSDOS, the MacIntosh, the Amiga, the Atari ST and the Apple II, and I have done it. That approach could have been taken a few years back, and our established software base on all machines would be greater. Unfortunately, we take the egocentric approach and just worry about the machine at hand. I would like to use Northern Bytes to help us reconsider that approach, and make programming really versatile.

Another reason for doing and thinking in terms of compatibility is the old "reinvent the wheel" syndrome. When working with higher level languages, it is much easier to pull a useful block of code from an existing application and adapt it to your application or machine. This is particularly important in a public domain publication such as Northern Bytes. Our goal is to improve the programming performance of every reader. This can be achieved by providing a wide selection of reusable routines that can be used without fear of copyright violation. We don't need to make everyone understand every single byte of code; we simply need to tell them how to use it.

Northern Bytes has always tended to focus on solving problems relating to the TRS-80 I/III and 4. Frequently, the solutions have tended to utilize the Z80 microprocessor assembly language. Z80 assembly language or any microprocessor assembly language is not a language that new users master quickly. There is a very high initial learning curve when learning your first assembly language. A single command can introduce a brand new set of computer concepts that must be understood before that command can be effectively used. When working in an assembly language, many simple functions that we take for granted must be written, often requiring hundreds, if not thousands of lines of code. Even the transition from TRSDOS to CP/M (both use the Z80 microprocessor) can be very tedious even though the differences represent only about 10% of the actual code. Translation from Z80 to other processor chips generally call for an entire rewrite. Z80 assembly language is a "low level" language.

In high level languages, one simple command may call into play hundreds or thousands of lines of preprogrammed code. There has been a plethora of compilers and assemblers and intepreters and various combinations thereof that allowed you to get the job done, albeit with a little hacking, on a TRS-80. The problem is that nobody really designed the tools right, as evidenced by the number of hackers who tried to do it better.

MSDOS users have a set of high level languages that will quickly generate extremely fast compiled code. Even things that exercise the low level attributes of the machine, like monitors and terminal programs, are being written in high-level languages. None of the hacking that was required on the TRS-80 is required on MSDOS. In the "real" world, I am working with a set of 93 dBase programs that walk and talk. There has never been a really outstanding data base manager for the TRS-80.

It would seem that the solution to the Northern Bytes problem can be resolved by the computer itself, by concentrating on higher level languages. Not as high level as dBase, but definitely as high level as BASIC, Pascal and C. People are much more likely to tweak or find someone who can tweak a BASIC program than a machine language program. We will give special attention to programs written in as generic a style as possible. Your support with this project would be appreciated.

WANTED your TRS-80 classifieds

only 50¢ a word CALL (517) 482-8270 or drop us letter!

FILE TRANSFER UTILITY

A program by Ted Baker

This program was written by Ted Baker of B.G.&T. Software, and first appeared in USR(80), the Journal of the Vancouver (British Columbia, Canada) TRS-80 Users Group. The program allows the transfer of files from NEWDOS/80 diskettes to TRSDOS 6.2 diskettes. The full capabilities of the program can be determined by reading the extensive remarks alongside the code. Some of the code lines are overlength and wrap around to the following line, so be sure to check the hex byte field of the listing if you have any doubts as to proper character spacing.

As this program is relatively long, you may wish to wait for it to appear on a TAS Public Domain Library disk, instead of keying it in yourself. We have received a few other programs that were written by Ted and those will probably be on the PD Library disk as well.

```
006A          00100 CKBRKC  EQU     106     ;CHECK BREAK BIT AND CLEAR IT
0021          00110 CKDRV   EQU     33      ;CHECK DRIVE
003C          00120 CLOSE   EQU     60      ;CLOSE A FILE OR DEVICE
0019          00130 CMNDR   EQU     25      ;EXECUTE COMMAND
005D          00140 DIV8    EQU     93      ;8-BIT DIVIDE
005E          00150 DIV16   EQU     94      ;16 BIT BY 8 BIT DIVIDE
0022          00160 DODIR   EQU     34      ;DO DIRECTORY DISPLAY/BUFFER
0002          00170 DSP     EQU     2       ;DISPLAY A CHARACTER
000A          00180 DSPLY   EQU     10      ;DISPLAY MESSAGE LINE
0016          00190 EXIT    EQU     22      ;EXIT TO TRSDOS
001A          00200 ERROR   EQU     26      ;ENTRY TO POST AN ERROR MESSAGE
004E          00210 FSPEC   EQU     78      ;ASSIGN FILE OR DEVICE SPECIFICATION
0051          00220 GTDCT   EQU     81      ;GET DRIVE CODE TABLE
0061          00230 HEXDEC  EQU     97      ;CONVERT BINARY TO DECIMAL ASCII
0062          00240 HEX8    EQU     98      ;CONVERT 1 BYTE TO HEX ASCII
0063          00250 HEX16   EQU     99      ;CONVERT 2 BYTES TO HEX ASCII
0064          00260 HIGH    EQU     100     ;GET OR ALTER HIGH$ OR LOW$
003A          00270 INIT    EQU     58      ;OPEN OR INITIALIZE FILE
0008          00280 KBD     EQU     8       ;SCAN KEYBOARD AND RETURN
0001          00290 KEY     EQU     1       ;SCAN *KI DEVICE, WAIT FOR CHARACTER
0009          00300 KEYIN   EQU     9       ;ACCEPT A LINE OF INPUT
0040          00310 LOF     EQU     64      ;CALCULATE THE EOF LOGICAL RECORD NUMBER
005A          00320 MUL8    EQU     90      ;8-BIT MULTIPLICATION
005B          00330 MUL16   EQU     91      ;16-BIT BY 8-BIT MULTIPLICATIION
003B          00340 OPEN    EQU     59      ;OPEN EXISTING FILE OR DEVICE
0042          00350 POSN    EQU     66      ;POSITION FILE
000E          00360 PRINT   EQU     14      ;PRINTS MESSAGE LINE
0006          00370 PRT     EQU     6       ;SEND CHARACTER TO PRINTER
0023          00380 RAMDIR  EQU     35      ;GET DIRECTORY RECORD OR FREE SPACE
0031          00390 RDSEC   EQU     49      ;READ SECTOR
0055          00400 RDSSC   EQU     85      ;READ SYSTEM SECTOR
0043          00410 READ    EQU     67      ;READ A RECORD
0039          00420 REMOV   EQU     57      ;REMOVE FILE OR DEVICE
0068          00430 SOUND   EQU     104     ;SOUND GENERATION
000F          00440 VDCTL   EQU     15      ;VIDEO FUNCTIONS
0049          00450 VER     EQU     73      ;WRITE AND VERIFY A RECORD
004A          00460 WEOF    EQU     74      ;WRITE END OF FILE
0048          00470 WRITE   EQU     75      ;WRITE A RECORD
0036          00480 WRSSC   EQU     54      ;WRITE A SYSTEM SECTOR
3000          00490         ORG     3000H
0100          00500 BUFFER  DEFS    256
3100 53       00510 SYSRES  DEFM    'SYSTEM (SYSRES='
     59 53 54 45 4D 20 28 53 59 53 52 45 53 3D
310F 31       00520 SN      DEFM    '1)'
     29
3111 0D       00530         DEFB    0DH
3112 53       00540 SYSRS1  DEFM    'SYSTEM (SYSRES=12)'
     59 53 54 45 4D 20 28 53 59 53 52 45 53 3D 31 32
     29
3124 0D       00550         DEFB    0DH
3125 0A       00560 NODIR   DEFB    0AH
3126 43       00570         DEFM    'CANNOT LOCATE DIRECTORY.'
     41 4E 4E 4F 54 20 4C 4F 43 41 54 45 20 44 49 52
     45 43 54 4F 52 59 2E
313E 0D       00580         DEFB    0DH
313F 0A       00590 FNDIR   DEFB    0AH
3140 44       00600         DEFM    'DIRECTORY FOUND.'
     49 52 45 43 54 4F 52 59 20 46 4F 55 4E 44 2E
3150 0D       00610         DEFB    0DH
3151 45       00620 ERR     DEFM    'ERROR ='
     52 52 4F 52 20 3D
3158 2020     00630 B2      DEFW    2020H   ;BUFFER TWO
315A 0D       00640         DEFB    0DH
315B 1C1F     00650 TITLE   DEFW    1F1CH   ;CLEAR SCREEN
315D 43       00660         DEFM    'COPY NEWDOS FILE TO TRSDOS 6.2.X COMPATIABLE DOS'
     4F 50 59 20 4E 45 57 44 4F 53 20 46 49 4C 45 20
     54 4F 20 54 52 53 44 4F 53 20 36 2E 32 2E 58 20
     43 4F 4D 50 41 54 49 41 42 4C 45 20 44 4F 53
318D 0A       00670         DEFB    0AH
318E 7E       00680         DEFM    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
31BE 0A0A     00690         DEFW    0A0AH
31C0 62       00700         DEFM    'by Ted Baker'
     79 20 54 65 64 20 42 61 6B 65 72
31CC 0A       00710         DEFB    0AH
31CD 42       00720         DEFM    'B.G.&T. Software - May 1985 - Ver 2.0'
     2E 47 2E 26 54 2E 20 53 6F 66 74 77 61 72 65 20
     2D 20 4D 61 79 20 31 39 38 35 20 2D 20 56 65 72
     20 32 2E 30
31F2 0A0A     00730         DEFW    0A0AH
31F4 4E       00740         DEFM    'NEWDOS 2.0 :'
     45 57 44 4F 53 20 32 2E 30 20 3A
3200 31       00750 SD1     DEFM    '1 <SOURCE>'
     20 3C 53 4F 55 52 43 45 3E
320A 0A       00760         DEFB    0AH
320B 54       00770         DEFM    'TRSDOS 6.2 :'
     52 53 44 4F 53 20 36 2E 32 20 3A
3217 30       00780 DD1     DEFM    '0 <DESTINATION>'
     20 3C 44 45 53 54 49 4E 41 54 49 4F 4E 3E
3226 0A       00790         DEFB    0AH
3227 50       00800         DEFM    'PRESS <SPACE BAR> TO CHANGE.'
     52 45 53 53 20 3C 53 50 41 43 45 20 42 41 52 3E
     20 54 4F 20 43 48 41 4E 47 45 2E
3243 0A0A     00810 ENTER   DEFW    0A0AH
3245 50       00820         DEFM    'PRESS <ENTER> TO BEGIN.'
     52 45 53 53 20 3C 45 4E 54 45 52 3E 20 54 4F 20
     42 45 47 49 4E 2E
325C 0F       00830         DEFB    15
325D 0A0D     00840         DEFW    0D0AH
              00850 ;NEWDOS DIRECTORY ON TRACK 9 SEC 8  1 SIDE
              00860 ;NEWDOS DIRECTORY ON TRACK 4 SEC 26 2 SIDES
325F 00       00870 EODR    DEFB    0       ;END OF NEWDOS DIRECTORY IN RAM
3260 00       00880 LDIRS   DEFB    0       ;LENGTH OF DIRECTORY IN SECTORS
3261 00       00890 LDIR    DEFB    0       ;LENGTH OF DIRECTORY IN GRANS
3262 00       00900 MTRK    DEFB    00      ;MAX TRACKS
3263 01       00910 SFLAG   DEFB    01      ;ONE SIDE NEWDOS
3264 00       00920 SEC     DEFB    0
3265 00       00930 TRK     DEFB    0
3266 00       00940 MSEC    DEFB    0
3267 00       00950 MNSEC   DEFB    0
3268 00       00960 DIRB    DEFB    0
3269 01       00970 SDRV    DEFB    1
326A 00       00980 DIRT    DEFB    0
326B 00       00990 CT      DEFB    0
326C 0A       01000 WMES    DEFB    0AH
326D 44       01010         DEFM    'DIRECTORY IS AT TRACK '
     49 52 45 43 54 4F 52 59 20 49 53 20 41 54 20 54
     52 41 43 4B 20
0005          01020 WMES1   DEFS    5
3288 20       01030         DEFM    '     SECTOR '
     20 20 20 20 20 53 45 43 54 4F 52 20
0005          01040 WMES2   DEFS    5
329A 0D       01050         DEFB    0DH
329B 1C1F     01060 CBQM    DEFW    1F1CH
329D 51       01070         DEFM    'Query NEWDOS File Before Copy to TRSDOS Disk'
     75 65 72 79 20 4E 45 57 44 4F 53 20 46 69 6C 65
     20 42 65 66 6F 72 65 20 43 6F 70 79 20 74 6F 20
     54 52 53 44 4F 53 20 44 69 73 6B
32C9 0A       01080         DEFB    0AH
32CA 7E       01090         DEFM    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
32F6 0A0D     01100         DEFW    0D0AH
32F8 1C1F     01110 CBAM    DEFW    1F1CH
32FA 43       01120         DEFM    'Copy All NEWDOS Files to TRSDOS Disk'
     6F 70 79 20 41 6C 6C 20 4E 45 57 44 4F 53 20 46
     69 6C 65 73 20 74 6F 20 54 52 53 44 4F 53 20 44
     69 73 6B
331E 0A       01130         DEFB    0AH
331F 7E       01140         DEFM    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
```

```
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
     7E 7E 7E
3343 0A0D      01150        DEFW    0D0AH
3345 20        01160 CQM    DEFM    ' Copy File? '
     43 6F 70 79 20 46 69 6C 65 3F 20
3351 03        01170        DEFB    3
0002           01180 CBAHL  DEFS    2
000B           01190 PFILEN DEFS    11
335F 3A6B32    01200 CBA    LD      A,(CT)
3362 FE00      01210        CP      0
3364 2805      01220        JR      Z,CBAA
3366 219B32    01230        LD      HL,CBQM
3369 1803      01240        JR      CBAAA
336B 21F832    01250 CBAA   LD      HL,CBAM
336E 3E0A      01260 CBAAA  LD      A,DSPLY
3370 EF        01270        RST     28H
3371 3E0F      01280        LD      A,VOCTL
3373 0607      01290        LD      B,7             ;SCROLL PROTECT
3375 0E03      01300        LD      C,3             ;3 LINES
3377 EF        01310        RST     28H
3378 210060    01320        LD      HL,DIRBUF       ;COPY ALL FILES - DIRECTORY BUFFER
337B 24        01330        INC     H
337C 24        01340        INC     H
337D 7E        01350 CB1A   LD      A,(HL)          ;CHECK FILE STATUS,SYS,EXT,KILLED,ETC
337E CB77      01360        BIT     6,A
3380 2008      01370        JR      NZ,CB1B
3382 CB7F      01380        BIT     7,A
3384 2004      01390        JR      NZ,CB1B
3386 CB67      01400        BIT     4,A
3388 200F      01410        JR      NZ,FOKFC
338A 012000    01420 CB1B   LD      BC,32
338D 09        01430        ADD     HL,BC
338E 3A5F32    01440        LD      A,(EODR)
3391 94        01450        SUB     H
3392 FE00      01460        CP      0
3394 CAF835    01470        JP      Z,BQDIR
3397 18E4      01480        JR      CB1A
3399 E5        01490 FOKFC  PUSH    HL
339A 010500    01500        LD      BC,5
339D 09        01510        ADD     HL,BC           ;POINTS TO FILE NAME
339E E5        01520        PUSH    HL
339F 060B      01530        LD      B,11
33A1 7E        01540 PFCAF  LD      A,(HL)
33A2 4F        01550        LD      C,A
33A3 3E02      01560        LD      A,DSP
33A5 EF        01570        RST     28H
33A6 23        01580        INC     HL
33A7 10F8      01590        DJNZ    PFCAF
33A9 0604      01600        LD      B,4
33AB 3E02      01610 CAFLP1 LD      A,DSP
33AD 0E20      01620        LD      C,20H
33AF EF        01630        RST     28H
33B0 10F9      01640        DJNZ    CAFLP1
33B2 3A6B32    01650        LD      A,(CT)
33B5 FE00      01660        CP      0
33B7 2816      01670        JR      Z,CAFF
33B9 3E0A      01680        LD      A,DSPLY
33BB 214533    01690        LD      HL,CQM
33BE EF        01700        RST 28H
33BF CD4234    01710        CALL    GETKEY
33C2 FE59      01720        CP      'Y'
33C4 2809      01730        JR      Z,CAFF
33C6 FE79      01740        CP      'y'
33C8 2805      01750        JR      Z,CAFF
33CA E1        01760        POP     HL
33CB E1        01770        POP     HL
33CC C31034    01780        JP      NFHFQ
33CF E1        01790 CAFF   POP     HL
33D0 115433    01800        LD      DE,PFILEN
33D3 010B00    01810        LD      BC,11
33D6 EDB0      01820        LDIR                    ;MOVE FILENAME TO FILE NAME QUESTION
33D8 215433    01830        LD      HL,PFILEN
33DB 116A3A    01840        LD      DE,FCB2         ;FILE CONTROL BUFFER
33DE 0608      01850        LD      B,8             ;FIRST 8 CHARACTERS OF FILE NAME
33E0 7E        01860 PL1CAF LD      A,(HL)
33E1 FE20      01870        CP      ' '
33E3 2802      01880        JR      Z,CBFA1
33E5 12        01890        LD      (DE),A
33E6 13        01900        INC     DE
33E7 23        01910 CBFA1  INC     HL
33E8 10F6      01920        DJNZ    PL1CAF
```

```
33EA 3E2F      01930        LD      A,'/'
33EC 12        01940        LD      (DE),A
33ED 13        01950        INC     DE
33EE 0603      01960        LD      B,3             ;FIRST 8 CHARACTERS OF FILE NAME
33F0 7E        01970 CBFA3  LD      A,(HL)
33F1 FE20      01980        CP      ' '
33F3 2802      01990        JR      Z,CBFA2
33F5 12        02000        LD      (DE),A
33F6 13        02010        INC     DE
33F7 23        02020 CBFA2  INC     HL
33F8 10F6      02030        DJNZ    CBFA3
33FA 3A1632    02040        LD      A,(DD1-1)       ;ADD DESTINATION DRIVE NUMBER
33FD 12        02050        LD      (DE),A          ;TO FILE NAME
33FE 13        02060        INC     DE
33FF 3A1732    02070        LD      A,(DD1)
3402 12        02080        LD      (DE),A
3403 13        02090        INC     DE
3404 3E0D      02100        LD      A,0DH
3406 12        02110        LD      (DE),A
3407 E1        02120        POP     HL              ;NECCESSARY FOR BSFCH
3408 E5        02130        PUSH    HL              ;LEAVE IN
3409 CDB742    02140        CALL    BSFCH
340C E1        02150        POP     HL
340D C26C39    02160        JP      NZ,EXT
3410 3E02      02170 NFHFQ  LD      A,DSP
3412 0E0D      02180        LD      C,0DH
3414 EF        02190        RST     28H
3415 3E6A      02200        LD      A,CKBRKC
3417 EF        02210        RST     28H
3418 C2F835    02220        JP      NZ,BQDIR
341B C38A33    02230        JP      CB1B
341E 2A6A32    02240 WHERE  LD      HL,(DIRT)
3421 2600      02250        LD      H,0
3423 118332    02260        LD      DE,WMES1
3426 3E61      02270        LD      A,HEXDEC
3428 EF        02280        RST     28H
3429 2A6732    02290        LD      HL,(MNSEC)
342C 2600      02300        LD      H,0
342E 2B        02310        DEC     HL
342F 119532    02320        LD      DE,WMES2
3432 3E61      02330        LD      A,HEXDEC
3434 EF        02340        RST     28H
3435 216C32    02350        LD      HL,WMES
3438 3E0A      02360        LD      A,DSPLY
343A EF        02370        RST     28H
343B C30A36    02380        JP      GHT1
343E 3E68      02390 BEEP   LD      A,SOUND
3440 EF        02400        RST     28H     ;BEEP
3441 C9        02410        RET
3442 3E01      02420 GETKEY LD      A,KEY
3444 EF        02430        RST     28H             ;GET CHARACTER IN A
3445 C5        02440        PUSH    BC
3446 4F        02450        LD      C,A
3447 3E02      02460        LD      A,DSP
3449 EF        02470        RST     28H
344A 79        02480        LD      A,C
344B C1        02490        POP     BC
344C C9        02500        RET
344D 0A        02510 CONF2  DEFB    0AH
344E 43        02520        DEFM    'CONFIGURATION'
     4F 4E 46 49 47 55 52 41 54 49 4F 4E
345B 0A0A      02530        DEFW    0A0AH
345D 4E        02540        DEFM    'NEWDOS 2.0 :'
     45 57 44 4F 53 20 32 2E 30 20 3A
3469 03        02550        DEFB    03H
346A 0A        02560 CONF1  DEFB    0AH
346B 54        02570        DEFM    'TRSDOS 6.2 :'
     52 53 44 4F 53 20 36 2E 32 20 3A
3477 03        02580        DEFB    03H
3478 0600      02590 CONFGR LD      B,0
347A CD3E34    02600        CALL    BEEP
347D 214D34    02610        LD      HL,CONF2
3480 3E0A      02620        LD      A,DSPLY
3482 EF        02630        RST     28H
3483 CD4234    02640        CALL    GETKEY
3486 320032    02650        LD      (SD1),A
3489 329E34    02660        LD      (CONCP+1),A
348C DE30      02670        SBC     A,30H
348E 326932    02680        LD      (SDRV),A
3491 216A34    02690        LD      HL,CONF1
3494 3E0A      02700        LD      A,DSPLY
```

```
3496 EF       02710          RST     28H
3497 CD4234   02720          CALL    GETKEY
349A 321732   02730          LD      (DD1),A
349D FE00     02740 CONCP    CP      0
349F 20D0     02750          JR      NZ,NOLOAD
34A1 0603     02760          LD      B,3
34A3 CD3E34   02770          CALL    BEEP
34A6 3E0A     02780          LD      A,DSPLY
34A8 21AE34   02790          LD      HL,CONER
34AB EF       02800          RST     28H
34AC 18CA     02810          JR      CONFGR
34AE 0A0A     02820 CONER    DEFW    0A0AH
34B0 45       02830          DEFM    'ERROR, '
     52 52 4F 52 2C 20
34B7 10       02840          DEFB    16                      ;REVERSE VIDEO
34B8 53       02850          DEFM    'SOURCE'
     4F 55 52 43 45
34BE 11       02860          DEFB    17                      ;REVERSE VIDEO OFF
34BF 20       02870          DEFM    ' CANNOT EQUAL '
     43 41 4E 4E 4F 54 20 45 51 55 41 4C 20
34CD 10       02880          DEFB    16
34CE 44       02890          DEFM    'DESTINATION'
     45 53 54 49 4E 41 54 49 4F 4E
34D9 11       02900          DEFB    17
34DA 2E       02910          DEFM    '...'
     2E 2E
34DD 0D       02920          DEFB    0DH
34DE 210030   02930 START    LD      HL,BUFFER
34E1 24       02940          INC     H
34E2 7C       02950          LD      A,H             ;EOB
34E3 323B38   02960          LD      (EOB1+1),A
34E6 324F39   02970          LD      (EOB2+1),A
34E9 3E64     02980          LD      A,HIGH          ;GET HIGH MEMORY
34EB 210000   02990          LD      HL,0
34EE 0600     03000          LD      B,0
34F0 EF       03010          RST     28H
34F1 7C       03020          LD      A,H             ;IF MODULES RESIDE MEMORY LESS <F3
34F2 FEE9     03030          CP      0E9H
34F4 2818     03040          JR      Z,NOLOAD
34F6 0604     03050          LD      B,4             ;/SYS NUMBER
34F8 78       03060 LPA      LD      A,B
34F9 C630     03070          ADD     A,30H           ;MAKE ASCII NUMBER
34FB 320F31   03080          LD      (SN),A          ;COMMAND SYNTAX
34FE 210031   03090          LD      HL,SYSRES       ;'SYSTEM (SYSRES='A')'
3501 3E19     03100          LD      A,CMNDR
3503 C5       03110          PUSH    BC
3504 EF       03120          RST     28H
3505 C1       03130          POP     BC
3506 10F0     03140          DJNZ    LPA             ;LOOP TILL 3 /SYS IN MEMORY
3508 211231   03150          LD      HL,SYSRS1
350B 3E19     03160          LD      A,CMNDR
350D EF       03170          RST     28H
350E 215B31   03180 NOLOAD   LD      HL,TITLE
3511 3E0A     03190          LD      A,DSPLY
3513 EF       03200          RST     28H             ;DISPLAY TITLE
3514 3E07     03210          LD      A,7
3516 E607     03220 G1       AND     7
3518 47       03230          LD      B,A
3519 CD3E34   03240          CALL    BEEP
351C 3E01     03250          LD      A,KEY           ;GET A KEY
351E EF       03260          RST     28H
351F FE20     03270          CP      ' '
3521 CA7834   03280          JP      Z,CONFGR
3524 FE0D     03290          CP      13              ;ENTER
3526 2807     03300          JR      Z,SEARCH
3528 FE80     03310          CP      80H
352A CA5436   03320          JP      Z,ENDQ
352D 18E7     03330          JR      G1
352F 1600     03340 SEARCH   LD      D,0             ;NEWDOS  INFORMATION SECTOR
3531 1E02     03350          LD      E,2
3533 3A6932   03360          LD      A,(SDRV)
3536 4F       03370          LD      C,A             ;DRIVE
3537 210030   03380          LD      HL,BUFFER
353A 3E31     03390          LD      A,RDSEC         ;READ SECTOR
353C EF       03400          RST     28H
353D C26C39   03410          JP      NZ,EXT
3540 210330   03420          LD      HL,BUFFER+3
3543 7E       03430          LD      A,(HL)          ;NUMBER OF TRACKS
3544 3D       03440          DEC     A               ;LESS ONE
3545 326232   03450          LD      (MTRK),A
3548 210930   03460          LD      HL,BUFFER+9
```

```
354B 7E       03470          LD      A,(HL)
354C 326132   03480          LD      (LDIR),A
354F 4F       03490          LD      C,A
3550 1E05     03500          LD      E,5             ;5 SECTORS PER GRAN-NEWDOS
3552 3E5A     03510          LD      A,MUL8
3554 EF       03520          RST     28H
3555 326032   03530          LD      (LDIRS),A
3558 010400   03540          LD      BC,4
355B 210030   03550          LD      HL,BUFFER
355E 09       03560          ADD     HL,BC
355F 7E       03570          LD      A,(HL)
3560 326632   03580          LD      (MSEC),A
3563 329243   03590          LD      (RSM+1),A       ;SECTORS PER TRACK
3566 32E335   03600          LD      (EODF+1),A
3569 325145   03610          LD      (EODFW+1),A
356C 23       03620          INC     HL
356D 0B       03630          DEC     BC
356E 7E       03640          LD      A,(HL)          ;GPL
356F 327B35   03650          LD      (PGPLH+1),A
3572 09       03660          ADD     HL,BC           ;HL POINTS TO DDSL
3573 4E       03670          LD      C,(HL)          ;C=DDSL
3574 210500   03680          LD      HL,5            ;5 SECTORS PER GRAN
3577 3E5B     03690          LD      A,MUL16
3579 EF       03700          RST     28H             ;HL,A = DIRECTORY START IN SECTORS
357A 0E00     03710 PGPLH    LD      C,0             ;GPL
357C 65       03720          LD      H,L
357D 6F       03730          LD      L,A
357E 3E5B     03740          LD      A,MUL16
3580 EF       03750          RST     28H             ;5*DDSL*GPL=DIRECTORY SECTOR
3581 65       03760          LD      H,L
3582 6F       03770          LD      L,A
3583 3A6632   03780          LD      A,(MSEC)
3586 4F       03790          LD      C,A             ;SECTORS PER TRACK
3587 3E5E     03800          LD      A,DIV16
3589 EF       03810          RST     28H             ;HL=TRACK,A=SECTOR OF DIRECTORY
358A 326832   03820          LD      (DIRB),A        ;SECTOR
358D 326432   03830          LD      (SEC),A         ;OF DIRECTORY START
3590 3C       03840          INC     A
3591 326732   03850          LD      (MNSEC),A       ;HIT TABLE SECTOR
3594 7D       03860          LD      A,L             ;TRACK START
3595 326532   03870          LD      (TRK),A
3598 326A32   03880          LD      (DIRT),A
359B 3A6632   03890          LD      A,(MSEC)
359E 5F       03900          LD      E,A
359F 0E12     03910          LD      C,18
35A1 3E5D     03920          LD      A,DIV8
35A3 EF       03930          RST     28H             ;MSEC/18=SIDES 18SEC PER SIDE
35A4 326332   03940          LD      (SFLAG),A
35A7 3A6932   03950          LD      A,(SDRV)        ;SOURCE DRIVE
35AA 4F       03960          LD      C,A
35AB 3E51     03970          LD      A,GTDCT
35AD EF       03980          RST     28H             ;GET DRIVE 1 TABLE ADDRESS
35AE FD4E04   03990          LD      C,(IY+4)
35B1 3A6332   04000          LD      A,(SFLAG)
35B4 FE01     04010          CP      1
35B6 2804     04020          JR      Z,NDS
35B8 CBE9     04030          SET     5,C             ;SET FOR DOUBLE SIDED OPERATION
35BA 1802     04040          JR      SBCH
35BC CBA9     04050 NDS      RES     5,C             ;SET FOR SINGLE SIDED OPERATION
35BE FD7104   04060 SBCH     LD      (IY+4),C
35C1 3A6232   04070          LD      A,(MTRK)        ;NUMBER OF TRACKS
35C4 FD7706   04080          LD      (IY+6),A
35C7 CD8C38   04090          CALL    CMPARE  ;SEE IF 40 ON 80
35CA 3A6032   04100          LD      A,(LDIRS)
35CD 47       04110          LD      B,A
35CE 3A6932   04120          LD      A,(SDRV)
35D1 4F       04130          LD      C,A
35D2 3A6832   04140          LD      A,(DIRB)
35D5 5F       04150          LD      E,A
35D6 3A6A32   04160          LD      A,(DIRT)
35D9 57       04170          LD      D,A
35DA 210060   04180          LD      HL,DIRBUF
35DD 3E55     04190 NSOFD    LD      A,RDSSC
35DF EF       04200          RST     28H
35E0 7B       04210          LD      A,E
35E1 3C       04220          INC     A
35E2 FE00     04230 EODF     CP      0
35E4 2003     04240          JR      NZ,IA
35E6 3E00     04250          LD      A,0
35E8 14       04260          INC     D               ;INC TRACK      *************
35E9 5F       04270 IA       LD      E,A
```

```
35EA 24        04280          INC    H
35EB 10F0      04290          DJNZ   NSOFD
35ED 7C        04300          LD     A,H
35EE 325F32    04310          LD     (EODR),A
35F1 24        04320          INC    H
35F2 227D45    04330          LD     (STDIR),HL
35F5 CD0F44    04340          CALL   SORT
35F8 3E0F      04350 BQDIR    LD     A,VDCTL
35FA 0607      04360          LD     B,7              ;SCROLL PROTECT
35FC 0E00      04370          LD     C,0              ;0 LINES
35FE EF        04380          RST    28H
35FF 218B36    04390          LD     HL,QDIR1
3602 3E0A      04400          LD     A,DSPLY
3604 EF        04410          RST    28H
3605 0600      04420          LD     B,0
3607 CD3E34    04430          CALL   BEEP
360A 3E01      04440 GHT1     LD     A,KEY
360C EF        04450          RST    28H
360D FE31      04460          CP     '1'
360F CABE37    04470          JP     Z,RDIR
3612 FE32      04480          CP     '2'
3614 CA4C36    04490          JP     Z,NCBF           ;COPY BY FILE
3617 FE33      04500          CP     '3'
3619 CA4436    04510          JP     Z,AFC
361C FE35      04520          CP     '5'
361E CACA3B    04530          JP     Z,DIRECT
3621 FE36      04540          CP     '6'
3623 CA0E35    04550          JP     Z,NOLOAD
3626 FE51      04560          CP     'Q'
3628 282A      04570          JR     Z,ENDQ
362A FE71      04580          CP     'q'
362C 2826      04590          JR     Z,ENDQ
362E FE46      04600          CP     'F'
3630 CA2C3D    04610          JP     Z,FREE
3633 FE66      04620          CP     'f'
3635 CA2C3D    04630          JP     Z,FREE
3638 FE3F      04640          CP     '?'
363A CA1E34    04650          JP     Z,WHERE
363D FE17      04660          CP     23               ;CTRL W
363F CA3B45    04670          JP     Z,WRITED
3642 18C6      04680          JR     GHT1
3644 3E00      04690 AFC      LD     A,0
3646 326B32    04700          LD     (CT),A
3649 C35F33    04710          JP     CBA
364C 3E01      04720 NCBF     LD     A,1
364E 326B32    04730          LD     (CT),A
3651 C35F33    04740          JP     CBA
3654 3E0A      04750 ENDQ     LD     A,DSPLY
3656 216D36    04760          LD     HL,ENDQ1
3659 EF        04770          RST    28H
365A 0600      04780          LD     B,0
365C CD3E34    04790          CALL   BEEP
365F 3E01      04800          LD     A,KEY
3661 EF        04810          RST    28H
3662 0E00      04820          LD     C,0
3664 3E21      04830          LD     A,CKDRV          ;CHECK DRIVE
3666 EF        04840          RST    28H
3667 C26C39    04850          JP     NZ,EXT
366A 3E16      04860          LD     A,EXIT
366C EF        04870          RST    28H              ;RETURN TO DOS
366D 0A0A      04880 ENDQ1    DEFW   0A0AH
366F 0E        04890          DEFB   14
3670 50        04900          DEFM   'PUT SYSTEM DISK IN DRIVE 0'
     55 54 20 53 59 53 54 45 4D 20 44 49 53 4B 20 49
     4E 20 44 52 49 56 45 20 30
368A 03        04910          DEFB   3
368B 1C1F      04920 QDIR1    DEFW   1F1CH
368D 43        04930          DEFM   'COPY NEWDOS FILE TO TRSDOS 6.2.X COMPATIABLE DOS'
     4F 50 59 20 4E 45 57 44 4F 53 20 46 49 4C 45 20
     54 4F 20 54 52 53 44 4F 53 20 36 2E 32 2E 58 20
     43 4F 4D 50 41 54 49 41 42 4C 45 20 44 4F 53
36BD 0A        04940          DEFB   0AH
36BE 7E        04950          DEFM   '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
     7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E 7E
36EE 0A0A      04960          DEFW   0A0AH
36F0 62        04970          DEFM   'by Ted Baker'
     79 20 54 65 64 20 42 61 6B 65 72
36FC 0A        04980          DEFB   0AH
36FD 42        04990          DEFM   'B.G.&T. Software - May 1985'
```

```
     2E 47 2E 26 54 2E 20 53 6F 66 74 77 61 72 65 20
     2D 20 4D 61 79 20 31 39 38 35
3718 0A0A      05000          DEFW   0A0AH
371A 3C        05010          DEFM   '<1>  Copy One File'
     31 3E 20 20 43 6F 70 79 20 4F 6E 65 20 46 69 6C
     65
372C 0A        05020          DEFB   0AH
372D 3C        05030          DEFM   '<2>  Query Each File'
     32 3E 20 20 51 75 65 72 79 20 45 61 63 68 20 46
     69 6C 65
3741 0A        05040          DEFB   0AH
3742 3C        05050          DEFM   '<3>  Copy All Files'
     33 3E 20 20 43 6F 70 79 20 41 6C 6C 20 46 69 6C
     65 73
3755 0A        05060          DEFB   0AH
3756 3C        05070          DEFM   '<5>  Directory of Drive'
     35 3E 20 20 44 69 72 65 63 74 6F 72 79 20 6F 66
     20 44 72 69 76 65
376D 0A        05080          DEFB   0AH
376E 3C        05090          DEFM   '<6>  Change NEWDOS Disk'
     36 3E 20 20 43 68 61 6E 67 65 20 4E 45 57 44 4F
     53 20 44 69 73 6B
3785 0A        05100          DEFB   0AH
3786 3C        05110          DEFM   '<F>  NEWDOS Free Space Map'
     46 3E 20 20 4E 45 57 44 4F 53 20 46 72 65 65 20
     53 70 61 63 65 20 4D 61 70
37A0 0A        05120          DEFB   0AH
37A1 3C        05130          DEFM   '<Q>  Return to TRSDOS 6.2'
     51 3E 20 20 52 65 74 75 72 6E 20 74 6F 20 54 52
     53 44 4F 53 20 36 2E 32
37BA 0A        05140          DEFB   0AH
37BB 20        05150          DEFM   ' '
37BC 0E        05160          DEFB   14
37BD 03        05170          DEFB   3
37BE 3A6A32    05180 RDIR     LD     A,(DIRT)
37C1 326532    05190          LD     (TRK),A
37C4 3A6B32    05200          LD     A,(DIRB)
37C7 326432    05210          LD     (SEC),A
37CA 3A6532    05220          LD     A,(TRK)
37CD 00        05230 DBL1     NOP
37CE 57        05240          LD     D,A
37CF 3A6432    05250          LD     A,(SEC)
37D2 5F        05260          LD     E,A
37D3 3A6932    05270          LD     A,(SDRV)
37D6 4F        05280          LD     C,A              ;DRIVE NUMBER
37D7 210030    05290          LD     HL,BUFFER
37DA CD7042    05300          CALL   DODISK
37DD C26C39    05310          JP     NZ,EXT           ;EXIT ON ERROR
37E0 21D030    05320          LD     HL,BUFFER+0D0H   ;DISK NAME
37E3 11103A    05330          LD     DE,NAME
37E6 010800    05340          LD     BC,8
37E9 EDB0      05350          LDIR
37EB 11233A    05360          LD     DE,DATE
37EE 010800    05370          LD     BC,8
37F1 EDB0      05380          LDIR
37F3 21083A    05390          LD     HL,DIR           ;DISPLAY DIRECTORY
37F6 3E0A      05400          LD     A,DSPLY
37F8 EF        05410          RST    28H
37F9 3A6632    05420          LD     A,(MSEC)
37FC 320538    05430          LD     (MSEC1+1),A
37FF 3A6432    05440          LD     A,(SEC)
3802 3C        05450          INC    A
3803 3C        05460 NSEC     INC    A
3804 FE00      05470 MSEC1    CP     0
3806 C21238    05480          JP     NZ,OK1
3809 3A6532    05490          LD     A,(TRK)
380C 3C        05500          INC    A
380D 326532    05510          LD     (TRK),A
3810 3E00      05520          LD     A,0
3812 326432    05530 OK1      LD     (SEC),A
3815 5F        05540          LD     E,A
3816 3A6932    05550          LD     A,(SDRV)
3819 4F        05560          LD     C,A              ;DRIVE NUMBER
381A 3A6532    05570          LD     A,(TRK)
381D 00        05580 DBL2     NOP
381E 57        05590          LD     D,A
381F 210030    05600          LD     HL,BUFFER
3822 CD7042    05610          CALL   DODISK
3825 C25E38    05620          JP     NZ,DIRDNE        ;EXIT ON ERROR
3828 7E        05630 LKRC     LD     A,(HL)
3829 CB77      05640          BIT    6,A
```

```
382B 2008      05650          JR     NZ,NXT
382D CB7F      05660          BIT    7,A
382F 2004      05670          JR     NZ,NXT
3831 CB67      05680          BIT    4,A
3833 200F      05690          JR     NZ,PRNT
3835 012000    05700 NXT      LD     BC,32
3838 09        05710 NXT1     ADD    HL,BC              ;NEXT RECORD
3839 7C        05720          LD     A,H
383A FEEF      05730 EOB1     CP     0EFH               ;END OF SECTOR
383C 20EA      05740          JR     NZ,LKRC
383E 3A6432    05750          LD     A,(SEC)
3841 C30338    05760          JP     NSEC
3844 010500    05770 PRNT     LD     BC,5               ;START OF FILE NAME
3847 09        05780          ADD    HL,BC
3848 010B00    05790          LD     BC,11
384B 112D3A    05800          LD     DE,B3
384E EDB0      05810          LDIR                      ;COPY FILE NAME TO BUFFER
3850 E5        05820          PUSH   HL
3851 212D3A    05830          LD     HL,B3
3854 3E0A      05840          LD     A,DSPLY            ;DISPLAY FILE NAME
3856 EF        05850          RST    28H
3857 E1        05860          POP    HL
3858 011000    05870          LD     BC,16
3858 C33838    05880          JP     NXT1               ;NEXT RECORD
385E 213E3A    05890 DIRDNE   LD     HL,FIL             ;ASCII FOR FILE NAME
3861 3E0A      05900          LD     A,DSPLY
3863 EF        05910          RST    28H
3864 0600      05920          LD     B,0                ;TONE
3866 CD3E34    05930          CALL   BEEP
3869 3E09      05940          LD     A,KEYIN            ;GET FILNAME
386B 21EA39    05950          LD     HL,FN              ;BUFFER FOR FILENAME
386E 060F      05960          LD     B,15               ;MAX LENGTH FOR FILENAME
3870 0E00      05970          LD     C,0
3872 EF        05980          RST    28H
3873 78        05990          LD     A,B
3874 FE00      06000          CP     0
3876 CA6C39    06010          JP     Z,EXT              ;NOTHING ENTERED ERROR
3879 48        06020          LD     C,B
387A 0600      06030          LD     B,0
387C 09        06040          ADD    HL,BC
387D 3A1632    06050          LD     A,(DD1-1)
3880 77        06060          LD     (HL),A
3881 23        06070          INC    HL
3882 3A1732    06080          LD     A,(DD1)
3885 77        06090          LD     (HL),A
3886 23        06100          INC    HL
3887 3E0D      06110          LD     A,0DH
3889 77        06120          LD     (HL),A
388A 3E4E      06130          LD     A,FSPEC            ;PUT FILESPEC INTO MFCB1
388C 21EA39    06140          LD     HL,FN              ;BUFFER CONTAINING FILENAME
388F 116A3A    06150          LD     DE,FCB2            ;MY FCB
3892 EF        06160          RST    28H
3893 281D      06170          JR     Z,CNT1
3895 0607      06180          LD     B,7                ;TONE
3897 CD3E34    06190          CALL   BEEP
389A 21A238    06200          LD     HL,FERR            ;FILENAME ERROR
389D 3E0A      06210          LD     A,DSPLY
389F EF        06220          RST    28H
38A0 18BC      06230          JR     DIRDNE
38A2 46        06240 FERR     DEFM   'Filename Error.'
     69 6C 65 6E 61 6D 65 20 45 72 72 6F 72 2E
38B1 0D        06250          DEFB   0DH
38B2 21EA39    06260 CNT1     LD     HL,FN              ;CORRECTED FILE NAME FROM INPUT
38B5 116A3A    06270          LD     DE,FCB2            ;SEARCH BUFFER FOR DISK SEARCH
38B8 0608      06280          LD     B,8
38BA 1A        06290 LK1      LD     A,(DE)
38BB FE03      06300          CP     3
38BD 280B      06310          JR     Z,SF
38BF FE2F      06320          CP     '/'
38C1 2807      06330          JR     Z,SF               ;CHANGE FILE/CMD
38C3 77        06340          LD     (HL),A             ;TO     FILE    CMD
38C4 23        06350          INC    HL
38C5 13        06360          INC    DE
38C6 10F2      06370          DJNZ   LK1
38C8 1812      06380          JR     NEXTZ
38CA 3620      06390 SF       LD     (HL),32            ;PAD WITH BLANKS
38CC 23        06400          INC    HL
38CD 10FB      06410          DJNZ   SF
38CF FE03      06420          CP     3
38D1 2009      06430          JR     NZ,NEXTZ
38D3 0603      06440          LD     B,3
```

```
38D5 3620      06450 LK2      LD     (HL),32
38D7 23        06460          INC    HL
38D8 10FB      06470          DJNZ   LK2
38DA 1808      06480          JR     NXTA1
38DC 0603      06490 NEXTZ    LD     B,3                ;FILE EXTENT
38DE 13        06500 NEXTZA   INC    DE
38DF 1A        06510          LD     A,(DE)
38E0 77        06520          LD     (HL),A
38E1 23        06530          INC    HL
38E2 10FA      06540          DJNZ   NEXTZA
38E4 3A6732    06550 NXTA1    LD     A,(MNSEC)          ;BEGINNING OF DIRECTOR
38E7 320739    06560          LD     (MNSEC1+1),A
38EA 3A6A32    06570          LD     A,(DIRT)
38ED 320F39    06580          LD     (QDIRT1+1),A
38F0 3A6432    06590 NXT2     LD     A,(SEC)
38F3 3D        06600          DEC    A
38F4 FEFF      06610          CP     0FFH
38F6 2008      06620          JR     NZ,OK1B
38F8 3A6532    06630          LD     A,(TRK)
38FB 3D        06640          DEC    A
38FC 326532    06650          LD     (TRK),A
38FF 3A6632    06660          LD     A,(MSEC)
3902 3D        06670          DEC    A
3903 326432    06680 OK1B     LD     (SEC),A
3906 FE00      06690 MNSEC1   CP     0
3908 C21639    06700          JP     NZ,OK1C
390B 3A6532    06710          LD     A,(TRK)
390E FE00      06720 QDIRT1   CP     0
3910 CA5539    06730          JP     Z,NFOUND
3913 3A6432    06740          LD     A,(SEC)
3916 210030    06750 OK1C     LD     HL,BUFFER          ;SECTOR BUFFER
3919 5F        06760          LD     E,A                ;SECTOR
391A 3A6532    06770          LD     A,(TRK)
3910 00        06780 DBL3     NOP
391E 57        06790          LD     D,A                ;TRACK
391F 3A6932    06800          LD     A,(SDRV)
3922 4F        06810          LD     C,A                ;DRIVE
3923 CD7042    06820          CALL   DODISK
3926 C26C39    06830          JP     NZ,EXT             ;EXIT IF ERROR
3929 7E        06840 NL1A     LD     A,(HL)
392A CB77      06850          BIT    6,A
392C 201B      06860          JR     NZ,NXT3A
392E CB67      06870          BIT    4,A
3930 2002      06880          JR     NZ,NL1
3932 1815      06890          JR     NXT3A
3934 010500    06900 NL1      LD     BC,5               ;START OF FILENAME
3937 E5        06910          PUSH   HL                 ;SAVE HL
3938 09        06920          ADD    HL,BC
3939 060B      06930          LD     B,11               ;NUMBER OF CHARACTERS
393B 11EA39    06940          LD     DE,FN              ;FILE NAME TO FIND
393E 1A        06950 NL       LD     A,(DE)             ;FIRST LETTER IN A
393F BE        06960          CP     (HL)               ;FILE NAME IN DIRECTOR
3940 2006      06970          JR     NZ,NXT3            ;TRY NEXT ONE
3942 13        06980          INC    DE                 ;NEXT LETTER
3943 23        06990          INC    HL                 ; "     "
3944 10F8      07000          DJNZ   NL                 ;LOOP
3946 1860      07010          JR     FOUND              ;MATCHED
3948 E1        07020 NXT3     POP    HL                 ;NO MATCH
3949 012000    07030 NXT3A    LD     BC,32              ;NEXT DIRECTORY RECORD
394C 09        07040          ADD    HL,BC
394D 7C        07050          LD     A,H
394E FEEF      07060 EOB2     CP     0EFH
3950 CAF038    07070          JP     Z,NXT2             ;END OF SECTOR GET NEXT ONE
3953 18D4      07080          JR     NL1A               ;COMPARE NEXT RECORD
3955 3E0A      07090 NFOUND   LD     A,DSPLY
3957 210E39    07100          LD     HL,NMAT
395A EF        07110          RST    28H
395B 0604      07120          LD     B,4
395D CD3E34    07130          CALL   BEEP
3960 219A39    07140 PETR     LD     HL,EXITQ
3963 3E0A      07150          LD     A,DSPLY
3965 EF        07160          RST    28H
3966 CD4234    07170          CALL   GETKEY
3969 C3F835    07180          JP     BQDIR
396C CBFF      07190 EXT      SET    7,A
396E CBF7      07200          SET    6,A
3970 4F        07210          LD     C,A
3971 C5        07220          PUSH   BC
3972 0E0A      07230          LD     C,0AH
3974 3E02      07240          LD     A,DSP
3976 EF        07250          RST    28H
```

```
3977 3E02       07260        LD      A,DSP
3979 EF         07270        RST     28H
397A C1         07280        POP     BC
397B 3E1A       07290        LD      A,ERROR                  ;ERROR SVC
397D EF         07300        RST     28H
397E 3E0A       07310 EXT1AB LD      A,DSPLY
3980 219A39     07320        LD      HL,EXITQ
3983 EF         07330        RST     28H
3984 0600       07340        LD      B,0
3986 CD3E34     07350        CALL    BEEP
3989 3E01       07360        LD      A,KEY
398B EF         07370        RST     28H
398C FE0D       07380        CP      13
398E CAF835     07390        JP      Z,BQDIR
3991 FE50       07400        CP      80
3993 2802       07410        JR      Z,QUIT
3995 18E7       07420        JR      EXT1AB           ;ASK AGAIN
3997 3E16       07430 QUIT   LD      A,EXIT
3999 EF         07440        RST     28H
399A 0A         07450 EXITQ  DEFB    0AH
399B 50         07460        DEFM    'PRESS <ENTER> TO CONTINUE'
     52 45 53 53 20 3C 45 4E 54 45 52 3E 20 54 4F 20
     43 4F 4E 54 49 4E 55 45
39B4 03         07470        DEFB    3
39B5 21D539     07480 FOUND  LD      HL,FNDM
39B8 3E0A       07490        LD      A,DSPLY
39BA EF         07500        RST     28H
39BB E1         07510        POP     HL
39BC CDB742     07520        CALL    BSFCH
39BF C26C39     07530        JP      NZ,EXT
39C2 C32F35     07540        JP      SEARCH
39C5 0000       07550 COUNT  DEFW    0
39C7 43         07560 COPYQ  DEFM    'COPY?'
     4F 50 59 3F
39CC 03         07570        DEFB    3
39CD 0000       07580 CSEC   DEFW    0                ;NUMBER OF CONTIGUOUS SECTORS
39CF 0000       07590 STRK   DEFW    0                ;STARTING TRACK
39D1 0000       07600 SSEC   DEFW    0                ;STARTING SECTOR+GRANULE OFFSET
39D3 0000       07610 LENGTH DEFW    0                ;TOTAL LENGTH OF FILE
39D5 0A0A       07620 FNDM   DEFW    0A0AH
39D7 46         07630        DEFM    'Found.'
     6F 75 6E 64 2E
39DD 0D         07640        DEFB    0DH
39DE 0A0A       07650 NMAT   DEFW    0A0AH
39E0 4E         07660        DEFM    'No Match.'
     6F 20 4D 61 74 63 68 2E
39E9 0D         07670        DEFB    0DH
001E            07680 FN     DEFS    30               ;FILENAME PLUS 1
3A08 1C1F       07690 DIR    DEFW    1F1CH            ;CLS
3A0A 4E         07700        DEFM    'NAME: '
     41 4D 45 3A 20
0008            07710 NAME   DEFS    8
3A18 20         07720        DEFM    '     DATE: '
     20 20 20 20 44 41 54 45 3A 20
0008            07730 DATE   DEFS    8
3A2B 0A0D       07740        DEFW    0D0AH
3A2D 20         07750 B3     DEFM    '           '
     20 20 20 20 20 20 20 20 20 20
3A38 20         07760        DEFM    '     '
     20 20 20 20
3A3D 03         07770        DEFB    3
3A3E 0A0A       07780 FIL    DEFW    0A0AH
3A40 46         07790        DEFM    'Filename:'
     69 6C 65 6E 61 6D 65 3A
3A49 03         07800        DEFB    03H
0020            07810 MFCB1  DEFS    32
0020            07820 FCB2   DEFS    32
3A8A 0000       07830 WFCB   DEFW    0
0100            07840 BUFFR1 DEFS    256
3B8C 1601       07850 CMPARE LD      D,1              ;IF TRACK ONE = TRACK 0
3B8E 1E02       07860        LD      E,2                ;THEN DOUBLE STEP
3B90 3A6932     07870        LD      A,(SDRV)
3B93 4F         07880        LD      C,A              ;DRIVE
3B94 218C3A     07890        LD      HL,BUFFR1
3B97 3E31       07900        LD      A,RDSEC          ;READ SECTOR
3B99 EF         07910        RST     28H
3B9A 0600       07920        LD      B,0              ;256
3B9C 118C3A     07930        LD      DE,BUFFR1
3B9F 210030     07940        LD      HL,BUFFER
3BA2 ED52       07950 COMP1  SBC     HL,DE
3BA4 2008       07960        JR      NZ,NOTSM
```

```
3BA6 23         07970        INC     HL
3BA7 13         07980        INC     DE
3BA8 10F8       07990        DJNZ    COMP1
3BAA 3E87       08000        LD      A,87H      ;DOUBLE STEP FOR READING 40TRKS ON A 80
3BAC 1802       08010        JR      BG1
3BAE 3E00       08020 NOTSM  LD      A,0
3BB0 32CD37     08030 BG1    LD      (DBL1),A
3BB3 321D38     08040        LD      (DBL2),A
3BB6 321D39     08050        LD      (DBL3),A
3BB9 326943     08060        LD      (DBL4),A
3BBC 32CD43     08070        LD      (DBL5),A
3BBF C9         08080        RET
3BC0 0A0A       08090 DIREC1 DEFW    0A0AH
3BC2 44         08100        DEFM    'DRIVE :'
     52 49 56 45 20 3A
3BC9 03         08110        DEFB    3
3BCA 3E0A       08120 DIRECT LD      A,DSPLY
3BCC 21C03B     08130        LD      HL,DIREC1
3BCF EF         08140        RST     28H
3BD0 3A0032     08150        LD      A,(SD1)
3BD3 32DA3B     08160        LD      (DIREC2+1),A
3BD6 CD4234     08170        CALL    GETKEY
3BD9 FE00       08180 DIREC2 CP      0
3BDB CA153D     08190        JP      Z,OK1DIR
3BDE F5         08200        PUSH    AF
3BDF 3E02       08210        LD      A,DSP
3BE1 0E0D       08220        LD      C,0DH
3BE3 EF         08230        RST     28H
3BE4 F1         08240        POP     AF
3BE5 D630       08250        SUB     30H
3BE7 4F         08260        LD      C,A
3BE8 3E21       08270        LD      A,CKDRV
3BEA EF         08280        RST     28H
3BEB C26C39     08290        JP      NZ,EXT
3BEE 3E51       08300        LD      A,GTDCT
3BF0 EF         08310        RST     28H
3BF1 C5         08320        PUSH    BC
3BF2 FD6E06     08330        LD      L,(IY+6)
3BF5 2C         08340        INC     L
3BF6 2600       08350        LD      H,0
3BF8 119532     08360        LD      DE,WMES2
3BFB 3E61       08370        LD      A,HEXDEC
3BFD EF         08380        RST     28H
3BFE 219532     08390        LD      HL,WMES2
3C01 11CA3C     08400        LD      DE,CATM+31
3C04 010500     08410        LD      BC,5
3C07 EDB0       08420        LDIR
3C09 C1         08430        POP     BC
3C0A FD5609     08440        LD      D,(IY+9)
3C0D 210030     08450        LD      HL,BUFFER
3C10 3E55       08460        LD      A,RDSSC
3C12 1E00       08470        LD      E,0
3C14 EF         08480        RST     28H
3C15 3E30       08490        LD      A,30H
3C17 81         08500        ADD     A,C
3C18 32B43C     08510        LD      (CATM+9),A
3C1B C5         08520        PUSH    BC
3C1C 21D030     08530        LD      HL,BUFFER+0D0H  ;DISK NAME
3C1F 11B73C     08540        LD      DE,CATM+12
3C22 010800     08550        LD      BC,8
3C25 EDB0       08560        LDIR
3C27 13         08570        INC     DE
3C28 13         08580        INC     DE
3C29 010800     08590        LD      BC,8
3C2C EDB0       08600        LDIR
3C2E C1         08610        POP     BC
3C2F 41         08620        LD      B,C
3C30 0EFF       08630        LD      C,255           ;GET FREE SPACE
3C32 21A73C     08640        LD      HL,FSTD
3C35 3E23       08650        LD      A,RAMDIR
3C37 EF         08660        RST     28H
3C38 C5         08670        PUSH    BC
3C39 119532     08680        LD      DE,WMES2
3C3C 2AA93C     08690        LD      HL,(FSTD+2)     ;FREE SPACE
3C3F 3E61       08700        LD      A,HEXDEC
3C41 EF         08710        RST     28H
3C42 11E13C     08720        LD      DE,CATM+54
3C45 219532     08730        LD      HL,WMES2
3C48 010500     08740        LD      BC,5
3C4B EDB0       08750        LDIR
3C4D 2AA73C     08760        LD      HL,(FSTD)
```

```
3C50 ED4BA93C 08770          LD      BC,(FSTD+2)
3C54 09       08780          ADD     HL,BC
3C55 119532   08790          LD      DE,WMES2
3C58 3E61     08800          LD      A,HEXDEC
3C5A EF       08810          RST     28H
3C5B 11ED3C   08820          LD      DE,CATM+66
3C5E 219532   08830          LD      HL,WMES2
3C61 010500   08840          LD      BC,5
3C64 EDB0     08850          LDIR
3C66 21AB3C   08860          LD      HL,CATM
3C69 3E0A     08870          LD      A,DSPLY
3C6B EF       08880          RST     28H
3C6C C1       08890          POP     BC
3C6D 0E00     08900          LD      C,0              ;GET DIRECTORY RECORDS OF VIS FILES
3C6F 2A7D45   08910          LD      HL,(STDIR)
3C72 3E23     08920          LD      A,RAMDIR
3C74 EF       08930          RST     28H
3C75 C26C39   08940          JP      NZ,EXT
3C78 E5       08950          PUSH    HL               ;FILE NAME PRINT - LOOP 3
3C79 0610     08960 FNPL3    LD      B,16
3C7B 7E       08970 FNPL1    LD      A,(HL)           ;FILE NAME PRINT - LOOP 1
3C7C FE3A     08980          CP      ':'
3C7E 2807     08990          JR      Z,FNPL2          ;COLON FOUND
3C80 4F       09000          LD      C,A
3C81 3E02     09010          LD      A,DSP
3C83 EF       09020          RST     28H
3C84 23       09030          INC     HL
3C85 10F4     09040          DJNZ    FNPL1
3C87 3E02     09050 FNPL2    LD      A,DSP            ;FILE NAME PRINT - LOOP 2
3C89 0E20     09060          LD      C,20H
3C8B EF       09070          RST     28H
3C8C 10F9     09080          DJNZ    FNPL2
3C8E E1       09090          POP     HL
3C8F 011600   09100          LD      BC,22
3C92 09       09110          ADD     HL,BC
3C93 3E2B     09120          LD      A,'+'
3C95 BE       09130          CP      (HL)
3C96 2803     09140          JR      Z,OK1DNE
3C98 E5       09150          PUSH    HL
3C99 18DE     09160          JR      FNPL3
3C9B 3E0A     09170 OK1DNE   LD      A,DSPLY
3C9D 21F83C   09180          LD      HL,ENTER1
3CA0 EF       09190          RST     28H
3CA1 CD4234   09200          CALL    GETKEY
3CA4 C3F835   09210          JP      BQDIR
0004          09220 FSTD     DEFS    4
3CAB 1C1F     09230 CATM     DEFW    1F1CH
3CAD 44       09240          DEFM    'Drive :?  ????????  ????????
                                       ?? Cyl, DDEN, Free =   ??.00K /  ???.00K'
     72 69 76 65 20 3A 3F 20 20 3F 3F 3F 3F 3F 3F 3F
     3F 20 20 3F 3F 3F 3F 3F 3F 3F 3F 20 20 20 20 3F
     3F 20 43 79 6C 2C 20 44 44 45 4E 2C 20 46 72 65
     65 20 3D 20 20 20 3F 3F 2E 30 30 4B 20 2F 20 20
     20 3F 3F 3F 2E 30 30 4B
3CF6 0A0D     09250          DEFW    0D0AH
3CF8 0A0A     09260 ENTER1   DEFW    0A0AH
3CFA 50       09270          DEFM    'PRESS <ENTER> TO CONTINUE'
     52 45 53 53 20 3C 45 4E 54 45 52 3E 20 54 4F 20
     43 4F 4E 54 49 4E 55 45
3D13 0D       09280          DEFB    0DH
0001          09290 OK1BUF   DEFS    1
3D15 3A5E3B   09300 OK1DIR   LD      A,(DIRDNE)
3D18 32143D   09310          LD      (OK1BUF),A
3D1B 3EC9     09320          LD      A,0C9H
3D1D 325E3B   09330          LD      (DIRDNE),A
3D20 CDBE37   09340          CALL    RDIR
3D23 3A143D   09350          LD      A,(OK1BUF)
3D26 325E3B   09360          LD      (DIRDNE),A
3D29 C39B3C   09370          JP      OK1DNE
3D2C 210541   09380 FREE     LD      HL,FREEM1+30
3D2F 3A6332   09390          LD      A,(SFLAG)
3D32 C630     09400          ADD     A,30H
3D34 77       09410          LD      (HL),A
3D35 3A6332   09420          LD      A,(SFLAG)
3D38 FE01     09430          CP      1
3D3A 2804     09440          JR      Z,F1S
3D3C 3E90     09450          LD      A,144
3D3E 1802     09460          JR      FRCNT
3D40 3E48     09470 F1S      LD      A,72
3D42 328B3D   09480 FRCNT    LD      (CFLT+1),A
3D45 21433E   09490          LD      HL,FREEM2+7
3D48 3A0032   09500          LD      A,(SD1)
3D4B 77       09510          LD      (HL),A
3D4C CD0842   09520          CALL    FRESET
3D4F 210030   09530          LD      HL,BUFFER
3D52 3A6A32   09540          LD      A,(DIRT)
3D55 57       09550          LD      D,A
3D56 3A6832   09560          LD      A,(DIRB)
3D59 5F       09570          LD      E,A
3D5A 3A6932   09580          LD      A,(SDRV)
3D5D 4F       09590          LD      C,A
3D5E CD7042   09600          CALL    DODISK           ;READ NEWDOS GAT TABLE
3D61 21D030   09610          LD      HL,BUFFER+0D0H   ;DISK NAME
3D64 11463E   09620          LD      DE,FREEM2+10
3D67 010800   09630          LD      BC,8
3D6A EDB0     09640          LDIR
3D6C 13       09650          INC     DE
3D6D 13       09660          INC     DE
3D6E 010800   09670          LD      BC,8
3D71 EDB0     09680          LDIR
3D73 110030   09690          LD      DE,BUFFER
3D76 21E43E   09700          LD      HL,FREEM3+8
3D79 1A       09710 NB0      LD      A,(DE)
3D7A CB47     09720          BIT     0,A
3D7C 2802     09730          JR      Z,NB1
3D7E 3678     09740          LD      (HL),'x'
3D80 23       09750 NB1      INC     HL
3D81 CB4F     09760          BIT     1,A
3D83 2802     09770          JR      Z,NB2
3D85 3678     09780          LD      (HL),'x'
3D87 23       09790 NB2      INC     HL
3D88 13       09800          INC     DE
3D89 7B       09810          LD      A,E
3D8A FE00     09820 CFLT     CP      0
3D8C 2813     09830          JR      Z,LTF
3D8E 7E       09840          LD      A,(HL)
3D8F FE04     09850          CP      4
3D91 2808     09860          JR      Z,NBS
3D93 FE05     09870          CP      5
3D95 280D     09880          JR      Z,PRFREE
3D97 23       09890          INC     HL
3D98 23       09900          INC     HL
3D99 18DE     09910          JR      NB0
3D9B 010B00   09920 NBS      LD      BC,11
3D9E 09       09930          ADD     HL,BC
3D9F 18D8     09940          JR      NB0
3DA1 CD1C42   09950 LTF      CALL    CLROUT
3DA4 CD4442   09960 PRFREE   CALL    FRESPC
3DA7 11353E   09970          LD      DE,FREBUF
3DAA 3E61     09980          LD      A,HEXDEC
3DAC EF       09990          RST     28H
3DAD 11693E   10000          LD      DE,FREEM2+45
3DB0 21353E   10010          LD      HL,FREBUF
3DB3 010300   10020          LD      BC,3
3DB6 EDB0     10030          LDIR
3DB8 13       10040          INC     DE
3DB9 010200   10050          LD      BC,2
3DBC EDB0     10060          LDIR
3DBE CD2942   10070          CALL    TOTSPC
3DC1 11353E   10080          LD      DE,FREBUF
3DC4 3E61     10090          LD      A,HEXDEC
3DC6 EF       10100          RST     28H
3DC7 11733E   10110          LD      DE,FREEM2+55
3DCA 21353E   10120          LD      HL,FREBUF
3DCD 010300   10130          LD      BC,3
3DD0 EDB0     10140          LDIR
3DD2 13       10150          INC     DE
3DD3 010200   10160          LD      BC,2
3DD6 EDB0     10170          LDIR
3DD8 210030   10180          LD      HL,BUFFER
3DDB 3A6932   10190          LD      A,(SDRV)
3DDE 4F       10200          LD      C,A
3DDF 3A6832   10210          LD      A,(DIRB)
3DE2 3C       10220          INC     A
3DE3 5F       10230          LD      E,A
3DE4 3A6A32   10240          LD      A,(DIRT)
3DE7 57       10250          LD      D,A
3DE8 CD7042   10260          CALL    DODISK                   ;LOAD IN HIT TABLE
3DEB 3A6132   10270          LD      A,(LDIR)                 ;DIR LENGTH IN GRANS
3DEE 4F       10280          LD      C,A
3DEF 1E05     10290          LD      E,5                      ;5 SECTORS PER GRAN
3DF1 3E5A     10300          LD      A,MUL8
3DF3 EF       10310          RST     28H
```

```
3DF4 D602     10320         SUB     2                  ;-2, 1 FOR GAT,1 FOR HIT
3DF6 4F       10330         LD      C,A                ;NUMBER OF DIR RECORDS
3DF7 210800   10340         LD      HL,8               ;8 FILES ON EACH RECORO
3DFA 3E5B     10350         LD      A,MUL16
3DFC EF       10360         RST     28H
3DFD 65       10370         LD      H,L
3DFE 6F       10380         LD      L,A                ;TOTAL NUMBER OF FILES
3DFF 22333E   10390         LD      (LDIR1),HL
3E02 3E61     10400         LD      A,HEXDEC
3E04 11353E   10410         LD      DE,FREBUF
3E07 EF       10420         RST     28H
3E08 21373E   10430         LD      HL,FREBUF+2
3E0B 11883E   10440         LD      DE,FREEM2+76
3E0E 010300   10450         LD      BC,3
3E11 EDB0     10460         LDIR
3E13 2A333E   10470         LD      HL,(LDIR1)
3E16 CD5942   10480         CALL    FREDIR
3E19 11353E   10490         LD      DE,FREBUF
3E1C 3E61     10500         LD      A,HEXDEC
3E1E EF       10510         RST     28H
3E1F 21373E   10520         LD      HL,FREBUF+2
3E22 11843E   10530         LD      DE,FREEM2+72
3E25 010300   10540         LD      BC,3
3E28 EDB0     10550         LDIR
3E2A 213A3E   10560         LD      HL,FREEM
3E2D 3E0A     10570         LD      A,DSPLY
3E2F EF       10580         RST     28H
3E30 C36039   10590         JP      PETR
0002          10600 LDIR1   DEFS    2
0005          10610 FREBUF  DEFS    5
3E3A 1C1F     10620 FREEM   DEFW    1F1CH
3E3C 44       10630 FREEM2  DEFM    'Drive :?  ????????  ????????   Free Space =  ???.??K/  ???.??K  Files = ???/??? '
     72 69 76 65 20 3A 3F 20 20 3F 3F 3F 3F 3F 3F 3F
     3F 20 20 3F 3F 3F 3F 3F 3F 3F 3F 20 20 20 46 72
     65 65 20 53 70 61 63 65 20 3D 20 20 3F 3F 3F 2E
     3F 3F 4B 2F 20 20 3F 3F 3F 2E 3F 3F 4B 20 20 46
     69 6C 65 73 20 3D 20 3F 3F 3F 2F 3F 3F 3F 20
3E8C 2D       10640         DEFM    '------------------------------------------------------------------------------- '
     2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
     2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
     2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
     2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
     2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 20
3EDC 20       10650 FREEM3  DEFM    '  0- 17 ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..'
     20 30 2D 20 31 37 20 2E 2E 20 20 2E 2E 20 20 2E
     2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E
     2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E
     2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E
     2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E
3F2A 04       10660         DEFB    4
3F2B 20       10670         DEFM    '   18- 35 ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..'
     20 20 31 38 2D 20 33 35 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E
3F7B 04       10680         DEFB    4
3F7C 20       10690         DEFM    '   36- 53 ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..'
     20 20 33 36 2D 20 35 33 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E
3FCC 04       10700         DEFB    4
3FCD 20       10710         DEFM    '   54- 71 ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..'
     20 20 35 34 2D 20 37 31 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E
4010 04       10720         DEFB    4
401E 20       10730         DEFM    '   72- 89 ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..'
     20 20 37 32 2D 20 38 39 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E
406E 04       10740         DEFB    4
406F 20       10750         DEFM    '   90-107 ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..'
     20 20 39 30 2D 31 30 37 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E
40BF 04         10760         DEFB    4
40C0 20         10770         DEFM    ' 108-125 .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..'
     20 31 30 38 2D 31 32 35 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E
4110 04         10780         DEFB    4
4111 20         10790         DEFM    ' 126-143 .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..'
     20 31 32 36 2D 31 34 33 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E 20
     20 2E 2E 20 20 2E 2E 20 20 2E 2E 20 20 2E 2E
4161 0505       10800         DEFW    0505H
4163 0505       10810         DEFW    0505H
4165 20         10820         DEFM    ' ---------------------------------------------------------------------------------'
     20 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
     2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
     2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
     2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
     2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D
     20
41B7 54         10830 FREEM1  DEFM    'Type =>  5" Floppy    Heads = ?   Density = DOUBLE   Note - 1 Position =  1.25K '
     79 70 65 20 3D 3E 20 20 35 22 20 46 6C 6F 70 70
     79 20 20 20 20 48 65 61 64 73 20 3D 20 3F 20 20
     20 44 65 6E 73 69 74 79 20 3D 20 44 4F 55 42 4C
     45 20 20 20 4E 6F 74 65 20 2D 20 31 20 50 6F 73
     69 74 69 6F 6E 20 3D 20 20 31 2E 32 35 4B 20
4207 0D         10840         DEFB    0DH
4208 21DC3E     10850 FRESET  LD      HL,FREEM3
420B 23         10860 FRLP1   INC     HL
420C 7E         10870         LD      A,(HL)
420D FE05       10880         CP      5
420F C8         10890         RET     Z
4210 FE78       10900         CP      'x'
4212 2804       10910         JR      Z,MP
4214 FE80       10920         CP      80H
4216 20F3       10930         JR      NZ,FRLP1
4218 362E       10940 MP      LD      (HL),'.'
421A 18EF       10950 NMP     JR      FRLP1
421C 23         10960 CLROUT  INC     HL
421D 7E         10970         LD      A,(HL)
421E FE05       10980         CP      5
4220 C8         10990         RET     Z
4221 FE2E       11000         CP      '.'
4223 20F7       11010         JR      NZ,CLROUT
4225 3680       11020         LD      (HL),80H
4227 18F3       11030         JR      CLROUT
4229 11DC3E     11040 TOTSPC  LD      DE,FREEM3
422C 210000     11050         LD      HL,0
422F 017D00     11060         LD      BC,125
4232 13         11070 FRLP3   INC     DE
4233 1A         11080         LD      A,(DE)
4234 FE05       11090         CP      5
4236 C8         11100         RET     Z
4237 FE2E       11110         CP      '.'
4239 2806       11120         JR      Z,TOTADD
423B FE78       11130         CP      'x'
423D 2802       11140         JR      Z,TOTADD
423F 18F1       11150         JR      FRLP3
4241 09         11160 TOTADD  ADD     HL,BC
4242 18EE       11170         JR      FRLP3
4244 11DC3E     11180 FRESPC  LD      DE,FREEM3
4247 210000     11190         LD      HL,0
424A 017D00     11200         LD      BC,125
424D 13         11210 FRLP2   INC     DE
424E 1A         11220         LD      A,(DE)
424F FE05       11230         CP      5
4251 C8         11240         RET     Z
4252 FE2E       11250         CP      '.'
4254 2001       11260         JR      NZ,NAP
4256 09         11270         ADD     HL,BC
4257 18F4       11280 NAP     JR      FRLP2
4259 110030     11290 FREDIR  LD      DE,BUFFER
425C 210000     11300         LD      HL,0
425F 1A         11310 FRDR1   LD      A,(DE)
4260 FE00       11320         CP      0
```

```
4262 2801     11330        JR     Z,NFROR
4264 23       11340        INC    HL
4265 13       11350 NFROR  INC    DE
4266 7B       11360        LD     A,E
4267 FE00     11370        CP     0
4269 20F4     11380        JR     NZ,FROR1
426B C9       11390        RET
0002          11400 BUFSAV DEFS   2
0002          11410 SECCYL DEFS   2
4270 226C42   11420 DODISK LD     (BUFSAV),HL
4273 ED536E42 11430        LD     (SECCYL),DE
4277 C5       11440        PUSH   BC
4278 3A6A32   11450        LD     A,(DIRT)
427B 47       11460        LD     B,A
427C 7A       11470        LD     A,D
427D 90       11480        SUB    B
427E 67       11490        LD     H,A              ;DIRECTORY TRACK-RAM
427F 3A6B32   11500        LD     A,(DIRB)
4282 47       11510        LD     B,A
4283 7B       11520        LD     A,E
4284 90       11530        SUB    B
4285 6F       11540        LD     L,A              ;DIRECTORY SECTOR-RAM
4286 7C       11550        LD     A,H
4287 4F       11560        LD     C,A
4288 3A6632   11570        LD     A,(MSEC)
428B 5F       11580        LD     E,A
428C 3E5A     11590        LD     A,MUL8
428E EF       11600        RST    28H
428F 85       11610        ADD    A,L
4290 67       11620        LD     H,A
4291 2E00     11630        LD     L,0
4293 010060   11640        LD     BC,DIRBUF
4296 09       11650        ADD    HL,BC
4297 3A5F32   11660        LD     A,(EODR)
429A 94       11670        SUB    H
429B 2004     11680        JR     NZ,NEOD
429D 3E11     11690        LD     A,17
429F 180B     11700        JR     RETDD
42A1 ED5B6C42 11710 NEOD   LD     DE,(BUFSAV)
42A5 010001   11720        LD     BC,256
42A8 EDB0     11730        LDIR
42AA 3E00     11740        LD     A,0              ;-NOERROR
42AC C1       11750 RETDD  POP    BC
42AD 2A6C42   11760        LD     HL,(BUFSAV)
42B0 ED5B6E42 11770        LD     DE,(SECCYL)
42B4 FE00     11780        CP     0
42B6 C9       11790        RET
42B7 114A3A   11800 BSFCH  LD     DE,MFCB1
42BA 012000   11810        LD     BC,32
42BD EDB0     11820        LDIR
42BF 214A3A   11830        LD     HL,MFCB1         ;MY FCB1
42C2 011400   11840        LD     BC,20
42C5 09       11850        ADD    HL,BC            ;FILE LENGTH BYTES
42C6 5E       11860        LD     E,(HL)
42C7 23       11870        INC    HL
42C8 56       11880        LD     D,(HL)
42C9 ED53D339 11890        LD     (LENGTH),DE      ;SAVE
42CD 23       11900        INC    HL               ;NOW POINTS TO FIRST EXTENT
42CE 4E       11910 NFXDE1 LD     C,(HL)           ;TRACK (10 SEC/TRACK-NEWDOS)
42CF E5       11920        PUSH   HL
42D0 210A00   11930        LD     HL,10
42D3 3E5B     11940        LD     A,MUL16
42D5 EF       11950        RST    28H     ;A=C*HL:STARTING SECTOR LESS SECTOR OFFSET
42D6 65       11960        LD     H,L
42D7 6F       11970        LD     L,A
42D8 22CF39   11980        LD     (STRK),HL        ;STORE HERE TEMP.
42DB E1       11990        POP    HL
42DC 23       12000        INC    HL               ;THIS BYTE CONTAINS
42DD 7E       12010        LD     A,(HL)           ; FIRST 5-BITS-NUMBER OF CONTIGUOUS
42DE E61F     12020        AND    31               ; GRANULES LESS ONE
42E0 3C       12030        INC    A
42E1 E5       12040        PUSH   HL
42E2 4F       12050        LD     C,A
42E3 210500   12060        LD     HL,5             ;5 SECTORS PER GRANULE
42E6 3E5B     12070        LD     A,MUL16
42E8 EF       12080        RST    28H
42E9 65       12090        LD     H,L
42EA 6F       12100        LD     L,A              ;HL EQUAL CONTIGUOUS SECTORS
42EB 22CD39   12110        LD     (CSEC),HL        ;SAVE
42EE E1       12120        POP    HL
42EF 7E       12130        LD     A,(HL)           ;FIRST 3-BITS-STARTING GRANULE
```

```
42F0 228A3A   12140        LD     (WFCB),HL        ;SAVE HL POINTER
42F3 E6E0     12150        AND    224              ;OFFSET FROM START (TRACK/SECTOR)
42F5 07       12160        RLCA                    ;EG. 1010000
42F6 07       12170        RLCA
42F7 07       12180        RLCA                    ;BECOMES 0000101
42F8 4F       12190        LD     C,A              ;5 SECTORS=1 GRAN
42F9 1E05     12200        LD     E,5
42FB 3E5A     12210        LD     A,MUL8
42FD EF       12220        RST    28H              ;A=SECTOR OFFSET
42FE 5F       12230        LD     E,A
42FF 1600     12240        LD     D,0
4301 2ACF39   12250        LD     HL,(STRK)        ;STARTING SECTOR LESS SECTOR OFFSET
4304 19       12260        ADD    HL,DE            ;ACTUAL STARTING SECTOR IN HL
4305 3A6632   12270        LD     A,(MSEC)         ;SECTORS PER TRACK
4308 4F       12280        LD     C,A
4309 3E5E     12290        LD     A,DIV16
430B EF       12300        RST    28H
430C 32D139   12310        LD     (SSEC),A         ;A=REMAINDER-SECTOR
430F 7D       12320        LD     A,L              ;HL=QUOTIENT-TRACK NUMBER
4310 32CF39   12330        LD     (STRK),A
4313 C34243   12340        JP     BCOPY
4316 0F       12350 CSM1   DEFB   15
4317 20       12360        DEFM   ' Copying Record '
     43 6F 70 79 69 6E 67 20 52 65 63 6F 72 64 20
4327 03       12370 CSM2   DEFB   3
4328 0000     12380        DEFW   0
432A 00       12390        DEFB   0
432B 48       12400        DEFM   'H'
432C 1818     12410        DEFW   1818H
432E 1818     12420        DEFW   1818H
4330 1818     12430        DEFW   1818H
4332 1818     12440        DEFW   1818H
4334 1818     12450        DEFW   1818H
4336 1818     12460        DEFW   1818H
4338 1818     12470        DEFW   1818H
433A 1818     12480        DEFW   1818H
433C 18       12490        DEFB   18H
433D 1818     12500        DEFW   1818H
433F 1818     12510        DEFW   1818H
4341 03       12520        DEFB   3
4342 3E3A     12530 BCOPY  LD     A,INIT           ;OPEN/INITIALIZE FILE
4344 210030   12540        LD     HL,BUFFER
4347 116A3A   12550        LD     DE,FCB2
434A 0600     12560        LD     B,0              ;256 LRL
434C EF       12570        RST    28H
434D ED5BC539 12580 RLP1   LD     DE,(COUNT)
4351 13       12590        INC    DE
4352 ED53C539 12600        LD     (COUNT),DE
4356 212743   12610        LD     HL,CSM2
4359 3E63     12620        LD     A,HEX16
435B EF       12630        RST    28H
435C 211643   12640        LD     HL,CSM1
435F 3E0A     12650        LD     A,DSPLY
4361 EF       12660        RST    28H
4362 3AD139   12670        LD     A,(SSEC)
4365 5F       12680        LD     E,A              ;SECTOR
4366 3ACF39   12690        LD     A,(STRK)
4369 00       12700 DBL4   NOP
436A 57       12710        LD     D,A              ;TRACK
436B 3A6932   12720        LD     A,(SDRV)
436E 4F       12730        LD     C,A              ;DRIVE 1
436F 210030   12740        LD     HL,BUFFER
4372 3E31     12750        LD     A,RDSEC
4374 EF       12760        RST    28H
4375 C0       12770        RET    NZ
4376 3E4B     12780        LD     A,WRITE          ;WRITE RECORD TO FILE
4378 116A3A   12790        LD     DE,FCB2
437B EF       12800        RST    28H
437C C0       12810        RET    NZ
437D 2ACD39   12820        LD     HL,(CSEC)
4380 2B       12830        DEC    HL
4381 22CD39   12840        LD     (CSEC),HL
4384 7D       12850        LD     A,L
4385 84       12860        ADD    A,H
4386 FE00     12870        CP     0
4388 281B     12880        JR     Z,NFXDE
438A 3AD139   12890 RLP2   LD     A,(SSEC)
438D 3C       12900        INC    A
438E 32D139   12910        LD     (SSEC),A
4391 FE00     12920 RSM    CP     0                ;MAX SECTORS PER TRACK
4393 C24D43   12930        JP     NZ,RLP1
```

```
4396 3E00      12940          LD     A,0
4398 32D139    12950          LD     (SSEC),A
439B 3ACF39    12960          LD     A,(STRK)
439E 3C        12970          INC    A
439F 32CF39    12980          LD     (STRK),A
43A2 C34D43    12990          JP     RLP1
43A5 2A8A3A    13000 NFXDE    LD     HL,(WFCB)
43A8 23        13010          INC    HL
43A9 7E        13020          LD     A,(HL)
43AA FEFF      13030          CP     0FFH
43AC CAFC43    13040          JP     Z,CLEND
43AF FEFE      13050          CP     0FEH
43B1 2803      13060          JR     Z,NRECD
43B3 C3CE42    13070          JP     NFXDE1
43B6 23        13080 NRECD    INC    HL
43B7 7E        13090          LD     A,(HL)
43B8 E607      13100          AND    7        ;LAST 3 BITS POINT DIRECTOR SECTOR LESS 2
43BA C602      13110          ADD    A,2
43BC 47        13120          LD     B,A
43BD 3A6832    13130          LD     A,(DIRB)
43C0 80        13140          ADD    A,B
43C1 E5        13150          PUSH   HL
43C2 5F        13160          LD     E,A              ;SECTOR
43C3 3A6932    13170          LD     A,(SDRV)
43C6 4F        13180          LD     C,A              ;DRIVE NO
43C7 210030    13190          LD     HL,BUFFER
43CA 3A6532    13200          LD     A,(TRK)
43CD 00        13210 DBLS     NOP
43CE 57        13220          LD     D,A
43CF CD7042    13230          CALL   DODISK
43D2 C0        13240          RET    NZ
43D3 E1        13250          POP    HL
43D4 7E        13260          LD     A,(HL)   ;POINTS TO NEXT DIRECTORY EXTENT
43D5 E6E0      13270          AND    224      ;1ST 3 BITS POINT TO DIRECTOR RECORD
43D7 07        13280          RLCA            ;EG. 1010000
43D8 07        13290          RLCA
43D9 07        13300          RLCA                     ;BECOMES 0000101
43DA 4F        13310          LD     C,A
43DB 1E20      13320          LD     E,32
43DD 3E5A      13330          LD     A,MUL8
43DF EF        13340          RST    28H              ;FIND DIRECTOR RECORD
43E0 5F        13350          LD     E,A
43E1 1600      13360          LD     D,0
43E3 210030    13370          LD     HL,BUFFER
43E6 19        13380          ADD    HL,DE
43E7 114A3A    13390          LD     DE,MFCB1
43EA 012000    13400          LD     BC,32
43ED EDB0      13410          LDIR
43EF 214A3A    13420          LD     HL,MFCB1
43F2 111500    13430          LD     DE,15H           ;1 BYTE BEFOR 1ST EXTENT
43F5 19        13440          ADD    HL,DE
43F6 228A3A    13450          LD     (WFCB),HL        ;STORE IN MY POINTER
43F9 C3A543    13460          JP     NFXDE
43FC 116A3A    13470 CLEND    LD     DE,FCB2
43FF 3E3C      13480          LD     A,CLOSE          ;CLOSE FILE
4401 EF        13490          RST    28H
4402 110000    13500          LD     DE,0             ;SET SECTOR COUNT TO 0
4405 ED53C539  13510          LD     (COUNT),DE
4409 3E03      13520          LD     A,3
440B 322743    13530          LD     (CSM2),A
440E C9        13540          RET                      ;END OF COPY FILE
440F 211F61    13550 SORT     LD     HL,DIRBUF+11FH   ;RESET BYTE 1F, IN HIT TABLE ******
4412 3600      13560          LD     (HL),0    ;IT'S 14H IN NONSTANDARD FORMAT    ******
4414 210060    13570          LD     HL,DIRBUF        ;BEGINNING OF SORT
4417 ED5B7D45  13580          LD     DE,(STDIR)
441B 3E00      13590 LP1W     LD     A,0              ;BEGINNING OF SORT
441D 12        13600          LD     (DE),A
441E 23        13610          INC    HL
441F 13        13620          INC    DE
4420 3A5F32    13630          LD     A,(EODR)
4423 BC        13640          CP     H
4424 C21B44    13650          JP     NZ,LP1W
4427 0620      13660          LD     B,32
4429 215D45    13670          LD     HL,FBA
442C 36FF      13680 LP1X     LD     (HL),0FFH
442E 23        13690          INC    HL
442F 10FB      13700          DJNZ   LP1X
4431 010001    13710          LD     BC,256
4434 210060    13720          LD     HL,DIRBUF
4437 ED5B7D45  13730          LD     DE,(STDIR)
443B EDB0      13740          LDIR
```

```
443D 24        13750          INC    H
443E 14        13760          INC    D
443F CB66      13770 NRECR    BIT    4,(HL)           ;Z = KILLED
4441 CA7344    13780          JP     Z,NREC
4444 CB7E      13790          BIT    7,(HL)           ;NZ = EXT
4446 C25644    13800          JP     NZ,CIN
4449 CB76      13810          BIT    6,(HL)           ;NZ = SYS
444B C25644    13820          JP     NZ,CIN
444E CB5E      13830          BIT    3,(HL)           ;NZ = INV
4450 C25644    13840          JP     NZ,CIN
4453 C37344    13850          JP     NREC             ;NEXT RECORD
4456 E5        13860 CIN      PUSH   HL
4457 D5        13870          PUSH   DE
4458 7C        13880          LD     A,H              ;H IS DIR SEC IN MEMROY
4459 D662      13890          SUB    62H              ;60 IS GAT,61 IS HIT,A=SECTOR
445B 85        13900          ADD    A,L
445C 6F        13910          LD     L,A
445D ED5B7D45  13920          LD     DE,(STDIR)
4461 14        13930          INC    D
4462 5F        13940          LD     E,A
4463 2661      13950          LD     H,61H            ;GAT TABLE
4465 7E        13960          LD     A,(HL)
4466 12        13970          LD     (DE),A
4467 3600      13980          LD     (HL),0           ;ZERO SINCE COPIED
4469 D1        13990          POP    DE
446A E1        14000          POP    HL
446B 012000    14010          LD     BC,32            ;COPY FILE TO NEW BUF
446E EDB0      14020          LDIR
4470 C37A44    14030          JP     CIE
4473 012000    14040 NREC     LD     BC,32
4476 09        14050          ADD    HL,BC
4477 EB        14060          EX     DE,HL
4478 09        14070          ADD    HL,BC
4479 EB        14080          EX     DE,HL
447A 3A5F32    14090 CIE      LD     A,(EODR)
447D BC        14100          CP     H
447E 2803      14110          JR     Z,NSRTD
4480 C33F44    14120          JP     NRECR
4483 210062    14130 NSRTD    LD     HL,6200H
4486 CB66      14140 NRECR1   BIT    4,(HL)           ;Z = KILLED   -- BEGINING OF SORT
4488 CABD44    14150          JP     Z,NREC1
448B CB7E      14160          BIT    7,(HL)           ;NZ = EXT
448D C2BD44    14170          JP     NZ,NREC1
4490 CB76      14180          BIT    6,(HL)           ;NZ = SYS
4492 C2BD44    14190          JP     NZ,NREC1
4495 CB5E      14200          BIT    3,(HL)           ;NZ = INV
4497 C2BD44    14210          JP     NZ,NREC1
449A E5        14220          PUSH   HL               ;FILE IS VALID
449B 010500    14230          LD     BC,5
449E 09        14240          ADD    HL,BC            ;HL POINTS TO FILE NAME
449F 060B      14250          LD     B,11
44A1 116245    14260          LD     DE,FBA+5         ;DE POINTS TO OLD FILE NAME
44A4 1A        14270 LP2W     LD     A,(DE)
44A5 BE        14280          CP     (HL)
44A6 3814      14290          JR     C,NLTO           ;M=NEG IF A-(HL)= (-) THEN JUMP
44A8 280E      14300          JR     Z,NLP1
44AA 012000    14310          LD     BC,32
44AD E1        14320          POP    HL
44AE 227F45    14330          LD     (FPIH),HL
44B1 115D45    14340          LD     DE,FBA
44B4 EDB0      14350          LDIR                     ;TRANSFER FILE TO FBA
44B6 1809      14360          JR     CIE1
44B8 13        14370 NLP1     INC    DE
44B9 23        14380          INC    HL
44BA 10E8      14390          DJNZ   LP2W
44BC E1        14400 NLTO     POP    HL
44BD 012000    14410 NREC1    LD     BC,32
44C0 09        14420          ADD    HL,BC
44C1 3A5F32    14430 CIE1     LD     A,(EODR)
44C4 BC        14440          CP     H
44C5 2803      14450          JR     Z,CFND
44C7 C38644    14460          JP     NRECR1
44CA 2A7D45    14470 CFND     LD     HL,(STDIR)       ;NO PUT FILE SOMEPLACE
44CD 24        14480          INC    H                ;HIT TABLED
44CE 54        14490          LD     D,H
44CF 14        14500          INC    D                ;END OF HIT
44D0 3E00      14510 TG       LD     A,0              ;0 IS EMPTY
44D2 BE        14520          CP     (HL)
44D3 280C      14530          JR     Z,ZFH            ;ZERO FOUND HERE
44D5 012000    14540          LD     BC,20H
44D8 09        14550          ADD    HL,BC
```

```
44D9 7C        14560           LD      A,H
44DA BA        14570           CP      D
44D8 20F3      14580           JR      NZ,TG              ;TRY AGAIN
44DD 25        14590           DEC     H
44DE 2C        14600           INC     L
44DF 18EF      14610           JR      TG                 ;TRY AGAIN
44E1 EB        14620 ZFH       EX      DE,HL              ;DE NOW POINTS TO EMPTY TABLE
44E2 2A7F45    14630           LD      HL,(FPIH)
44E5 7C        14640           LD      A,H                ;H IS DIR SEC IN MEMROY
44E6 D662      14650           SUB     62H                ;60 IS GAT,61 IS HIT,A=SECTOR
44E8 85        14660           ADD     A,L
44E9 6F        14670           LD      L,A
44EA 2661      14680           LD      H,61H              ;HIT TABLE
44EC 7E        14690           LD      A,(HL)
44ED 3600      14700           LD      (HL),0
44EF 12        14710           LD      (DE),A             ;NEW HIT NOW CONTAINS PROPER ENTRY
44F0 2A7D45    14720           LD      HL,(STDIR)
44F3 7B        14730           LD      A,E                ;SEE PAGE 202 TECH MANUAL
44F4 E61F      14740           AND     1FH
44F6 C602      14750           ADD     A,2
44F8 47        14760           LD      B,A
44F9 7B        14770           LD      A,E
44FA E6E0      14780           AND     0E0H
44FC 4F        14790           LD      C,A
44FD 09        14800           ADD     HL,BC
44FE 115D45    14810           LD      DE,FBA
4501 EB        14820           EX      DE,HL
4502 012000    14830           LD      BC,32
4505 EDB0      14840           LDIR
4507 0620      14850           LD      B,32
4509 215D45    14860           LD      HL,FBA
450C 36FF      14870 LP2X      LD      (HL),0FFH
450E 23        14880           INC     HL
450F 10FB      14890           DJNZ    LP2X
4511 2A7F45    14900           LD      HL,(FPIH)
4514 3600      14910           LD      (HL),0
4516 210061    14920           LD      HL,6100H
4519 3E00      14930 CNHE      LD      A,0
4518 BE        14940           CP      (HL)
451C C28344    14950           JP      NZ,NSRTD
451F 23        14960           INC     HL
4520 3E62      14970           LD      A,62H
4522 BC        14980           CP      H
4523 20F4      14990           JR      NZ,CNHE            ;CHECK NEXT HIT ENTRY
4525 ED5B7D45  15000           LD      DE,(STDIR)
4529 210060    15010           LD      HL,DIRBUF
452C 1A        15020 TDBTS     LD      A,(DE)             ;TRANSFER DIRECTORY BACK TO START
452D 77        15030           LD      (HL),A
452E 23        15040           INC     HL
452F 13        15050           INC     DE
4530 3A5F32    15060           LD      A,(EODR)
4533 BC        15070           CP      H
4534 C22C45    15080           JP      NZ,TDBTS
4537 C9        15090           RET                              ;END OF SORT
4538 3A6032    15100 WRITED    LD      A,(LDIRS) ;NUMBER OF DIR SEC'S - WRITE DIRECTORY
453B 47        15110           LD      B,A
453C 3A6932    15120           LD      A,(SDRV)           ;SOURCE DRIVE
453F 4F        15130           LD      C,A
4540 3A6832    15140           LD      A,(DIRB)           ;START OF DIR SEC'S
4543 5F        15150           LD      E,A
4544 3A6A32    15160           LD      A,(DIRT)           ;DIR TRACK
4547 57        15170           LD      D,A
4548 210060    15180           LD      HL,DIRBUF
454B 3E36      15190 NSOFDW    LD      A,WRSSC            ;WRITE SECTOR
454D EF        15200           RST     28H
454E 7B        15210           LD      A,E
454F 3C        15220           INC     A
4550 FE00      15230 EODFW     CP      0
4552 2002      15240           JR      NZ,IAW
4554 3E00      15250           LD      A,0
4556 5F        15260 IAW       LD      E,A
4557 24        15270           INC     H
4558 10F1      15280           DJNZ    NSOFDW
455A C3F835    15290           JP      BQDIR              ;RETURN FROM RIGHT
0020           15300 FBA       DEFS    32
0002           15310 STDIR     DEFS    2
0002           15320 FPIH      DEFS    2
6000           15330           ORG     6000H
0001           15340 DIRBUF    DEFS    1                  ;DIRECTORY BUFFER
34DE           15350           END     START
00000 TOTAL ERRORS
```

```
AFC     3644  B2      3158  B3      3A2D  BCOPY   4342  BEEP    343E
BG1     3880  BQDIR   35F8  BSFCH   42B7  BUFFER  3000  BUFFR1  3A8C
BUFSAV  426C  CAFF    33CF  CAFLP1  33AB  CATM    3CAB  CB1A    337D
CB1B    338A  CBA     335F  CBAA    3368  CBAAA   336E  CBAHL   3352
CBAM    32F8  CBFA1   33E7  CBFA2   33F7  CBFA3   33F0  CBQM    329B
CFLT    308A  CFNO    44CA  CIE     447A  CIE1    44C1  CIN     4456
CKBRKC  006A  CKDRV   0021  CLEND   43FC  CLOSE   003C  CLROUT  421C
CMNDR   0019  CMPARE  388C  CNHE    4519  CNT1    3882  COMP1   38A2
CONCP   349D  CONER   34AE  CONF1   346A  CONF2   344D  CONFGR  3478
COPYQ   39C7  COUNT   39C5  CQM     3345  CSEC    39CD  CSM1    4316
CSM2    4327  CT      326B  DATE    3A23  DBL1    37CD  DBL2    3810
DBL3    391D  DBL4    4369  DBL5    43CD  DD1     3217  DIR     3A08
DIRB    3268  DIRBUF  6000  DIRDNE  385E  DIREC1  38C0  DIREC2  3BD9
DIRECT  38CA  DIRT    326A  DIV16   005E  DIV8    0050  DODIR   0022
DODISK  4270  DSP     0002  DSPLY   000A  ENDQ    3654  ENDQ1   366D
ENTER   3243  ENTER1  3CF8  EOB1    383A  EOB2    394E  EODF    35E2
EODFW   4550  EODR    325F  ERR     3151  ERROR   001A  EXIT    0016
EXITQ   399A  EXT     396C  EXT1AB  397E  F1S     3D40  FBA     455D
FCB2    3A6A  FERR    38A2  FIL     3A3E  FN      39EA  FNDIR   313F
FNDM    3905  FNPL1   3C7B  FNPL2   3C87  FNPL3   3C79  FOKFC   3399
FOUND   39B5  FPIH    457F  FRCNT   3D42  FRDR1   425F  FREBUF  3E35
FREDIR  4259  FREE    3D2C  FREEM   3E3A  FREEM1  41B7  FREEM2  3E3C
FREEM3  3EDC  FRESET  4208  FRESPC  4244  FRLP1   420B  FRLP2   424D
FRLP3   4232  FSPEC   004E  FSTD    3CA7  G1      3516  GETKEY  3442
GHT1    360A  GTDCT   0051  HEX16   0063  HEX8    0062  HEXDEC  0061
HIGH    0064  IA      35E9  IAW     4556  INIT    003A  KBD     0008
KEY     0001  KEYIN   0009  LDIR    3261  LDIR1   3E33  LDIRS   3260
LENGTH  3903  LK1     388A  LK2     38D5  LKRC    3828  LOF     0040
LP1W    441B  LP1X    442C  LP2W    44A4  LP2X    450C  LPA     34F8
LTF     3DA1  MFCB1   3A4A  MNSEC   3267  MNSEC1  3906  MP      4218
MSEC    3266  MSEC1   3804  MTRK    3262  MUL16   005B  MUL8    005A
NAME    3A10  NAP     4257  NB0     3D79  NB1     3D80  NB2     3D87
NB5     3D9B  NCBF    364C  NDS     35BC  NEOD    42A1  NEXTZ   38DC
NEXTZA  380E  NFHFQ   3410  NFOUND  3955  NFRDR   4265  NFXDE   43A5
NFXDE1  42CE  NL      393E  NL1     3934  NL1A    3929  NLP1    4488
NLTO    44BC  NMAT    390E  NMP     421A  NODIR   3125  NOLOAD  350E
NOTSM   3BAE  NREC    4473  NREC1   448D  NRECD   43B6  NRECR   443F
NRECR1  4486  NSEC    3803  NSOFD   35DD  NSOFDW  454B  NSRTD   4483
NXT     3835  NXT1    3838  NXT2    38F0  NXT3    3948  NXT3A   3949
NXTA1   38E4  OK1     3812  OK1B    3903  OK1BUF  3D14  OK1C    3916
OK1DIR  3D15  OK1DNE  3C9B  OPEN    003B  PETR    3960  PFCAF   33A1
PFILEN  3354  PGPLH   357A  PL1CAF  33E0  POSN    0042  PRFREE  3DA4
PRINT   000E  PRNT    3844  PRT     0006  QDIR1   368B  QDIRT1  390E
QUIT    3997  RAMDIR  0023  RDIR    37BE  RDSEC   0031  RDSSC   0055
READ    0043  REMOV   0039  RETDD   42AC  RLP1    434D  RLP2    438A
RSM     4391  SBCH    35BE  SD1     3200  SDRV    3269  SEARCH  352F
SEC     3264  SECCYL  426E  SF      38CA  SFLAG   3263  SN      310F
SORT    440F  SOUND   0068  SSEC    39D1  START   34DE  STDIR   457D
STRK    39CF  SYSRES  3100  SYSRS1  3112  TDBTS   452C  TG      44D0
TITLE   315B  TOTADD  4241  TOTSPC  4229  TRK     3265  VDCTL   000F
VER     0049  WEOF    004A  WFCB    3A8A  WHERE   341E  WMES    326C
WMES1   3283  WMES2   3295  WRITE   004B  WRITED  4538  WRSSC   0036
ZFH     44E1
```

ZAP TO DIS/CMD

by Jack Decker

If you have tried to use DIS/CMD (the alphabetized directory program for use under NEWDOS/80 that appears on TAS Public Domain Library disk #008, side B) in a configuration that permits more than four logical drives (such as with the RAMDISK program from NORTHERN BYTES Volume 7, Number 4 installed, or with Alan Johnstone's NEWDOS/80 mods installed, or under NEWDOS/80 version 2.5), you have no doubt found that the highest drivespec that DIS/CMD allows is :3. The following one byte change to DIS/CMD will permit it to access up to ten drives (numbered 0 - 9):

DIS/CMD, File Relative Sector 1, byte 4A: change 34 to 3A

This changes a CP 34H instruction to CP 3AH, thus allowing up to ten drives to be accessed, which should be adequate under any conceivable situation.

TRS-80 MODEL I AND SYSTEM 80 SINGLE/DOUBLE DENSITY 5¼"/8" ADAPTER

Designed and Documented by Maurice Abbott
66 Airlie Road, Montmorency, Victoria 3094, Australia
26th October 1986

[Editor's note: The printed circuit board layout for this article is not included because, as Mr. Abbott explains, "I have had a number of PCB's manufactured to enable home constructors to easily build the doubler. Having outlaid cash to have the PCB's made I would like to get my money back before distributing the layout. Many hours of work went into the design of this doubler over a period of about 18 months... I can supply a PCB, including postage, for $20 Australian. With the exchange rate the way it is, that is a bargain." Since Australian dollars can be bought for around 62¢ at this writing, that would put the cost of the Printed Circuit Board at somewhere around $12.40 U.S., which we agree is not a bad price at all. If you wish to order a PCB, send your order directly to Mr. Abbott at the address shown above.]

CONTENTS

DISCLAIMER: No liability is accepted for loss or damage resulting from the use of this information.

1. INTRODUCTION

I have owned a Model I TRS-80 for a number of years and have designed and built a few hardware add-ons to improve my system. I would like to upgrade to a more modern computer but my economic situation will have to improve before that can occur. The hardware construction project I am about to describe will enable Model I and SYSTEM 80 single density disk systems to be upgraded to double density for less than half the cost of an imported American double density adapter (unless you can purchase a discarded Model I or SYSTEM 80 equipped with double density cheaply). An American adapter currently (June 1986) costs about $180 Australian. The adapter described here uses the WD2791 or WD2793 floppy disk controller (FDC), the latest and third generation, Western Digital single density/double density (SD/DD) disk drive controller. The WD2791 or WD2793 includes within the IC data separation and write precompensation for SD/DD, as required, and the ability to interwork with both 5¼" and 8" disk drives.

The adapter PCB plugs into the existing single density controller socket and may require a flying lead or two depending upon the user requirements. The adapter PCB provides a number of switch selectable options to the user or these options may be strapped permanently when this flexibility is not required. LED's can be provided on the front panel of the expansion interface to indicate to the user if SD or DD has been selected or if a 5¼" or an 8" drive has been selected.

The adapter PCB uses two disk controller IC's, the original WD1771 for SD and a WD2791 or WD2793 for DD and optionally SD. Another eight common inexpensive IC's are used to provide the decoding and switching between densities and type of drive. Circuitry is also required to enable the two controller IC's to be connected in parallel. The IC count is less than other DD adapter PCB's, while providing extra facilities. The American adapters use the WD1771 and WD1791 disk controllers with additional circuitry to provide data separation and write precompensation.

This Doubler can be built for approximately $45 (Australian) excluding the FDC chip. The chip can be purchased from Danever Australia Pty. Ltd. by mail order using your credit card for approximately 18 - 41 dollars. The price depends on the chip type (WD2791 or WD2793) and the quantity required. The WD2793 can also be imported directly from B.G. MICRO, P.O. Box 280298, Dallas, Texas, United States of America, Phone (214) 271-5546 to use your credit card, for about 15 - 20 dollars (Australian, June 1986) including duty, in one off quantities.

At the time of writing, the Doubler has been installed with complete success, in the later version Tandy expansion interface, both types of SYSTEM 80 (Video Genie, PMC-80) expansion interface, and the LNW expansion interface for the Model I. The doubler has also been used with 40 and 80 track drives and with the CPU clock frequency increased by two, again with complete success.

2. ADDRESS DECODING

The decoding used to select density and type of drive is compatible with that used in the LNW 5/8 adapter. As far as I'm aware the AEROCOMP and PERCOM doublers use the same decoding for density selection. The adapter has been used with NEWDOS/80 Version 2 and MULTIDOS with satisfactory results. The wait state facility provided by the LNW 5/8 Doubler has not been provided in this 5/8 Doubler.

Selection of single density :
Write F8 or FA or FC or FE into 37EC.
FDC command Register address.

Selection of double density :
Write F9 or FB or FD or FF into 37EC.
FDC command Register address.

Selection of 5¼" drive :
Write C0 into 37EE. Bits 0-5 don't care.
FDC Sector Register address.

Selection of 8" drive :
Write 80 into 37EE. Bits 0-5 don't care.
FDC Sector Register address.

When booting the system, SD or DD and 5¼" or 8" drive are selected by either user accessible switches or by permanent straps. Hardware hackers may prefer to have control of all functions while other users may prefer permanent straps.

The TANDY double density adapter uses different decoding (naturally) to select SD or DD and most DOS's provide a driver to handle the TANDY doubler.

3. DATA ADDRESS MARKS

The FDC provided in the TRS-80 is designed for use with soft-sector formatted floppy disks. In this type of disk operation, the location of the space occupied by a sector is denoted by special data patterns and identification fields written on the disk during the formatting process. The soft sectored format was defined by IBM with the introduction of eight inch disk drives. A variation of this format is used in the TRS-80. The special data pattern consists of an index mark, track and sector identification, a data address mark (DAM), a sector of data and checksum fields (CRC's). There is a gap between each sector to enable the FDC to separate the end of one sector from the identification field of the next sector.

The data address mark was defined by IBM to be one of two values, FB to indicate that the sector contains data, or F8 to indicate the data in the sector has been deleted. While designing the WD1771 FDC, Western Digital allocated an extra two DAM's (F9 and FA) by using a spare bit in the write sector command. When the WD1791/WD279X was designed, this bit was reclaimed to provide additional facilities. Therefore the WD1791/WD279X can only generate the FB and F8 DAM's.

Tandy used DAM's FA and FB to define the directory sectors and non directory sectors respectively when Model I TRSDOS was designed. This may have been an unintentional deviation due to an error in the WD1771 data sheet that reversed the two bits identifying the DAM read from the disk.

Data address marks have caused compatibility problems since doublers were introduced into the Model I. The incompatibility problems may occur when attempting to interchange Model I SD disks with a Model III due to the Model III using a WD1793 FDC (DAM's FB and F8) and the Model I a WD1771 FDC (DAM's FB and FA).

The latest versions of MULTIDOS, DOSPLUS and LDOS use DAM's FB and F8 in both Model I and Model III and therefore the problem described above has been overcome. Model I NEWDOS/80

Version 2 will use DAM's F8 and FB if the SYSTEM command parameter BN = Y, this should be the normal value used for BN. TRSDOS 2.3 uses DAM's FA and FB, and therefore a WD1771 must be used with this DOS unless the DOS has been modified (nobody uses it anyhow).

The American magazine 80-MICRO published a hardware article in the December 1982 issue describing construction of a Model I expansion interface. The expansion interface disk controller PCB consisted of a large board with a smaller extension PCB containing the two FDC's and some associated logic. It appears that the designer originally produced the board with only a WD1791 and no extension PCB. It is my guess that upon firing up the original version, the designer was very surprised to find difficulty in reading (TRSDOS) SD disks due to the previously discussed DAM incompatibility problems. Because the PCB was produced using a taped layout and no space was available for the additional FDC, the mezzanine extension board was introduced. Only a small section of the PCB layout needed to be rearranged to connect to the extension PCB with the additional circuitry. The preceding ramble is my own opinion because no valid reason was given in the article for the extension PCB.

Since I have been using the Doubler I have found that Super Utility Plus insists that two FDC'S must be accessible before it will configure for DD. SU+ appears to write and read toward both FDC's in SD and DD to check the type of Doubler, if any, installed. Therefore, if only the WD279X was provided for both SD and DD, SU+ would not operate in DD.

I have never seen an article on how to build a Doubler for the Model I in any computer magazines or newsletters, so even after all these years, this may still be a first.

4. CIRCUIT DESCRIPTION

The Doubler uses a WD1771 (IC1) for SD operation connected in parallel with a WD2791 or WD2793 (IC2) for DD and/or SD and the associated address decoding to control two flipflops (IC4) for selection of density and type of drive. By inverting the read and write data for the WD2793, bus transceiver IC10 enables a WD2793 to be used in lieu of a WD2791. The WD2793 with a true data bus is the commonly used FDC and is therefore sometimes about half the cost of a WD2791 with an inverted data bus. If the WD2793 can be purchased significantly cheaper than the WD2791, provide IC10 to invert the bus for the WD2793. If not, replace IC10 with straps and use a WD2791, this is the preferred option. The WD1771 uses an inverted bus and hence the existing TRS-80 bus circuitry that the Doubler must interface to, expects the FDC data I/O to be inverted.

The chip select input to each FDC is used to enable the FDC by connecting or disconnecting the FDC data leads to and from the bus. The STEP, DIRC, WG and WD outputs from the FDC to the disk drive bus are not controlled by the FDC chip select input and are therefore always active. To enable these signals to be switched from the selected FDC to the disk drive bus, a quad 2 input noninverting multiplexer, IC3, is used. The switching function of IC3 (pin 1) is controlled by the density selection flipflop IC4a or optionally by a front panel switch during SD operation via NOR gates IC8d and IC8c. The WD2791 or WD2793 is always selected for DD operation by IC4a. During SD operation the WD1771 is selected by 0 volts applied at point "B" on the circuit diagram. The WD2791 or WD2793 is selected for SD if point "B" is open circuit.

When the reset switch is operated, density and type of drive are selected by IC4a and IC4b being preset by MR* via the boot selection switches or permanent straps. It is possible to boot in SD or DD using a 5¼" or an 8" drive. NEWDOS/80 Version 2 enables use of a system disk that is completely DD format, i.e., the first track does not need to be SD (PDRIVE parameter TI=K).

The DOS manipulates IC4 to switch densities and type of drive by writing to the FDC as described in the section on address decoding. A number of inverters (IC7) and open collector inverters (IC5, IC6), arranged in a "wired or" configuration, are used to decode the various data and address conditions to set and reset the flipflops in IC4.

The WD2791 or WD2793 simplifies the design of a FDC by including data separation and write precompensation within the chip. C11 is used to adjust the centre frequency, R14 the write precompensation and R15 the read pulse width. The PUMP circuit consisting of R13, C10 and D1 is used to inhibit over-responsiveness to jitter and to prevent an extremely wide lock-up response by the internal phase-detector. C10 should be 0.1µF for 8" drives or 0.22µF for 5¼" drives or a compromise of 0.15µF if both types of drive are being used.

The INTRQ output from each FDC cannot be connected directly together because the WD2791 or WD2793 uses active high and low output whereas the WD1771 requires a pullup resistor. This is not spelt out clearly in the WD2791 or WD2793 data sheet as a variation from the WD1771 and WD1791 FDC's. Inverters IC9e and IC9f overcome this incompatibility.

Visual indication of SD or DD and 5¼" or 8" selection is available via LED's D3 and D2 driven by OR gates IC8b and IC9c. D3 'on' indicates DD while 'off' indicates SD . D2 'on' indicates 8" while 'off' indicates 5¼".

The type of drive is normally selected via IC4b, however, it is possible to perform this selection from the drive select signal from the disk drive bus. An input via point "C" and IC7a can be used to select the type of drive. A low (0 volts) applied to point "C" selects 8" drive mode.

A disk inhibit switch option is provided to enable the keyboard reset switch to act as in LEVEL II rather than initiating a boot up. This will only work correctly when a WD2793 is used, and if the WD2793 is selected, when the reset switch is operated. This facility is not available when the WD2791 is used.

The TG43* output at point "D" has been provided for drives requiring an indication when the track number is greater than 42. A spare wire in the disk drive bus must be utilised if this facility is required. NOR gate IC8a generates the TG43* signal at point "D" when DD is selected or if the track number is greater than 42.

The WD2791 or WD2793 can accept either a 1 or 2 MHz clock, however, if operation with eight inch drives is required, a 2 MHz clock must be used in lieu of the existing 1 MHz clock. The WD279X has an internal divide by two controlled by the ENMF* FDC input. If a 2 MHz clock is required the divide by two option must be used when accessing a 5¼" drive.

Single density data separation for the WD1771 has not been provided on the adapter PCB. Most TRS-80 users will be using a plug-in SD data separator, this separator can be used with the DD adapter. The DD adapter is plugged into the exiting FDC socket, the SD data separator is plugged into the adapter WD1771 socket and the WD1771 is plugged into the SD data separator. The SYSTEM 80 has an in-built single density data separator for use with the WD1771. This data separator can be utilised in conjunction with the adapter PCB via strap h-i and a flying lead from point "E" to a RAW DATA connection on the SYSTEM 80 expansion interface board. The connections associated with SD data separation (pins 25, 26 and 27) are connected directly to the WD1771 on the adapter PCB, therefore SD operation is unchanged with the Doubler installed. The WD279X provides SD data separation internally and can be utilised where a Model I data separator is not available.

Testpoints 1, 2 and 3 (TP1, TP2, TP3) have been provided to enable easy access to connection points used during the initial adjustment of the adapter in conjunction with the test link. Refer to the adjustment section for details.

5. CONSTRUCTION

The double density adapter is built using a 6" x 4" single sided PCB. The PCB was designed with the aid of a Computer Aided Design (CAD) software package. Updating of the PCB during development of the adapter was made easy by the editing facilities of the CAD package. The use of a single sided PCB results in a larger sized board than if a double sided board had been designed. However, a single sided board is cheaper to manufacture and is easier for the home constructor to handle. The PCB requires the drilling of 393 1/32" holes before construction can start. Reasonable care must be taken when handling the PCB due to the small track width used to enable some of the tracks to pass between IC pads. A PCB for the Doubler is available from the author.

Before commencing construction of the Doubler it is worth having a look at the insides of your expansion interface to determine the most suitable connection and mounting method for the Doubler in your particular box. The board was originally designed for my 1978 TRS-80 Model I but will also fit into both versions of the SYSTEM 80 expansion interface and the LNW expansion interface. There is a lot of space above the TRS-80 expansion interface board allowing the Doubler to be plugged directly into the existing FDC socket. A SD data separator may also be plugged into the Doubler providing that clearance between boards is adjusted accordingly during construction.

The main board in both SYSTEM 80 expansion interfaces is mounted with only about 20 mm clearance between the top side of the board and the case. This makes plugging the Doubler into the

the FDC socket difficult because of the height of the crystal and some 0.1 µF disc ceramic bypass capacitors mounted on the main board. If the crystal is removed and placed on the other side of the main board and the existing bypass capacitors are replaced with modern monolithic 0.1 µF ceramic capacitors, the Doubler may be plugged into the existing FDC socket. If you do not wish to touch your SYSTEM 80 expansion interface main board, the doubler may be connected to the main board using a 40 wire ribbon cable and a 40 pin DIL insulation displacement connector. The ribbon cable is soldered directly to pads on the under side of the Doubler PCB. The multicolour ribbon cable with a very thin clear plastic covering is the best type for this application as it allows the wires to be easily separated for soldering. Rod Irving, Melbourne, stocks this type of cable. The cable should be as short as possible. I have not as yet used the doubler with a speed up modification.

The SYSTEM 80 FDC socket can also pose a problem due to some sockets being the type that come up the side of the FDC chip. Normal 40 pin IDC or solder type headers will not plug into this type of socket because the pins are too short. Where the Doubler plugs directly into the FDC socket, this socket problem can be overcome by using long square pins cut from a wire wrap socket to provide the Doubler to FDC connection. Where the ribbon cable connection option is to be used, a 40 pin wire wrap socket with square pins can be pruned to provide a interface between the 40 pin IDC DIL header and the main board FDC socket, i.e. insert the wire wrap socket into the FDC socket of the main board and then insert the IDC header into wire wrap socket. The wire wrap socket pins must be long enough to provide a good connection but short enough to ensure adequate clearance between the IDC header and the expansion interface case.

Once you have selected the way the Doubler is to be connected to the main board, commence construction of the Doubler by inserting all straps. The straps can be bare wire where there is no likelihood of short circuits. Component pigtail offcuts are good for this purpose. Where the straps are close together, insulated fine stiff single conductor wire should be used. Insert all IC sockets, except the WD1771 socket, (good quality only) and check for correct orientation before soldering. Mount and solder resistors, capacitors and diodes, noting that polarities are correct where applicable. Insert the trimmer capacitor (C11) and trimpots (R14, R15) and set according to Appendix 3.

Install the terminal posts on the upper side of the Doubler. I suggest that all terminal posts be provided so that if you change your mind later, e.g. due to acquiring an 8" drive at a bargain price, it can be pressed into service by a simple strapping change. I have found that some stiff component pigtail offcuts make very good strapping terminal posts. The posts can be installed using 1/32 holes (same as for IC's) if pigtail offcuts are used. This is an advantage due to the fine tracks used on the Doubler PCB.

The remaining part of the PCB construction will vary depending upon the type of computer and the selected connection method.

i. TRS-80 Model I

Install the terminal posts on the lower side of the PCB used to connect the 40 pin header for plugging into the existing SD FDC. The under side posts must be long enough to ensure the adapter PCB clears components on the main expansion interface PCB. Install the WD1771 socket and solder the 40 pin solder connection header to pins. Insert the header into a 40 pin socket during soldering to prevent misalignment of pins due to the application heat. Ensure that alignment is accurate before soldering the header to the posts. The PCB holes may require enlargement to accommodate the posts you are using, however, minimum sized holes must be used.

ii. SYSTEM 80 Direct (Socket up side of FDC chip)

Drill PCB to take posts of wire wrap socket, insert into PCB until required length of post protrudes on the lower side of the PCB. Solder posts and then cut posts on the upper side of the PCB as close as possible to the surface of the PCB. Insert the WD1771 FDC socket into the PCB and solder. Take care to ensure that the posts protruding on the lower side of the PCB are not bent. Refer to an earlier section of this document for changes to the main board when using this method.

iii SYSTEM 80 Extension cable

Insert the WD1771 FDC socket and solder in place. Decide where the Doubler is to be mounted (at the back is OK) and determine the length of 40 wire cable required and cut a piece to length. Install the 40 pin IDC header on one end of the length of cable. Strip back the thin plastic sheath on the other end of the cable about 50 mm. Strip and tin the individual wires to enable the correct wires to be soldered to the PCB pads. After double checking the header orientation and wire to pad connections complete the remaining soldering.

iv. SYSTEM 80 (Direct) (Normal FDC socket)

Insert the WD1771 FDC socket into the PCB and solder. A 40 pin solder connection header is used to connect the PCB to the main board. The header connection tags must be tapered to reduce the likelihood of track short circuits occurring. If the solder tag has two points, break one off to taper the tag. Insert the header into a 40 pin socket during soldering to prevent misalignment of pins due to the application of heat. The space between the header and the PCB must be kept to a minimum.

All Methods

Depending upon the options selected and the flexibility required (refer to strapping option section), insert straps and/or prepare sections of rainbow cable to provide switch and LED options. Check track side of PCB for cracks and short circuits. Mask off the component side of the PCB and apply a coat of lacquer to the track side of the board.

6. INSTALLATION

Disconnect the power before inserting or removing IC's or the adapter PCB and handle the FDC IC's with care. Check your work and then plug the adapter into the expansion interface. Connect switches and LED's as required. If the Doubler is being installed in a SYSTEM 80 connect the RAW DATA input to the point designated in the terminal post section. Connect the 2 MHz lead if 8" drives are to be used. Verify power potentials on each IC socket before inserting any IC's. Insert all IC's except the FDC's and check that the LED's indicate the correct response when the reset button is operated. Insert the single density FDC into the correct FDC socket of the adapter, power up, and check that a single density system disk will boot using the WD1771 FDC.

The WD2791 or WD2793 can now be inserted in the adapter and if the necessary Cathode Ray Oscilloscope and frequency counter are available, the adapter can be adjusted as per the relevant section in this document. If a double density system disk is available, attempt to boot using this disk. Adjust C11 to obtain boot-up if it is not achieved initially. Proper adjustment will need to be performed when the test equipment is available.

When the adapter has been adjusted, verify all facilities provided and check operation with your software. GOOD LUCK !!

7. PARTS LIST

IC's

IC1	WD1771	Single density disk controller.
IC2	WD2791 or 2793	Single/double density disk controller.
IC3	74LS157	Quad 2 input multiplexer, noninverting.
IC4	74LS74	Dual D flip flop.
IC5	74LS05	Hex inverter, open collector.
IC6	74LS05	Hex inverter, open collector.
IC7	74LS04	Hex inverter.
IC8	74LS02	Quad two input NOR gate.
IC9	74LS05	Hex inverter, open collector.
IC10	74LS640	Octal transceiver, 3 state, inverting.

Danever Australia Pty. Ltd. 03 5985622 for WD2791 or WD2793 (a significant discount for quantities of 10 2793's is offered) [This note for the benefit of our readers in Australia and New Zealand -editor].

Resistors

R1, R2, R3, R4, R5,	4.7K ohm, 5%, ¼W.
R6, R7, R8, R9, R10.	4.7K ohm, 5%, ¼W.
R11, R12	470 ohm, 5%, ¼W.
R13	1K ohm, 5%, ¼W.
R14	10K Trimpot, horizontal PCB, Cermet.
R15	50K Trimpot, horizontal PCB, Cermet.
R16	330 ohm, 5%, ¼W.

Capacitors

C1, C2, C3, C4, C5, C6	0.1 µF Monolithic bypass.
C7, C8	100 pF ceramic disk.
C9	47 µF 10 volt electrolytic.

C10	0.22 µF ceramic disk or Greencap.
C11	4.5-60 pF trimmer (Phillips), or Murata 9.8-60 pF to provide a lower profile when installing the doubler in a SYSTEM 80.

Diodes

D1	1N914
D2, D3 (Optional)	Red LED. SD/DD & 5¼/8" indicators.

Sockets

14 pin DIL	quantity 6.
16 pin DIL	quantity 1.
20 pin DIL	quantity 1. Required if WD2793 used.
40 pin DIL	quantity 2.

Headers

40 pin DIL, solder connections - quantity 1.

The System 80 interface may require a 40 wire extension cable if the adapter card will not plug directly into the existing 1771 socket due to lack of space.

Depending on the type of socket used for the existing FDC a 40 pin wire wrap socket with square pins may also be required. If this is the case, the extension cable method of connection will be used. Refer to construction section.

Miscellaneous

Terminal posts	several required.
Printed Circuit Board	- quantity 1 (PCB available from the author at address shown at start of this article).

Wire for straps (wire wrap).
Rainbow ribbon cable for switches and LED's.
40 wire ribbon cable. May be required for SYSTEM 80.
Switches. (Optional).

Disk inhibit	- SPST
Boot SD/DD	- SPDT
Boot 5¼"/8"	- SPDT
SD 1771/279X	- SPST

8. ADJUSTMENTS

Write precompensation:

(a) Set up for Double Density 5¼" drive, disconnect the test link and power up the system, i.e. set boot switch to DD and 5¼/8" switch to 5¼".

(b) Insert the test link and observe the pulse width on TP1 using a Cathode Ray Oscilloscope.

(c) Adjust the write precompensation trimpot R14 for a pulse width of 125-150 (or as specified for your drives) nanoseconds at TP1. Remove the test link.

Data Separator:

(a) Set up for Double Density 5¼" drive, disconnect the test link and power up the system, i.e. set boot switch to DD and 5¼/8" switch to 5¼".

(b) Insert test link and observe pulse width on TP2 using a Cathode Ray Oscilloscope.

(c) Adjust read pulse width trimpot R15 to obtain a pulse width of 500 nanoseconds at TP2.

(d) Connect a frequency counter to TP3 and adjust trimmer C11 to obtain a frequency of 250 KHz.

(e) Remove test link and set switches normally.

9. DESCRIPTION OF TERMINAL POSTS

Testpoints:

TP1	WD pin 31 of 279X, write precompensation.
TP2	TG43 pin 29 of 279X, data separator.
TP3	DIRC pin 16 of 279X, centre frequency.

Posts:

A.	2 M Hz clock input for 8" drive applications.
B.	WD1771 or WD279X selection for single density.
C.	Selection of 5¼" or 8" via drive select.
D.	TG43* to 8" drive via disk drive bus cable.
E.	Input for RAW DATA when installed in a SYSTEM 80.
F.	SD/DD LED indicator.
G.	5¼"/8" LED indicator.

10. STRAPPING OPTIONS

Selection of 1 or 2 MHz clock.

1 MHz	insert a-b, remove c-d.
2 MHz	insert c-d, remove a-b.

Selection of 5¼" or 8" drive operation.

DOS	insert e-f, remove e-g.
Drive select	insert e-g, remove e-f.

SYSTEM 80 or TRS-80 option.

TRS-80	insert h-i.
SYSTEM 80	remove h-i.

Boot Density Selection. May be a switch.

Double Density	insert x-y, remove y-z.
Single Density	insert y-z, remove x-y.

Boot Drive Type Selection. May be a switch.

5¼"	insert v-w, remove u-v.
8"	insert u-v, remove v-w.

Disk Disable.

Insert j-k if a WD2791 FDC is used or if the facility is not required when a WD2793 is used.

11. NOTES (Refer to circuit diagram).

Note 1.

A 2 MHz clock must be obtained via a flying lead from the main expansion interface PCB when 8" drives are to be used.

Connect Adapter PCB point "A" to:
(i) TRS-80 early interface (Buffered cable): Z28 pin 3
(ii) TRS-80 later interface: Z25 pin 3
(iii) SYSTEM 80 early interface, X-4010: Z8 pin 3
(iv) SYSTEM 80 later interface, X-4020: Z54 pin 8 or J2 pin 2

Where only 5¼" drives are to be used the flying lead from point "A" is not required. A 1 MHz clock is used by inserting strap a-b and removing strap c-d.

Note 2.

Selection of 5¼" or 8" drive at power up is determined by the boot switch position. By inserting strap e-g and removing e-f 5¼" or 8" can be selected by a flying lead from "C" to the drive select signals. The 5¼/8 switch and/or the 5¼/8 flipflop manipulated by the DOS are not used with the above straps. Normally type of drive is selected via the DOS.

Note 3.

TG43 is provided for use with 8" drives if required. A spare wire in the disk drive bus cable is used to connect this signal to the 8" drive(s).

Note 4.

S1 closed selects the WD1771 for SD operation and the WD279X for DD. S1 open selects the WD279X for both SD and DD operation. A strap can be substituted to always select the WD1771 for SD.

Note 5.

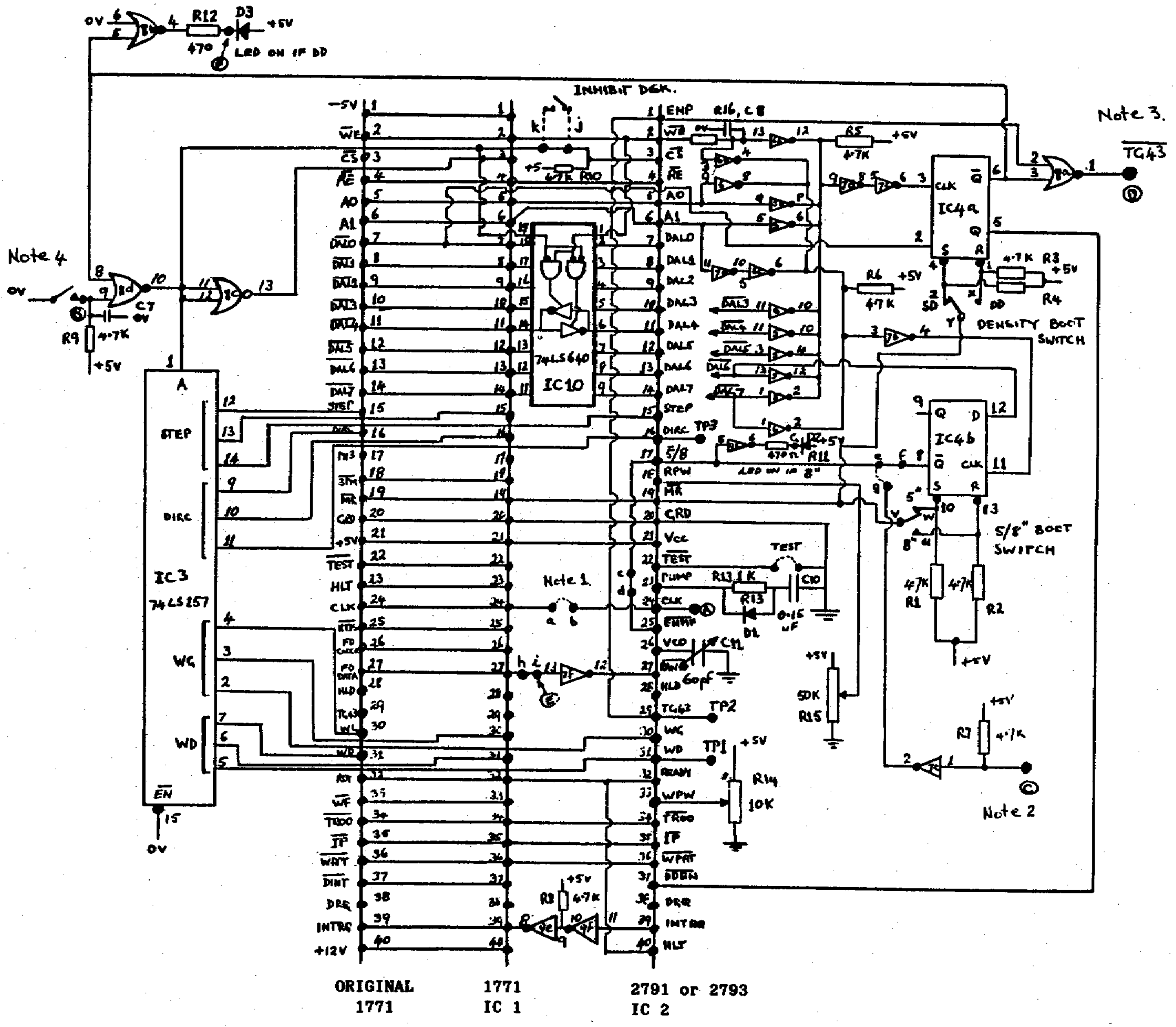
A single density external data separator for the TRS-80 WD1771 is provided, if required, by plugging the existing SD data separator into the conversion card WD1771 socket. The WD1771 would be required when reading a single density disk of another user and the data address marks cannot be read by the WD2791 or WD2793.

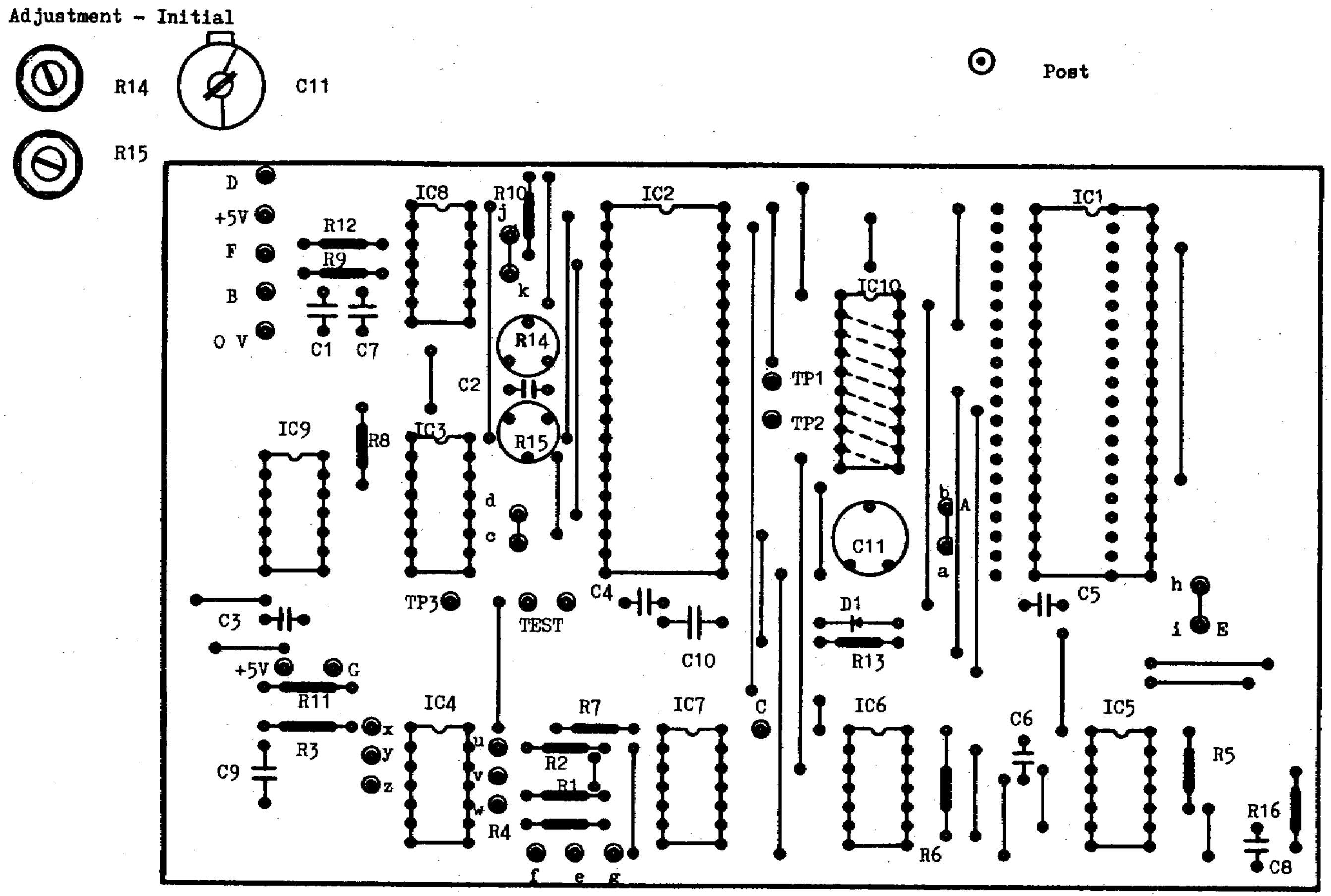
The SYSTEM 80 has a built in SD data separator and this may be utilised by removing strap h-i on the adapter PCB and connecting a flying lead from "E" to Z11 pin 13 (X-4010) or Z58 pin 2 / J1 pin 2 (X-4020) of the expansion interface. This connection provides an alternative path for RAW DATA to the WD279X, bypassing the SYSTEM 80 SD data separator.

The WD279X contains built-in data separators and write precompensation for both SD and DD operation. It may therefore be possible to get by without a data separator for the WD1771 in the TRS 80.

APPENDIX 1: TRS-80 MODEL I SS/DD 5¼"/8" PLUG IN CONVERSION CARD

APPENDIX 1: TRS-80 MODEL I SS/DD 5¼"/8" PLUG IN CONVERSION CARD





APPENDIX 3: DOUBLE DENSITY ADAPTER – TOP VIEW (COMPONENT SIDE)

APPENDIX 3: DOUBLE DENSITY ADAPTER - TOP VIEW (COMPONENT SIDE)

TRSPRE - A TRS-80 BASIC PRE-PROCESSOR

by Gil Spencer
Box 300, Spit Junction, New South Wales 2088, AUSTRALIA
Phone: (02) 969-7060 (IDDD 011+61+2+969-7060) / Ham Radio: VK2JK

Adapted for the TRS-80 from: "A BASIC Preprocessor" by James L. Shearer (Microsystems Volume 5, Number 5 - May, 1984).

BASIC programs are hard to maintain because of the inherently poor structure, the GOTOs and GOSUBs, and the remarks and white space that slow performance and use up memory. Also, the incessant line numbers clutter up the code and are tough to type.

The "TRSPRE" preprocessor overcomes most of these shortcomings. The BASIC source program can be written without line numbers, using your favorite editor such as your word processor or text editor. As TRSPRE is currently written, your text editor must be able to generate a caret, "^", ASCII 94, as a spec character. The original version (not written with TRS-80 BASIC in mind) used "@" for this spec char, but this conflicts with PRINT@, etc. in TRS80 BASIC. When writing your BASIC source, use alphanumeric labels (example: ^branch1) for GOTO, GOSUB, and other branching statements.

When you run the preprocessor (TRSPRE), it strips away the REM statements and unnecessary white space, inserts BASIC line numbers, substituting the appropriate ones for labels, and produces a file of compact interpreter-readable code.

The syntactical requirements for this preprocessor are few. The BASIC source code must have the correct syntax according to the requirements your BASIC version. The preprocessor, TRSPRE, does no syntax checking. Line numbers must NOT be used. Where GOTO, GOSUB and other branching statements indicate the need for a line number, replace it with a label named in a manner to help clarify your code. The labels are 8-character strings whose first character must be the caret, "^", and whose subsequent characters may be letters, numerals, or virgules, "/".

Examples:

Acceptable	Unacceptable
`^PRINT/1`	`^WRITE#4 (# is illegal character)`
`^EXIT`	`EXIT/6 (no leading ^)`
`^age/sr`	`^POSITION/35 (too long)`

Actually, labels longer than 8 characters are allowed, but only the first 8 characters are recognised.

REM statements that are to be skipped by this preprocessor must have REM or rem as the FIRST string on the line, i.e. TRSPRE will not skip REM statements if they are imbedded within the line.

Examples:

```
REM This is a valid usage of the REM function.
rem This is also valid.
' This is NOT a valid REM.
^TEST PRINT"Example of invalid REM":REM Invalid imbedded "REM"
    REM Invalid "REM" as it is preceeded by white space.
```

USAGE

First, compile TRSPRE/C on a C compiler like Alcor's. Result will be a compiled runtime file, say, TRSPRE/OBJ. TRSPRE/OBJ can be run with a runtime utility like Alcor's RUNC/CMD. Alternatively, TRSPRE/OBJ can be optimized and converted to a CMD file to operate direct from DOS.

Second, write your BASIC "source" program as described, being sure NOT to use line numbers. Use plenty of white space and remarks. They won't slow your program.

Third, save your program using an extension like /SRC or /PRE (anything but /BAS). Suppose your source is named SAMPLE/PRE. Run the preprocessor by typing from DOS:

With object file (TRSPRE/OBJ):

```
RUNC TRSPRE SAMPLE/PRE SAMPLE/BAS
```

With /CMD file (TRSPRE/CMD):

```
TRSPRE SAMPLE/PRE SAMPLE/BAS
```

Your source file SAMPLE/PRE will be read, the labels will be converted to line numbers, REM and rem and extra white-space eliminated, and the resulting BASIC program will be written to disk as SAMPLE/BAS. The disk file, SAMPLE/BAS, can now be RUN like any other BASIC program and/or SAVEd in tokenized format.

It is also possible to specify the starting line number and the line number increment if the default values of 10 and 10 are not satisfactory.

To process a file to give a starting line number of 100 and the default line number increment of 10, type:

```
[RUNC] TRSPRE SAMPLE/PRE SAMPLE/BAS 100
```

To process a file to give a starting line number of 50 and a line number increment of 25, type:

```
[RUNC] TRSPRE SAMPLE/PRE SAMPLE/BAS 50 25
```

The following files are part of this suite of programs:

TRSPRE/DOC - This document.
TRSPRE/C - The source file for this program converted for Alcor C and the TRS-80 Model III.
SAMPLE/PRE - A sample source for this preprocessor.
SAMPLE/BAS - What you should get when you run SAMPLE/PRE through TRSPRE.

TRSPRE/C

```
/*  TRSPRE    ( TRS-80 BASIC Pre-Processor )    v 1.2-860222  */
/*  ________________________________________________________
   |  This Source written in Alcor's C Language  v.2.01.00  |
   | Hacker:Gil Spencer / Box 300 / Spit Jct NSW 2088 AUSTA |
   |       Phone: 61 (02) 969-7060 / Ham Radio: VK2JK       |
   | Environment:                                           |
   | 48k 2disk TRS80 Mod III / NEWDOS80(2.0) / FAX80 printer|
   |________________________________________________________| */

/* This source was written in another country, on another C,
   for a different BASIC.  It was written on a different
   processor by a guy I never heard of.  I don't even under-
   stand half the Source code!
   Yet, thru the magic of C, the whole thing compiled and
   executed "with a minimum of fuss" with Alcor C for the
   TRS-80 Mod3.  Amazing!  First though, MBPRE/C was filtered
   thru FIXCRLF to remove the #0A (VLF bytes).  Next, MBPRE/C
   was filtered thru PRETTY to make the Source more readable.
   To try it out on your trombone, first, read TRSPRE/DOC.
   Then, compile this file, TRSPRE/C, using the Alcor C
   compiler, CC/CMD, with this command line from DOS:
         cc trspre  (File, "stdio", must be available to CC)
   A new file, TRSPRE/OBJ will be written.  This is the
   executable P-Code.  Now, using the Alcor C runtime file,
   RUNC/CMD, enter this command line from DOS:
         runc trspre sample/pre sample/bas 5000 10
   where runc       = The Alcor C 'runtime' pgm.
         trspre     = The target C object file.
         sample/pre = trspre's target.
         sample/bas = The new destination file.
         5000       = Starting BASIC line number.
         10         = BASIC line increment.
   The file, sample/bas, can then RUN on the TRS80 BASIC
   interpreter and/or SAVEd in token form.  My goodness!      */

/* MBPRE    (Acquired from Sydney Zeta board on 851008)       */
/*************************************************************/
/*                                                            */
/* mbpre.c  Version 1.3 (gs)                                  */
/*                                                            */
/* Microsoft Basic pre-processor                              */
/*                                                            */
/* by James Shearer                                           */
/*                                                            */
/* from Microsystems May, 1984                                */
/*                                                            */
/*************************************************************/

/* 11-Oc-85 Source converted to Alcor C for TRS-80 Mod 3.     */
/*          Label prefix changed from '@' to '^' to avoid     */
/*          stumbles on PRINT@, etc.  Version to 1.3 (gs)     */

/*  6-09-84 Modified to allow user specified starting line    */
/*          number, line number increment and accepts lower   */
/*          "rem" as being a remark.  Version to 1.2  (kw)    */

/*  3-09-84 Modified to show version number within COM file. */
/*          Version number changed to 1.11  (kw)              */
```

```
/* 31-
/*

/* 11-
/*

#includ

#define
#define

char la
int lab
static

MAIN( a
int arg
char *a
{
   int 
   stat
   char
   char
   stat

   IF (
      pr

      pr
      pr
      ex
   }
  f1 = 

/*******
/*  Firs
/*
/*******

    line
    line

    IF (
       l

    IF (
       l

    linen

    WHILE
    {
       g
       I

       {



       }
    }

    fclose
    f1 = f
    IF ( (
       pr

    lineno

/**********
/* Second 
/* essary 
/*
/**********



   WHILE(
   {
      get
```

```
/* 31-08-84  Modified to allow TAB characters within the     */
/*           source file.  Version changed to 1.1.   (kw)    */

/* 11-08-84  This programme written to compile under C/80    */
/*           compiler obtained from Microsystems magazine.   */

#include "stdio"        /* Standard I/O file for Alcor C. */

#define MAXLEN 255              /* Max line length.      */
#define MAXLBL 500              /* Room for 500 labels.  */

char label[ 9 * MAXLBL ];       /* arrays for labels-        */
int labelno[ MAXLBL ];          /* -and their line numbers. */
static int lblcnt = 0;

MAIN( argc, argv ) /* "command line", cf Alcor C book, pg R65.*/
int argc;      /* # of "command line" args. */
char *argv[ ]; /* array of ptrs to "command line" args. */
{
   int length;
   static int f1, f2;
   char line[ MAXLEN ];
   char word[ MAXLEN ];
   static int lineno, linest, lineinc;

   IF ( (argc < 3 ) || ( argc > 5 ) )      {
      printf("\nBASIC pre-processor version 1.3 : 11-Oc-85\n\n"

      printf( "Usage: TRSPRE srcfile/ext:d outfile/ext:d " );
      printf( "[1st line] [line inc]\n" );
      exit( 0 );
    }
   f1 = fileopn( argv[ 1 ] );    /* File to be read.       */

/**************************************************************/
/*  First pass. Assign line nos, build label table, drop REM */
/*                          lines.                            */
/**************************************************************/


    linest = 10;
    lineinc = 10;

    IF ( argc == 5 )
        lineinc = atoi( argv[ 4 ] );

    IF ( argc >= 4 )
        linest = atoi( argv[ 3 ] );

    lineno = linest - lineinc;

    WHILE ( ( length = getln( line, MAXLEN, f1 ) ) > 0 )
    {
        getwrd( word, line );
        IF ( length > 1 && ( strcmp( word, "REM" )
                             && strcmp( word, "rem" ) ) )
        {
            lineno = lineno + lineinc;
            IF ( *line == '^' )
                bldlabel( line, lineno );
        }
    }

    fclose( f1 );
    f1 = fileopn( argv[ 1 ] );
    IF ( ( f2 = fopen( argv[ 2 ], "w" ) ) == NULL )
        printf( "Can't open: %s\n", argv[ 2 ] );

    lineno = linest - lineinc;

/**************************************************************/
/* Second pass. Replace labels with nos, squeeze out unnec-   */
/* essary white space, write statement lines with numbers     */
/*                        to outfile.                         */
/**************************************************************/


    WHILE( ( length = getln( line, MAXLEN, f1 ) ) > 0 )
    {
        getwrd( word, line );
```

```
            IF ( length > 1 && ( strcmp( word, "REM" )
                                 && strcmp( word, "rem" ) ) )
            {
                lineno = lineno + lineinc;
                writeln( line, lineno, f2 );
            }
        }
        finish( f2 );
    }  /* MAIN */

fileopn( fname ) /* Open file fname for binary read. Return-  */
char *fname;    /*  -file number. Print err mesg if no file.  */
    {
    int i;
    i = fopen( fname, "r" );
    IF ( i != NULL )
        RETURN i;
    ELSE
    {
        printf( "Can't open: %s\n", fname );
        exit( 0 );
    }
}  /* fileopn */

getln( s, lim, f1 )         /* Reads line from file f1 to s. */
int lim, f1;                /* Returns line len (0 if EOF).  */
char s[ ];
{
    int c, i;

    i = 0;

    DO ( c = getc( f1 ) );          /* pass white space */
    WHILE ( isspace( c ) );

    IF ( c != EOF && c != '\n' )
    {
        s[ i++ ] = c;
        WHILE ( --lim > 0 && ( c=getc( f1 ) ) != EOF
                          && c != '\n' )
            s[ i++ ] = c;
    }
    IF ( c == '\n' )
        s[ i++ ] = c;
    s[ i ] = '\0';
    RETURN( i );
}  /* getln */

bldlabel( s, n )            /* Put label from line, s, and-    */
char *s;                    /* -line number, n, into arrays.   */
int n;
{
    IF ( lblcnt <= MAXLBL );
    {
        wrdcpy( label + 9 * lblcnt, s );
           /* limit lbl to 8 chars */
            label[ 9 * lblcnt + 8 ] = '\0';
        labelno[ lblcnt ] = n;
        lblcnt = lblcnt + 1;
    }
}  /* bldlabel */

writeln( s, n, f2 ) /* Write line s to outfile f2 inserting-  */
int f2, n;          /* -line number n. Also replace internal- */
char s[ ];          /* -labels with appropriate line numbers. */
{
    int i, j;
    char t[ MAXLEN ];

    i = 0;
    IF ( s[ i ] == '^' ) /* It's a label if 1st char = "^". */
    {
        WHILE ( s[ i ] != ' ' && s[ i++ ] != '\t' ) /* pass it*/
           ;
        WHILE ( isspace( s[ i ] ) )     /* and white space. */
            ++i;
    }
    fprintf( f2, "%d ", n );            /* write line number. */
    WHILE( s[ i ] != '\n' )
    {
        IF ( s[ i ] == '"' )                 /* check for quotes   */
        {
            putc( s[ i ], f2 );              /* write first "      */
            WHILE ( s[ ++i ] != '"' )        /* and all in between*/
                putc( s[ i ], f2 );
            putc( s[ i++ ], f2 );            /* then ending "      */
        }
        ELSE IF ( s[ i ] == '^' )            /* check for label    */
        {
            getwrd( t, &s[ i ] );            /* get label into t   */
            i = i + strlen( t );             /* move ptr past lbl  */
            t[ 8 ] = '\0';                   /* only 8 char signf  */
            FOR ( j = 0; j <= lblcnt; j++ )
                IF ( strcmp( t, label + 9 * j ) == 0 )
                {                    /* write line # if match      */
                    fprintf( f2, "%d ", labelno[ j ] );
                    break;
                }
            IF (j > lblcnt  )                /* no match.          */
            {
                printf( "\n##### No match for label " );
                printf( "%s on line %d\n", t, n );
                fprintf( f2, "?%s?", t );
            }
        }
        ELSE
            putc( s[ i++ ], f2 );            /* 'regular' char.    */
    }
    putc( s[ i ], f2 );                      /* EOL char. */
}  /* writeln */

getwrd( t, s )     /* Get next word from s, put it in t.      */
char s[ ], t[ ];   /* For our purposes, words start with      */
{                  /* "^", or alpha and then alpha, digits,   */
    int i, j;      /* embedded '/' and opt final type char    */

    i = 0;
    IF (s[ i ] == '^' || isalpha( s[ i ] ) )
    {
        ++i;
        WHILE( isalpha( s[ i ] ) || isdigit( s[ i ] )
                                 || s[ i ] == '/' )
            i++;
        IF ( s[ i ] == '%' || s[ i ] == '!' || s[ i ] == '#'
                          || s[ i ] == '$' )
            i++;
    }
    FOR ( j = 0; j <=i; j++ )
        t[ j ] = s[ j ];
    t[ i ] = '\0';
}  /* getwrd */

wrdcpy( s, t )  /* copy next word (see getwrd) from t to s. */
char *s, *t;
{
    WHILE ( isalpha( *t ) || isdigit( *t )
                          || *t == '^' || *t == '/' )
    {
        *s = *t;
        s++;
        t++;
    }
}  /* wrdcopy */

finish( f2 )
int f2;
{
    int j;
    fclose( f2 );
    FOR ( j = 0; j < lblcnt; ++j )
        printf( "\n%4d %10s", labelno[ j ], label +9 *j );
    printf( "\n" );
    exit( 0 );
}  /* finish */

/* End of TRSPRE/C. */
```

SAMPLE/PRE

```
REM   SAMPLE/PRE -A demo pgm to test TRSPRE BASIC preprocessor.
REM   v 1.1 - 851012.
rem   Read about TRSPRE BASIC preprocessor in file TRSPRE/DOC.

REM  ________________________________________________________
REM |     This file written with Alcor's Blaise Text Editor
REM |  Hacker:Gil Spencer / Box 300 / Spit Jct NSW 2088 AUSTA |
REM |        Phone: 61 (02) 969-7060 / Ham Radio: VK2JK       |
REM |  Environment:                                           |
REM |  48k 2disk TRS80 Mod III / NEWDOS80(2.0) / FAX80 printer |
REM |_________________________________________________________|

rem   NOTE: There's a flow control logic error in this pgm,
rem      of the kind that often creeps into BASIC pgms due to
rem      the GOSUBs and GOTOs.  The bug isn't easy to find.  To
rem      force the symptom, enter "0" (exit), then "N" (don't
rem      exit), then "0", (exit).

REM   Establish strings.
go$ = " << Hit any key to continue >> "
YN$ = " << Hit 'Y' or 'N' ONLY, please >> "

REM   fn UC$ = Convert lcalfa to ucalfa ELSE no change.
deffn UC$(Z$)=chr$(asc(Z$)+32*((asc(Z$)>90) AND(asc(Z$)<123)))

rem   MAIN.
   GOSUB ^start   REM SAMPLE/OUT represents what you should
   GOSUB ^body    REM get when you filter SAMPLE/PRE through
   GOSUB ^exit    REM the BASIC preprocessor, TRSPRE.
   END

^start    cls
          FOR X = 0 TO 12 STEP 2
             PRINT@ ( 64 * X ) + ( 2 * X ), "Hello, ";
             PRINT "I'm a sample program!"
          NEXT X
          PRINT@ 960, go$;
          GOSUB ^getcap
      RETURN

^body     PRINT
          PRINT
          INPUT "Enter a number ( 5 thru 100 - 0 to quit ) "; X
          IF X = 0 THEN RETURN
          IF ( X < 5 ) OR ( X > 100 ) THEN GOTO ^wrong
          Y = X [ 2
          Z = x [ 3
          PRINT "Your number is ", X
          PRINT "Its square is ",  Y
          PRINT "Its cube is ",    Z
          print@ 960, go$;
          GOSUB ^getcap
      GOTO ^body

^exit     cls
^exit1    PRINT
          PRINT "Are you sure you want to Quit ( Y/N ) ?"
          GOSUB ^getcap
          IF Z$ = "Y" THEN RETURN
          IF Z$ = "N" THEN GOTO ^body
          PRINT YN$
          GOTO ^exit1

^wrong    PRINT , "<< WRONG! >>"
      GOTO ^body

^getcap  Z$ = inkey$
         IF Z$ = "" THEN GOTO ^getcap ELSE Z$ = fnUC$( Z$ )
         RETURN

REM   End of SAMPLE/PRE.
```

SAMPLE/BAS

```
1000 go$ = " << Hit any key to continue >> "
1010 YN$ = " << Hit 'Y' or 'N' ONLY, please >> "
1020 deffn UC$(Z$)=chr$(asc(Z$)+32*((asc(Z$)>90) AND(asc(Z
$)<123)))
1030 GOSUB 1070    REM SAMPLE/OUT represents what you shou
ld
1040 GOSUB 1150     REM get when you filter SAMPLE/PRE thr
ough
```

```
1050 GOSUB 1280     REM the BASIC preprocessor, TRSPRE.
1060 END
1070 cls
1080 FOR X = 0 TO 12 STEP 2
1090 PRINT@ ( 64 * X ) + ( 2 * X ), "Hello, ";
1100 PRINT "I'm a sample program!"
1110 NEXT X
1120 PRINT@ 960, go$;
1130 GOSUB 1380
1140 RETURN
1150 PRINT
1160 PRINT
1170 INPUT "Enter a number ( 5 thru 100 - 0 to quit ) "; X
1180 IF X = 0 THEN RETURN
1190 IF ( X < 5 ) OR ( X > 100 ) THEN GOTO 1360
1200 Y = X [ 2
1210 Z = x [ 3
1220 PRINT "Your number is ", X
1230 PRINT "Its square is ",  Y
1240 PRINT "Its cube is ",    Z
1250 print@ 960, go$;
1260 GOSUB 1380
1270 GOTO 1150
1280 cls
1290 PRINT
1300 PRINT "Are you sure you want to Quit ( Y/N ) ?"
1310 GOSUB 1380
1320 IF Z$ = "Y" THEN RETURN
1330 IF Z$ = "N" THEN GOTO 1150
1340 PRINT YN$
1350 GOTO 1290
1360 PRINT , "<< WRONG! >>"
1370 GOTO 1150
1380 Z$ = inkey$
1390 IF Z$ = "" THEN GOTO 1380  ELSE Z$ = fnUC$( Z$ )
1400 RETURN
```

ADD EXTERNAL DRIVES TO THE MODEL 4P

by Art Rasmussen

Here is a hardware modification for the Model 4P to allow it to use two external drives. This modification is for the latest version of the 4P (green screen and clustered arrow keys). The main PCB board modification or revision number will be REV - (that is a dash or minus sign, there is no letter designation).

Parts needed:

About 3 feet of 26-30 gauge wire. I used 30 gauge wire wrap wire (Radio Shack part number 278-503).

One 34 position female header connector (Radio Shack part number 276-1525).

Three 34 pin female card-edge connectors (Radio Shack part number 276-1564).

About 3 feet of flat 34 conductor cable (Alpha part number 3580/34).

One extra long (2'-3') drive extender cable--male card-edge on one end and one or two female (276-1564) card-edge connectors on the other. You can get by with only one female connector at the end if you are using two half-height drives in one case.

About 6 inches of 1/4" (inner diameter) tubing.

Remove the 4P from its case, 2 screws on either side and 2 screws under the handle. Remove the drives and drive case from the 4P. Remove the screws holding the bottom metal cover to the computer. This contains the main PCB. Be careful when removing the bottom cover, there are three connectors and a ground wire that are attached to it near the front and they do not lend themselves to easy access for removal. Remove the PCB from the bottom cover.

Cut, strip the ends, and solder the 30 gauge wire to make the following connections:

On the top side (component side) of the PCB:
1. Pin 7 of U34 to pin 1 of U77.
2. Pin 10 of U34 to pin 3 of U77.

On the bottom side of the PCB:
1. Pin 2 of U77 to pin 14 of the disk drive header connector.
2. Pin 4 of U77 to pin 6 of the disk drive header connector.

Replace the PCB and bottom cover. BE CAREFUL. Radio Shack has seen fit to place a stand-off insulator on the bottom of the main chassis which comes very close to the 4P's speaker and it is very easy to smash the speaker when attempting to put the bottom cover and its connectors on... I know, I did!

Now, if you are rich and famous, or at least rich, you may want to replace the Tandon drives with ones a little more reliable... that shouldn't be too hard. The Tandon drives have trouble at a 6 ms stepping rate on some software, and they won't even run at 12 ms. An excellent replacement would be the Teac 55A (single sided) or the Teac 55B (double sided), (PC'S Limited in Austin, Texas has the 55B's for $95.00 apiece... (512) 452-0323). Be aware, if you do use the Teac drives the following changes are necessary:

1. You may have to drill some new holes in the drive cover to match up with the screw holes in the Teac drives.

2. You may need to put one or two spacer washers between the top of drive cover and the computer so that the computer case does not interfere with the disk latch on drive 0.

3. The positioning of the card-edge connectors on the ribbon cable for drive 0 and drive 1 will be different than the original cable.

4. You may need to rewire the power connectors or add an extension in order to get the power cable to reach the drive connectors since they are in a different position on the Teacs.

Assuming you are going to use the original Tandons, position the new 34 conductor cable next to the old one, making sure the stripe on the edge is on the same side. Take the 34 pin header connector and attach it to the end of the new cable, (you may use the old header if you carefully remove it from the old cable). Make sure the pin numbers are in exactly the same position as on the old cable. Attach the card-edge connectors for drives 0 and 1 at the same positions as on the old cable, again make sure that the pin numbers on the connectors are in the same positions (you should not use the old card-edge connectors if you plan on using double sided internal drives. Radio Shack has pulled pin 32 on the connectors which is needed for side selection). DO NOT cut the cable off at the end of drive 1.

The drives will now need to be programmed for drive 0 and 1 operation. On the main drive PCB there are two empty 16 pin dip sockets next to the 34 pin card-edge. The socket closest to the card-edge is the termination resistor socket, the other one is the programmable shunt socket and this is the one needed. On drive 0, pin 2 needs to be shunted to pin 15 and pin 8 shunted to pin 9 (pins 8 & 9 are the ones closest to the edge of the PCB). On drive 1, pin 3 needs to be shunted to pin 14 and pin 8 to pin 9. A staple, cut and properly bent, works perfectly as a shunt.

Attach the cable to the header connector and drives 0 and 1. Loop the extra cable over the top edge on the rear of the computer chassis. Cut the cable so that when the remaining card-edge connector is attached it will hang next to and in line with the RS-232 port. Make sure it is attached with the pin numbers in the same positions as drives 0 and 1.

Slit the piece of 1/4" hose down one side and slide it over the rear edge of the computer housing so that the drive cable sits on top of the hose. This will help prevent it from being cut by the housing. Now tape the cable in place. Test out the drive operation and then reassemble the computer.

Now you can use the drive extender cable to connect the card-edge connector on the rear of the computer to your external drives. Double sided operation for all four drives is supported.

Art Rasmussen
612 West Hillcrest
Keene, Texas 76059
Phone (817) 641-8922

REVIEW: SEATRONIC'S SPEEDUP BOARD FOR THE TRS-80 MODEL 4

by Ichiro Nohara
3423-1 Itano Ikeda Miyoshi, Tokushima 778, JAPAN

I am pleased to report on a speedup board that has made my TRS-80 Model 4 the fastest machine in the town.

This March, I purchased a speed-up board from Holland. It's really efficient (better than the Holmes sprinter, according to my experience). The company name is 'Seatronic' and it costs $129.99 in U.S. dollars. Their speedup board is small (approximately 2.5 inches square) and the Z80-H was placed in the middle of the board.

Installing the speedup board is easy, in my case, I finished all within two hours (I'm not a technical person). Also, you may use 120 or 150 nanosecond RAMs (my choice was 100 nanosecond). The speedup board offers 4 modes, 2, 4, 5.3 and 8 MHz. The following table is the result of simple test with each mode:

mode(MHz)	runtime(sec.)
2	162.0
4	81.0
5.3	61.5
8	43.0

```
10 ' SPEED TEST
20 '
30 FOR A = 0 TO 10000
40 PRINT @470,A;
50 NEXT
```

How about it... Sounds good ?

By the way, running at 8 MHz is not easy. In my case, I could not read/write disk with the 8MHz mode at first. This kind of problem is caused by a FDC problem. Wait timing is different with each FDC controller's tolerance. It looks as though the wait timing is too short. But, when running NEWDOS/80, it is easy to solve the problem. Refer your NEWDOS/80 manual zap 54, and try to zap as follows:

```
SYS0/SYS,02,C3  change  01 40 A3  to  01 50 A3
```

Now, your NEWDOS/80 will boot/read/write with the 8 MHz mode as normal.

Depending on machine tolerance, the above '50' is not always efficient. If you still have trouble, change the 50 to 60 (less than 4F, and you can't access 8 MHz mode; greater than 86, it appears to still work but some programs that use rapid RAM fetching, like a RAMDISK program, will not work correctly.

Also, I received a letter from Seatronic, regarding another solution to the above problem. The fix to the problem is change the timing resistor on FDC controller (on MDX6 board change timing resistor R10 from 47K ohm to 100K ohm). I still cannot read/write on TRSDOS 6.x or DOSPLUS 4a in the 8 MHz mode, so will try to change the resistor soon.

Finally, my recommendation is that their board is a good buy, and their board is well made, easy to install, comes with a good manual, and works efficiently.

Below is the listing for a speed control program for the above speedup board, for use with the TRS-80 Model I/III/4.

Documentation for SP/CMD

Format: SP XY

This program will run with line command X and Y. Space separation between X and Y is not allowed. X and Y enable configuration, parameters for X and Y as follows:

```
2,4,5,8: Direct speed change
      B: for Boot
      C: for copyright notice
      ?: for Help
      *: for present speed (mod 3/4 only)
      /: speed change with nothing printout
         (2nd parameter only - it's good for BASIC)

Examples:
SP 25    => speed to 2MHz then 5MHz (no meaning)
SP *  or SP  *    => display present speed
SP 8/    => speed to 8MHz but nothing printout
SP *4    => show present speed then turn to 4MHz
```

SP/ASM - SOURCE CODE FOR SP/CMD

```
             ;-----------------------------------------------
             ; Speedup8  clock rate change program  >>>> SP/CMD
             ;     Work with Seatronic's speedup board
             ;
             ;  For TRS80 Model 1/3/4  with NEWDOS80 v.2
             ;
             ; release >>>  04/08:V1.01       04/12:V1.02
             ;              04/13:V1.05       04/27:V1.07
             ; programed by Ichiro Nohara   dated  04/08/1986
             ;
             ; 3423-1 Itano Ikeda  Miyoshi Tokushima 778 JAPAN
             ;                                Tel  0883 72 5520
             ;-----------------------------------------------
5200                 ORG    5200H
01C9         CLS     EQU    01C9H
4467         PRINT   EQU    4467H
00EC         PORT    EQU    0ECH
000D         ENTER   EQU    0DH
000A         CR      EQU    0AH
00C4         TAB     EQU    196
4210         SPCHECK EQU    4210H         ;speed check for Model3/4
             ;----------save command line---------------
5200  7E     START   LD     A,(HL)        ;pointer
5201  32E855         LD     (CMDBUF),A    ;get parameter & save it
5204  23             INC    HL            ;next pointer
5205  7E             LD     A,(HL)        ;get next parameter
5206  32E955         LD     (CMDBUF+1),A  ;save it
             ;----------main routine--------------------
5209  F5             PUSH   AF            ;save flag
520A  CDA852         CALL   JOBCLS        ;set dammy flag 'no job done'
520D  21E855         LD     HL,CMDBUF     ;1st parameter
5210  CD2752         CALL   MAIN1         ;key check & do job
5213  C46C52         CALL   NZ,COMMON     ;if no job done then print out
5216  CDA852         CALL   JOBCLS        ;set dammy flag 'no job done'
5219  21E955         LD     HL,CMDBUF+1   ;next paragraph
521C  CD7F52         CALL   NOMORE        ;is it a <CR> ?
521F  CD2752         CALL   MAIN1         ;key check & do job again
5222  C46C52         CALL   NZ,COMMON     ;if no job done then print out
5225  F1     RETURN  POP    AF            ;restore AF register
5226  C9             RET                  ;return to DOS/BASIC
             ;----------key check-----------------------
5227  7E     MAIN1   LD     A,(HL)        ;what is the 1st paragraph
5228  FE2A           CP     '*'           ; * ?
522A  CA0853         JP     Z,PRESENT           ;so;present speed
522D  FE42           CP     'B'           ; B ?
522F  CA0000         JP     Z,0H          ;if so go boot
5232  FE62           CP     'b'           ;also lower case
5234  CA0000         JP     Z,0H          ;
5237  FE3F           CP     '?'           ; ? for help
5239  285E           JR     Z,HELP        ;if so help message
523B  FE43           CP     'C'           ;is C ?
523D  2846           JR     Z,COPYRT      ;copyright
523F  FE63           CP     'c'           ;also lower case
5241  2842           JR     Z,COPYRT
5243  FE20           CP     ' '           ;null work nothing but not ERROR
5245  285B           JR     Z,NULL        ;no paragraph, it is O.K.
5247  CD4052         CALL   DONE          ;so, job was done
524A  C45352         CALL   NZ,KEY        ;if no job then numeric
             ;----------dammy was used ?----------------
524D  3AEA55 DONE    LD     A,(JOB)       ;dammy set when job complete
5250  FEFF           CP     0FFH          ;ffh is dammy
5252  C9             RET                  ;return to caller
             ;----------check command-------------------
5253  7E     KEY     LD     A,(HL)        ;1st paragraph
5254  FE32           CP     '2'           ;is it 2 ?
5256  2856           JR     Z,SET2        ;then set for 2 mhz
5258  FE34           CP     '4'
525A  285D           JR     Z,SET4        ;set for 4 mhz
525C  FE35           CP     '5'
525E  2864           JR     Z,SET5        ;set for 5 mhz
5260  FE38           CP     '8'
5262  286B           JR     Z,SET8        ;set for 8 mhz
5264  FE2F           CP     '/'           ;no-need, but not error
5266  C8             RET    Z
5267  21CA55 ERROR   LD     HL,ERRMSG     ;else point to error message
526A  1833           JR     DISPLY        ;go display & return
             ;----------set speed to port---------------
526C  3AEB55 COMMON  LD     A,(BITSET)    ;get speed value
526F  D3EC           OUT    (PORT),A      ;set speed (model1)
5271  3AE955         LD     A,(CMDBUF+1)  ;2nd para.(must 2nd)
5274  FE2F           CP     '/'           ;is / ?
```

```
5276  C47A52           CALL   NZ,NOTICE      ;if not much, go print out
5279  C9               RET                   ;return to caller
                 ;----------speed display-------------------
527A  217455   NOTICE  LD     HL,SPDMSG      ;print Clock was change..
527D  1820             JR     DISPLY         ;go display & return
                 ;----------check enter key (2nd para)------
527F  7E       NOMORE  LD     A,(HL)
5280  FE0D             CP     0DH            ;no more param. ?
5282  281E             JR     Z,NULL         ;if so, goto null
5284  C9               RET                   ;            for jobmark set
                 ;----------copyright-----------------------
5285  CDE352   COPYRT  CALL   M1OR3          ;which model 1 or 3
5288  CC9352           CALL   Z,M1           ;if mod1 then m1
528B  CDC901           CALL   CLS            ;clear screen
528E  214A53           LD     HL,CPYMSG      ;copyright print routine
5291  180C             JR     DISPLY         ;go display & return
5293  3E31     M1      LD     A,'1'          ;if model1
5295  320254           LD     (MODELX),A     ;set 1 for print buffer
5298  C9               RET                   ;return to caller
                 ;----------help----------------------------
5299  CDC901   HELP    CALL   CLS            ;cls screen
529C  211E54           LD     HL,HELPMSG     ;point to help message
529F  CD6744   DISPLY  CALL   PRINT          ;print it
52A2  3EFF     NULL    LD     A,0FFH         ;null it is also one of job
52A4  32EA55           LD     (JOB),A        ;set dammy as job done
52A7  C9               RET                   ;return to caller
                 ;----------clear job buffer----------------
52A8  3E00     JOBCLS  LD     A,0            ;clear dammy job flag
52AA  32EA55           LD     (JOB),A
52AD  C9               RET                   ;return to caller
                 ;----------set for speed-------------------
52AE  3E32     SET2    LD     A,'2'          ;set 2mhz print
52B0  CDDC52           CALL   SETCHR         ;write it & check m1 or 3
52B3  204C             JR     NZ,M3SP2       ;if so,go model3 routine
52B5  3E00             LD     A,0            ;set 2mhz with model1
52B7  181F             JR     SETUP          ;prepare to return
52B9  3E34     SET4    LD     A,'4'          ;4 charcter for buffer
52BB  CDDC52           CALL   SETCHR         ;write it
52BE  2038             JR     NZ,M3SP4       ;if mod3 go model3 routine
52C0  3E40             LD     A,40H          ;set 4mhz
52C2  1814             JR     SETUP          ;go save it
52C4  3E35     SET5    LD     A,'5'          ; >> 5mhz
52C6  CDDC52           CALL   SETCHR         ;write '5'
52C9  2024             JR     NZ,M3SP5       ;if model3 go model3 routine
52CB  3E80             LD     A,80H          ;set to 5mhz
52CD  1809             JR     SETUP          ;go save it
52CF  3E38     SET8    LD     A,'8'          ; >> 8mhz
52D1  CDDC52           CALL   SETCHR         ;write '8'
52D4  2012             JR     NZ,M3SP8       ;else model3 then m3speed8mhz
52D6  3EC0             LD     A,0C0H         ;set to 8mhz
52D8  32EB55   SETUP   LD     (BITSET),A     ;save value of speed
52DB  C9               RET                   ;and return to caller
                 ;----------set characters------------------
52DC  328055   SETCHR  LD     (SPDBUF),A     ;for character set
52DF  CDE352           CALL   M1OR3          ;check m1 or m3
52E2  C9               RET                   ;return to caller
                 ;----------model 1 or 3--------------------
52E3  3A5400   M1OR3   LD     A,(54H)        ;get byte from ROM
52E6  3D               DEC    A              ;check it
52E7  C9               RET                   ;return to caller
                 ;----------model 3 speed set---------------
52E8  3A1042   M3SP8   LD     A,(SPCHECK)              ;how in DOS
52EB  F6C0             OR     0C0H           ;set 8mhz
52ED  1817             JR     M3SETOK        ;return
52EF  3A1042   M3SP5   LD     A,(SPCHECK)              ;how's in DOS
52F2  E6BF             AND    0BFH           ;set to
52F4  F680             OR     80H            ;    5mhz
52F6  180E             JR     M3SETOK        ;save and setup
52F8  3A1042   M3SP4   LD     A,(SPCHECK)              ;how's in DOS
52FB  F640             OR     40H            ;set to
52FD  E67F             AND    7FH            ;    4mhz
52FF  1805             JR     M3SETOK        ;save and setup
5301  3A1042   M3SP2   LD     A,(SPCHECK)              ;how's in DOS
5304  E63F             AND    3FH            ;set 2mhz
5306  321042   M3SETOK LD     (SPCHECK),A              ;set DOS's require
5309  18CD             JR     SETUP
                 ;----------present speed-------------------
530B  CDE352   PRESENT CALL   M1OR3          ;check model 1 3
530E  280D             JR     Z,CANT         ;if model 1 can't proceed
5310  CD2353           CALL   HOW            ;else read present speed
```

```
5313  218855           LD     HL,PRNTMSG     ;and print it out
5316  1887             JR     DISPLY         ;display & return
5318  21B855   WHAT    LD     HL,HOWMSG      ;unknown spped parameter ??
531B  1882             JR     DISPLY         ;display & return
531D  21A355   CANT    LD     HL,CANTMSG     ;sorry model 1
5320  C39F52           JP     DISPLY         ;display & return
                 ;----------check present speed (only mod3)-
5323  3A1042   HOW     LD     A,(SPCHECK)              ;check RAM
5326  FE28             CP     28H            ;is it 2 mhz
5328  280E             JR     Z,READY2       ;then write 2mhz
532A  FE68             CP     68H            ;is it 4mhz?
532C  280E             JR     Z,READY4       ;then write '4'
532E  FEA8             CP     0A8H           ;is it 5mhz?
5330  280E             JR     Z,READY5       ;then write '5'
5332  FEE8             CP     0E8H           ;is it8mhz?
5334  280E             JR     Z,READY8       ;then write '8'
5336  18E0             JR     WHAT           ;goto error 'UNKNOWN'
                 ;----------present speed set for mod3------
5338  3E32     READY2  LD     A,'2'          ;put 2 on present buffer
533A  180A             JR     READY          ;printout
533C  3E34     READY4  LD     A,'4'          ;put 4 on present buffer
533E  1806             JR     READY          ;printout
5340  3E35     READY5  LD     A,'5'          ;put 5 on present buffer
5342  1802             JR     READY          ;printout
5344  3E38     READY8  LD     A,'8'          ;put 8 on present buffer
5346  328F55   READY   LD     (PRNTCHR),A    ;save it
5349  C9               RET                   ;return to caller
                 ;----------copyright message---------------
534A  0A000A00 CPYMSG  DW      CR,CR,CR
534E  0A00
5350  0AC895C2         DEFM    CR,200,149,194,'Clock rate change
utility',195,'version 1.1',194,170
5354  436C6F636B20726174652063686E67
5364  652075746696C697479C376657273696F
5374  6E20312E31C2AA
537B  0AC895C2         DEFM    CR,200,149,194,'Mod 1/3/4, NEWDOS80 & Speedup8
required',194,170
537F  4D6F6420312F332F342C204E4557444F
538F  533830202620537065656475703820 72
539F  65717569726564C2AA
53A8  0AC895C2         DEFM    CR,200,149,194,'Created by Ichiro
Nohara',197,'04/09/1986',194,170
53AC  43726561746564206279204963686972
53BC  6F204E6F68617261C530342F30392F31
53CC  393836C2AA
53D1  0AC895C9         DEFM    CR,200,149,201,'Ikeda
Tokushima',194,'JAPAN',203,170
53D5  496B656461202054 6F6B757368696D61
53E5  C24A4150414ECBAA
53ED  0AC895EB         DEFM    CR,200,149,235,170
53F1  AA
53F2  0AC895C4         DEFM    CR,200,149,196,'Enjoy Model '
53F6  456E6A6F79204D6F64656C20
5402  3300     MODELX  DW      '3'
5404  20776974         DW      ' with speed up board',198,170,ENTER
5408  682073706565642075702062 6F617264
5418  C600AA000D00
                 ;----------help message--------------------
541E  C1436C6F HELPMSG DEFM    193,'Clock rate utility for NEWDOS80 with
Seatronic',39,'s SPEEDUP8'
5422  636B20726174652075746696C69747920
5432  666F72204E4557444F53383020776974
5442  6820536561747 26F6E69632773205350
5452  454544555038
5458  0A0AD2B7         DEFM    CR,CR,210,183,194,'Format: SP XY',194,187,CR,CR
545C  C2466F726D61743A2053502058 59C2BB
546C  0A0A
546E  C4506172         DEFM    196,'Parameter X and Y able to mix/configure.',CR
5472  616D6574657220582061 6E6420592061
5482  626C6520746F206D69782F636F6E6669
5492  677572652E0A
5498  C4426574         DEFM    196,'Between X-Y,',195,'space separation will not
allowed.',CR
549C  7765656E20582D592CC373706163652 0
54AC  73657061726174696F6E2077696C6C20
54BC  6E6F7420616C6C6F7765642E0A
54C9  C44E756D         DEFM    196,'Numeric always priority for secure.',CR,CR
54CD  6572696320616C7761797320707269 6F
54DD  72697479206666F7220736563757265 2E
54ED  0A0A
```

```
54EF 50617261          DEFM    'Parameters
B:',TAB,'Boot',TAB,TAB,'C:',TAB,'Copyright',CR
54F3 6D65746572732020204 23AC4426F6F74
5503 C4C4433AC4436F7079726967687 40A
5512 CD3F3AC4          DEFM    205,'?:',TAB,'Help',TAB,TAB,'*:',TAB,'Present
speed',CR
5516 48656C70C4C42A3AC4507265736 56E74
5526 20737065656 40A
552D CD2F3AC4          DEFM    205,'/:',TAB,'Without printout (2nd para.only)',CR
5531 576974686F757420707 2696E746F7574
5541 2028326E6420706172612E6F6E6C7929
5551 0A
5552 C7322C34          DEFM    199,'2,4,5,8:',TAB,'direct speed definition',ENTER
5556 2C352C383AC464697265637420737065
5566 65642064656669 6E6974696F6E0D
               ;----------speed message-------------------
5574 B7002020  SPDMSG  DW      183,'  Set to',194
5578 5365742074 6FC200
5580 3F00      SPDBUF  DW      '?'
5582 4D487A20          DW      'MHz  ',187,ENTER
5586 20BB000D00
5588 B7002000  PRNTMSG DW      183,' '
558F 3F00      PRNTCHR DW      '?'
5591 4D487A20          DW      'MHz Presented ',187,ENTER
5595 50726573656E74656420BB000D00
               ;----------error message-------------------
55A3 536F7272  CANTMSG DEFM    'Sorry, I don',39,'t know.',ENTER
55A7 792C204920646F6E277420686E6F772E
55B7 0D
55B8 53706565  HOWMSG  DEFM    'Speed is UNKNOWN.',ENTER
55BC 6420697320554E4B4E4F574E2E0D
55CA 4D697373  ERRMSG  DEFM    'Missing parameter(s)... sorry',ENTER
55CE 696E672070617261 6D65746572287329
55DE 2E2E2E20736F72727 90D
               ;----------buffers-------------------------
55E8 0000      CMDBUF  DW      0000            ;buffer for speed para.
55EA 00        JOB     DB      00              ;job dammy buffer
55EB 00        BITSET  DB      00              ;for bitset buffer
               ;----------end of program------------------
5200                   END     START
               ;------------------------------------------

SYMBOL TABLE
CLS     01C9   PRINT   4467   PORT    00EC   ENTER   000D   CR      000A
TAB     00C4   SPCHECK 4210   START   5200   RETURN  5225   MAIN1   5227
DONE    524D   KEY     5253   ERROR   5267   COMMON  526C   NOTICE  527A
NOMORE  527F   COPYRT  5285   M1      5293   HELP    5299   DISPLY  529F
NULL    52A2   JOBCLS  52A8   SET2    52AE   SET4    52B9   SET5    52C4
SET8    52CF   SETUP   52D8   SETCHR  52DC   M1OR3   52E3   M3SP8   52E8
M3SP5   52EF   M3SP4   52F8   M3SP2   5301   M3SETOK 5306   PRESENT 530B
WHAT    5318   CANT    5310   HOW     5323   READY2  5338   READY4  533C
READY5  5340   READY8  5344   READY   5346   CPYMSG  534A   MODELX  5402
HELPMSG 541E   SPDMSG  5574   SPDBUF  5580   PRNTMSG 5588   PRNTCHR 558F
CANTMSG 55A3   HOWMSG  55B8   ERRMSG  55CA   CMDBUF  55E8   JOB     55EA
BITSET  55EB
```

MODEL 4 POWER SUPPLY

by Mel Patrick

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

I have noticed an ever increasing amount of TRS-80 Model 4 owners who are installing additional drives inside their computers. While I am not going to condone this procedure, there are a few facts you should be aware of.

The normal TRS-80 Model 4 power supply (of which there are three different versions), is rated at 65 Watts maximum. The electrical formula for calculating the required amount is:

$$W = E * I$$

Where W=Watts, E=Volts, I=Current. In a simple example, if you had a 12 Volt light bulb and it required 1 Amp to work, that would translate to W=12*1, or 12 Watts of power required.

Using the same idea we will move to the Model 4. The power supply, as previously mentioned, can supply a continuous 65 Watts. Now let's look at what the computer requires.

Computer	+5 @	4.0A	-	20.0 Watts
RS232	+12 @	.1A	-	1.2 Watts
RS232	-12 @	.1A	-	1.2 Watts
Video	+12 @	1.0A	-	12.0 Watts
Disk Card	+12 @	.3A	-	3.6 Watts
Disk Card	+5 @	.4A	-	2.0 Watts
1 Drive {	+5 @	.7A	-	3.5 Watts
	+12 @	1.0A	-	12.0 Watts

This totals out to 55.2 Watts for a one drive system. Add a second drive to that and add another 15.5 Watts. This would then total 70.7 Watts.....

This shows clearly that at full power the computer requires more than the computer can supply.

Fortunately, this condition only exists when you are doing disk access. This is only an intermittent operation, so you can subtract the 12V supply line values for each of the disk drives when they are idle. This will save 24 Watts of power leaving us with a total of 47 Watts, well within design limitations of the supply. Since disk drives are only rated for intermittent operation, we only have heavy current draw while they are active. The heaviest draw occurs during a **FORMAT** operation. You may have noticed that your screen "pulses" when formatting. By adding another disk drive you increase the drain on the power supply.

If you checked out the price of a new power supply it may also change your thinking somewhat. I know, I had to buy one, not a Radio Shack one, they are far too expensive, but from another company. You can expect to pay at least 150 dollars.

If you are presently overtaxing your power supply and experiencing nothing, remember that anything can be abused. However in due time it may fail.

The last problem with over loading concerns heat. The heavier the load on the power supply, the warmer it will operate. Add the increased heat from an additional drives you also have in the case and it spells bad news. Integrated circuits are affected adversely in two major areas. First is static electricity, second is heat. If you are going to load a poorly ventilated Model 4 case, I suggest a fan at the very least. You can buy a lot of fans for the price of one power supply repair bill.

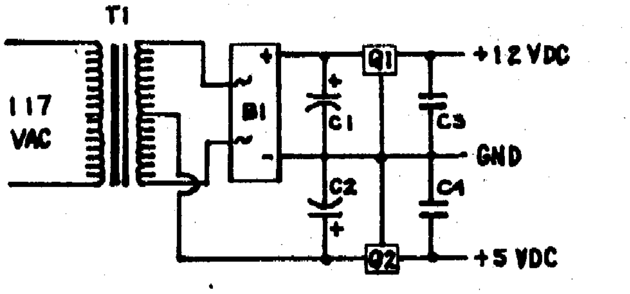
Now that you have all this bad news, I have a suggestion. Power supplies are very simple to make. If you want to add drives to your computer the circuit shown below will more than adequately power a drive (single or double sided). As for a disk drive case, try to obtain a Hammond case to fit, or make your own out of metal or plexiglass.

The parts may be bought from your local Radio Shack store, or another distributor of electronic parts. I have provided the Radio Shack part numbers where applicable.

PARTS LIST

B1	- Bridge Rectifier 4A 100V	- RS 276-1171
C1 & C2	- Capacitors 1000 µf 35V	- RS 272-1019
C3 & C4	- Capacitors .1 µf 50V	- RS 272-135
Q1	- Voltage Regulator 7812	- RS 276-1771
Q2	- Voltage Regulator 7805	- RS 276-1770
T1	- Transformer 12.6 V CT 1.2 A	- RS 273-1505

The only additional material required would be the necessary tools to assemble the supply and a line cord to plug it in.



UNDOCUMENTED AND UNTESTED ZAPS

The following zaps are mostly of spurious origin (translation: they were found buried under a pile of papers on the editor's desk), and therefore should be considered unreliable until thoroughly tested.

1) Zap to Model I Scripsit so that initialization character is not sent:

```
  SCRIPSIT/CMD,00,47  from  0A 32 E8 37 AF  to  0A 00 00 00 AF
```

2) Optional zap to Radio Shack's SuperScripsit Version 1.0 for the Model I. The SuperScripsit module PROOF/CTL assumes you have three drives as it creates (and later kills) a file named MISSPELL/CTL on Drive 2. Further, the file WORDS/CTL (from Scripsit Dictionary) is supposed to be mounted on Drive 1. This zap removes the drive number specifiers from the PROOF/CTL program, thereby allowing WORDS/CTL to be mounted on any drive and MSSSPELL/CMD to be created on the first available drive.

```
  PROOF/CTL,11,30  change  4C 3A 32 0D 00  to  4C 0D 00 00 00
  PROOF/CTL,11,67  change  4C 3A 31 0D 00  to  4C 0D 00 00 00
```

3) The Microsoft/Radio Shack FORTRAN package does not work properly under the Model III version of NEWDOS/80. Specifically, the -E switch for L80/CMD does not work. The following corrects this problem and, presumably, resolves other problems as well. Using zap 050 (11/09/81) as a guide, it appears that address 4313 in TRSDOS became 4479 in NEWDOS, and there is no equivalent to TRSDOS address 4315. This zap is for the Model III version of FORTRAN only, but NEWDOS Model I zap 053 implies that there is not a problem there.

```
EDIT/CMD,18,AA  change   22 13 43 3E C3 32 15 43
                to       22 79 44 3E C3 00 00 00
EDIT/CMD,42,7A
chg 32 15 43 22 19 53 2A 13 43 22 52 78 21 4E 78 22 13 43
to  00 00 00 22 19 53 2A 79 44 22 52 78 21 4E 78 22 79 44
F80/CMD,05,BD
change   32 15 43 2A 13 43 22 F1 7A 21 ED 7A 22 13 43
to       00 00 00 2A 79 44 22 F1 7A 21 ED 7A 22 79 44
F80/CMD,45,55  change   32 15 43 2A F1 7A 22 13 43
               to       00 00 00 2A F1 7A 22 79 44
L80/CMD,00,08
change   32 15 43 2A 13 43 22 A1 6F 21 9D 6F 22 13 43
to       00 00 00 2A 79 44 22 A1 6F 21 9D 6F 22 79 44
L80/CMD,07,18
change   32 15 43 2A 01 02 00 59 A1 6F 22 13 43
to       00 00 00 2A 01 02 00 59 A1 6F 22 79 44
```

4) Proposed zaps to Model I/III SuperScripsit version 1.3.0 for use with NEWDOS/80 version 2.0 and 2.5.

a) Disables a call to a TRSDOS routine which returns the amount of free space on a diskette and replaces it with code to return "lots". Note that this patch merely reports that there is always plenty of available disk space, even though this may not be the case in fact. The user MUST remain aware of this, as a DISK FULL condition may cause a non-recoverable error with the current output file.

```
  SCRIPSIT/CMD,25,E1  change  C9 C5 D5 E5  to  C9 3F FF C9
```

b) The first patch below installs a "DIR 0" text at address 92CAH, replacing another text not needed by NEWDOS/80. The second patch overwrites code to display TRSDOS's directory with a DOS call to 4419H using the text installed by the first patch.

```
SCR17/CTL,07,9F  change   92 1C 1F 44 52 49 56 45
                 to       92 44 49 52 20 30 0D 45
SCR17/CTL,00,85
change  E5 32 A8 92 3A 25 01 FE 49 20 0D 3A A8 92
to      E5 32 CE 92 21 CA 92 CD 19 44 C3 1A 8C 92
```

c) The first patch below allows the "Display Directory" routine to accept drives numbered 0-9 instead of 0-3. The second modifies the "Which Drive" prompt similarly.

```
    SCR17/CTL,00,7A  change  FE 34 30  to  FE 3A 30
    SCR17/CTL,08,31  change  2D 33 29  to  2D 39 29
```

d) Further notes: SuperScripsit version 1.3.0 DOES NOT respect NEWDOS/80's HIMEM pointer. This will work OK with version 2.5 because SuperScripsit will keep out of the upper 2K anyway, but users with high memory drivers or other routines must be aware of this fact. Also, these zaps have not been tested with the PROOFREAD function, as this was not available to the zap developer.

5) Apparat issued zap number 83 for zapping the Model I SuperScripsit for TRSDOS to enable it to run with Model I NEWDOS/80. However, the data in these zaps does not apply to SuperScripsit version 1.2.0. To enable this TRSDOS version to run under NEWDOS it is necessary to make the following changes. This zap is by Jack Bognuda of Queensland, Australia (from Bits & Bytes #26).

```
SCRIPSIT/CMD,09,1F                      change  C93A
        D9AB 4FCD F04A C021 004D 0123 00CB 1E38
        0104 CB1E 3801 040D 2320 F204 0528 0105
        2105 00CD 6666 0604 CD46 667D 3222 7EAF
        C900 0000
                                        to      C906
        0411 333C 21DC AC1A 7713 D630 FE0A 3003
        0102 005B 2310 F036 8400 0000 0605 CD66
        6606 04CD 4666 7D24 2528 023E FF32 227E
        AFC9 0000
SCR17/CTL,00,31
change  69FE 3038 F9FE 3430 F5CD 9E75 0E00 3271
        42CD 1944 3E0F CD33 0021
to      6932 728D 216E 8DCD 1944 2805 F6C0 CD09
        4418 0644 4952 2030 0D21
SCR17/CTL,02,A2  change  57 08CB 8F52
                 to      57 4450 8D52
```

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

6) Another set of zaps to Model III SuperScripsit, this time for version 1.2.3 for use with NEWDOS/80 instead of zap 077 supplied by Apparat. These zaps will, in case of a "full diskette", refrain from attempting to write the file to disk. It will instead allow you to select another diskette and try again. Thus, you can recover from the error and the file will (hopefully) not be lost. This zap is by Kees Walter of Landsmeer, Holland (from Remarks #46).

```
SCR17/CTL,00,30
change  69FE 3038 F9FE 3430 F5CD 8975 0E00 3271
        42CD 1944 3E0F CD33 0021
to      6932 728D 216E 8DCD 1944 2805 F6C0 CD09
        4418 0644 4952 2030 0D21
SCR17/CTL,02,68 change 2054 5253 444F
                to     204E 4557 444F
SCRIPSIT/CMD,09,0A         change  AB47 0EFF 219B
        ACCD 9042 C023 2346 2103 00CD 5166 0604
        CD31 667D 3222 7EAF C900 0000 0000 0000
        0000 0000 0000 0000 0000 0000 0000 0000
        0000 0000 0000 0000 0000 0000 0000 0000
        0000 0000 0000 0000 0000 00E5
                           to      ABC6 3032 2B5B
        F521 275B CD19 44C0 210B 3C06 2536 2D23
        10FB 363E CD49 0000 F121 DCAC 7706 043E
        8477 2310 FC77 CDB2 6806 05CD 5166 0604
        CD31 667D 2425 2802 3EFF 3222 7EAF C944
        4952 2030 2C2F 5139 510D 00E5
```

7) Zap to the Model III version of NEWDOS/80 to make it boot automatically in the 4 MHz (high speed) mode on the Model 4. Any routine which is controlled by the interrupts (including keyboard repeat and cursor blink) will be activated twice as often as normal when this zap is applied. The first four bytes of this zap are loader control information, and most of the new bytes are loader control. The zap modifies the loader control information in SYS0/SYS so that a value of 64H is loaded into memory location 4210H, which the ROM then outputs to port 236 later in the boot routine, invoking the 4 MHz clock speed. SYSTEM OPTION BJ MUST BE SET TO 2 WHEN THIS ZAP IS APPLIED!!!

```
  SYS0/SYS,14,B2  change  01 48 A6 51 00 00 00 00 00 00
                  to      01 03 10 42 68 01 43 A6 51 00
```

8) NEWDOS/80 Model I lock in run-only mode:

```
          SYS0/SYS,02,7B  change  BF  to  00
```

CONTEST!

We will give a complete set of our public domain library diskettes to anyone who submits a PD catalog program that will enable us to easily catalog all public domain programs. Must not be memory dependent! Submissions must be received by July 1st, 1987. Send to: TAS, 704 N Pa., Lansing, MI, 48906-5319.

DISKMAP PRINT PROGRAM

by Ross Placing

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

FOR all of you people who don't want to read a long list of file locations here is a quick & dirty program that should print out a map of a disk similar to this:

```
          Disk name <text1txt>  auto<>
Cyl    GRAN 0          GRAN 1          GRAN 2          GRAN 3
 0 < BOOT    SYS >< BOOT    SYS ><                ><                >
 1 < MAPPER  TXT >< MAPPER  TXT >< MAPPER  TXT >< MAPPER  TXT >
 2 < MAPPER  TXT >< MAPPER  TXT >< MAPPER  TXT >< MAPPER  TXT >
 3 < MAPPER  TXT >< MAPPER  TXT >< MAPPER  TXT >< MAPPER  TXT >
 4 < MAPPER  TXT >< MAPPER  TXT >< MAPPER  TXT ><                >
 5 <             ><             ><             ><                >
.....
19 <             ><             ><             ><                >
20 < DIR     SYS >< DIR     SYS >< DIR     SYS >< DIR     SYS >
21 <             ><             ><             ><                >
.....
```

and here is the program listing:

```
10 CLEAR15000
11 DEFINT A-Z
12 DIMPO(201,7),R$(7),RR$(300),BI(7)
13 FORA=0TO7
14 BI(A)=2[A
15 NEXTA
16 CLS
17 DEFFNGR(B)=(BAND31)
18 DEFFNGS(B)=INT(B/32)
19 DEFFNB2(S$)=ASC(MID$(S$,2))
20 DEFFNGC(S$)=FNGR(FNB2(S$))
21 DEFFNSG(S$)=FNGS(FNB2(S$))
22 DEFFNF$(R$,S)=MID$(R$,21+S*2,2)
111 DEFFNNR(T$)=FNGC(T$)*8+FNSG(T$)+1
112 T$="  "+""
140 PRINT@256,"Which drive ";
141 FORA=1TO1
142 K$=INKEY$
143 A=LEN(K$)*INSTR(CHR$(0)+"01234567",K$)
144 NEXTA
145 DR$=K$
146 PRINTK$;
147 IFDR$=CHR$(13)ORDR$=CHR$(31)THEN140
220 DR$=":"+DR$
221 CMD"I "+DR$+",M"
222 OPEN"R",1,"DIR/SYS"+DR$
223 CLS
224 PRINT"DISKMAP FOR DOSPLUS, LDOS & TRSDOS 2.3 disks- Drive "
;RIGHT$(DR$,1)
225 CO$=CHR$(15)
226 OF$=CHR$(18)
227 PRINTSTRING$(64,95);
228 PRINT@256,"Reading directory on Drive ";RIGHT$(DR$,1)
229 GET1,1
320 FIELD 1,204AS C1$,1AS CL$,3ASD$,8ASNA$,8ASD$,32ASAU$
321 GOSUB760
322 FORII=0TO7
323 FIELD 1,(II*32)ASD$,32ASR$(II)
324 NEXTII:IFLOF(1)>34THENDF=34ELSEDF=LOF(1)
325 FORA=3TODF
326 GET1,A
327 FORB=0TO7
328 RR$((A-3)*8+1+B)=R$(B)
329 NEXTB,A
330 CLOSE
331 FORA=1TO300
332 IFLEN(RR$(A))<32THENGOTO450
420 B=A
430 IFASC(RR$(B))>127ANDASC(MID$(RR$(B),6,1))< 64THENB=FNNR(RR$(B
))
440 IFASC(RR$(B))>1ANDASC(MID$(RR$(B),6,1))>64THENT$=FNF$(RR$(B),5
):IFASC(T$)=254THENMID$(RR$(FNNR(T$)),6,11)=MID$(RR$(B),6,11):B= FNN
R(T$):GOTO430
450 NEXTA
451 RR$(300)="     locked out "+STRING$(16,255)
452 RR$(0)=STRING$(32,32)
453 FORA=1TO300
454 IFLEN(RR$(A))<32THENGOTO670
500 IFASC(RR$(A))=0THENGOTO670
510 IFASC(RR$(A))>127ANDASC(MID$(RR$(A),6,11))<64THENGOTO670
520 T$=MID$(RR$(A),6,11)
521 IFT$<STRING$(12,32)THENGOTO670
540 PRINT@256,CHR$(31)"Working on "MID$(RR$(A),6,11);
541 LSET T$=FNF$(RR$(A),1)
542 A1=ASC(T$)
543 IFA1<CL+1THENA2=FNSG(T$):A3=FNGC(T$):GOSUB690ELSE670
580 LSETT$=FNF$(RR$(A),2)
581 A1=ASC(T$)
582 IFA1<CL+1THENA2=FNSG(T$):A3+FNGC(T$):GOSUB690ELSE670
610 LSETT$=FNF$(RR$(A),3)
611 A1=ASC(T$)
612 IFA1<CL+1THENA2=FNSG(T$):A3=FNGC(T$):GOSUB690ELSE670
640 LSETT$=FNF$(RR$(A),4)
641 A1=ASC(T$)
642 IFA1<CL+1THENA2=FNSG(T$):A3=FNGC(T$):GOSUB690
670 NEXTA
671 GOTO1030
690 PRINTA1;A2;A3",";
691 FORAA=0TOA3
710 P=A2+AA
711 IFP>GMTHENA1=A1+1:A2=-AA:GOTO710
730 PO(A1,P)=A
731 NEXTAA
732 RETURN
760 CL=ASC(CL$)
761 C2=CL
762 CL=C2+34
763 IFCL<96THENTT=ASC(MID$(C1$,97,1))ELSETT=255
840 FORA=1TOCL
841 TN=ASC(MID$(C1$,A,1))
842 IFTN<TTTHENTT=TN:NEXTAELSENEXTA
870 TP=256-TT
871 A=INSTR(AU$,CHR$(13))
872 IFA>0THENUA$=LEFT$(AU$,INSTR(AU$,CHR$(13))-1)ELSEUA$=AU$
900 DN$=NA$
901 PRINT@128,"<"DN$"> auto <"UA$">";
902 PRINT@192,CL+1"cylinders";
903 FORA=8TO1STEP-1
904 IF(2[A-1)>TPTHENNEXTAELSEGM=A-1
950 PRINTGM+1"grans per cyl";
951 FORA=0TOCL
952 TT=ASC(MID$(C1$,A+1,1))
953 PRINT@320,"working on cyl"A;
954 FORB=0TOGM
955 IF(TTANDBI(B)) THENPO(A,B)=300
1010 NEXTB,A
1011 RETURN
1030 LPRINTTAB(10)"Disk name <"DN$">  auto <"UA$">"
1031 LPRINTCO$" Cyl    ";
1032 FORA=0TOGM
1033 LPRINTUSING"%                %";"GRAN"+STR$(A);
1034 NEXTA
1035 LPRINT
1036 FORA=0TOCL
1037 LPRINTUSING"### ";A;
1038 FORAA=0TOGM
1039 LPRINTUSING"< %              %>";MID$(RR$(PO(A,AA)),6,11);
1040 NEXTAA
1041 LPRINT
1132 NEXTA
1133 LPRINTOF$
1134 POKE14312,12
1135 CLS
1136 PRINT@590,"Press (D) for DOS or any other key for next disk";
1137 FORA=1TO1
1138 A$=INKEY$
1139 A=LEN(A$)
1140 NEXTA
1141 IFA$="D"ORA$="d"THEN CMD"S"
1230 RUN
```

The limitations are that the program will read MULTIDOS, DOSPLUS, TRSDOS Model I, LDOS, and any standard DOS (exceptions CP/M, MSDOS, NEWDOS 80 DDEN, ETC...). The program has been tested with MULTIDOS, LDOS, TRSDOS and DOSPLUS 3.5 diskettes (note you might have to change the access password to '9642' on DIR/SYS entry in the directory).

SYDTRUG NEWS EDITOR'S NOTE: Ross uses DOSPLUS BASIC which has a number of differences to that used with TRSDOS and LDOS. To be specific the following changes should be made if you are not using DOSPLUS:

```
        Line 221 should be deleted
          "  329 should read NEXT B : NEXT A
          " 1010 should read NEXT B : NEXT A
```

After looking over Ross's program, I decided to adapt it for operation on the Model 4 under TRSDOS 6.x so that I wouldn't have to boot LDOS just to map a diskette. So for those of you who, like me, have the advantage of a Mod 4 the following program will allow you to map any of the disk types that Ross has stated above.

This program takes note of the fact that the AUTO feature is no longer stored in the GAT sector of the directory with TRSDOS 6.x (it is actually stored in BOOT/SYS sector 02 beginning at byte 20H) and so it checks the version number before trying to find a valid AUTO statement. If the version is 60 to 62 the AUTO statement is retrieved from BOOT/SYS rather than from DIR/SYS as for other DOS's (this also means that should TRSDOS 6 be raised in version from the current 6.2, the test in line 1560 should be adjusted to account for the change).

```
1000 REM ******************************************
1010 REM *                                        *
1020 REM *              DISKMAP6/BAS              *
1030 REM *            by  Ross Placing            *
1040 REM *                                        *
1050 REM * Adapted and Modified for the Model 4 *
1060 REM *             by Gary Bryce              *
1070 REM *                                        *
1080 REM ******************************************
1090 CLEAR 15000
1100 LPRINT CHR$(27);"v66.";:  REM Sets FORM Length
1110 COMP$=CHR$(27)+"Q":       REM Sets COMPRESSED Mode
1120 NORM$=CHR$(27)+"N":       REM Sets NORMAL Mode
1130 DEFINT A-Z
1140 DIM PO(201,7),R$(7),RR$(485),BI(7)
1150 FOR A=0 TO 7
1160   BI(A)=2^A
1170 NEXT A
1180 CLS
1190 DEF FN GR(B)=(B AND 31)
1200 DEF FN GS(B)=INT(B/32)
1210 DEF FN B2(S$)=ASC(MID$(S$,2))
1220 DEF FN GC(S$)=FN GR(FN B2(S$))
1230 DEF FN SG(S$)=FN GS(FN B2(S$))
1240 DEF FN F$(R$,S)=MID$(R$,21+S*2,2)
1250 DEF FN NR(T$)=FN GC(T$)*8+FN SG(T$)+1
1260 T$="  "+""
1270 PRINT @ 480,"Which drive ";
1280 FOR A = 1 TO 1
1290   KEYIN$=INKEY$
1300   A=LEN(KEYIN$)*INSTR(CHR$(0)+"01234567",KEYIN$)
1310 NEXT A
1320 DRV$=KEYIN$
1330 PRINT KEYIN$;
1340 IF DRV$=CHR$(13)OR DRV$=CHR$(31)
       THEN GOTO 1270
1350 DRV$=":"+DRV$
1360 OPEN "R",1,"DIR/SYS"+DRV$
1370 CLS
1380 PRINT @ 20,"**************** DISKMAP ****************"
1390 PRINT @ 95,"for TRSDOS 6, LDOS and compatible disks";
1400 PRINT " - Drive ";RIGHT$(DRV$,1)
1410 PRINT STRING$(80,95);
1420 PRINT @ 480,"Reading DIR/SYS on Drive ";RIGHT$(DRV$,1)
1430 GET 1,1
1440 FIELD 1,203 AS C1$,1 AS VER$,1 AS XCYLS$,3 AS D$
        ,8 AS DISKN$,8 AS D$,32 AS AUTX$
1450 VER=ASC(VER$)-36
1460 DNAME$=DISKN$
1470 XCYLS=ASC(XCYLS$)
1480 CYLS=XCYLS+34
1490 IF CYLS<96
      THEN TT=ASC(MID$(C1$,97,1))
      ELSE TT=255
1500 FOR A =1 TO CYLS
1510   TN=ASC(MID$(C1$,A,1))
1520   IF TN<TT
         THEN TT=TN
1530 NEXT A
1540 TP=256-TT
1550 REM ** CHECK FOR TRSDOS 6.0 TO 6.2 **
1560 IF VER<60 OR VER>63 THEN AUT$=AUTX$:GOTO 1610
1570 OPEN "R",2,"BOOT/SYS"+DRV$
1580 GET 2,3
1590 FIELD 2,32 AS D$,80 AS AUT6$
1600 AUT$=AUT6$
1610 A=INSTR(AUT$,CHR$(13))
1620 IF A>0 THEN AUT0$=LEFT$(AUT$,INSTR(AUT$,CHR$(13))-1)
       ELSE AUT0$=CHR$(0)
1630 PRINT @ 240,"<"DNAME$"> AUTO <"AUT0$">";
1640 PRINT @ 360,CYLS+1"cylinders";
1650 FOR A =8 TO 1 STEP -1
1660   IF (2^A-1)>TP
         THEN NEXT A
       ELSE GRANS=A-1
1670 PRINT GRANS+1"grans per cyl";
1680 FOR A =0 TO CYLS
1690   TT=ASC(MID$(C1$,A+1,1))
1700   PRINT @ 601,"working on cyl"A;
1710   FOR B=0 TO GRANS
1720     IF (TT AND BI(B))
           THEN PO(A,B)=300
1730   NEXT B:
     NEXT A
1740 FOR II=0 TO 7
1750   FIELD 1,(II*32) AS D$,32 AS R$(II)
1760 NEXT II
1770 IF LOF(1)>34 THEN DFCB=34 ELSE DFCB=LOF(1)
1780 FOR A =3 TO DFCB
1790   GET 1,A
1800   FOR B=0 TO 7
1810     RR$((A-3)*8+1+B)=R$(B)
1820   NEXT B:
     NEXT A
1830 CLOSE
1840 FOR A =1 TO 300
1850 IF LEN(RR$(A))<32
       THEN GOTO 1890
1860 B=A
1870 IF ASC(RR$(B))>127 AND ASC(MID$(RR$(B),6,1))<64
       THEN B=FN NR(RR$(B))
1880 IF ASC(RR$(B))>1 AND ASC(MID$(RR$(B),6,1))>64
       THEN T$=FN F$(RR$(B),5):
       IF ASC(T$)=254
         THEN MID$(RR$(FNNR(T$)),6,11)=MID$(RR$(B),6,11):
         B=FN NR(T$):
         GOTO 1870
1890 NEXT A
1900 RR$(300)="      locked out "+STRING$(16,255)
1910 RR$(0)=STRING$(32,32)
1920 FOR A =1 TO 300
1930   IF LEN(RR$(A))<32
         THEN GOTO 2110
1940   IF ASC(RR$(A))=0
         THEN GOTO 2110
1950   IF ASC(RR$(A))>127 AND ASC(MID$(RR$(A),6,11))<64
         THEN GOTO 2110
1960   T$=MID$(RR$(A),6,11)
1970   IF T$<STRING$(12,32)
         THEN GOTO 2110
1980 PRINT @ 480,CHR$(31)"Working on "MID$(RR$(A),6,11);
1990   LSET T$=FN F$(RR$(A),1)
2000   A1=ASC(T$)
2010   IF A1<CYLS+1
         THEN A2=FN SG(T$):
         A3=FN GC(T$):
         GOSUB 2390
       ELSE GOTO 2110
2020   LSET T$=FN F$(RR$(A),2)
2030   A1=ASC(T$)
2040   IF A1<CYLS+1
         THEN A2=FN SG(T$):
         A3=FN GC(T$):
         GOSUB 2390
        ELSE GOTO 2110
2050    LSET T$=FN F$(RR$(A),3)
2060    A1=ASC(T$)
2070    IF A1<CYLS+1
          THEN A2=FN SG(T$):
          A3=FN GC(T$):
          GOSUB 2390
        ELSE GOTO 2110
2080    LSET T$=FN F$(RR$(A),4)
2090    A1=ASC(T$)
2100    IF A1<CYLS+1
          THEN A2=FN SG(T$):
          A3=FN GC(T$):
          GOSUB 2390
2110 NEXT A
2120 LPRINT COMP$
2130 LPRINT TAB(10)"Disk name <"DNAME$">  Auto <"AUT0$">"
2140 LPRINT " Cyl   ";
2150 FOR A =0 TO GRANS
2160   LPRINT USING "\                  \";"GRAN"+STR$(A);
2170 NEXT A
2180 LPRINT
2190 FOR A =0 TO CYLS
2200   LPRINT USING "### ";A;
2210   FOR AA=0 TO GRANS
2220     LPRINT USING "< \              \>";MID$(RR$(PO(A,AA)),6,11);
2230   NEXT AA
2240   LPRINT
2250 NEXT A
2260 LPRINT NORM$
2270 LPRINT CHR$(12)
2280 CLS
2290 PRINT @ 1102,"Press <D> to return to DOS
        or any other key for NEXT disk";
2300 FOR A =1 TO 1
2310   A$=INKEY$
2320   A=LEN(A$)
2330 NEXT A
2340 IF A$="D"OR A$="d"THEN SYSTEM
2350 RUN
2390 PRINT A1;A2;A3",";
2400 FOR AA=0 TO A3
2410   P=A2+AA
2420   IF P>GRANS
         THEN A1=A1+1:
         A2=-AA:
         GOTO 2410
2430   PO(A1,P)=A
2440 NEXT AA
2450 RETURN
```

SHORT & SWEET SUBROUTINE

by Bert C. Guffens (COMPU-80)

Kasteelstraat 28, B-1800 Vilvoorde, Belgium

In all my programs that need a facility to "fill in the blanks" I use the following routine. Before I begin writing any new program, I load a program containing this and some other routines, and start by deleting what I am not going to need. This one I always need. It is located at line 100 and all my subroutines will have line numbers below 500, which is where I usually start the program.

```
100 ZZ=15360:A3=0:AA#=0:AA$="":W$=""
102 IF A2=1 THEN A4=48:A5=57
104 IF A2<>1 THEN A4=32:A5=122
106 FOR II=0 TO A1-1:POKEZZ+A0+II,95:NEXT II
108 W$=INKEY$:IFW$=""THEN108
110 IF W$=CHR$(91)THEN AA$=W$:RETURN
112 IF W$=CHR$(64)THEN AA$=W$:RETURN
114 IF W$=CHR$(13)THEN FOR II=A3 TO A1:POKE ZZ+A0+II,128:NEXT II:F
OR II=0 TO A3-1:AA$=AA$+CHR$(PEEK(ZZ+A0+II)):NEXT II:GOTO 124
116 IF W$=CHR$(8) THEN W$="":IFA3>0 THEN A3=A3-1:POKE ZZ+A0+A3,9
5: GOTO 108
118 IF W$=CHR$(24) THEN 100
120 IF W$<CHR$(A4) OR W$> CHR$(A5) THEN W$="":GOTO 108
122 IF A3=A1 THEN 108 ELSE POKE ZZ+A0+A3,ASC(W$):A3=A3+1:GOTO 1
08
124 IF A2=1 THEN AA#=VAL(AA$)
126 RETURN
```

How it works

Whenever you need to fill in a field, put the location in A0, put the length in A1, and give A2 the value of 1 if only numeric entries are allowed or 0 if any alphanumeric entry is to be excepted. Example: A0=405:A1=8:A2=1:GOSUB 100 will print a line of eight underscore characters and position the cursor (invisibly) at the beginning of the field. Upon return AA$ will contain all information in string form and in case A2 was 1, A# will also contain the numeric value.

Line 100 initializes the variables and incidentally, ZZ=15360 which is the start of your screen (position 0) can also be placed early in your program.

Line 102 limits your entries to numeric characters.

Line 104 extends the limits to alphanumeric entries.

Line 106 pokes a line at A0, with a length of A1 characters, in this case underline (CHR$(95)). You can change it to any character you like but if you do, also adjust line 116.

Line 108 waits for your input.

Line 110 - If you entered an up-arrow, which in my program is used to point at the previous entry, AA$ will cause it to be handled as required in your program.

Line 112 works the same as 110 but upon entering the '@' sign. I use this to escape from the present program part back to the menu.

Line 114, pressing ENTER ends all entries. All unused underlines are cleared, leaving only your entry. AA$ is concatenated with all the information you typed and program flows to 124 where, if A2 was 1 then AA# takes the value of AA$ and line 126 returns with a valid field.

Line 116 checks to see if you pressed the left arrow. If you did, the last entered character is cleared and replaced by an underline.

Line 118 checks to see if you pressed shift/left arrow. If you did then you restart in line 100 as you entered the subroutine.

Line 120 verifies if the entered character is valid, if not it is refused and you return to line 108 for the next input.

Line 122 if you have used up all your space then you are continuously directed to line 108 for an escape character. As long as your field length is not used up, you poke the ASCII value of W$ in its proper place, move the cursor one position and goto 108 for a new input.

You must recover the contents of AA$ before calling the subroutine again.

A small demonstration program (subroutine in place) for an address list (Belgian style).

```
10 CLS:CLEAR 2000:P$=CHR$(91):GOTO500 'jump all subroutines
500 PRINT@386,"NAME........:"
510 PRINT@450,"ADDRESS.....:"
520 PRINT@514,"POSTAL CODE.:"
530 PRINT@578,"TOWN........:"
540 OPEN"R",1,"ADDRESS/BEL:1",74:FIELD 1,20 AS A1$,30 AS A2$,4 AS A
3$,20 AS A4$:X=LOF(1)
550 X=X+1
560 A0=400:A1=20:A2=0:GOSUB100
570 IF AA$=CHR$(64) THEN CLOSE:CLS:END
580 LSET A1$=AA$
590 A0=464:A1=30:A2=0:GOSUB100
600 IF AA$=P$ THEN PRINT@A0,STRING$(A1,32);:GOTO560
610 LSET A2$=AA$
620 A0=528:A1=4:A2=1:GOSUB100
630 IFAA$=P$ THEN PRINT@A0,STRING$(A1,32);:GOTO590
640 LSET A3$=AA$
650 A0=592:A1=20:A2=0:GOSUB100
660 IF AA$=P$ THEN PRINT@A0,STRING$(A1,32);:GOTO620
670 LSET A4$=AA$
680 PUT 1,X:GOTO 550
```

This example is not intended to be a stand-alone program, it just demonstrates the usefulness of the subroutine. It works for the Models I/III. For CoCo change ZZ=1024 and adapt your screen addresses. If you are not going to bother with numbers forget A1 altogether, if you do not use it elsewhere in the program, it will be zero anyway.

TRY IT, YOU'LL LIKE IT.

MODEL 4 MULTIDOS BASIC CHANGES
Information provided by Vern Hester

The new Model 4 version of MULTIDOS is not a "true" Model 4 DOS in the same sense as the Model 4 versions of TRSDOS and DOSPLUS - it's actually closer to being a Model III DOS with enhancements to use all the capabilities of the Model 4. For memory-mapping purposes, the Model 4 version of MULTIDOS runs in the Model III mode.

But the Model 4 MULTIDOS is not just the Model III version of MULTIDOS with a 24x80 screen driver added. Many enhancements have been added to take advantage of the capabilities of the Model 4. One of the more noticable areas of change is the Model 4 version of SUPERBASIC. I asked Vern Hester to provide me with a list of the changes between the Model I/III and the Model 4 versions of SUPERBASIC, because I figured many of our readers might be interested in the new ehnancements. Here are the features of the new Model 4 SUPERBASIC (note: all numbers are decimal unless suffixed with an H):

When editing a line the <RIGHT-ARROW> moves the cursor one space as the <SPACE-BAR>. The <E>xit function has been deleted.

The REFERENCE utility is invoked with the "@" key.

Under BASIC shorthand the user can backspace to use the single keystroke commands. ([<SHIFT>]up-arrow, [<SHIFT>]down-arrow, comma, period, and slash)

Single letter commands

A[n][,m]	Auto line numbering starting at n (default 10) incrementing by m (default 10)
C	Continue program execution after STOP.
D[n][-m]	Delete lines from n to m. (Changed to have similar syntax to LIST. m does not have to be an existing line; however, at least one line should be between n and m.)
E[n]	Edit line n. (Default current line)
I	Insert. Invokes (psuedo) A .+1,1
K"program	Remove program
L[n][-m]	List lines from n to m
L"program	Load program
Mn,m	Move line n to m
Nn,m	Duplicated line n at m
P[n]	List page of lines from n (defaults current line)
R[q]	Run program starting at line q (see note)
R	Run program starting at the first line
R"program	Load and run program
S"program	Save program

The period "." may be used for n or m to represent the current line. NOTE: q = line number or a mandatory space LABEL"label" (RUN LABEL"DOGGY")

TRON has 7 additional functions when followed by the numbers 1-7

TRON or TRON 0 = Trace in the upper right corner of display.
TRON 1 = trace to line printer.
TRON 2 = display the BASIC statement in the lower left corner BEFORE it is executed.
TRON 3 = single step with delay. Delay is controlled via <CTRL><D> to increase delay, and <CTRL><F> to decrease delay. (need <CTR><S> between each delay change - see below)
TRON 4 = single step line.
TRON 5 = single step instruction.
TRON 6 = single step off.
TRON 7 = display erroneous statement.

Once TRON is envoked, <CTRL><Q> thru <CTRL><W> will modify a TRON from TRON 1 to TRON 7 respectively. i.e. <CTRL><S> will modify a TRON 2 to be TRON 2 and TRON 3. A <CTRL><V> will remove a TRON 2.

TRON or TRON 0 and TRON 1 are mutually exclusive.

TRON 4 and TRON 5 are mutually exclusive.

CMD functions

B "Soft" disable of "BREAK" key.
C Invoke "SPACE COMPRESSION" utility.
D Invoke debug.
E Interrogate last disk related error after BASIC initialized.
P Invoke "PACKER" UTILITY.
Q Dual/single dimension string sort.
S Exit BASIC.
U Invoke "UNPACKER" utility.
V Display active scalar variables.
X Invoke "REM REMOVER" utility.

CMD "uuuuu" requires a minimum of 6080 free bytes to execute. If less than 6080 bytes are free, then the message

ignored!

will be displayed and a return to the next statement, (if any), will occur.

KEYWORD changes

CLEAR - Clears all variables, closes all OPENed files, resets execution pointer, nullifies all FOR-NEXT loops, and GOSUBS; resets ON ERROR/STOP GOTOs, resets all variables to their default type, and activates the <BREAK> key.
CLEAR nnnnn - (nnnnn = 0 to 32767, -32768 to -1) Changes the amount of space allowed for string storage, and nullifies all FOR-NEXT loops and GOSUBs. Although nnnnn can be less than - 32768, a number much less than -25000 will produce an "Out of Memory" error.
ON STOP GOTO line number/"label" - This command will deactivate the <BREAK> key in a user input mode, and cause a branch to line number/"label" if the <BREAK> key is pressed during program execution.
GOTO "label"
GOSUB "label"
RESTORE line number/"label"
ON ERROR GOTO "label"
ON n GOTO/GOSUB "label"/line number,"label"/line number,etc...
RESUME "label"
IF-THEN-ELSE "label"
RUN LABEL "label".
CLS val - Homes the cursor and sets all of video refresh RAM to value of val (0 to 255).

Keywords removed from BASIC

AUTO	- USE A
CLOAD	- NOT USED
CONT	- USE C
CSAVE	- NOT USED
DELETE	- USE D
EDIT	- USE E
SYSTEM	- NOT USED
'	- USE REM

Keywords added to BASIC

LABEL	LABEL "label" - defines current line as being "label".
EXIT	EXIT line number/"label" - satisfies FOR-NEXT loop without FOR parameter reaching limit.
SORT	SORT var(0) - single dimension array sort. (will add later).
IND(	PRINT IND(n) - prints n spaces from current cursor position. (n = 0 to 255)
ERASE	ERASE var(0) - removes var array from RAM. (CMD"L"var(0) - v1.6)
ZERO	ZERO var(0) - sets all elements in var array to zero if numeric or null if string. (CMD"K"var(0) v1.6)
LPOS	LPOS(0) - returns the position of the printer under software control.
HEX$	HEX$(integer val) - returns a 4 character string equivalent to integer val.
BIN$	BIN$(integer val) - returns a 16 character string equivalent to integer val.
CALL	CALL integer val - executes code to located at integer val.
WPEEK	WPEEK(integer val) - returns the WORD located at integer val.

EXAMPLES of new BASIC commands

PROBLEM: To obtain the hexadecimal equivalent in a RAM location.

```
OLD
    20 DEFINT A-Z
    30 H$= "0123456789ABCDEF"
    40 X= PEEK(N)
    50 Y= PEEK(N+1)
    60 L1= INT (X/16)
    70 L2= X-L1*16
    80 M1= INT (Y/16)
    90 M2= Y-M1*16
   100 A$=MID$(H$,M1+1,1)+MID$(H$,M2+1,1)+
     MID$(H$,L1+1,1)+MID$(H$,L2+1,1)
```

```
NEW
    10 A$=HEX$(WPEEK(N))
```

PROBLEM: Prematurely exit a for-next loop.

```
OLD
     200 FOR X= 1 TO N
     210 IF A$(X)= "MATCH" THEN Y=X:X=N: NEXT:
      GOTO 400
     220 NEXT X
     ...
     400 PRINT A$(Y)

NEW
     200 FOR X=1 TO N
     210 IF A$(X)="MATCH" THEN EXIT 400
     220 NEXT X
     ...
     400 PRINT A$(X)
```

While programs are being developed, it is easier to use LABEL's to define various routines, than to assign and remember specific line numbers.

Rules for labels:

1. LABEL must be the first statement in a line.
2. The "label" referenced must match in character length and case.
3. Any character other than 0 and 34 is permitted.

```
60 IF A$="R"THEN GOSUB"NEW BOARD"
.. more program lines
480 LABEL"NEW BOARD"
```

The labels may be removed after a program is developed via the use of the "*" command. This command invokes RESOLVE/BOL and replaces references to labels with line numbers. The LABEL"label" is removed from each line.

PROBLEM: Limit/control an input to a numeric variable.

OLD: Varies with the limits of the acceptable input characters.

```
NEW: 10 CLEAR
     20 CLS
     30 AC$ = "012345"
     .. menu printed here with 5 options
     80 INPUT @ 704,1,95, USING A$, "SELECTION ";S
```

SYNTAX:

```
INPUT@pos,[#]len,char,[NOT][USINGexp$],["prompt";]varu
LINEINPUT @pos, [#] len, char, [NOT] [USING exp$], ["prompt";] var$
INPUT !col, row, [#] len, char, [NOT] [USING exp$], ["prompt";] varu
LINEINPUT!col,row,[#]len,char,[NOT][USINGexp$],["prompt";]var$
```

SYMBOL	MEANING
@pos	Specifies exactly - in terms of Video Display positions - where the INPUT prompt or INPUT field - if no prompt, will begin printing. Integer expression between 0 and 1023 (64X16), or 0 and 1919 (80X24).
!col,row	Specifies exactly which column (col), and which row the INPUT prompt or INPUT field - if no prompt, will begin printing. Integer expressions between 0 and 63 for "col", and 0 to 15 for "row" (64X16), or between 0 and 79 for "col", and 0 to 23 for "row" (80X24).
#	Specifies automatic <ENTER> when INPUT field is full.
len	This is the length of the INPUT field. Integer expression between 1 and 255.
char	This is the field character. Integer expression between 1 and 255. (These are Video "POKE" values not "PRINT" values).
NOT	Mask reject indicator. (see below)
USING	Mask indicator. (see below)
exp$	String expression representing the mask characters. USING exp$ = Only use characters in exp$. NOT USING exp$ = Do not use any characters in exp$.
"prompt";	The optional prompt message.
varu	Numeric or string variable, or numeric/string variable list.
var$	A single string variable.

MODEM PROBLEMS AND PHONE LINES

by Mel Patrick

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

Had an interesting thing happen to a friend the other day. He had purchased an auto answer modem and couldn't get it to auto answer. After building up some real anger he stormed off to the store and demanded some satisfaction.

The manager took the modem, plugged it into the phone line, set the switches for auto answer and called it from another in-store line. When it answered perfectly, you might say that the wind went out of his sails. Nevertheless, the store provided him with a brand new modem and even tested it to make sure that it worked before allowing him to take it home.

Upon arriving at home he proceeded to plug in the modem and dial it up. No, it still didn't answer. But why? He was considering calling B.C. Tel and complaining or returning the modem for a complete refund, with no doubt some ill feelings towards the store.

About this time he phoned me and asked if I had any ideas. Contrary to what you might think or have heard at the club meetings, I didn't have any brainstorms, although I did say it could be done. A few ideas were tossed around and he tried them without any success.

We then discussed a few other ideas and low and behold we hit on the cause of the problem. He has a single phone line entering his house and is somewhat of a phone fanatic. On that line there are a total of five phones, which as it turned out, dropped the bell signal level low enough so that the modem couldn't detect it. Consequently it wouldn't answer when it got the low ring signal.

The moral of this story is a simple one. I didn't solve his problem, nor did he. But between the two of us, we managed to work out a solution. That is what computing usually lacks. Someone has a problem and they can't solve it so they give up and write it off as impossible. All that was required was a little insight from another person to help out. If you have a computing problem ask someone about it. If they have already had it happen to them it could save a great deal of anguish.

[NORTHERN BYTES Editor's Note: To expand on the above, almost every telephone or telephone device contains some sort of ring signal detect circuit. In the older style phones, this consists of a capacitor in series with the coils of the electromagnets on the ringer, thus forming a tuned circuit (usually tuned to 20 Hz on most systems). Newer "electronic" phones may substitute a solid state circuit of some sort. The important thing to remember is that the circuit, be it mechanical or electronic, remains connected to the phone line when the associated phone is on the hook. And, because these are often tuned circuits, they have the potential to absorb certain audio frequencies, while not affecting others.

The point is that if you are having ANY sort of problem in using a modem on your line that does not occur when you take your modem to a friend's house, try unplugging all other phones and phone devices on your line before taking any drastic measures (like calling the phone company). It just may be that the circuitry in one or more of the other phones is forming an audio "trap" that is absorbing the modem frequencies that are produced and/or received by your modem. Obviously, this sort of problem is rare, but it never hurts to eliminate all possibilities before giving up completely...]

DATA COMPRESSION TECHNIQUES USING SQ AND USQ

by Mark DiVecchio

[This article has been in so many user group newsletters that I'm not sure where it originated, though most sources give original credit to the San Diego IBM PC Users Group. It is very apparent that the author was working in an MS-DOS environment, but the principles discussed here are just as applicable to the TRS-80 versions of SQ and USQ.]

In any computer system, efficient management of the file storage space is important. The two programs SQ and USQ reduce the size of data files by using the Huffman Data Compression Algorithm.

A file is composed of a sequence of bytes. A byte is composed of 8 bits. That byte can have a decimal value between 0 and 255. A typical text file, like a C language program, is composed of the ASCII character set (decimal 0 to 127). This character set only requires seven of the eight bits in a byte. Just think -- if there were a way to use that extra bit for another character, you could reduce the size of the file by 12.5%. For those of you who use upper case characters only, you use only about 100 of the ASCII codes from 0 to 255. You could save 60% of your storage if the bits not needed within each byte could be used for other characters. What if you could encode the most frequently used characters in the file with only one bit? For example, if you could store the letter "e" (the most commonly used letter in the English language) using only one bit: "1", you could save 87.5% of the space that the "e" would normally take if it were stored as the ASCII "0110 0101" binary.

The answer to these questions is the SQ and USQ programs.

The SQ program uses the Huffman coding technique to search for the frequency of use of each of the 256 possible byte patterns, and it then assigns a translation for each character to a bit string. All of these bit strings are placed end to end and written onto a disk file. The encoding information is also put on the file since the USQ program needs to know the character distribution of the original file.

The USQ program reads in the encoding information and then reads in the encoded file. It is a simple matter to scan the encoded file and produce an output file which is identical to the file that SQ started with.

HUFFMAN CODING TECHNIQUE

This is by far the most popular encoding technique in use today. The Huffman encoding replaces fixed bit characters with variable length bit strings. The length of the bit string is roughly inversely proportional to the frequency of occurrence of the character. For those of you inclined to such symbolism:

$$\text{Length of bit string} = \log_2 (\text{character probability})$$

The implementation of the algorithm which we will discuss encodes fixed bit strings of length 8.

The algorithm requires two passes through the input file. The first pass builds a table of 256 entries showing the frequency of each occurrence of each of the 256 possible values for a byte of information.

Once the counting is complete, the algorithm must decide which bit strings to associate with each of the 256 characters that were found in the file. Note that if a particular byte value was never used, no string association is needed.

The second pass through the input file converts each byte into its encoded string. Remember that when the output file is created, the information used for encoding must also be written on the file for use by the decoding program.

The decoding program reads the encoding information from the file and then starts reading the bit strings. As soon as enough bits are read to interpret a character, that character is written onto the final output file. See the next two sections on how SQ and USQ actually implement this.

Even though this article primarily has addressed ASCII input files, there is nothing which restricts this algorithm to ASCII. It will work on binary files (.COM or .EXE [or /CMD or /OBJ in TRS-80 usage]) as well. But since the length of the encoded bit string is approximately equal to the inverse of the frequency of occurrence of each 8 bit byte, a binary file may not compress very much. This is because a binary file most likely has a uniform distribution over the 256 values in a byte. A machine language program is not like the English language where the letter "e" is used far more than other letters. If the distribution is uniform, the encoded bit strings will all be the same length and the encoded file could be longer than the original (because of the encoding information on the front of the file). All of this has to be qualified, because machine language programs tend to use a lot of "MOV" instructions and have a lot of bytes of zeros so that encoding .COM and .EXE files does save some disk space.

SQ

The SQ program is an example of the Huffman algorithm.

The first thing that SQ does is read through the input file and create a distribution array for the 256 possible characters. This array contains counts of the number of occurrences of each of the 256 characters. The program counts these values in a 16 bit number. It makes sure that, if you are encoding a big file, counts do not exceed a 16 bit value. This is highly unlikely, but it must be accounted for.

At the same time, SQ removes strings of identical characters and replaces them with the ASCII character DLE followed by a character count of 2-255. SQ replaces the ASCII DLE with the pair of characters: DLE DLE. This is not related to the Huffman algorithm but just serves to compress the file a little more.

Once SQ has scanned the input file, it creates a binary tree structure containing this frequency occurrence information. The most frequently occurring characters have the shortest path from the root to the node, and the least frequently occurring characters have the longest path. For example, if your file were 'ABRACADABRA' (a very simple and magical example) the table of frequency of occurrences would be:

LETTER	# OF OCCURRENCES
A	5
B	2
C	1
D	1
R	2
all the rest	0

Since the letter "A" occurs most often, it should have the shortest encoded bit string. The letters "C" and "D" should have the longest. The other characters which don't appear in the input file don't need to be considered.

SQ would create a binary tree to represent this information. The tree might look something like this (for purposes of discussion only):

```
   root  <--- Computer trees are always upside down!
  /    \
 1      0  <--- This is called a node.
/      / \
A     1   0
     /   / \  <--- This is called a branch.
    B   1   0
       /   / \
      C   1   0
         /     \
        D       R  <--- A leaf.
```

From this our encoded bit strings which are kept in a translation table would be:

Table Entry	Character	Binary
1	A	1
2	B	01
3	C	001
4	D	0001
5	R	0000

The output file would be:

```
A  B  R     A  C   A  D    A  B  R    A
1  01 0000  1  001 1  0001 1  01 0000 1
(binary)

A1  31  A1
(hex)
```

We have reduced the size of your file from ten bytes to three bytes for a 70% savings. For this simple example, things aren't that well off since we must put the binary tree encoding information onto the file as well. So the file size grew a lot. But consider a file with the word ABRACADABRA repeated 100,000 times. Now the encoding information is going to be a very, very small percentage of the output file and the file will shrink tremendously.

SQ opens the output file and writes out the binary tree information. The SQ rewinds the input file and rereads it from the beginning. As it reads each character, it looks into the translation table and outputs the corresponding bit string.

SQ is a little more complicated than what I have outlined since it must operate in the real world of hardware, but this is a fairly complete description of the algorithm.

USQ

The USQ program is very straightforward. It reads in the encoding information written out by SQ and builds the identical binary tree that SQ used to encode the file.

USQ then reads the input file as if it were a string of bits. Starting at the root of the tree, it traverses one branch of the tree with each input bit. If it has reached a leaf, it has a character and that character is written to the output file. USQ then starts at the root again with the next bit from the input file.

WHAT DOES IT ALL MEAN?

Now that we understand the algorithm and a little about how the SQ and USQ programs work, we can use that knowledge to help us run our systems a little more efficiently. (So all of this theory is worth something after all)!

1. Files must be above a threshold size, or else the output file will be longer than the input file because of the decoding information put at the beginning of the compressed data. We don't know the exact size of the threshold because the encoding binary tree information depends on the distribution of the characters in a file. At least we know to check the size of the encoded file after we run SQ to make sure our file didn't grow.
2. Some files will not compress well if they have a uniform distribution of byte values, for example, .COM or .EXE files. This is because of the way SQ builds the tree. Remember that bytes with the same frequency of occurrence are at the some depth (usually) in the tree. So if all of the bytes have the same depth, the output strings are all the same length.
3. SQ reads the input file twice. If you can, use RAM disk at least for the input file and for both files if you have the room. The next best case is to use two floppy drives, one for input and one for output. This will cause a lot of disk starts and stops but not much head movement. Worst case is to use one floppy drive for both input and output. This will cause a lot of head movement as the programs alternate between the input and output files.

OTHER COMPRESSION TECHNIQUES

RUN-LENGTH ENCODING

Run-length encoding is a technique whereby sequences of identical bytes are replaced by the repeated byte and a byte count. As you might guess, this method is effective only on very specialized files. One good candidate is a screen display buffer. A screen is made up mostly of "spaces". A completely blank line could be reduced from 80 bytes of spaces to one space followed by a value of 80. To go from 80 bytes down to two bytes is a savings of almost 98%. You might guess that for text files or binary files, this technique does not work well at all.

ADAPTIVE COMPRESSION

This technique replaces strings of characters of code. For example, the string "ABRACADABRA" would be replaced by a code. Typical algorithms use a 12 bit code. The algorithm is unique in that it only requires a single pass through the input file as the encoding is taking place. The current incarnation of this procedure is called the LZW method (after co-inventors A. Lempel, J. Ziv and T. Welch). This algorithm claims a savings of 66% on machine language files and up to 83% on COBOL files.

[NORTHERN BYTES editor's note: The method used by the BASIC interpreter to tokenize BASIC keywords down to just one byte might be considered an example of a very simple form of adaptive compression.]

OTHER READING

If you are interested in reading more about data compression techniques, you may be interested in these articles:

H. K. Reghbati, "An Overview of Data Compression Techniques", Computer Magazine, Vol. 14, No. 4, April 1981, pp. 71-76.

T. A. Welch, "A Technique for High-performance Data Compression", Computer Magazine, Vol. 17, No. 6, June 1984, pp. 8-19.

J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. I1-23, No. 3, May 1977, pp. 337-343.

[NORTHERN BYTES editor's note: The TRS-80 versions of SQ and USQ (written by David Huelsmann) are available on TAS Public Domain Library disk #015 ($10 for #015 only, or $20 for the set #015-#017 which includes Mr. Huelsmann's Library Utility programs, plus $3 shipping/handling per order). Mr. Huelsmann is the Chairman of the TRS-80 Conference on the TBC BBS at (505) 821-7379 (1200-2400 bps anytime, 300 bps after 2300 Mountain Time. Verification required - Request entry to TRS-80 Conference SIG).

I'd like to especially encourage those who may be creating networking type BBS programs (the kind that send messages or files to other BBS's in the middle of the night) to consider using a data compression algorithm of some sort to automatically squeeze the data prior to transmission, and then unsqueeze it at the receiving end. This seems like a sensible thing to do, but few networking BBS's actually do it - yet such compression of data could save lots of money on the SYSOP's phone bill!]

NEVER RETYPE TEXT AGAIN!

One of the reasons that it is so difficult to do NORTHERN BYTES is because of the amount of material that we reprint from other publications. Most of this material has to be re-typed. I am a slow typist and hate to type anyway, but have sometimes been able to get articles re-typed by people at The Alternate Source. Unfortunately, when humans re-type an article, they usually make mistakes - ESPECIALLY on a long program listing, the very thing that absolutely MUST be accurate.

Unfortunately, most of the current generation of text readers are not much better. They work by comparing characters on the printed page against characters stored in the reader's memory. This MAY work properly if the characters on the printed page EXACTLY match the stored characters. Considering the hundreds of typefaces now in use in various publications, the odds against that are pretty large. Besides, most text readers cannot cope with proportionally spaced text, text that is in columns or zones on a page, text that wraps around illustrations, etc.

Now the Palantir Corporation (2500 Augustine Drive, Santa Clara, California 95054, telephone 408-986-8006) has introduced a device called the Compound Document Processor which not only overcomes all of the above limitations, it also has the capability to digitize images that may be on the page. In other words, it can read the text that wraps around an image and digitize the image so that, for example, a page of mixed text and graphics could be reconstructed after editing of the text. The text itself can be in just about any type font, and no, you don't have to pre-train the unit with a sample of the font - it works using a process called "feature extraction" that looks for unique features of characters to distinguish them, and is also context-sensitive so that it can distinguish between characters such as "l", "1", and "I" (but can it distinguish between the variable "I" and the constant "1" in a BASIC program listing? That would be a real test!).

Unfortunately, such sophistication does not come cheap. The current price of the system is $39,500 (now if we could only get 1000 newsletter editors to chip in $39.50 each...). So I doubt that the price of such technology will drop enough to save Northern Bytes (even at one-tenth the price, we couldn't afford it!). This would be an ideal item for libraries, print shops and copy centers to purchase as a convenience for their patrons (with a coin-operated option added), but the current price is prohibitive for even that use in most situations. Still, the technology is now available, so it's only a matter of time until the price drops. Maybe in ten or twenty years from now, the headline of this article will be a reality for all of us, not just the very rich!

USDATE/SYS OVERLAY FOR NEWDOS/80 MODEL I

by Mark Davey
P.O. Box 433, Sunnybank, Queensland 4109, Australia

In Northern Bytes Volume 5, Number 2 Jack Decker published a program called "SETDATE/ASM". This program would display a recent date on the computer's video display unit, at power up or reboot, and with a few key strokes it was possible to update the system date, and save the new date. The saved date would be recalled when the computer was rebooted.

In Northern Bytes Volume 5, Number 3, Jack Decker published an article on adding your own /SYS overlays to NEWDOS/80; whilst in Northern Bytes volume 5, number 4, a similar article was published entitled "A New Overlay Module For NEWDOS/80 Version 2". This article gave instructions on how to add a new JKLSYS/SYS file.

Why not, I thought, take Jack Decker's SETDATE program and turn it into a NEWDOS/80 "/SYS" overlay, that would be activated by the NEWDOS/80 SYSTEM options AY and AZ (AY sets date and time prompts after cold boot, whilst AZ sets date and time prompts after warm boot).

"USDATE/SRC" is the result of this effort, a new /SYS file for Model I NEWDOS/80 users. The new /SYS file is loaded into the BASIC overlay area starting 5200H by SYS0/SYS, under the conditions you have specified by SYSTEM options AY and AZ. In operation it is similar to the original SETDATE program, with a few exceptions. Most notable of these exceptions is the lack of displayed instructions (in order not to clutter the screen I decided to remove them all).

The instructions are included at the start of the program listing, and for the sake of brevity, will not be repeated here. I would, however, like to draw your attention to one particular change to the original program. When the <ENTER> key was pressed, SETDATE would take the displayed date and not only set the computer's date, but save that date on disk for future use. In this program, pressing <ENTER> will ONLY set the computer's date. In order to save the date to disk, as well as setting the computer's date, you must press <SHIFT><ENTER>, i.e. whilst holding the shift key down, press the enter key.

Why did I do this? Well I like to keep a write protect tab on my system disk, and if I do this I obviously can't write to the disk. Thus I needed to be able to set the computer's date without writing to disk. At the same time I wanted to be able to occasionally alter the date stored on the disk. Hence I needed to be able to update the computer's system date in two ways; one, without saving the date to disk; and two, saving the date to disk.

The installation procedure is similar to that described by Joachim Kelterbaum in his article "A New Overlay Module For NEWDOS/80 Version 2". First type in the assembly language listing using an editor/assembler program, and save it on a NEWDOS/80 system disk (I would recommend you make a backup of NEWDOS/80 for this procedure). Assemble the program and save it to the disk as USDATE/CMD.

To operate, the program MUST be positioned at a specific directory entry. I have chosen position 28, which is byte 60H of relative sector 6 in the directory. The easiest way to place the file in this position is to use the CREATE command to create a series of directory entries, e.g. CREATE TRY1:0; CREATE TRY2:0; etc. Now examine the directory to find which of the newly created files is at position 28 (byte 60H of relative sector 6 in the directory). If there is no entry at position 28, continue creating files until you have one there. When you do, note the name of this entry, then return to the NEWDOS/80 command level and kill all the unwanted files that you have created.

Now change the name of this entry to USDATE/SYS (using the NEWDOS/80 RENAME function), and copy USDATE/CMD to USDATE/SYS. Once copied, you may kill the USDATE/CMD file.

The next step is to examine the directory entry for USDATE/SYS, and make sure that it has only one extent (see the directory column headed "EXTS"). This is important as an overlay is always assumed to have only one extent. If the file has more than one extent you must make a new system disk by copying it to another disk with the command COPY 0,1,,FMT,CBF. Now check the directory entry for USDATE/SYS once again to make sure that it has only one extent.

Once the program is saved in the correct directory position, use SUPERZAP to change the first byte of the directory entry for USDATE/SYS to 5FH. NEWDOS/80 will now consider this file to be a standard /SYS file when displaying a directory listing.

NEWDOS/80 must now be patched to call the new overlay in accordance with SYSTEM command AY and AZ options. To do this load SUPERZAP and zap SYS0/SYS file relative sector 0012, byte 56. It should be 21 63 50, and you must change it to 3E 9C EF. If you wish, you can change bytes 59 through to 67 inclusive, to 00. This is optional, as it just zeros the balance of the normal Date input routine, which will no longer be used. The following is the sector from byte 50H onwards.

```
BEFORE:  CD 67 44 C3 05 44 21 63 50 CD 64 4F 01 9C 50 11 46 40
3E 2F CD 6F 4F 00

AFTER:  CD 67 44 C3 05 44 3E 9C EF 00 00 00 00 00 00 00 00 00
00 00 00 00 00
```

This above patch corresponds to the following machine code instructions:

```
3E 9C        LD        A,9C
EF           RST       28H
```

The 9C being comprised of AAA BBBBB, where BBBBB is 28 decimal, and AAA is 100 binary. The RST 28H instruction loads and runs the overlay at the directory entry specified by the BBBBB portion of the byte stored in the A register. The AAA portion of the byte must have at least one of the bits set. This is a very simplified explanation. To obtain a better understanding of how this works I would refer you to the articles mentioned above, and to comments on this subject made by Nathan W. Harrington which appeared in Northern Bytes Volume 5, Number 4, page 2.

As the program stands it will not jump to the time request. For those who would like to be able to enter the time as well as the date, the standard NEWDOS/80 time input can be activated by changing line 2430 in the source code from a "RET" to a "JP 4F50H", and your wish will be granted!

When I originally altered the SETDATE program, I changed it to allow date storage in the European format, i.e. DD/MM/YY. After successfully making the new /SYS file I altered the date storage back to the U.S. format, i.e. MM/DD/YY, for the majority of Northern Byte readers. I have made a copy of the European date version of the program available to Jack Decker, and if Jack is willing, you may be able to obtain a copy of the program by contacting him. [Editor's note: Jack's address is 1804 West 18th Street #155, Sault Ste. Marie, Michigan 49783-1268. When writing, you must enclose a blank diskette and a stamped, self-addressed disk mailer.]

For those who may be using Alan Johnstone's enhanced NEWDOS/80 (TAS disk ND-1), assemble the program with an origin of 6000H instead of 5200H (see line 550). All other installation procedures are the same.

```
00100 ;*****  USDATE/SRC  *****
00110 ;This program will set the computer system date
00120 ;in the U.S. format.
00130 ;It is called by SYS0/SYS as an overlay file
00140 ;which is loaded into the BASIC overlay area
00150 ;at 5200H.
00160 ;SYSTEM 0 options AY and AZ enable and disable
00170 ;this overlay in the same manner as they
00180 ;previously enabled and disabled the DATE prompt.
00190 ;
00200 ;INSTRUCTIONS
00210 ;The arrow keys may be used to advance or retard
00220 ;the date.  Shifted arrow keys increase the rate
00230 ;of change in the date.
00240 ;When the date is correct, press the <ENTER> key
00242 ;to update the system date only.
00244 ;To update the system date, and save the new
00246 ;date to disk, press <SHIFT><ENTER>.
00248 ;NOTE:  The disk must not be write protected
00249 ;       if you intend to save the date to disk!
00250 ;To bypass setting the system date, press either
00260 ;the <BREAK> or <CLEAR> key.
00270 ;
00280 ;Original program written and copyrighted in
00290 ;1983 by Jack Decker.
00300 ;Modifications made by Mark Davey,
00310 ;12th October, 1986.
00320 ;
00330 ;Original copyright and conditions are still
```

```
            00340 ;in force, i.e. the program may be reprinted
            00350 ;for non-commercial purposes or placed in
            00360 ;your club's computer library.
            00370 ;THE PROGRAM MAY NOT BE SOLD, OR INCLUDED AS
            00380 ;PART OF A PROGRAM PACKAGE THAT IS BEING SOLD.
            00390 ;
            00400 ;
5200        00550        ORG     5200H          ;MUST end with 00H
            00560 ;
            00570 ;String and date storage area used by program
            00580 ;
5200 2D52   00590 TABLE  DEFW    SUN            ;Table of string pointers
5202 3352   00600        DEFW    MON            ;  point to strings
5204 3952   00610        DEFW    TUE            ;  containing days of
5206 4052   00620        DEFW    WED            ;  the week
5208 4952   00630        DEFW    THU
520A 5152   00640        DEFW    FRI
520C 5752   00650        DEFW    SAT
520E 5F52   00660        DEFW    JAN            ;Table of string pointers
5210 6652   00670        DEFW    FEB            ;  point to strings
5212 6E52   00680        DEFW    MAR            ;  containing months of
5214 7352   00690        DEFW    APR            ;  the year
5216 7852   00700        DEFW    MAY
5218 7B52   00710        DEFW    JUN
521A 7F52   00720        DEFW    JUL
521C 8352   00730        DEFW    AUG
521E 8952   00740        DEFW    SEP
5220 9252   00750        DEFW    OCT
5222 9952   00760        DEFW    NOV
5224 A152   00770        DEFW    DEC
5226 FE     00780 MARKER DEFB    0FEH           ;Start date storage area
5227 56     00790 DATSTR DEFB    86D            ;Year storage
5228 0C     00800        DEFB    12D            ;Day storage
5229 0A     00810        DEFB    10D            ;Month storage
522A 00     00820        DEFB    0D             ;Day of week storage
522B 6C07   00830 CNTURY DEFW    1900D          ;Century storage
522D 53     00840 SUN    DEFM    'Sunda'        ;Strings containing days
     75 6E 64 61
5232 F9     00850        DEFB    'y'+80H        ;  of week
5233 4D     00860 MON    DEFM    'Monda'
     6F 6E 64 61
5238 F9     00870        DEFB    'y'+80H
5239 54     00880 TUE    DEFM    'Tuesda'
     75 65 73 64 61
523F F9     00890        DEFB    'y'+80H
5240 57     00900 WED    DEFM    'Wednesda'
     65 64 6E 65 73 64 61
5248 F9     00910        DEFB    'y'+80H
5249 54     00920 THU    DEFM    'Thursda'
     68 75 72 73 64 61
5250 F9     00930        DEFB    'y'+80H
5251 46     00940 FRI    DEFM    'Frida'
     72 69 64 61
5256 F9     00950        DEFB    'y'+80H
5257 53     00960 SAT    DEFM    'Saturda'
     61 74 75 72 64 61
525E F9     00970        DEFB    'y'+80H
525F 4A     00980 JAN    DEFM    'Januar'       ;Strings containing
     61 6E 75 61 72
5265 F9     00990        DEFB    'y'+80H        ;  months of year
5266 46     01000 FEB    DEFM    'Februar'
     65 62 72 75 61 72
526D F9     01010        DEFB    'y'+80H
526E 4D     01020 MAR    DEFM    'Marc'
     61 72 63
5272 E8     01030        DEFB    'h'+80H
5273 41     01040 APR    DEFM    'Apri'
     70 72 69
5277 EC     01050        DEFB    'l'+80H
5278 4D     01060 MAY    DEFM    'Ma'
     61
527A F9     01070        DEFB    'y'+80H
527B 4A     01080 JUN    DEFM    'Jun'
     75 6E
527E E5     01090        DEFB    'e'+80H
527F 4A     01100 JUL    DEFM    'Jul'
     75 6C
5282 F9     01110        DEFB    'y'+80H
5283 41     01120 AUG    DEFM    'Augus'
     75 67 75 73
```

```
5288 F4     01130        DEFB    't'+80H
5289 53     01140 SEP    DEFM    'Septembe'
     65 70 74 65 6D 62 65
5291 F2     01150        DEFB    'r'+80H
5292 4F     01160 OCT    DEFM    'Octobe'
     63 74 6F 62 65
5298 F2     01170        DEFB    'r'+80H
5299 4E     01180 NOV    DEFM    'Novembe'
     6F 76 65 6D 62 65
52A0 F2     01190        DEFB    'r'+80H
52A1 44     01200 DEC    DEFM    'Decembe'
     65 63 65 6D 62 65
52A8 F2     01210        DEFB    'r'+80H
52A9 44     01220 PROMPT DEFM    'DATE? '       ;Date prompt
     41 54 45 3F 20
52AF A0     01230        DEFB    ' '+80H
52B0 9A     01460 MSG    DEFB    1AH+80H        ;1AH moves down one line
52B1 1E     01470 ENDYR  DEFB    1EH            ;1EH clears to line end
52B2 9D     01480        DEFB    1DH+80H        ;1DH moves to line start
            01490 ;
            01500 ;Start of actual machine-language program
52B3 21A952 01730 START  LD      HL,PROMPT      ;Point to date prompt
52B6 CDED53 01740        CALL    DSPMSG         ;  and display it
52B9 012A52 01750        LD      BC,DATSTR+3    ;Point to day of wk byte
52BC 0A     01760        LD      A,(BC)         ;Get day of week (0-6)
52BD CDE553 01770        CALL    PRTSTR         ;Print day of week string
52C0 CD0454 01780        CALL    PRTCOM         ;Print comma and space
52C3 0B     01790        DEC     8C             ;Point to month byte
52C4 0A     01800        LD      A,(BC)         ;Get month (1-12)
52C5 C606   01810        ADD     A,6            ;Offset for string table
52C7 CDE553 01820        CALL    PRTSTR         ;Print month string
52CA CD0954 01830        CALL    PRTSPC         ;Print space
52CD 0B     01840        DEC     BC             ;Point to day byte
52CE 0A     01850        LD      A,(BC)         ;Get day (1-31)
52CF 6F     01860        LD      L,A            ;Put day in L
52D0 2600   01870        LD      H,0            ;HL = day
52D2 C5     01880        PUSH    BC             ;Save date storage pntr
52D3 CDFB53 01890        CALL    PRTNUM         ;Print day
52D6 C1     01900        POP     BC             ;Restore date storage ptr
52D7 CD0454 01910        CALL    PRTCOM         ;Print comma and space
52DA 0B     01920        DEC     BC             ;Point to year byte
52DB 0A     01930        LD      A,(BC)         ;Get year (0-99)
52DC 4F     01940        LD      C,A            ;Put year in C
52DD 0600   01950        LD      B,0            ;BC = last 2 digits year
52DF 2A2B52 01960        LD      HL,(CNTURY)    ;Get century offset
52E2 09     01970        ADD     HL,BC          ;HL = Year (all 4 digits)
52E3 CDFB53 01980        CALL    PRTNUM         ;Print year
52E6 21B152 01990        LD      HL,ENDYR       ;Point to ctrl chr string
52E9 CDED53 02000        CALL    DSPMSG         ;Output it to video
52EC 3A4038 02010 GETKEY LD      A,(3840H)      ;Get BREAK/CLEAR row
52EF E606   02020        AND     6              ;Mask out other keys
52F1 206E   02030        JR      NZ,EXIT        ;Exit if BREAK/CLEAR
52F3 CD2B00 02040        CALL    2BH            ;Get keystroke if any
52F6 B7     02045        OR      A              ;Z flag set if no key
52F7 28F3   02050        JR      Z,GETKEY       ;If no key was pressed
52F9 FE0D   02060        CP      0DH            ;Was it the ENTER key?
52FB 2076   02070        JR      NZ,NOTCR       ;Go if not ENTER
52FD 212752 02080        LD      HL,DATSTR      ;Point to prgram date
5300 E5     02090        PUSH    HL             ;Save program date pntr
5301 CD0F54 02100        CALL    GETBFR         ;Find memory date storage
5304 010300 02110        LD      BC,3           ;Number of bytes to move
5307 EDB0   02115        LDIR                   ;Move from program to mem
5309 3A8038 02116        LD      A,(3880H)      ;Get SHIFT key row
530C CB47   02117        BIT     0,A            ;Was SHIFT key pressed?
530E 2003   02118        JR      NZ,UPDATE      ;Save new date to disk
5310 E1     02119        POP     HL             ;Restore register
5311 184E   02120        JR      EXIT           ;No disk save, go exit
            02122 ;LINES 2124 - 2358 COMPRISE THE DISK I/O ROUTINE.
5313 010005 02124 UPDATE LD      BC,500H        ;Move SYS0/SYS
5316 115755 02126        LD      DE,SYS0        ;  initialisation code
5319 21004D 02127        LD      HL,4D00H       ;  out of the SYS2/SYS
531C EDB0   02128        LDIR                   ;  overlay area.
531E 0600   02130        LD      B,0            ;Logical Record lngth=256
5320 113754 02140        LD      DE,FCB         ;File control Block pntr
5323 215754 02150        LD      HL,FILBUF      ;File I/O Buffer pntr.
5326 CD2444 02160        CALL    4424H          ;DOS OPEN routine
5329 203D   02170        JR      NZ,ERREXT      ;Go if error
532B 13     02171        INC     DE             ;Point to FCB+1
532C 1A     02172        LD      A,(DE)         ;Get status byte from FCB
532D F640   02173        OR      40H            ;Set bit 6
```

```
532F 12         02174          LD      (DE),A            ;Re-save status byte
5330 1B         02175          DEC     DE                ;Point to FCB
5331 CD3644     02180          CALL    4436H             ;READ sector 0 to buffer
5334 2032       02190.         JR      NZ,ERREXT         ;Go if error
5336 E1         02200          POP     HL                ;Get program date pointer
5337 D5         02210          PUSH    DE                ;Save FCB pointer
5338 118154     02220          LD      DE,MARKER-TABLE+4+FILBUF
                02230 ;Above instruction points DE to first location in sector
                02240 ;zero of disk file that can possibly contain MARKER byte
533B 1A         02250 FNDDAT   LD      A,(DE)            ;Get byte frm disk sector
533C 13         02260          INC     DE                ;Bump pointer to next loc
533D FEFE       02270          CP      0FEH              ;MARKER byte found?
533F 20FA       02280          JR      NZ,FNDDAT         ;Try again if not
5341 010400     02290          LD      BC,4              ;Move 4 bytes frm program
5344 EDB0       02300          LDIR                      ;  storage to disk sector
5346 D1         02310          POP     DE                ;Restore FCB pointer
5347 CD3F44     02320          CALL    443FH             ;Reset to sector zero
534A 201C       02330          JR      NZ,ERREXT         ;Go if error
534C CD3944     02340          CALL    4439H             ;Write sector back to dsk
534F 2017       02345          JR      NZ,ERREXT         ;Go if error
5351 CD2844     02348          CALL    4428H             ;DOS CLOSE routine
5354 2012       02350          JR      NZ,ERREXT         ;Go if error
5356 010005     02352          LD      BC,500H           ;Move SYS0/SYS
5359 11004D     02354          LD      DE,4D00H          ;  initialisation code
535C 215755     02356          LD      HL,SYS0           ;  back to the SYS2/SYS
535F EDB0       02358          LDIR                      ;  overlay area.
5361 21B052     02410 EXIT     LD      HL,MSG            ;Drop down one line
5364 CDED53     02420          CALL    DSPMSG            ;Display message
5367 C9         02430          RET                       ;Cont. DOS initialisation
                02431 ;TO ENTER THE TIME AS WELL, CHANGE 'RET' IN 2430 TO
                02432 ;'JP     4F50H'
5368 F5         02433 ERREXT   PUSH    AF                ;Save error code
5369 21B052     02434          LD      HL,MSG            ;Drop down one line
536C CDED53     02436          CALL    DSPMSG            ;Display message
536F F1         02437          POP     AF                ;Restore error code
5370 C30944     02438          JP      4409H             ;If error use ERROR rtne
5373 FE5B       02440 NOTCR    CP      5BH               ;Was keystroke up-arrow?
5375 2814       02450          JR      Z,ARROW           ;Go if up-arrow
5377 47         02460          LD      B,A               ;Save keystroke in B
5378 F610       02470          OR      10H               ;Make shifted=unshifted
537A FE18       02480          CP      18H               ;Invalid keystroke if
537C DAEC52     02490          JP      C,GETKEY          ;  less than ASCII 18H
537F FE1C       02500          CP      1CH               ;Keystroke valid if less
5381 D2EC52     02510          JP      NC,GETKEY         ;  than ASCII 1CH
5384 A8         02520          XOR     B                 ;Check for shifted char
5385 2003       02530          JR      NZ,UNSHFT         ;If not shifted char
5387 323C40     02540          LD      (403CH),A         ;Zero arrow row storage
538A 78         02550 UNSHFT   LD      A,B               ;Restore original char
538B 21B352     02560 ARROW    LD      HL,START          ;RET addr for following
538E E5         02570          PUSH    HL                ;Save it on stack
538F 212A52     02580          LD      HL,DATSTR+3       ;Point to day-of-week
5392 0F         02590          RRCA                      ;Shift bit 0 into carry
5393 3030       02600          JR      NC,BACK           ;If key was back/down arr
5395 34         02610 ADV      INC     (HL)              ;Increment day of week
5396 7E         02620          LD      A,(HL)            ;Get day of week
5397 FE07       02630          CP      7                 ;Is it greater than 6?
5399 3802       02640          JR      C,SETDAY          ;Go if valid day
539B 3600       02650          LD      (HL),0            ;Else reset to Sunday
539D 2B         02660 SETDAY   DEC     HL                ;Point to month storage
539E 7E         02680          LD      A,(HL)            ;Get month
539F CD1B54     02690          CALL    MAXDAY            ;Get number days in month
53A2 2B         02700          DEC     HL                ;Point to day storage
53A3 7E         02710          LD      A,(HL)            ;Get current day
53A4 B9         02720          CP      C                 ;Compare with maximum
53A5 34         02730          INC     (HL)              ;Advance day of month
53A6 D8         02740          RET     C                 ;Finished if valid day
53A7 3601       02750          LD      (HL),1            ;Else first of new month
53A9 23         02760          INC     HL                ;Point to month storage
53AA 7E         02770          LD      A,(HL)            ;Get current month
53AB FE0C       02780          CP      12D               ;See if it's December
53AD 34         02790          INC     (HL)              ;Advance month count
53AE D8         02800          RET     C                 ;Finished if valid month
53AF 3601       02810          LD      (HL),1            ;Else January of new year
53B1 2B         02820          DEC     HL                ;Point to year
53B2 2B         02825          DEC     HL                ;  storage
53B3 7E         02830          LD      A,(HL)            ;Get current year
53B4 FE63       02840          CP      99D               ;See if last of century
53B6 34         02850          INC     (HL)              ;Advance year count
53B7 D8         02860          RET     C                 ;Finished if valid year
53B8 3600       02870          LD      (HL),0            ;Else reset year count
```

```
53BA 116400     02880          LD      DE,100D           ;Add 100 years to century
53BD 2A2B52     02890 CHGCEN   LD      HL,(CNTURY)       ;Get current century
53C0 19         02900          ADD     HL,DE             ;Adjust century offset
53C1 222B52     02910          LD      (CNTURY),HL       ;  and re-save it
53C4 C9         02920          RET                       ;Finished for sure
53C5 35         02930 BACK     DEC     (HL)              ;Decrement day of week
53C6 F2CB53     02940          JP      P,SETDA2          ;Go if valid day
53C9 3606       02950          LD      (HL),6            ;Else reset to Saturday
53CB 2B         02960 SETDA2   DEC     HL                ;Point to month storage
53CC 7E         02980          LD      A,(HL)            ;Get month
53CD 2B         02990          DEC     HL                ;Point to day storage
53CE 35         03000          DEC     (HL)              ;Decrement day of month
53CF C0         03010          RET     NZ                ;Finished if valid day
53D0 3D         03020          DEC     A                 ;Else A=# of previous mth
53D1 CD1B54     03030          CALL    MAXDAY            ;Get # days previous mnth
53D4 71         03040          LD      (HL),C            ;Day=last day prev. month
53D5 23         03050          INC     HL                ;Point to month storage
53D6 35         03060          DEC     (HL)              ;Decrement month count
53D7 C0         03070          RET     NZ                ;Finished if valid month
53D8 360C       03080          LD      (HL),12D          ;Else December prev. year
53DA 2B         03090          DEC     HL                ;Point to year
53DB 2B         03095          DEC     HL                ;  storage
53DC 35         03100          DEC     (HL)              ;Decrement year count
53DD F0         03110          RET     P                 ;Finished if valid year
53DE 3663       03120          LD      (HL),99D          ;Else last yr prev cntury
53E0 119CFF     03130          LD      DE,-100D          ;Subtract 100 frm century
53E3 18D8       03140          JR      CHGCEN            ;Adjust century & finish
53E5 07         03150 PRTSTR   RLCA                      ;A=A*2 (2 byte pointers)
53E6 6F         03160          LD      L,A               ;L=LSB string table addr
53E7 2652       03170          LD      H,TABLE<-8        ;H=MSB string table addr
53E9 7E         03180          LD      A,(HL)            ;A=LSB actual string addr
53EA 23         03190          INC     HL                ;Point to MSB string addr
53EB 66         03200          LD      H,(HL)            ;H=MSB actual string addr
53EC 6F         03210          LD      L,A               ;HL=string location addr
53ED 7E         03220 DSPMSG   LD      A,(HL)            ;Get byte to display
53EE B7         03230          OR      A                 ;See if zero terminator
53EF C8         03240          RET     Z                 ;Finished if zero byte
53F0 F5         03250          PUSH    AF                ;Save sign flag status
53F1 E67F       03260          AND     7FH               ;Mask off bit 7
53F3 CD3300     03270          CALL    33H               ;Display it on video
53F6 F1         03280          POP     AF                ;Restore sign flag
53F7 F8         03290          RET     M                 ;Finished if bit 7 set
53F8 23         03300          INC     HL                ;Advance string pointer
53F9 18F2       03310          JR      DSPMSG            ;Go print next byte
53FB CD9A0A     03320 PRTNUM   CALL    0A9AH             ;Number in HL to ACCUM
53FE CDBD0F     03330          CALL    0FBDH             ;Convert # to string
5401 23         03340          INC     HL                ;Skip leading space char
5402 18E9       03350          JR      DSPMSG            ;Display converted number
5404 3E2C       03360 PRTCOM   LD      A,','             ;Comma character in A
5406 CD3300     03370          CALL    33H               ;Display it on video
5409 3E20       03380 PRTSPC   LD      A,' '             ;Space character in A
540B CD3300     03390          CALL    33H               ;Display it on video
540E C9         03400          RET                       ;Back to calling routine
540F 114440     03410 GETBFR   LD      DE,4044H          ;DE=Mod I date storage
5412 3A5400     03420          LD      A,(54H)           ;Check which model TRS-80
5415 3D         03430          DEC     A                 ;A will be 0 on Model I
5416 C8         03440          RET     Z                 ;Return if Model I
5417 111A42     03450          LD      DE,421AH          ;DE=Mod III date storage
541A C9         03460          RET                       ;Pointing to Mod III date
541B 0E1E       03470 MAXDAY   LD      C,30D             ;C=# days in some months
541D FE04       03480          CP      4                 ;Is month April?
541F C8         03490          RET     Z                 ;Return if April
5420 FE06       03500          CP      6                 ;Is month June?
5422 C8         03510          RET     Z                 ;Return if June
5423 FE09       03520          CP      9                 ;Is month September?
5425 C8         03530          RET     Z                 ;Return if September
5426 FE0B       03540          CP      11D               ;Is month November?
5428 C8         03550          RET     Z                 ;Return if November
5429 0C         03560          INC     C                 ;C=31 (# days most mnths)
542A FE02       03570          CP      2                 ;Is month February?
542C C0         03580          RET     NZ                ;Return if 31 day month
542D 0E1C       03590          LD      C,28D             ;C=# days in February
542F 3A2752     03600          LD      A,(DATSTR)        ;Get current year
5432 E603       03610          AND     3                 ;Year divisible by 4?
5434 C0         03620          RET     NZ                ;If not leap year
5435 0C         03630          INC     C                 ;C=29 (# days leap Feb.)
5436 C9         03640          RET                       ;Finished
5437 55         03650 FCB      DEFM    'USDATE/SYS'      ;File Control Block area
     53 44 41 54 45 2F 53 59 53
5441 03         03660          DEFB    3                 ;  with program filename
```

```
0015          03670          DEFS   21D        ;  (total 32 bytes)
0100          03680 FILBUF   DEFS   100H       ;File I/O buffer area
0500          03690 SYS0     DEFS   500H       ;SYS0/SYS temp. storage
52B3          03700          END    START
00000 TOTAL ERRORS

ADV     5395    APR     5273    ARROW   538B    AUG     52B3    BACK    53C5
CHGCEN  538D    CNTURY  522B    DATSTR  5227    DEC     52A1    DSPMSG  53ED
ENDYR   52B1    ERREXT  5368    EXIT    5361    FCB     5437    FEB     5266
FILBUF  5457    FNDDAT  533B    FRI     5251    GETBFR  540F    GETKEY  52EC
JAN     525F    JUL     527F    JUN     527B    MAR     526E    MARKER  5226
MAXDAY  541B    MAY     5278    MON     5233    MSG     52B0    NOTCR   5373
NOV     5299    OCT     5292    PROMPT  52A9    PRTCOM  5404    PRTNUM  53FB
PRTSPC  5409    PRTSTR  53E5    SAT     5257    SEP     5289    SETDA2  53CB
SETDAY  539D    START   52B3    SUN     522D    SYS0    5557    TABLE   5200
THU     5249    TUE     5239    UNSHFT  538A    UPDATE  5313    WED     5240
```

DEFAULT CONFIGURATIONS FOR TRSDOS 6.2

by Dave Bower

TRSDOS 6.2 comes configured for two drives (DEVICE command). If you add one or two more drives you must enable the drives (SYSTEM command) and then create a configuration file (SYSGEN command). This takes up disk space and slows the booting of the system. The following patches will patch BOOT/SYS to default to drives :1, :2 and/or :3 enabled (or disabled). There are also patches to allow drives :1, :2 and/or :3 to default to SIDES=2 upon power-up.

I have provided the patches and an explanation. If the patches work and you don't give a hoot why, then read no further. If for some reason the patches don't work on your TRSDOS 6.2, or you want to know what is going on, then the rest of this article will explain how and why.

Of course the accuracy of this info is not guaranteed (hey, how can I prevent typos?!?) and nobody is perfect (especially the person applying these patches to your diskette), so try them on a back-up diskette first.

```
Patch #1
.
. This patch causes TRSDOS 6.2 to recognize :2 upon
. powerup without a configuration file.
.
PATCH BOOT/SYS.LSIDOS (D02,84=C3:F02,84=C9)
.

Patch #2
.
. This patch causes TRSDOS 6.2 to recognize :3 upon
. powerup without a configuration file.
.
PATCH BOOT/SYS.LSIDOS (D02,8E=C3:F02,8E=C9)
.

Patch #3
.
. This patch causes TRSDOS 6.2 to recognize :1 as
. SIDES=2 without a configuration file.
.
PATCH BOOT/SYS.LSIDOS (D02,7E=62:F02,7E=42)
.

Patch #4
.
. This patch causes TRSDOS 6.2 to recognize :2 as
. SIDES=2 without a configuration file.
.
PATCH BOOT/SYS.LSIDOS (D02,88=64:F02,88=44)
.

Patch #5
.
. This patch causes TRSDOS 6.2 to recognize :3 as
. SIDES=2 without a configuration file.
.
PATCH BOOT/SYS.LSIDOS (D02,92=68:F02,92=48)
.
```

What you have just done is modify the Drive Code Table (DCT). This table holds important information the system needs to interface with the drives.

There are eight ten-byte entries (one for each logical drive, 0-7). We are only concerned with the first four (floppy) drives, 0 thru 3.

Each ten-byte entry contains the following information about its respective drive position.

The first byte will be either a C3 or a C9. If it is a C3 the next two bytes are the address of the disk I/O routines. If it is a C9 (C9 is a RET, return instruction) the drive is disabled. This is the byte to change to disable or enable the drives. A C3 enables the drive, a C9 disables it.

The fourth byte (of each 10-byte entry) contains flags for the drive specifications for each drive. Bits 0 & 1 = step rate, bit 2 = time delay (1 or .5 sec.), bit 3 = hard or floppy, bit 4 = side information for the FDC, bit 5 = 8" or 1/4, bit 6 = DDEN or SDEN and bit 7 = software write protection status.

The fifth byte contains more drive information. Bits 0 thru 3 contain the physical drive address, bit 4 = flag for alien disk controller (talk about foresight, we don't even know if there is intelligent life in space).

BIT 5 of this byte contains the flag for double sided diskettes. A "1" indicates that the diskette in this drive is double sided, a "0" indicates that the diskette is single sided. This is the bit to set (or reset) to cause the system to default to your sides configuration.

Bit 6 = flag for double density capabilities. Bit 7 = special flag for disk accesses with TRSDOS 6.2

Byte 6 contains FDC information.

Byte 7 contains the highest cylinder number on the diskette.

Bytes 8 & 9 contain allocation information.

Byte 10 contains contains the directory location for this drive position.

This information is available, in greater detail, in the Model 4 Technical Reference Manual. What it doesn't tell, is where the darn DCT table is!

The DCT is located in the file BOOT/SYS.LSIDOS. On my diskette it is located on sector 2. It is easy to identify. Each ten-byte group starts with a C3 (the first two will on the distribution diskette) or a C9. The seventh byte will always be a hex 27. This will always be there because it tells the system that this drive position is a 40 track drive. I found it the first time by having Super Utility search out all the hex 27's on a minimum system diskette and then scoping out each one.

Use and enjoy!

THE BULLETIN BOARD OF MODEM USERS' GROUP

This is a BBS system for MUG, the Modem Users' Group. Membership is $20 per year and includes 6 issues of MODEM-LINES, their newsletter/ magazine. You can request membership information while on-line, or you can mail a check to the this address and the latest issue of MODEM-LINES and a membership application will be sent to you: MUG, Box 227, 132 Gazza Blvd., Farmingdale, New York 11735

The BBS number is (516) 742-0039 and operates at 300/1200/2400 bps., 24 hours per day. You do NOT need to be a member to access the BBS (at this writing). The purpose of the BBS is <u>information</u> - not downloading programs. There are some programs available for downloading, but messages and text information are the <u>main</u> purposes of the board. Interests of the board include telecommunications, modem use, tips, hints, help, and information about other systems. Users are requested to enter messages concerning other BBS systems, RCP/Ms, CompuServe, Delphi or The Source, packet news, or any tip on modem use you feel will help other modem users.

As far as I could tell, this is not a "pirate" or "cracker" board, but rather one that serves the legitimate interests of modem users. However, for some reason the message base is very underused, averaging about one message per day! So, if you call, why not leave a message concerning some aspect of telecommunications and help keep this board alive?

PURCHASE ORDER FORM GENERATOR

by John C. Adams, Jr.

Here are two versions of a user-oriented purchase order form generator which can be easily customized to meet anyone's ordering requirements. The necessary modifications by line number are clearly identified near the top of the program listing, and the printer control codes are commented throughout throughout the listing in case they need to be changed from the current Epson/Gemini values. The only differences between the Model I/III BASIC version and the Model 4, BASIC version involve use of the 80 column screen and position formatting with reverse video in the Model 4 version. Both versions are totally interactive with the user via screen prompts for necessary inputs; certain default information is built-in through the above-mentioned user modifications.

MODEL I/III BASIC VERSION

```
10 REM   *****************************************************
20 REM   *  PURCHASE ORDER FORM GENERATOR                    *
30 REM   *  FOR USE WITH TRS-80 MODEL I OR III DISK BASIC   *
40 REM   *****************************************************
50 REM   *  ORIGINAL VERSION BY R. ATHANASIOU               *
60 REM   *  MODIFIED BY J.C. ADAMS, JR. AND S.C. WILLIAMS  *
70 REM   *****************************************************
80 REM   *  PRINTER CONTROL CODES ARE FOR EPSON/GEMINI       *
90 REM   *  AND MAY REQUIRE MODIFICATION FOR OTHER PRINTERS  *
100 REM  *****************************************************
110 REM  *  USER MUST MODIFY THE FOLLOWING LINES:           *
120 REM  *       340     MASTERCARD NUMBER/EXPIRATION DATE   *
130 REM  *       350     VISA CARD NUMBER/EXPIRATION DATE    *
140 REM  *       360     AMERICAN EXPRESS CARD NUMBER/       *
150 REM  *                 EXPIRATION DATE                  *
160 REM  *       390     YOUR NAME                          *
170 REM  *       400     YOUR STREET ADDRESS                 *
180 REM  *       410     YOUR CITY, STATE, ZIP CODE         *
190 REM  *       420     YOUR TELEPHONE NUMBER               *
200 REM  *       740     YOUR AUTHORIZING SIGNATURE          *
210 REM  *       970     YOUR HARDWARE DESCRIPTION           *
220 REM  *       980     YOUR SYSTEM SOFTWARE DESCRIPTION    *
230 REM  *****************************************************
240 CLS :IF PEEK(14312) <> 61 THEN PRINT @214,"PRINTER OFF-LINE":
PRINT @335,"READY PRINTER AND PRESS <ENTER>";: LINE INPUT Z$:
CLS
250 CLEAR 3000: LPRINT CHR$(27); CHR$(64); : REM INITIALIZE
PRINTER
260 U$="$$###.##": REM DOLLAR AND CENTS FORMAT
270 REM FN CN$ CENTERS STRING BY PADDING LEFT BLANKS
280 DEF FN CN$(A$,A%)= STRING$(A%/2- LEN(A$)/2-.5," ")+A$
290 PRINT FN CN$("PURCHASE ORDER FORM GENERATOR",64):PRINT
300 PRINT :INPUT "IS THIS A CREDIT CARD (C) OR PERSONAL CHECK
($) PURCHASE ";M$
310 ON INSTR("C$",M$) GOTO 320  ,370  : GOTO 300
320 PRINT :INPUT "MASTERCARD (M) OR VISA (V) OR AMERICAN
EXPRESS (A)";K$
330 ON INSTR("MVA",K$) GOTO 340  ,350  ,360  : GOTO 320
340 C$(1)="1234 5678 8765 4321":C$(2)="12/34":C$(3)=" < MASTERCARD
>": GOTO 390
350 C$(1)=" **** *** *** *** ":C$(2)="**/**":C$(3)=" < VISA CARD
>": GOTO 390
360 C$(1)=" **** ****** ***** ":C$(2)="**/**":C$(3)=" < AMERICAN
EXPRESS CARD >":  GOTO 390
370 PRINT : INPUT "PERSONAL CHECK NUMBER "; CN$
380 REM INSERT YOUR ADDRESS HERE.. YOU ARE PROMPTED FOR NAME
390 PRINT : LINE INPUT "NAME OF PURCHASER: ";A$(1) : IF
LEN(A$(1))=0 THEN A$(1) = "JOHN Q. PUBLIC, ESQ."
400 A$(2)="9876 MAIN STREET"
410 A$(3)="ANYTOWN, US  98765"
420 PN$="(987) 654 - 3210"
430 PRINT :LINE INPUT "TODAY'S DATE (MONTH DAY YEAR): ";D$
440 PRINT :PRINT :LINE INPUT "TO: ";B$(1)
450 LINE INPUT "ADDRESS: ";B$(2)
460 LINE INPUT "CITY, STATE & ZIP (use comma): ";B$(3)
470 PRINT : F$="N" : INPUT "IS THIS AN ORDER FOR PERSONAL
COMPUTER SOFTWARE (Y/N) ";F$
480 PRINT
490 INPUT "TOTAL NUMBER OF ITEMS IN THIS ORDER ";N
500 IF N > 10 THEN PRINT "***  TOTAL NUMBER OF ITEMS IN THIS
ORDER MUST BE <= 10  ***": GOTO 480
510 PRINT
520 REM NOW ENTER THE QUANTITY, DESCRIPTION AND UNIT PRICE
530 FOR J=1 TO N
540 DL$="N"
550 CLS :PRINT "QUANTITY of item ";J;:INPUT Q(J)
560 PRINT : PRINT "DESCRIPTION of item ";J;" ?";
570 CP=PEEK(16416)+256*PEEK(16417) : POKE CP+16,2 : PRINT
580 LINE INPUT D$(J)
590 IF LEN(D$(J)) > 42 THEN PRINT "***  LINE TOO LONG - REENTER
***" : GOTO 560
600 PRINT : DL$="N" : INPUT "ANOTHER DESCRIPTION LINE NEEDED
(Y/N) ";DL$
610 IF INSTR("Yy",DL$) <> 0 THEN LINE INPUT DD$(J)
620 IF LEN(DD$(J)) > 42 THEN PRINT "***  LINE TOO LONG - REENTER
***" : GOTO 600
630 PRINT : INPUT "UNIT PRICE (OMIT '$') ";P(J)
640 NEXT J
650 CLS :INPUT "SALES TAX PERCENTAGE (e.g. 0.075) IF APPLICABLE
(ELSE 0) ";Z
660 NP = 0 : FOR J=1 TO N : NP = NP + Q(J)*P(J) : NEXT J
670 TX = NP*Z : TX = INT(TX*100 + .5)/100
680 FP = NP + TX
690 PRINT :PRINT "TOTAL COST OF THE ABOVE ITEMS IS ";: PRINT
USING U$;FP
700 PRINT :INPUT "ENTER THE SHIPPING AND HANDLING CHARGE (OMIT
'$') ";S
710 PRINT : SI$ = "N" : INPUT "ANY ORDER-SPECIFIC INSTRUCTIONS
(Y/N) "; SI$
720 IF INSTR("Yy",SI$) <> 0 LINE INPUT OS$
730 IF LEN(OS$) > 58 THEN PRINT "***  LINE TOO LONG - REENTER
***" : GOTO 710
740 PRINT : LINE INPUT "AUTHORIZING SIGNATURE: ";A$(6) : IF
LEN(A$(6))=0 THEN A$(6) = "John Q. Public, Esq."
750 IF LEN(A$(6)) > 32 THEN PRINT "***  NAME TOO LONG - REENTER
***" : GOTO 740
760 FOR CC=1 TO 2 :CLS : PRINT @ 404, "PRINTING PURCHASE ORDER
";
770 LPRINT  CHR$(27); CHR$(69); : REM EMPHASIZED PRINT
780  LPRINT CHR$(27); CHR$(71); :  REM DOUBLE-STRIKE PRINT
790 LPRINT FN CN$("P U R C H A S E    O R D E R", 80)
800 LPRINT CHR$(27); CHR$(72); : REM CANCEL DOUBLE-STRIKE PRINT
810 LPRINT : LPRINT : REM PRINTS YOUR HEADING
820 LPRINT FN CN$(A$(1),80)
830 LPRINT FN CN$(A$(2),80)
840 LPRINT FN CN$(A$(3),80)
850 LPRINT FN CN$(PN$,80)
860 LPRINT
870 LPRINT FN CN$(D$,80): REM PRINTS DATE
880 LPRINT :LPRINT :LPRINT
890 LPRINT "TO: ";: REM SUPPLIERS NAME AND ADDRESS
900 LPRINT TAB(5);B$(1)
910 LPRINT TAB(5);B$(2)
920 LPRINT TAB(5);B$(3)
930 LPRINT : LPRINT : LPRINT
940 LPRINT "PLEASE SHIP THE FOLLOWING ITEMS:"
950 REM IF SOFTWARE ITEMS, SYSTEM IS SPECIFIED
960 IF INSTR("Yy",F$) <> 0 THEN GOTO 970  ELSE GOTO 990
970 LPRINT :LPRINT "FOR USE WITH TRS-80 MODEL 4, 128K RAM,
RS-232C, DUAL FLOPPY DISK SYSTEM"
980 LPRINT "RUNNING TRSDOS 6.1/6.2, LDOS 5.1.4, TRSDOS 1.3, CP/M
2.2, AND CP/M PLUS"
990 LPRINT
1000 LPRINT "QTY ";
1010 LPRINT TAB(10);"DESCRIPTION";
1020 LPRINT TAB(50);"UNIT PRICE";
1030 LPRINT TAB(67);"AMOUNT"
1040 LPRINT  STRING$(79,"="):LPRINT
1050 T = 0 : REM INITIALIZE RUNNING TOTAL
1060 FOR J=1 TO N
1070 LPRINT Q(J);: REM QUANTITY
1080 LPRINT TAB(5);D$(J);: REM DESCRIPTION
1090 LPRINT TAB(50);
1100 LPRINT USING U$;P(J);: REM UNIT PRICE
1110 P=Q(J)*P(J): REM UNIT PRICE * QUANTITY
1120 LPRINT TAB(65);
1130 LPRINT USING U$;P: REM ITEM TOTAL
1140 T=T+P: REM RUNNING TOTAL
1150 IF  LEN(DD$(J)) > 0 THEN LPRINT TAB(5);DD$(J)
1160 LPRINT : REM SKIP LINE BEFORE NEXT ITEM
1170 NEXT J: REM GET NEXT ITEM
1180 LPRINT TAB(65);"----------"
```

```
1190 LPRINT TAB(30);"TOTAL FOR MERCHANDISE";
1200 LPRINT TAB(65)
1210 LPRINT USING U$;T
1220 T1=T*Z:T1= INT(T1*100+.5)/100: REM COMPUTE TAX
1230 LPRINT TAB(30)"SALES TAX";
1240 LPRINT TAB(65);
1250 LPRINT USING U$;T1
1260 LPRINT TAB(30);"SHIPPING AND HANDLING  ";
1270 LPRINT TAB(65)
1280 LPRINT USING U$;S
1290 LPRINT TAB(65);"----------"
1300 LPRINT TAB(58);"TOTAL";
1310 LPRINT TAB(65)
1320 LPRINT USING U$;T+S+T1: REM TOTAL + SHIPPING + TAX
1330 LPRINT
1340 IF M$="C" THEN 1350 ELSE 1380
1350 LPRINT :LPRINT "PLEASE CHARGE MY ";C$(3)
1360 LPRINT "CARD NUMBER: ";C$(1);"    ";"EXPIRATION DATE: ";C$(2)
1370 GOTO 1400
1380 LPRINT "ENCLOSED PLEASE FIND PERSONAL CHECK NUMBER "; CN$;
" FOR ";
1390 LPRINT USING U$;S+T+T1
1400 IF LEN(OS$) > 0 THEN LPRINT OS$
1410 LPRINT :LPRINT :LPRINT :LPRINT
TAB(5);STRING$(32,CHR$(95));"AUTHORIZING SIGNATURE"
1420 LPRINT  CHR$(27); CHR$(52);: REM ITALIC PRINT
1430 LPRINT TAB(5);FN CN$(A$(6),37)
1440 LPRINT  CHR$(27); CHR$(53);: REM CANCEL ITALIC PRINT
1450 LPRINT  CHR$(12);: REM FORM FEED TO NEXT PAGE
1460 NEXT CC
1470 CLS : O$="N" : INPUT "DO YOU HAVE ANOTHER ORDER (Y/N)";O$
1480 IF INSTR("Yy",O$) <> 0 THEN GOTO 240  ELSE PRINT
CHR$(27);CHR$(64); : CLS : END
```

MODEL 4 BASIC VERSION

```
10 REM
************************************************************
20 REM   *  PURCHASE ORDER FORM GENERATOR
*
30 REM   *  FOR USE WITH TRS-80 MODEL 4 DISK BASIC 1.00 OR 1.01
*
40 REM
************************************************************
50 REM   *  ORIGINAL TRS-80 MODEL III VERSION BY R. ATHANASIOU
*
60 REM   *  MODIFIED BY J.C. ADAMS, JR. AND S.C. WILLIAMS        *
70 REM
************************************************************
80 REM   *  PRINTER CONTROL CODES ARE FOR EPSON/GEMINI
*
90 REM   *  AND MAY REQUIRE MODIFICATION FOR OTHER PRINTERS
*
100 REM
************************************************************
110 REM  *  USER MUST MODIFY THE FOLLOWING LINES:
*
120 REM  *      340     MASTERCARD NUMBER/EXPIRATION DATE
*
130 REM  *      350     VISA CARD NUMBER/EXPIRATION DATE
*
140 REM  *      360     AMERICAN EXPRESS CARD NUMBER/
*
150 REM  *                EXPIRATION DATE
*
160 REM  *      390     YOUR NAME
*
170 REM  *      400     YOUR STREET ADDRESS
*
180 REM  *      410     YOUR CITY, STATE, ZIP CODE
*
190 REM  *      420     YOUR TELEPHONE NUMBER
*
200 REM  *      710     YOUR AUTHORIZING SIGNATURE
*
210 REM  *      930     YOUR HARDWARE DESCRIPTION
*
220 REM  *      940     YOUR SYSTEM SOFTWARE DESCRIPTION
*
230 REM
************************************************************
240 CLS :IF INP(248) <> 61 THEN PRINT @(3,30),"PRINTER OFF-LINE":
PRINT @(5,23),"READY PRINTER AND PRESS <ENTER>";: LINE INPUT Z$:
CLS
250 CLEAR : LPRINT CHR$(27); CHR$(64); : REM INITIALIZE PRINTER
260 U$="$$###.##": REM DOLLAR AND CENTS FORMAT
270 REM FN CN$ CENTERS STRING BY PADDING LEFT BLANKS
280 DEF FN CN$(A$,A%)= STRING$(A%/2- LEN(A$)/2-.5," ")+A$
290 PRINT FN CN$("PURCHASE ORDER FORM GENERATOR",80):PRINT
300 PRINT :INPUT "IS THIS A CREDIT CARD (C) OR PERSONAL CHECK
($) PURCHASE ";M$
310 IF INSTR(1,M$,"C") = 1 THEN GOTO 320 ELSE IF INSTR(1,M$,"$") =
1 THEN GOTO 370 ELSE GOTO 300
320 PRINT :INPUT "MASTERCARD (M) OR VISA (V) OR AMERICAN
EXPRESS (A)";K$
330 IF INSTR(1,K$,"M") = 1 THEN GOTO 340 ELSE IF INSTR(1,K$,"V") =
1 THEN GOTO 350 ELSE IF INSTR(1,K$,"A") = 1 THEN GOTO 360 ELSE
GOTO 320
340 C$(1)="1234 5678 8765 4321":C$(2)="12/34":C$(3)=" < MASTERCARD
>": GOTO 390
350 C$(1)=" **** *** *** *** ":C$(2)="**/**":C$(3)=" < VISA CARD
>": GOTO 390
360 C$(1)=" **** ****** ***** ":C$(2)="**/**":C$(3)=" < AMERICAN
EXPRESS CARD >":  GOTO 390
370 PRINT : INPUT "PERSONAL CHECK NUMBER "; CN$
380 REM INSERT YOUR ADDRESS HERE.. YOU ARE PROMPTED FOR NAME
390 PRINT : LINE INPUT "NAME OF PURCHASER: ";A$(1) : IF
LEN(A$(1))=0 THEN A$(1) = "JOHN Q. PUBLIC, ESQ."
400 A$(2)="9876 MAIN STREET"
410 A$(3)="ANYTOWN, US  98765"
420 PN$="(987) 654 - 3210"
430 PRINT :LINE INPUT "TODAY'S DATE (MONTH DAY YEAR): ";D$
440 PRINT :PRINT :LINE INPUT "TO: ";B$(1)
450 LINE INPUT "ADDRESS: ";B$(2)
460 LINE INPUT "CITY, STATE & ZIP (use comma): ";B$(3)
470 PRINT :INPUT "IS THIS AN ORDER FOR PERSONAL COMPUTER
SOFTWARE (Y/N) ";F$
480 PRINT
490 INPUT "TOTAL NUMBER OF ITEMS IN THIS ORDER ";N
500 IF N > 10 THEN PRINT "***  TOTAL NUMBER OF ITEMS IN THIS
ORDER MUST BE <= 10  ***": GOTO 490
510 PRINT
520 REM NOW ENTER THE QUANTITY, DESCRIPTION AND UNIT PRICE
530 FOR J=1 TO N
540 DL$="N"
550 CLS :PRINT "QUANTITY of item ";J;:INPUT Q(J)
560 PRINT "DESCRIPTION of item ";J;" ?"
570 RN=ROW(0) : PRINT @ (RN,0), STRING$(43,CHR$(143)); : PRINT @
(RN,0), CHR$(128);
580 LINE INPUT D$(J)
590 INPUT "ANOTHER DESCRIPTION LINE NEEDED (Y/N) ";DL$
600 IF INSTR(1,DL$,"Y") = 1 OR INSTR(1,DL$,"y") = 1 THEN RN=ROW(0)
: PRINT @ (RN,0), STRING$(43,CHR$(143)); : PRINT @ (RN,0), CHR$(128);
: LINE INPUT DD$(J)
610 INPUT "UNIT PRICE (OMIT '$') ";P(J)
620 NEXT J
630 CLS :INPUT "SALES TAX PERCENTAGE (e.g. 0.075) IF APPLICABLE
(ELSE 0) ";Z
640 NP = 0 : FOR J=1 TO N : NP = NP + Q(J)*P(J) : NEXT J
650 TX = NP*Z : TX = INT(TX*100 + .5)/100
660 FP = NP + TX
670 PRINT :PRINT "TOTAL COST OF THE ABOVE ITEMS IS ";: PRINT
USING U$;FP
680 PRINT :INPUT "ENTER THE SHIPPING AND HANDLING CHARGE (OMIT
'$') ";S
690 PRINT : INPUT "ANY ORDER-SPECIFIC INSTRUCTIONS (Y/N) "; SI$
700 IF INSTR(1,SI$,"Y") = 1 OR INSTR(1,SI$,"y") = 1 THEN RN=ROW(0) :
PRINT @ (RN,0), STRING$(58,CHR$(143)); : PRINT @ (RN,0), CHR$(128); :
LINE INPUT OS$
710 PRINT :PRINT "AUTHORIZING SIGNATURE: "; : RN=ROW(0) :
CP=POS(0) : PRINT @ (RN,CP), STRING$(33,CHR$(143)); : PRINT @
(RN,CP), CHR$(128); : LINE INPUT A$(6) :IF LEN(A$(6))=0 THEN A$(6) =
"John Q. Public, Esq."
720 FOR CC=1 TO 2 :CLS : PRINT @ (10,28), "PRINTING PURCHASE
ORDER ";
730 LPRINT  CHR$(27); CHR$(69); : REM EMPHASIZED PRINT
740  LPRINT CHR$(27); CHR$(71); :  REM DOUBLE-STRIKE PRINT
750 LPRINT FN CN$("P U R C H A S E    O R D E R", 80)
760 LPRINT CHR$(27); CHR$(72); : REM CANCEL DOUBLE-STRIKE PRINT
770 LPRINT : LPRINT : REM PRINTS YOUR HEADING
780 LPRINT FN CN$(A$(1),80)
```

```
790 LPRINT FN CN$(A$(2),80)
800 LPRINT FN CN$(A$(3),80)
810 LPRINT FN CN$(PN$,80)
820 LPRINT
830 LPRINT FN CN$(D$,80): REM PRINTS DATE
840 LPRINT :LPRINT :LPRINT
850 LPRINT "TO: ";: REM SUPPLIERS NAME AND ADDRESS
860 LPRINT TAB(6);B$(1)
870 LPRINT TAB(6);B$(2)
880 LPRINT TAB(6);B$(3)
890 LPRINT : LPRINT : LPRINT
900 LPRINT "PLEASE SHIP THE FOLLOWING ITEMS:"
910 REM IF SOFTWARE ITEMS, SYSTEM IS SPECIFIED
920 IF INSTR(1,F$,"Y") = 1 OR INSTR(1,F$,"y") = 1 THEN GOTO 930
ELSE GOTO 950
930 LPRINT :LPRINT "FOR USE WITH TRS-80 MODEL 4, 128K RAM,
RS-232C, DUAL FLOPPY DISK SYSTEM"
940 LPRINT "RUNNING TRSDOS 6.1/6.2, LDOS 5.1.4, TRSDOS 1.3, CP/M
2.2, AND CP/M PLUS"
950 LPRINT
960 LPRINT "QTY ";
970 LPRINT TAB(11);"DESCRIPTION";
980 LPRINT TAB(51);"UNIT PRICE";
990 LPRINT TAB(68);"AMOUNT"
1000 LPRINT  STRING$(79,"="):LPRINT
1010 T = 0 : REM INITIALIZE RUNNING TOTAL
1020 FOR J=1 TO N
1030 LPRINT Q(J);: REM QUANTITY
1040 LPRINT TAB(6);D$(J);: REM DESCRIPTION
1050 LPRINT TAB(51);
1060 LPRINT USING U$;P(J);: REM UNIT PRICE
1070 P=Q(J)*P(J): REM UNIT PRICE * QUANTITY
1080 LPRINT TAB(66);
1090 LPRINT USING U$;P: REM ITEM TOTAL
1100 T=T+P: REM RUNNING TOTAL
1110 IF  LEN(DD$(J)) > 0 THEN LPRINT TAB(6);DD$(J)
1120 LPRINT : REM SKIP LINE BEFORE NEXT ITEM
1130 NEXT J: REM GET NEXT ITEM
1140 LPRINT TAB(66);"----------"
1150 LPRINT TAB(31);"TOTAL FOR MERCHANDISE";
1160 LPRINT TAB(66)
1170 LPRINT USING U$;T
1180 T1=T*Z:T1= INT(T1*100+.5)/100: REM COMPUTE TAX
1190 LPRINT TAB(31)"SALES TAX";
1200 LPRINT TAB(66);
1210 LPRINT USING U$;T1
1220 LPRINT TAB(31);"SHIPPING AND HANDLING  ";
1230 LPRINT TAB(66)
1240 LPRINT USING U$;S
1250 LPRINT TAB(66);"----------"
1260 LPRINT TAB(59);"TOTAL";
1270 LPRINT TAB(66)
1280 LPRINT USING U$;T+S+T1: REM TOTAL + SHIPPING + TAX
1290 LPRINT
1300 IF M$="C" THEN 1310 ELSE 1340
1310 LPRINT :LPRINT "PLEASE CHARGE MY ";C$(3)
1320 LPRINT "CARD NUMBER: ";C$(1);"     ";"EXPIRATION DATE: ";C$(2)
1330 GOTO 1360
1340 LPRINT "ENCLOSED PLEASE FIND PERSONAL CHECK NUMBER "; CN$;
" FOR ";
1350 LPRINT USING U$;S+T+T1
1360 IF LEN(OS$) > 0 THEN LPRINT OS$
1370 LPRINT :LPRINT :LPRINT :LPRINT
TAB(6);STRING$(32,CHR$(95));"AUTHORIZING SIGNATURE"
1380 LPRINT  CHR$(27); CHR$(52);: REM ITALIC PRINT
1390 LPRINT TAB(6);FN CN$(A$(6),37)
1400 LPRINT  CHR$(27); CHR$(53);: REM CANCEL ITALIC PRINT
1410 LPRINT  CHR$(12);: REM FORM FEED TO NEXT PAGE
1420 NEXT CC
1430 CLS :INPUT "DO YOU HAVE ANOTHER ORDER (Y/N)";O$
1440 IF INSTR(1,O$,"Y") = 1 OR INSTR(1,O$,"y") = 1 THEN GOTO 240
ELSE LPRINT  CHR$(27); CHR$(64);:CLS:END
```

GETTING THE MOST OUT OF YOUR HARD DRIVE

by Martin Pollard

[Reprinted from Micro Notes, a publication of the Dearborn TRS-80 Users Group (P.O. Box 1942, Dearborn, Michigan 48121):]

As many of you with hard drives (running LDOS 5.1 or TRSDOS 6) know, DOS formats the hard drive very inefficiently. Taking a Tandy 5-megabyte hard drive for example, each platter is formatted with 32 sectors per cylinder, 16 sectors per granule. This means that each cylinder has only two granules. Simple math will tell you that 256 bytes per sector times 16 sectors equals 4096 bytes, or 4K per granule. If you have a small file that takes up only one sector (.25K), that means that 3.75K of usable disk space is wasted. Inexcusable! But here's a method of formatting your hard drive that will give you 4 sectors per granule and 8 granules per track, a more evenly distributed allocation system (only 1K per granule). Since there are too many hard drives out there to cover, I will concentrate on only the Tandy 5-megabyte hard drive, Tandy's drivers, and LDOS 5.1 and TRSDOS 6.2. Users of other hard drives, drivers, and DOSes, should be able to use the information below to apply it to their system.

1) Set up your hard drive with logical drives 4, 1, 2 and 3 assigned to heads 0, 1, 2 and 3, respectively, using the TRSHDx/DCT driver ("x" is either 3 or 5 for LDOS, 4 or 6 for TRSDOS 6).

2) You must now modify the DCT for the new granule allocation settings. Here is how the byte is defined:

```
Bits 7-5 = Number of granules per cylinder
Bits 4-0 = Number of sectors per granule
```

In this case, the byte value is X'E3' (11100011B), which translates to 8 granules, 4 sectors per granule. (Each value is offset from zero). The addresses to modify, in table form, are:

Drive	LDOS 5.1	TRSDOS 6.2
1	X'4712'	X'0482'
2	X'471C'	X'048C'
3	X'4726'	X'0496'
4	X'4730'	X'04A0'

Note that the values are known valid ONLY for LDOS 5.1 and TRSDOS 6.2. To modify the address, use the MEMORY command in the form:

```
MEMORY (ADD=X'nnnn',BYTE=X'E3')
```

3) Format each hard drive partition using the TRSFORMx/CMD utility (again, "x" is defined as above). TRSFORM will format the hard drive according to the specifications in the DCT, which is exactly what we want!

4) Copy the DOS onto drive 4 using the command:

```
BACKUP /SYS:0 :4 (S,I=N)
```

This copies the system files WITHOUT copying the CONFIG/SYS configuration file that might be present on your DOS disk.

5) If you are using LDOS 5.1, you must now create a CONFIG/SYS file. This is so that, when we finally transfer control to the hard drive, we can copy the final CONFIG/SYS file back to the boot disk so LDOS recognizes it. It is strange, I know, because you don't need this step under TRSDOS 6. Anyway, perform the following command:

```
SYSTEM (SYSGEN)
```

and continue on.

6) Transfer control to drive 4 by issuing the command:

```
SYSTEM (SYSTEM=4)
```

7) Copy any other files, utilities, etc. to the hard drive. After doing so, you must then save the current configuration, which will save the hard driver, any in-memory modules, etc., as well as the current DCTs. Under LDOS, you must issue the following commands:

```
SYSTEM (SYSGEN) COPY CONFIG/SYS.CCC:0 :4
```

It's easier under TRSDOS 6, just one command:

```
SYSGEN (DRIVE=4)
```

Your hard drive in now ready to go with your new format. I have been using my Tandy 5-megabyte hard drive with this format under TRSDOS 6.2.1 for several months now, and have not had one problem. If anyone tries this using LDOS 5.1, please let me know how error-free it is.

CHANGE LAZYWRITER 3.4a TO DISPLAY TRUE FREE GRANS ON 80 TRACK DOUBLE SIDED DISKS UNDER NEWDOS/80 VERSION 2 (MODEL I SD)

by Keith Stewart

P.O. Box 28-020, Kelburn, Wellington 5, New Zealand

With the DIR/CMD file accessed from EDIT with (clear/break) Lazywriter lets you use some DOS commands and compile a list of directories. It was this last use which started me on patching DIR/CMD to enable it to display correct free grans on 80 track double sided drives.

Upon execution, DIR/CMD first reads in the boot sector of the disk. It then gets the third byte of the boot sector, which tells the DOS where the directory track is, and it puts this byte in the high order byte of the File Control Block it has set up and then reads in the sector pointed to by this byte, i.e, the first sector of the directory. This contains the Granule Allocation Table of the disk. Then it reads the GAT byte by byte looking for FF or FE or FD or FC which indicate which sectors and how many sectors are allocated in the grans.

Starting with first byte in the buffer where the GAT sector is stored each byte is compared with FF. BC is used as a counter and is loaded with 50H. The value in BC is decremented for each byte examined. HL is loaded with 0000 and increments with 0 for FF and 1 for FE or FD and 2 for FC. If the GAT byte is not FF then BC value is put on stack and BC loaded with 08. RLCA eight times to check for FE (means second sector allocated) or FD (means first sector allocated) or FC (means first & second sectors free) This comparison continues until BC=0000. The value in HL then is computed and the resulting number is the number of free grans available on the disk.

Most TRSDOS compatible disks have 80 bytes (50H) allocated to the Gran Allocation Table, so that is why Lazywriter uses 50H as a counter. But NEWDOS/80 can extend the GAT if needed, and does so on 80 track double sided disks if we allocate 20 sectors per track. The counter has then to be changed to A0 or 160 from the default value of 50H.

Using the DFS function of SUPERZAP, find File Relative Sector 2 of DIR/CMD and MOD 68 (change byte 68) from 50 to A0. If you only have double sided drives then this is all you need to do to let LAZYWRITER display the correct free grans. This means, however, that Lazywriter will display false free grans on any single sided or 40 track double sided disks.

If you have a mixture of single sided and 80 track double sided drives then more than a zap is needed to make LAZYWRITER check for the type of drive and adjust the byte at 673DH as necessary. This patch uses the PDRIVE table which is stored at 4371H to determine how many tracks are allocated to each respective drive and if 80 tracks how many sectors per track are specified (thanks to SYDTRUG for this info.)

Assemble the source code as PATCH/CMD. You can set ORG at any address. This example uses FFACH. Using a utility which allows multiple origins when saving a file load PATCH/CMD and DIR/CMD (see note 2). Of course the utility must be out of the way of their loading areas. Change memory at 6689H-669BH from CD 36 44 to C3 AC FF (or whatever your ORG is). This links DIR/CMD to the PATCH. Then write out DIR/CMD with the following parameters:

```
Start  5262H  5285H  52CFH  64F0H  6E4BH  FFACH
End    5262H  5285H  52CFH  6E2CH  705DH  FFFFH
Entry  6509H
```

The patch loads in high memory starting at FFACH whenever DIR/CMD is called. To protect this memory set HIMEM before calling Lazywriter if your file is likely to be that large or set up a DO file to set HIMEM and then load L/CMD.

DIR/CMD reads the directory from drive 0 when you call it from Lazywriter. If you want it to default to another drive then change the byte at 7043H in DIR/CMD or using SUPERZAP File Relative Sector 11 MOD 74 change this byte to 01 or 02 or 03.

(Note 1) I had originally put the patch in an apparently empty memory area under LAZYWRITER so as to not have to worry about setting HIMEM. But when inserting tabs in EDIT the cursor inserted about three lines. That's why it is now in high memory. If you find another spare place then let me know.

(Note 2) A fine utility which loads and save multiple origins files, among other things, is available from Briggs Software, 14 Allan Berry Avenue, Napier, New Zealand. No I am not related to him.

[Editor's note: Although not written for the purpose, I suspect that this patch (perhaps with a slight modification) would also work with double density. If anyone gets this working under double density, please let us know what changes (if any) you had to make.]

```
               00100 ;A PATCH FOR LAZYWRITER 3.4 TO ENABLE IT TO GIVE CORRECT
               00110 ;FREE GRANS WHEN CALLING DIR/CMD WITH 80TRK DS DRIVE.
               00120 ;USE THIS PATCH ONLY IF YOU HAVE A MIX OF 80TRK DS
               00130 ;AND OTHER TYPES OF DRIVE I.E 35-40 SINGLE OR 40 DOUBLE
               00140 ;SIDED OR 80 SINGLE ELSE  USE MY ZAP FOR JUST 80TRK DS.
               00150 ;YOU MUST BE BE RUNNING NEWDOS 80 V2 AND SINGLE DENSITY
               00160 ;IT CAN ASSEMBLED ANYWHERE ABOVE THE TEXT. USE HIMEM TO
               00170 ;PROTECT IT.
               00180 ;ZAP FOR DIR/CMD WHEN YOU ONLY HAVE 80TRK DS DRIVES
               00190 ;DFS DIR/CMD RFS,2,MOD68 CHANGE 50 TO A0
               00200 ;
               00210 ;KEITH STEWART BOX 28-020 WELLINGTON NEW ZEALAND
               00220 ;
               00230 ;***TABLE*** REFERRED TO IN COMMENTS IS PDRIVE TABLE
               00240 ;STORED IN MEMORY.
FFAC           00250         ORG     0FFACH          ;ANYWHERE TO SUIT
FFAC D9        00260         EXX                     ;SAVE IMPORTANT REGISTERS
FFAD 3A4370    00270         LD      A,(7043H)       ;7TH BYTE OF FCB=DRIVE#
FFB0 FE03      00280         CP      03              ;IS IT DRIVE 3?
FFB2 2843      00290         JR      Z,TEST3         ;GO TO DRIVE 3 TEST IF Y
FFB4 FE02      00300         CP      02              ;IS IT DRIVE 2?
FFB6 2836      00310         JR      Z,TEST2         ;GO TO DRIVE 2 TEST IF Y
FFB8 FE01      00320         CP      01              ;IS IT DRIVE 1?
FFBA 2829      00330         JR      Z,TEST1         ;GO TO DRIVE 1 TEST IF Y
FFBC 3A7443    00340 TEST0   LD      A,(4374H)       ;# TRKS FROM TABLE DR 0
FFBF 217543    00350         LD      HL,4375H        ;# SECTORS HERE DRIVE 0
FFC2 FE50      00360 TEST    CP      50H             ;IS IT 80 TRK?
FFC4 280F      00370         JR      Z,TEST80        ;YES GO CHECK IF DSIDED
FFC6 3E50      00380 RET80   LD      A,0050H         ;MAKE SURE DEFAULT 50H
FFC8 323D67    00390         LD      (673DH),A       ;LOAD DEFAULT INTO PROG
FFCB 3B        00400 RETURN  DEC     SP              ;CORRECT STACK
FFCC 3B        00410         DEC     SP              ;CORRECT STACK
FFCD 218C66    00420         LD      HL,668CH        ;PC VALUE NEEDED
FFD0 E3        00430         EX      (SP),HL         ;PUT ON STACK
FFD1 D9        00440         EXX                     ;RESTORE REGISTERS
FFD2 C33644    00450         JP      4436H           ;CONTINUE DIR/CMD
FFD5 7E        00460 TEST80  LD      A,(HL)          ;# SECT/TRK IN HL ALREADY
FFD6 FE14      00470         CP      14H             ;IS IT 20 IE DOUBLE SIDED
FFD8 2803      00480         JR      Z,PUTIN         ;YES THEN CHANGE VALUE
FFDA C3C6FF    00490         JP      RET80           ;NO RETURN TO PROG.
FFDD 3EA0      00500 PUTIN   LD      A,00A0H         ;LOAD A WITH 160
FFDF 323D67    00510         LD      (673DH),A       ;PUT IN COUNTER ADDRESS
FFE2 C3CBFF    00520         JP      RETURN          ;RETURN TO PROGRAM
FFE5 3A7E43    00530 TEST1   LD      A,(437EH)       ;# TRKS FROM TABLE DR 1
FFE8 217F43    00540         LD      HL,437FH        ;# SECTORS HERE DRIVE 1
FFEB C3C2FF    00550         JP      TEST            ;GO & SEE IF 80 TRK
FFEE 3A8843    00560 TEST2   LD      A,(4388H)       ;# TRKS FROM TABLE DR 2
FFF1 218943    00570         LD      HL,4389H        ;# SECTORS HERE DRIVE 2
FFF4 C3C2FF    00580         JP      TEST            ;GO & SEE IF 80 TRK
FFF7 3A9343    00590 TEST3   LD      A,(4393H)       ;# TRKS FROM TABLE DR 3
FFFA 219243    00600         LD      HL,4392H        ;# SECTORS HERE DRIVE 3
FFFD C3C2FF    00610         JP      TEST            ;GO & SEE IF 80 TRK
0000           00620         END     0000
00000 TOTAL ERRORS

PUTIN   FFDD   RET80   FFC6   RETURN FFCB   TEST    FFC2   TEST0  FFBC
TEST1   FFE5   TEST2   FFEE   TEST3  FFF7   TEST80  FFD5
```

****

John Smith at (602) 279-1646 (Arizona) is seeking and willing to pay near list price for a copy of the VIVACE BASIC Compiler.

****

Mel Eisele, Route 4, Box 49-C, Portland, TN, 37148, has a 128K Model 4P he is willing to sell. Call Mel at (615) 325-9387.

HEX TO DECIMAL OR DECIMAL TO HEX 'U' ROUTINE FOR TASMON

by Nate Salsbury

[This is another of our assembly language program listings where the documentation appears as comments within the source code. For reasons that will be obvious as you study the source code, we are unable to print this in our normal "expanded" source code format (showing the hex bytes for each instruction). Here's the source code listing:]

```
00100 ;         ************************************
00110 ;         * HEX TO DECIMAL OR DECIMAL TO HEX *
00120 ;         *      'U' ROUTINE FOR TASMON      *
00130 ;         ************************************
00140 ;
00150 ;            NATE SALSBURY
00160 ;            610 MADAM MOORE'S LANE
00170 ;            NEW BERN, NC     28560
00180 ;
00190 ;++++++++++++++++++++++++++++++++++++++++++++++++++
00200 ;
00210 ;        This routine provides a 'U'ser function to TASMON
00220 ;        to convert 4 hex digits to decimal or 5 decimal
00230 ;        digits to their hex equivalent.
00240 ;
00250 ;        OPERATION OF MODIFIED TASMON (AFTER HITTING 'U')
00260 ;
00270 ;        TO CONVERT DECIMAL TO HEX, HIT 'D' FOLLOWED BY
00280 ;        5 DECIMAL DIGITS (0 - 9). NUMBERS LESS THAN
00290 ;        10,000 MUST HAVE LEADING ZEROS E.G.  00183
00300 ;
00310 ;        TO CONVERT HEX TO DECIMAL, HIT 'H' FOLLOWED BY
00320 ;        4 HEX CHARACTERS (0 - 9 OR A - F).
00330 ;
00340 ;        YOU ARE RETURNED TO THE TASMON PROMPT AFTER
00350 ;        EACH CONVERSION.
00360 ;
00370 ;++++++++++++++++++++++++++++++++++++++++++++++++++
00380 ;
00390 ;        There are a number of different versions of
00400 ;        TASMON which have been published.  In addition,
00410 ;        you may have relocated yours to other than the
00420 ;        distribution ORG.
00430 ;
00440 ;        The working part of this program is added beyond
00450 ;        the end of TASMON.  It uses several TASMON rou-
00460 ;        tines for keyboard entry.  For each EQUate, I
00470 ;        show addresses for four versions of TASMON plus
00480 ;        a routine for locating the proper place if you
00490 ;        have a different version.
00500 ;
00510 ;        This can be done by using the TASMON 'Find' rou-
00520 ;        tine, starting at the loading address for your
00530 ;        version.  I'll describe a byte sequence to 'Find'
00540 ;        with instructions on what to do then.
00550 ;
00560 ;        First, load your version and with the 'View' com-
00570 ;        mand, find the starting, ending and transfer add-
00580 ;        ress for your version.  Record them here.
00590 ;
00600 START   EQU     nnnnH           ;Your starting address
00610 FINISH  EQU     nnnnH           ;Your ending address
00620 TRANS   EQU     nnnnH           ;Your transfer address
00630 ;
00640 ;Since the TASMON addresses used are for the distribution
00650 ;versions, it will be necessary to compute an 'offset'.
00660 ;
00670 ;Remove the semicolon for the version you use.
00680 ;
00690 ;OFFSET EQU     START-6000H     ;Model III, 'D5' version
00700 ;OFFSET EQU     START-6000H     ;Model III, 'D7' version
00710 ;OFFSET EQU     START-0E000H    ;Model 4, Version 1.10
00720 ;OFFSET EQU     START-3000H     ;Model 4, Version 1.12
00730 ;
00740 ;This program uses the following TASMON routines
00750 ;Remove the initial semicolon for the EQU you need
00760 ;for your version of TASMON.
00770 ;
00780 ;++++++++++++++++++++++++++++++++++++++++++++++++++
00790 ;HEXINP EQU     6541H+OFFSET    ;MIII, 'D5'
00800 ;HEXINP EQU     6567H+OFFSET    ;MIII, 'D7'
00810 ;HEXINP EQU     0E56FH+OFFSET   ;M4, 1.10
00820 ;HEXINP EQU     3503H+OFFSET    ;M4, 1.12
00830 ;
00840 ;Or, use the 'Find' command from your START address
00850 ;and look for CB 27 67 CD.  Disassemble the code at
00860 ;the address shown and then back up (using the '-' key)
00870 ;to find the following sequence:
00880 ;
00890 CD nnnn 20 FB CD nnnn CD nnnn CB 27 CB 27 CB 27 CB 27 67
00900 ;
00910 ;The address you want is that of the first 'CD'.
00920 ;++++++++++++++++++++++++++++++++++++++++++++++++++
00930 ;
00940 ;KYBORD EQU     60E1H+OFFSET    ;MIII, 'D5'
00950 ;KYBORD EQU     60E6H+OFFSET    ;MIII, 'D7'
00960 ;KYBORD EQU     0E0F1H+OFFSET   ;M4, 1.10
00970 ;KYBORD EQU     315AH+OFFSET    ;M4, 1.12
00980 ;
00990 ;Use the 'Find' command from your START to locate:
01000 ;
01010 ;FE 2A CC.  Disassemble at that address and back
01020 ;up to find:   CD nnnn FE 2A CC.  The address you
01030 ;should use is the one for the CD instruction.
01040 ;++++++++++++++++++++++++++++++++++++++++++++++++++
01050 ;
01060 ;VALIDH EQU     6584H+OFFSET    ;MIII, 'D5'
01070 ;VALIDH EQU     65AAH+OFFSET    ;MIII, 'D7'
01080 ;VALIDH EQU     0E582H+OFFSET   ;M4, 1.10
01090 ;VALIDH EQU     3616H+OFFSET    ;M4, 1.12
01100 ;
01110 ;Or, use the 'Find' command to locate:
01120 ;
01130 ;FE 00 28 F9.  Disassemble at that address and back
01140 ;up to locate:  CD nnnn FE 00 28 F9.  The address of
01150 ;the CD is the one you want.
01160 ;++++++++++++++++++++++++++++++++++++++++++++++++++
01170 ;
01180 ;WR     EQU     7A1CH+OFFSET    ;MIII, 'D5'
01190 ;WR     EQU     7B12H+OFFSET    ;MIII, 'D7'
01200 ;WR     EQU     0F870H+OFFSET   ;M4, 1.10
01210 ;WR     EQU     4B8DH+OFFSET    ;M4, 1.12
01220 ;
01230 ;Use the 'Find' command to locate: E5 DD E5 D5.
01240 ;Disassemble at the address shown and back up one byte.
01250 ;You should find C5 E5 DD E5 D5 DD 21.  The address
01260 ;you want is that of the C5 instruction. Check that
01270 ;you have the correct group of these bytes by disas-
01280 ;sembling the earlier instructions to find:
01290 ;
01300 ;23 EB 47 70 23 1B 7A B3 20 F9 C3 nnnn C5 etc.
01310 ;
01320 ;++++++++++++++++++++++++++++++++++++++++++++++++
01330 ;
01340 ;WL     EQU     695BH+OFFSET    ;MIII, 'D5'
01350 ;WL     EQU     69C6H+OFFSET    ;MIII, 'D7'
01360 ;WL     EQU     0E9A2H+OFFSET   ;M4, 1.10
01370 ;WL     EQU     39F7H+OFFSET    ;M4, 1.12
01380 ;
01390 ;This address is harder to find because there are
01400 ;no 'address independent' bytes nearby to locate.
01410 ;One way is to determine the address for the WR routine
01420 ;above. Then, after noting the absolute value for WR,
01430 ;use your 'Find' command to locate:  CD (LSB) (MSB) C3
01440 ;where (LSB) and (MSB) are the usual Z-80 address entry
01450 ;for WR in reverse order.  The address located in this
01460 ;manner is the one you want for WL.
01470 ;
01480 ;++++++++++++++++++++++++++++++++++++++++++++++++++
01490 ;
01500 ;The last address needed is the location of the 'jump'
01510 ;address for the 'U' command.
01520 ;JUMP   EQU     60DCH+OFFSET    ;MIII, 'D5'
01530 ;JUMP   EQU     60E1H+OFFSET    ;MIII, 'D7'
01540 ;JUMP   EQU     0E0ECH+OFFSET   ;M4, 1.10
01550 ;JUMP   EQU     3155H+OFFSET    ;M4, 1.12
01560 ;
01570 ;Or, use the 'Find' routine to look for  FE 55 CA.
01580 ;The address you want is that of the byte FOLLOWING
01590 ;the CA byte.
```

```
01600 ;
01610 ;++++++++++++++++++++++++++++++++++++++++++++++++++
01620 ;
01630 ;       For reference, here is a list of the functions
01640 ;       of the TASMON routines used.
01650 ;
01660 ;HEXINP         TASMON 4-CHAR HEX INPUT
01670 ;KYBORD         GET KBRD CHARACTER AND
01680                  ;PROCESS 'BREAK' KEY
01690 ;VALIDH         GET HEX CHARACTER
01700 ;WL             DISPLAY (A) + SPACE
01710 ;WR             DISPLAY (A)
01720 ;
01730 ;++++++++++++++++++++++++++++++++++++++++++++++++++
01740 ;
01750 ;To install this utility, assemble it to disk as, say,
01760 ;TASMON/OVL.  Make a note of the address of LAST at
01770 ;the very end of this listing..
01780 ;
01790 ;Activate your TASMON module and Load TASMON/OVL.  After
01800 ;testing your new 'U' function, you should then 'W'rite
01810 ;the modified version back to disk using your START, the
01820 ;noted address for LAST as the ending address and START
01830 ;as the transfer address.
01840 ;
01850 ;++++++++++++++++++++++++++++++++++++++++++++++++++
01860         ORG     JUMP+OFFSET     ;'U' JP ADDR IN TASMON
01870         DEFW    CONVRT          ;ADDR FOR THIS ROUTINE
01880 ;
01890         ORG     FINISH+10H      ;JUST PAST END OF TASMON
01900 CONVRT  CALL    WL              ;DISPLAY 'U' + SPACE
01910         XOR     A               ;ZERO ACCUMULATOR
01920         LD      (CHARCT),A      ;SET # OF CHARS = 0
01930 GETCHR  CALL    KYBORD          ;GET A CHARACTER
01940         CP      'H'             ;CONVERTING HEX?
01950         JR      Z,HEXDEC        ;GO IF SO
01960         CP      'D'             ;CONVERTING DECIMAL?
01970         JR      NZ,GETCHR       ;INVALID KEY-GET ANOTHER
01980 ;
01990 ;       CONVERT DECIMAL INPUT TO HEX
02000 ;
02010         CALL    WL              ;DISPLAY 'D' + SPACE
02020         LD      BC,BUFFER       ;POINT TO DECIMAL STORAGE
02030 GETDEC  CALL    VALIDH          ;Z FLAG SET IF HEX CHAR
02040         JR      NZ,GETDEC       ;NOT VALID HEX CHAR
02050         CP      3AH             ;> 9 ?
02060         JR      NC,GETDEC       ;NOT 0-9 SO GET ANOTHER
02070         CALL    WR              ;DISPLAY DIGIT 0 - 9
02080         LD      (BC),A          ;AND STORE IT
02090         INC     BC              ;POINT TO NEXT SPOT
02100         LD      A,(CHARCT)      ;GET # DIGITS SO FAR
02110         INC     A               ;BUMP COUNT
02120         LD      (CHARCT),A      ;SAVE COUNT
02130         SUB     5               ;MAX # OF CHARACTERS
02140         JR      NZ,GETDEC       ;IF < 5, GET ANOTHER
02150 ;
02160 ;THIS DECIMAL TO HEX CONVERSION ROUTINE TAKEN FROM
02170 ;"MORE TRS 80 ASSEMBLY LANGUAGE PROGRAMMING" BY
02180 ;BILL BARDEN. PAGE 313. RADIO SHACK CAT. NO 62-2075
02190 ;
02200 ;THE ORIGINAL ROUTINE IS MODIFIED TO REJECT DECIMAL
02210 ;INPUTS OVER 65535.
02220 ;
02230 ;
02240         LD      HL,BUFFER       ;POINT TO STORED #
02250         LD      B,5             ;NUMBER OF DIGITS
02260         LD      IX,0
02270         OR      A               ;RESET CARRY
02280 DECHEX  ADD     IX,IX           ;INTERMEDIATE *2
02290         RET     C               ;>65535, BACK TO TASMON
02300         PUSH    IX              ;SAVE
02310         ADD     IX,IX           ;*4
02320         JR      C,TOOBIG        ;> 65535
02330         ADD     IX,IX           ;*8
02340         JR      C,TOOBIG        ;>65535
02350         POP     DE              ;RETRIEVE *2
02360         ADD     IX,DE           ;*10
02370         RET     C               ;>65535, BACK TO TASMON
02380         LD      A,(HL)          ;GET DIGIT
02390         SUB     30H             ;REMOVE ASCII BIAS
```

```
02400         LD      E,A             ;STORE IN E
02410         LD      D,0             ;(DE) HAS HEX VALUE
02420         ADD     IX,DE           ;MERGE
02430         RET     C               ;>65535, BACK TO TASMON
02440         INC     HL              ;POINT TO NEXT DIGIT
02450         DJNZ    DECHEX          ;DO THIS FOR 5 DIGITS
02460         PUSH    IX              ;NOW, TRANSFER RESULT
02470         POP     HL                ;TO HL
02480         LD      A,20H           ;PRINT A SPACE
02490         CALL    WR
02500         LD      A,H             ;FIRST TWO HEX DIGITS
02510         CALL    SHIFT           ;SHOW HI NIBBLE AS ASCII
02520         LD      A,H             ;GO FOR LOW NIBBLE OF (H)
02530         AND     0FH             ;STRIP 4 HIGH BITS
02540         CALL    ASCII           ;DISPLAY ASCII VALUE
02550         LD      A,L             ;DO THE SAME FOR (L)
02560         CALL    SHIFT
02570         LD      A,L
02580         AND     0FH
02590         CALL    ASCII
02600         RET
02610 TOOBIG  POP     DE              ;FIX STACK
02620         RET                     ;>65535 - BACK TO TASMON
02630 ;
02640 ;       CONVERT HEX INPUT TO DECIMAL
02650 ;
02660 HEXDEC  CALL    WL              ;DISPLAY 'H' + SPACE
02670         CALL    HEXINP          ;GET 4 VALID HEX CHARAC-
02680                                  ;TERS INTO HL VIA TASMON
02690         LD      A,20H           ;BLANK SPACE
02700         CALL    WR              ;PRINT IT
02710 ;
02720 ;(HL) = 4 HEX DIGITS HERE. THE FOLLOWING HEX TO
02730 ;DECIMAL CONVERSION IS TAKEN FROM:
02740 ;"ASSEMBLY LANGUAGE LIBRARY #5" BY CRAIG LINDLEY
02750 ;IN "THE ALTERNATE SOURCE", VOLUME III, NUMBER 5, P 136.
02760 ;
02770         LD      IX,TENTBL       ;POINT TO TABLE
02780 DEC011  LD      B,(IX+1)        ;MSB OF DECIMAL VAL
02790         LD      C,(IX)          ;LSB OF DECIMAL VALUE
02800         OR      A               ;CLEAR CARRY
02810         LD      A,30H           ;ASCII '0'
02820 DEC012  SBC     HL,BC           ;SUBTRACT POWR OF 10
02830         JR      C,DEC013        ;GO WHEN RESULT < 1
02840         INC     A               ;BUMP DECIMAL VALUE
02850         JR      DEC012          ;KEEP GOING
02860 DEC013  ADD     HL,BC           ;FIX REMAINDER
02870         CALL    WR              ;SHOW DECIMAL DIGIT
02880         LD      A,C             ;CHECK FOR END OF TABLE
02890         CP      1                 ;WHEN (C) = 1
02900         RET     Z               ;TO TASMON
02910         INC     IX              ;ELSE, POINT TO
02920         INC     IX                ;NEXT ITEM IN TABLE
02930         JR      DEC011          ;AND PROCESS NEXT CHARAC
02940 SHIFT   SRL     A               ;SHIFT HI NIBBLE TO LOW
02950         SRL     A
02960         SRL     A
02970         SRL     A
02980 ASCII   CP      0AH             ;CHECK IF 0-9
02990         JR      C,DECM          ;GO IF SO
03000         ADD     A,7             ;ADJUST FOR HEX CHARACTER
03010 DECM    ADD     A,30H           ;CONVERT TO ASCII
03020         CALL    WR              ;AND DISPLAY IT
03030         RET
03040 TENTBL  DEFW    10000
03050         DEFW    1000
03060         DEFW    100
03070         DEFW    10
03080         DEFW    1
03090 CHARCT  DEFB    0                ;COUNT OF DECIMAL DIGITS
03100 BUFFER  DEFS    5                ;STORAGE FOR DEC. INPUT
03110 LAST    EQU     $
03120         END
```

ACCESSING U.S. ONLINE INFORMATION SERVICES FROM OTHER COUNTRIES

It's often difficult enough for the novice modem user to figure out how to get online with a mainframe information utility here in the United States. First, you have to figure out how to get your modem hooked up properly, and how to use your terminal program. Then you have to know how to access the information utility you want to reach, usually by way of a packet switching network. It's all really simple once you know how to do it, but learning how can be quite frustrating for some users.

Well, it can be even more frustrating for users outside of the United States, who have the additional complication of going through the local packet switching network (provided by the local PTT) and then interconnecting with a U.S. packet network via a "gateway". Most online services are able to instruct their overseas users on how to make the connection to that point. But then you get online, and certain types of file transfers (particularly those using XMODEM protocol) just don't seem to work right!

Some CompuServe subscribers have figured out how to make their local packet networks behave correctly. Since the commands for packet network operation tend to be similar around the world, these instructions may be useful to those in other countries. If the local packet network provides any kind of instruction sheet or "quick reference" card, by all means try to obtain one and then compare the commands used with those shown below. You may find that the commands you need to use are very similar.

CANADA

For those of us in Canada that access U.S. based information services via Datapac and would like to upload or download with XMODEM protocol, here are the commands that Datapac requires for proper transmission via XMODEM:

Note: underscore indicates a space and is mandatory!

```
Control-P
PROF_1  <Return>
SET_126:004,003:000,004:004,001:000   <Return>
<Return>
GOODBYE  <Return>
```

That's all it takes. Happy downloading!

JLG (Joe Gagnon)

JAPAN

For those of us in Japan that access U.S. based information services via KDD's Venus-P and would like to upload or download with XMODEM protocol, here are the commands that Venus-P requires for proper transmission via XMODEM:

Rules for resetting for Packet: After connecting to CIS, and CIS gives "!" Prompt mark, then it's the proper time to change parameters.

Easy (for downloading only) method:

```
Control-P
SET?3:126,4:4,9:0,13:0,14:0,15:0,20:176 <Return>
<Return>
GO [to wherever you want to be in CompuServe] <Return>
```

Another more difficult (for down/uploading) method:

```
Control-P
SET?1:0,2:0,3:126,4:4,12:0,9:0,13:0,14:0,15:0 <Return>
<Return>
Go [to wherever you want to be in CompuServe] <Return>
```

That's all it takes. Happy downloading from fareast Japan!

Kiyomasa Ono [70127,247] at Tokyo, Japan

UK EUROPE INTERNATIONAL PACKET SWITCHING X.25 PAD PARAMETER

Text document giving information about PAD parameters in the UK, how to set them and what works for XMODEM. Try 'Command-p PROF TP' for a transparent mode. Experience gained through trial and error, parameters from BT Reference Card No. 2, comments and corrections solicited.

James Putnam 70346,1372

A few notes about accessing Compuserve from the UK (Parameters Follow):

British Telecom Packet Switch System (PSS) dial-up PADs (these are the devices you call up - the full name is Packet Assembler Disassembler) have only 18 parameters numbered 1 to 18. While the login procedure usually gives you a usable set of parameters, I have noticed some differences when using the same procedure, so if you want to be sure then set up a macro and set the parameters every time you log in.

To get the PAD's attention send a 'control-P'. Note that how you do this may be different with different software. For example with Jazz type 'shift-command-P' with most other software it's 'command-P'. Next type 'PAR?<return>' to get a list of current parameters which will look like: '1: 01,2:0,3: 126,...18:18'. You can ask for only certain parameters with 'cntl-P PAR?1,3,6...<return>'

Next set the parameters to the values you want with 'cntl-P SET 1:1,3:4,6:1...<return>'. These are example numbers not for use. Only the parameters you set will be affected. Check your work with another 'cntl-P PAR?<return>'.

There is a short cut if you want to use one of the 20 standard profiles. These are A1-A9,B1,B2,D1,D2,V1-V5,SP,TP. These are set with the PROF command: 'PROF TP<return>'. I think TP is the most important because it sets up Transparent mode which lets everything go through both ways. Be careful because once this command is set the only way to reset it is to hang up. If you want to check out the various profiles (except TP) first set the profile with the PROF command and then read back the parameters with the PAR? command. The TP profile is 0,0,0,20,0,0,2,0,0,0,x,0,0,0,0,127,24,18.

Parameter Summary

No.	Function Description	Values	Effect
1	Escape from Data Transfer	0	Escape off
		1	Escape on
2	Echo	0	Echo off
		1	Echo on
3	Data Forward Characters	0	None
		1	A-Z,a-z,0-9
	(Note: values may be summed	2	CR
	to achieve a combination.	4	ESC,BEL,ENQ,ACK
	eg 126 = All characters	8	DEL,CAN,DC2
	between NUL and US of	16	ETX,EOT
	International Alphabet No.5.)	32	HT,LF,VT,FF
	(IA5)	64	All chars. between NUL and US of IA5 not included above.
4	Data forwarding timeout	0	None
		1-255	1-255 x 1/20 seconds timeout
5	Ancillary device control	0	Control off
		1	Control on
6	Suppression of PAD service signals	0	Suppression on
		1	Suppression off
7	Action of PAD on receipt of the Break Signal from the DTE-C	0	No action
		1	PAD transmits INTERRUPT Pkt.
		2	PAD transmits RESET Pkt.
		5	PAD transmits INTERRUPT Pkt. and indication of Break.
		8	PAD escapes from Data Transfer state.
		21	PAD transmits INTERRUPT Pkt. and indication of Break and sets PAD parameter 8 to 1.
8	Suppression of data	0	Suppression off
		1	Suppression on
9	Padding after CR	0	Terminal speed dependent
		1-7	1-7 padding characters
10	Line Folding	0	No line folding
		1-255	Line folding after 1-255 characters
11	Terminal Speed (a 'read only' parameter)	0	110 bit/s
		2	300 bt/s
		3	1200/1200 bit/s

		11	1200/75 bit/s
12	Flow control by DTE-C	0	Flow control off
		1	Flow control on
13	Line feed insertion after carriage return	0	No LF insertion
		1	LF inserted after CR from DTE-P
		4	LF inserted after CR echoed to DTE-C
		5	As for Value 1 + 4
		6	As for 4 + LF inserted in data to DTE-P
		7	Value 1 + 6
14	Padding inserted after line feed character	0	No padding characters
		1-7	1-7 padding characters after LF
15	Editing	0	Editing off
		1	Editing on
16	Character delete character	0	Character delete off
		1-255	Decimal code of delete char.
17	Buffer delete character	0	Buffer display off
		1-255	Decimal code of delete char.
18	Buffer display character	0	Buffer display off
		1-255	Decimal code of display char.

That's it. Good luck and long live x.25!

James Putnam [70346,1372]

KEEPING THE CORRECT TIME

by Jack Decker

This article covers both a hardware and a software solution to keeping the correct time (particularly when the fast clock speed of the Model 4 is used in the Model III mode).

First, the hardware modification: It's called GCLOCK and there are models available for the Model III, 4, or 4P, for external or external mounting (you have to specify which computer you have, and which mounting style you want, when ordering). The internal unit requires a little experience in working with hardware (mostly soldering) to install, but leaves the external bus connector free in case you need it (to run a hard disk drive, or some other external hardware unit). It has been designed to not interfere with other internally mounted boards (such as a graphics board, etc.).

GCLOCK uses a standard lithium battery from Radio Shack that is fairly inexpensive and that should last for years. But I think the best thing about it is that the time is read directly from ports 30H through 3FH, so that software routines that access the clock do not have to be unnecessarily complicated. Model 4P users that do a lot of work in the Model III mode really have it made, because a routine that reads the GCLOCK and then transfers the correct time to the normal real-time clock storage locations can be patched right into the MODELA/III ROM image, thus keeping the system clock always accurate no matter which DOS is used. Patches are also available for use with TRSDOS 6.x / LS-DOS 6.3.

Steve Wemyss, the designer of GCLOCK, says that a few people have even trashed their Alpha NEWCLOCKs in favor of his design. I can believe it, it works very well and appears to be a really well-designed unit. For more information, contact GSOFT, 89 Peachey Road, Elizabeth Fields, South Australia 5113, Australia. The price in Australia is $85.00 but since the Australian dollar is worth quite a bit less than the U.S. dollar, the cost would be less here in the U.S. even after the additional shipping charges are figured in (if you want to go ahead and place an order, the shipping charges will probably be about $5.00 additional. Your bank can give you the current exchange rate on Australian money, and U.S. checks can be cashed in Australia, or some banks can give you a bank draft in Australian dollars).

Now the software solution, which is a lot less expensive but not nearly as accurate, for Model 4 users that run NEWDOS/80 in the Model III mode.

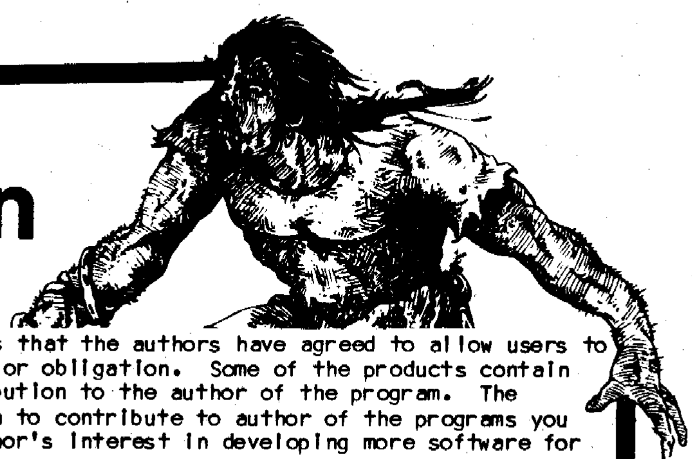
Many methods have been published to invoke the 4 MHz CPU speed of the Model 4 from the Model III mode. The process is relatively simple, in that one need only set bit 6 of the byte at memory location 4210H and then output that byte to port 0ECH. This has only one major drawback, and that is that the system clock runs twice as fast.

The following program invokes the fast clock speed and patches the Model III version of NEWDOS/80 to keep the correct time (well, as correct as it ever was under the slow speed, anyway). It uses only nine bytes of high memory. Once the source code shown below is assembled to a /CMD file, you can simply execute that /CMD file from NEWDOS/80 READY and it will take care of the rest. This program has been used with the Syslink BBS in Sault Ste. Marie, Ontario (705-253-5366, 300/1200 bps) to permit much faster operation while still keeping the correct time. Just one more reminder, this program runs under NEWDOS only and is just about guaranteed to crash the system if you try and run it under any other DOS! If this doesn't meet your needs try the GCLOCK, it's a lot more accurate and doesn't have to be reset every time you power-up the computer.

```
              00100 ;THE FOLLOWING PATCH IS TO NEWDOS/80 (MODEL III)
              00110 ;TO SPEED UP THE CPU CLOCK SPEED TO 4 MHZ WHILE STILL
              00120 ;KEEPING THE SYSTEM TIME CORRECT.
              00130
6000          00140        ORG    6000H          ;JUST HERE TIL RELOCATED
6000 3A5400   00150 INTLZE LD     A,(54H)        ;GET BYTE FROM ROM
6003 3D       00160        DEC    A              ;CHECK IF ON MODEL I
6004 C8       00170        RET    Z              ;BAIL OUT IF MODEL I
6005 213E60   00180        LD     HL,END         ;END OF UNRELOCATED PGRM
6008 ED5B1144 00190        LD     DE,(4411H)     ;END OF UNPROTECTED MEM
600C 010900   00200        LD     BC,END-PATCH+1 ;LENGTH OF MAIN PROGRAM
600F EDB8     00210        LDDR                  ;MOVE THE PROGRAM
6011 ED531144 00220        LD     (4411H),DE     ;SAVE NEW HIMEM POINTER
6015 13       00230        INC    DE             ;DE=START RELOCATED PRGRM
6016 F3       00240        DI                    ;DISABLE INTERRUPTS
6017 ED535C45 00250        LD     (455CH),DE     ;REPLACE CALL ADDRESS
601B 213160   00260        LD     HL,NEWCOD      ;CODE TO BE RELOCATED
601E 115745   00270        LD     DE,4557H       ;PLACE TO STUFF IT
6021 010500   00280        LD     BC,5           ;BYTES TO MOVE
6024 EDB0     00290        LDIR                  ;MOVE THE CODE
6026 211042   00300        LD     HL,4210H       ;POINT TO PORT 0ECH MASK
6029 7E       00310        LD     A,(HL)         ;GET MASK BYTE
602A F640     00320        OR     40H            ;SET BIT 6
602C 77       00330        LD     (HL),A         ;RE-SAVE MASK BYTE
602D D3EC     00340        OUT    (0ECH),A       ;& OUTPUT IT TO PORT 0ECH
602F FB       00350        EI                    ;RE-ENABLE INTERRUPTS
6030 C9       00360        RET                   ;EXIT RELOCATOR ROUTINE
              00370
              00380 ;CODE TO BE RELOCATED
              00390
6031 3E1E     00400 NEWCOD LD     A,1EH          ;ROM CLOCK COUNTER VALUE
6033 BE       00410        CP     (HL)           ;VALUE JUST UPDATED?
              00420 ;FOLLOWING INSTRUCTION IS XOR'ED WITH ITSELF IN LINE 540,
              00430 ;TO ALTERNATELY MAKE IT AN 'RLCA' OR 'NOP' INSTRUCTION.
              00440 ;THIS MEANS THAT ON ALTERNATE PASSES THE VALUE IN A WILL
              00450 ;EITHER BE DOUBLED (FROM 1EH TO 3CH) OR LEFT AS IS.
6034 07       00460        RLCA                  ;DOUBLE IT (OR NOP HERE)
6035 CC       00470        DEFB   0CCH           ;START OF "CALL Z" INST.
              00480
              00490 ;THIS IS THE ACTUAL PATCH CODE
              00500
6036 77       00510 PATCH  LD     (HL),A         ;RELOAD VALUE (1E OR 3C)
6037 215A45   00520        LD     HL,455AH       ;POINT TO "RLCA" OR "NOP"
603A 7E       00530        LD     A,(HL)         ;GET CURRENT INSTRUCTION
603B EE07     00540        XOR    7              ;CHANGE IT TO THE OTHER
603D 77       00550        LD     (HL),A         ;STORE IT BACK
603E C9       00560        RET
              00570
603E          00580 END    EQU    $-1            ;USED BY RELOCATOR
              00590
6000          00600        END    INTLZE
00000 TOTAL ERRORS

END     603E     INTLZE  6000     NEWCOD  6031     PATCH   6036
```

Public Domain



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

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

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

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

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

New PD's 23 & 24!

VISA MasterCard

Yes, please send me the following public domain diskettes:

Code	Price	Code	Price

Shipping:

Total Amount:

Charge Card Number:

Expiration Date:

Your Name:

Address:

Address:

City/State/Zip:

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

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

NORTHERN BYTES

Subscription Information

Northern Bytes is published on an irregular basis by The Alternate Source Information Outlet. Back issues are available starting with Volume 5, Number 1. Issues prior to that are not available. Currently there are eight back issues available for Volume 5, and eight back issues for Volume 6, as well as all issues from Volume 7. All back issues prior to Volume 7, Number 3 are $2 each. Back issues starting with Volume 7, Number 3 are $4 each (prices include all shipping charges unless you live overseas and request airmail delivery, in which case we may charge extra to cover postage).

It is very easy to be placed on the Northern Bytes **REGULAR** list. Simply place your address, Visa or Mastercard number and expiration date on file with us. We will start with the issue you request. We do not bill you for **ANY ISSUE** until that issue has been mailed. This way, we can continue to offer you top quality information with absolutely no risk to you. There's no question of what to do about unfulfilled issues if we decide to quit publishing. Unless otherwise requested, we presume your subscription will extend through the month of your credit card expiration date. **PLEASE NOTE** your card expiration date (which will appear on your mailing label) and be sure to send us your new card number and/or expiration date when you receive it to assure uninterrupted delivery of Northern Bytes.

Don't have a charge card, huh? We understand the myriad of reasons for not having them and we feel that a "To-Be-Invoiced" policy could help increase the demand for Northern Bytes. The Alternate Source maintains a regular list of readers that have asked **TO BE BILLED** for each issue. You then send a check (made payable to The Alternate Source Information Outlet) for each issue as you receive it. If you don't send a check, we presume that your interest has died and discontinue your subscription. The only requirement for getting onto the list is to pay for the first issue up front; the next will be mailed automatically, if you request. You are assured that you will receive top of the line TRS-80 information as it is released. Ask to be placed on the **NO RISK** "To-Be-Invoiced Northern Bytes List".

Call or write, but SIGN UP TODAY!
The Alternate Source Information Outlet
704 North Pennsylvania Avenue
Lansing, Michigan 48906-5319
Telephone: (517) 482-8270
EMAIL: CompuServe: 72167,161 / MCI Mail: 109-7407 / Telex: 6501097407 MCI

NORTHERN BYTES

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

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

POSTMASTER:
Address Correction Requested
Return Postage Guaranteed

To: