```
                                page    40,132
                        ;_____________________________________________________________________________
                        ;       T0DRIVE is part of T0DOS. Copyright (C) 2021 Ton Daas
                        ;       T0DOS is free software: you can redistribute it and/or modify
                        ;       it under the terms of the GNU General Public License as published by
                        ;       the Free Software Foundation, either version 3 of the License,
                        ;       or any later version.
                        ;
                        ;       T0DOS is distributed in the hope that it will be useful,
                        ;       but WITHOUT ANY WARRANTY; without even the implied warranty of
                        ;       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
                        ;       See the GNU General Public License for more details.
                        ;
                        ;       You should have received a copy of the GNU General Public License
                        ;       along with this program.  If not, see <https://www.gnu.org/licenses/>.
                        ;_____________________________________________________________________________
                        ; Device-driver for unused FAT space on Bootdisk Track 0, Head 0
                        ; With T0FORMAT this space is formatted as a 160 KB DOS diskette
                        ; of which only the first 31 KB can be used. Remaining area is marked unusable.
                        ;=============================================================================
0000                    T0DRIVE SEGMENT
                                ASSUME  CS:T0DRIVE

                        ;Device header
0000  FF FF FF FF       NXT_DEV DD      -1
0004  2000                      DW      2000h   ;Block device, no IOCTRL, non IBM format
0006  0150 R                    DW      DEV_STR ;Pointer to device strategy
0008  0171 R                    DW      DEV_INT ;Pointer to device interrupt handler
000A  01 07 [                   DB      1,7 dup (0)     ;Number of block devices
              00
                  ]


= 0002                  RH_CMD  EQU     2       ;byte
= 0003                  RH_STAT EQU     3       ;word
```

```
                        RH_UNTS EQU     13      ;byte
                        RH_EADR EQU     14      ;dword
                        RH_DTA  EQU     14      ;dword
                        RH_PBPB EQU     18      ;dword
                        RH_CNT  EQU     18      ;word
                        RH_OFS  EQU     20      ;word
                        RH_DRV  EQU     22      ;byte

                        ST_ERR  EQU     8000H
                        ST_DONE EQU     100H
                        ST_WERR EQU     0Ah
                        ST_RERR EQU     0Bh

                        ; BIOS Parameter Block
                        MEDIA   EQU     0F8h
                        FAT_NBR EQU     2
                        DIR_SIZ EQU     4
                        FATSIZE EQU     (63-3-dir_siz)*3/2+3

0012    03 [            BOOTSEC DB      3 DUP (0)
                00
                    ]

0015    54 30 2D 44 52 49       DB      'T0-DRIVE'      ;8 char OEM ID
        56 45
001D    0200            BPB1    DW      512     ;Bytes per Sector
001F    01                      DB      1       ;Sectors per Cluster
0020    0001                    DW      1       ;Reserved Sectors
0022    02                      DB      FAT_NBR ;Number of FATs
0023    0040                    DW      DIR_SIZ*16      ;Number of ROOT Directory entries
0025    003F            VOL_SIZ DW      63      ;Total number of sectors
0027    F8                      DB      MEDIA   ;Media descriptor
0028    0001                    DW      1       ;Sectors per FAT
002A    003F            TRK_SIZ DW      63      ;Sectors per track
002C    0001                    DW      1       ;Heads per cyl
```

```
002E  0000                        DW      0       ;Hidden sectors preceding the boot sector
= 0013                      BPB_SIZ EQU     $-BPB1
0030  0000                        DW      0       ;Hidden Sectors high word for LBA
0032  3F 00 00 00           LBA_SIZ DD      63      ;Large sectors (for LBA type)
0036  80                    DRIVE   DB      80h     ;Physical Drive Number
0037  00                            DB      0       ;Current Head (Boot record track)
0038  00                            DB      0       ;Signature
0039  ????????              SERIAL  DD      ?       ;ID (random serial #)
003D  28 43 29 54 6F 6E             DB      '(C)Ton Daas'  ;Volume Label (no longer in use)
      20 44 61 61 73
0048  46 41 54 31 32 20            DB      'FAT12   '     ;System ID (either FAT12 or FAT16)
      20 20
= 003E                      BS_SIZ  EQU     $-BOOTSEC

0050  001D R               BPB_PTR DW      BPB1    ;pointer aray with only one entry

                            ; Do the actual read, write or verify
0052                        DO_13   PROC    NEAR
0052  2E: 8B 16 002A R             MOV     DX,TRK_SIZ      ;get track size in DL, DH=0
0057  3B CA                       CMP     CX,DX   ;make sure we stay on cyl 0
0059  73 20                       JNB     I13CF
005B  2A D1                       SUB     DL,CL   ;remaining sectors on track
005D  3A C2                       CMP     AL,DL   ;is count within remaining?
005F  76 02                       JBE     I13VAL
0061  8A C2                       MOV     AL,DL   ;reduce count to valid value
0063  41                    I13VAL: INC     CX      ;make start sector 1 origin
0064  2E: 8A 16 0036 R             MOV     DL,DRIVE        ;drive
                            ;       MOV     DH,0    ;head 0 (is aleady 0)
0069  BF 0003                     MOV     DI,3    ;set retry count
006C  96                          XCHG    SI,AX   ;save AX
006D  8B C6                 I13CMD: MOV     AX,SI   ;restore command and sector count
006F  CD 13                       INT     13h     ;read/write sectors with transfer area
0071  73 0C                       JNC     I13OK
0073  B4 00                       MOV     AH,0
0075  CD 13                       INT     13h     ;reset disk
```

```
106      0077  4F                        DEC     DI
107      0078  75 F3                     JNZ     I13CMD  ;retry
108      007A  49                        DEC     CX
109      007B  B0 00               I13CF: MOV     AL,0    ;set count to 0
110      007D  F9                        STC
111      007E  C3                        RET
112
113      007F  96                  I13OK: XCHG    AX,SI   ;get command and sector count back in AX
114      0080  49                        DEC     CX      ;make sector # 0 origin again
115      0081  C3                        RET
116      0082                      DO_13  ENDP
117
118      0082                      INT13  PROC    NEAR
119      0082  57                        PUSH    DI
120      0083  06                        PUSH    ES
121      0084  26: 8A 45 12              MOV     AL,BYTE PTR ES:[DI+RH_CNT]    ;get sector count
122      0088  26: 8B 4D 14              MOV     CX,ES:[DI+RH_OFS]      ;starting logical sector
123      008C  26: C4 5D 0E              LES     BX,dword ptr ES:[DI+RH_DTA]   ;get disk transfer address
124      0090  E3 0B                     JCXZ    INT13b  ;jump to bootsector handler
125      0092  83 F9 03                  CMP     CX,3    ;is request for sector higher then fat?
126      0095  72 28                     JB      INT13f  ;skip to fat handler if not the boot sector
127      0097  E8 0052 R           INT13S: CALL    DO_13
128      009A  07                  I13RET: POP     ES
129      009B  5F                        POP     DI
130      009C  C3                        RET
131
132                                ; handle bootsector access (cx=0)
133      009D  80 FC 02            INT13B: CMP     AH,2h   ;request for read of bootsector?
134      00A0  75 15                     JNE     I13NXT  ;do nothing if write or verify with bootsector
135      00A2  51                        PUSH    CX
136      00A3  BE 0012 R                 MOV     SI,OFFSET BOOTSEC
137      00A6  8B FB                     MOV     DI,BX
138      00A8  B9 003E                   MOV     CX,BS_SIZ
139      00AB  F3/ A4                    REP MOVSB       ;copy bpb into bootsector buffer on read
140      00AD  96                        XCHG    SI,AX   ;save AX
```

```
00AE  B9 01C2                   MOV     CX,512-BS_SIZ
00B1  B0 00                     MOV     AL,0
00B3  F3/ AA                    REP STOSB
00B5  96                        XCHG    AX,SI   ;restore AX
00B6  59                        POP     CX
00B7  80 C7 02          I13NXT: ADD     BH,2    ;skip to next block in user buffer
00BA  41                        INC     CX      ;next sector
00BB  FE C8                     DEC     AL      ;decrease sector count
00BD  76 DB                     JBE     I13RET  ;nothing more to do, also return if CF=1

                        ; handle fat sector access, write fat handling
00BF  80 FC 03          INT13F: CMP     AH,3h   ;is it a write request?
00C2  77 D3                     JA      INT13S  ;verify request can be handled normally
00C4  74 33                     JE      WRTFAT  ;go to handler for writing fat

                        ; read fat handling. Do no disk access, but transfer from our buffer
00C6  83 F9 02          RDFATN: CMP     CX,2    ;is this the second fat?
00C9  77 CC                     JA      INT13S  ;jump to handle normal data sector
00CB  BE 0521 R                 MOV     SI,OFFSET FAT1+1
00CE  72 03                     JB      RDFAT
00D0  BE 0721 R                 MOV     SI,OFFSET FAT2+1
00D3  8B FB             RDFAT:  MOV     DI,BX   ;point to user buffer
00D5  51                        PUSH    CX
00D6  50                        PUSH    AX
00D7  B0 F8                     MOV     AL,MEDIA
00D9  AA                        STOSB
00DA  B9 0056                   MOV     CX,FATSIZE-1
00DD  F3/ A4                    REP MOVSB       ;fill user buffer with fat
00DF  B9 008D                   MOV     CX,(512-FATSIZE)/3
00E2  B8 7FF7           BADFAT: MOV     AX,7FF7h
00E5  AB                        STOSW
00E6  B0 FF                     MOV     AL,0FFh
00E8  AA                        STOSB
00E9  E2 F7                     LOOP    BADFAT
                                REPT    (512-FATSIZE) MOD 3
```

```
                                        STOSB
                                        ENDM
00EB  AA                        +       STOSB
00EC  AA                        +       STOSB
00ED  58                                POP     AX
00EE  59                                POP     CX
00EF  80 C7 02                          ADD     BH,2    ;should end with CF=0
00F2  41                                INC     CX
00F3  FE C8                             DEC     AL
00F5  77 CF                             JA      RDFATN
00F7  EB A1                     II13R:  JMP     SHORT I13RET

                                ; copy fat from user buffer into our saved sectors and write them instead
00F9  06                        WRTFAT: PUSH    ES
00FA  1E                                PUSH    DS
00FB  8C C6                             MOV     SI,ES
00FD  8C DF                             MOV     DI,DS
00FF  8E C7                             MOV     ES,DI
0101  8E DE                             MOV     DS,SI
                                        ASSUME  DS:NOTHING
0103  51                                PUSH    CX
0104  8A F0                             MOV     DH,AL   ;save sectorcount
0106  B0 00                             MOV     AL,0    ;initialize sector count for us to write
0108  8A D1                             MOV     DL,CL   ;save first sector #
010A  80 FA 02                  CPFATN: CMP     DL,2    ;fat 2 or not?
010D  77 1B                             JA      W13FAT
010F  BF 0523 R                         MOV     DI,OFFSET FAT1+3
0112  72 03                             JB      CPFAT
0114  BF 0723 R                         MOV     DI,OFFSET FAT2+3
0117  8D 77 03                  CPFAT:  LEA     SI,[BX+3]
011A  B9 0054                           MOV     CX,FATSIZE-3
011D  F3/ A4                            REP MOVSB       ;copy user fat into our first fat sector
011F  80 C7 02                          ADD     BH,2
0122  FE C0                             INC     AL      ;count # sectors for us to write
0124  FE C2                             INC     DL      ;increment to next sector
```

```
0126  FE CE                        DEC     DH      ;sector count remaining
0128  75 E0                        JNZ     CPFATN
012A  59                  W13FAT:   POP     CX      ;get original logical sector to start with
012B  1F                            POP     DS
012C  53                            PUSH    BX
012D  52                            PUSH    DX
012E  BB 0520 R                     MOV     BX,OFFSET FAT1 ;point to our own fats
0131  83 F9 02                      CMP     CX,2
0134  72 03                         JB      W13F
0136  BB 0720 R                     MOV     BX,OFFSET FAT2
0139  E8 0052 R           W13F:    CALL    DO_13
013C  5A                            POP     DX
013D  8A CA                         MOV     CL,DL   ;set to next sector to do
013F  8A C6                         MOV     AL,DH   ;set to remaining sector count
0141  5B                            POP     BX
0142  07                           POP     ES
0143  72 B2                        JC      II13R
0145  84 C0                        TEST    AL,AL   ;any more sectors to write?
0147  74 AE                        JZ      II13R
0149  E9 0097 R                    JMP     INT13S
014C                       INT13    ENDP

                           ;=============================================================================
                           ;Device strategy routine

014C                       RH_PTR   LABEL   DWORD   ;Request header offset and segment
014C  ????                 RH_ADR   DW      ?
014E  ????                 RH_SEG   DW      ?

0150                       DEV_STR  PROC    FAR
0150  2E: 89 1E 014C R             MOV     CS:RH_ADR,BX    ;Save offset of request-header
0155  2E: 8C 06 014E R             MOV     CS:RH_SEG,ES    ;Save segment of request header
015A  CB                           RET
015B                       DEV_STR  ENDP
```

```
246      015B  0200 R              CMD_TBL DW      INIT
247      015D  01A3 R                      DW      MEDCHK
248      015F  01C4 R                      DW      BLDBPB
249      0161  01A8 R                      DW      RET_OK  ;IOCTL input (unsupported)
250      0163  01D5 R                      DW      READ
251      0165  01A8 R                      DW      RET_OK  ;NON-DESTRUCTIVE INPUT NO WAIT (char. dev. only)
252      0167  01A8 R                      DW      RET_OK  ;INPUT STATUS (char. dev. only)
253      0169  01A8 R                      DW      RET_OK  ;INPUT FLUSH (char. dev. only)
254      016B  01E1 R                      DW      WRITE
255      016D  01E1 R                      DW      WR_VER
256      016F  01A8 R              CMD_END DW      RET_OK  ;OUTPUT STATUS, OUTPUT FLUSH, IOCTL output (unsuported)
257
258                                ;Device interrupt routine
259      0171                      DEV_INT PROC    FAR
260      0171  06                          PUSH    ES
261      0172  1E                          PUSH    DS
262      0173  57                          PUSH    DI
263      0174  56                          PUSH    SI
264                                ;unused PUSH    BP
265      0175  53                          PUSH    BX
266      0176  51                          PUSH    CX
267      0177  52                          PUSH    DX
268      0178  50                          PUSH    AX
269      0179  8C C8                       MOV     AX,CS
270      017B  BB 0520 R                   MOV     BX,OFFSET MYSTACK
271      017E  8C D2                       MOV     DX,SS
272      0180  FA                          CLI
273      0181  8E D0                       MOV     SS,AX
274      0183  87 E3                       XCHG    SP,BX
275      0185  FB                          STI
276      0186  52                          PUSH    DX
277      0187  53                          PUSH    BX
278      0188  9C                          PUSHF
279      0189  FC                          CLD
280      018A  8E D8                       MOV     DS,AX
```

```
                                        ASSUME  DS:T0DRIVE
018C  C4 3E 014C R                      LES     DI,RH_PTR
0190  26: 8A 45 02                      MOV     AL,ES:[DI+RH_CMD]      ;Get command
0194  98                                CBW
0195  D1 E0                            SAL     AX,1
0197  BB 0014                          MOV     BX,CMD_END-CMD_TBL
019A  3B C3                            CMP     AX,BX
019C  73 01                            JNB     JMPTBL
019E  93                               XCHG    BX,AX
019F  FF A7 015B R                JMPTBL: JMP     CMD_TBL[BX]

                                  ; Return Media not changed flag
01A3  26: C6 45 0E 01             MEDCHK: MOV     BYTE PTR ES:[DI+14],1  ;flag media not changed
01A8  33 D2                       RET_OK: XOR     DX,DX   ;pass no error
01AA  81 CA 0100                  DONE:   OR      DX,ST_DONE
01AE  26: 89 55 03                        MOV     WORD PTR ES:[DI+RH_STAT],DX   ;Set status done
01B2  9D                                  POPF
01B3  5B                                  POP     BX
01B4  58                                  POP     AX
01B5  FA                                  CLI
01B6  8E D0                               MOV     SS,AX   ;restore DOS stack
01B8  87 E3                               XCHG    SP,BX
01BA  FB                                  STI
01BB  58                                  POP     AX
01BC  5A                                  POP     DX
01BD  59                                  POP     CX
01BE  5B                                  POP     BX
                                  ;unused POP     BP
01BF  5E                                  POP     SI
01C0  5F                                  POP     DI
01C1  1F                                  POP     DS
01C2  07                                  POP     ES
01C3  CB                                  RET

                                  ; return pointer to BPB
```

```
01C4  2E: C4 3E 014C R       BLDBPB: LES     DI,CS:RH_PTR   ;load request header address
01C9  26: C7 45 12 001D R            MOV     WORD PTR ES:[DI+RH_PBPB],OFFSET BPB1
01CF  26: 8C 4D 14                   MOV     ES:[DI+RH_PBPB+2],CS
01D3  EB D3                          JMP     SHORT RET_OK

01D5  B4 02                  READ:   MOV     AH,2h  ;read
01D7  E8 0082 R                      CALL    INT13
01DA  73 CC                          JNC     RET_OK
01DC  BA 800B                        MOV     DX,ST_ERR+0Bh
01DF  EB 18                          JMP     SHORT RETCNT

01E1                         WR_VER:
01E1  B4 03                  WRITE:  MOV     AH,3h  ;specify write
01E3  E8 0082 R                      CALL    INT13
01E6  72 0E                          JC      WR_ERR
01E8  26: 80 7D 02 09                CMP     BYTE PTR ES:[DI+RH_CMD],9     ;was it a write verify command?
01ED  75 B9                          JNE     RET_OK
01EF  B4 04                          MOV     AH,4h  ;verify
01F1  E8 0082 R                      CALL    INT13
01F4  73 B2                          JNC     RET_OK
01F6  BA 800A                WR_ERR: MOV     DX,ST_ERR+0Ah  ;set write fault
01F9  98                     RETCNT: CBW
01FA  26: 89 45 12                   MOV     ES:[DI+RH_CNT],AX     ;return actual sectors rd/wr
01FE  EB AA                          JMP     SHORT DONE

                             ; Do we have a fixed disk drive?
0200  26: C4 7D 12           INIT:   LES     DI,dword ptr ES:[DI+RH_PBPB]
0204  B0 20                          MOV     AL,' '
0206  B9 0050                        MOV     CX,80
0209  F2/ AE                        REPNE SCASB
020B  E3 10                          JCXZ    INI_DR
020D  26: 8A 05                      MOV     AL,ES:[DI]
0210  B2 4F                        MOV     DL,80h-'1'
0212  02 D0                        ADD     DL,AL
0214  79 07                        JNS     INI_DR
```

```
0216  88 16 0036 R                MOV     DRIVE,DL
021A  A2 02DF R                   MOV     MSG_DSK,AL
021D  C4 3E 014C R          INI_DR: LES     DI,RH_PTR
0221  8A 16 0036 R                MOV     DL,DRIVE
0225  B4 08                       MOV     AH,08h  ;Read drive parms
0227  CD 13                       INT     13h
0229  72 71                       JC      NODEV
022B  84 D2                       TEST    DL,DL
022D  74 6D                       JZ      NODEV

                            ; Get tracksize and prepare drive information message
022F  91                    INICON: XCHG    AX,CX
0230  24 3F                       AND     AL,3Fh  ;mask max sectors value
0232  A2 0025 R                   MOV     BYTE PTR VOL_SIZ,AL
0235  A2 002A R                   MOV     BYTE PTR TRK_SIZ,AL
0238  A2 0032 R                   MOV     BYTE PTR LBA_SIZ,AL
023B  2C 07                       SUB     AL,7    ;reduce with MBR, FAT, DIR
023D  D0 E8                       SHR     AL,1    ;convert to KB
023F  73 05                       JNC     INISIZ
0241  C6 06 0304 R 35            MOV     BYTE PTR MSG_SIZ+3,'5'
0246  D4 0A                 INISIZ: AAM
0248  0D 3030                     OR      AX,'00'
024B  80 FC 30                    CMP     AH,'0'
024E  75 02                       JNE     INITEN
0250  B4 20                       MOV     AH,' '
0252  86 C4                 INITEN: XCHG    AL,AH
0254  A3 0301 R                   MOV     WORD PTR MSG_SIZ,AX
0257  26: 8A 45 16                MOV     AL,ES:[DI+RH_DRV]       ;get drive number
025B  3C 1A                       CMP     AL,26   ;if not a valid value (as with DOS 2),
025D  73 05                       JNB     BADDRV  ;then skip and keep '?' to display
025F  04 41                       ADD     AL,'A'
0261  A2 031B R                   MOV     MSG_DRV,AL

                            ; Read real FAT sectors to get a copy of the hidden code
0264  0E                    BADDRV: PUSH    CS
```

```
0265  07                            POP     ES      ;point to our own segment
0266  BB 0520 R                     MOV     BX,OFFSET FAT1
0269  B9 0001                       MOV     CX,1    ;logical sector 1
026C  B8 0202                       MOV     AX,202H ;read 2 sectors
026F  E8 0052 R                     CALL    DO_13
0272  C4 3E 014C R                  LES     DI,RH_PTR
0276  72 24                         JC      NODEV
0278  BA 031F R                     MOV     DX,OFFSET MSG_FAT
027B  80 3F FE                      CMP     BYTE PTR DS:[BX],0FEh
027E  75 1F                         JNE     NOFAT
0280  80 C7 02                      ADD     BH,2
0283  80 3F FE                      CMP     BYTE PTR DS:[BX],0FEh
0286  75 17                         JNE     NOFAT

                                ; Return BPB-table, number of devices, Resident memory size
0288  B0 01                             MOV     AL,1    ;set devicecount to 1
028A  BA 02B6 R                         MOV     DX,OFFSET MSG_OK
028D  BB 0920 R                         MOV     BX,OFFSET RES_SIZ
0290  26: C7 45 12 0050 R               MOV     WORD PTR ES:RH_PBPB[DI],OFFSET BPB_PTR
0296  26: 8C 5D 14                      MOV     ES:RH_PBPB+2[DI],DS
029A  EB 07                             JMP     SHORT INIMSG

029C  BA 035B R                 NODEV:  MOV     DX,OFFSET MSG_BAD
029F  32 C0                     NOFAT:  XOR     AL,AL   ;set devicecount to 0
02A1  33 DB                             XOR     BX,BX   ;return ending address
02A3  26: 88 45 0D              INIMSG: MOV     ES:RH_UNTS[DI],AL
02A7  26: 89 5D 0E                      MOV     ES:RH_EADR[DI],BX
02AB  26: 8C 5D 10                      MOV     ES:RH_EADR+2[DI],DS
02AF  B4 09                             MOV     AH,9h
02B1  CD 21                             INT     21h
02B3  E9 01A8 R                         JMP     RET_OK
02B6                            DEV_INT ENDP

02B6  54 30 2D 44 4F 53         MSG_OK  DB      'T0-DOS Drive (C) loaded using fixed disk '
      20 44 72 69 76 65
```

```
                20 28 43 29 20 6C
                6F 61 64 65 64 20
                75 73 69 6E 67 20
                66 69 78 65 64 20
                64 69 73 6B 20
02DF            30 2C 20 63 79 6C       MSG_DSK DB      '0, cyl 0, head 0',13,10,'  volume space: '
                20 30 2C 20 68 65
                61 64 20 30 0D 0A
                20 20 76 6F 6C 75
                6D 65 20 73 70 61
                63 65 3A 20
0301            20 30 2E 30 20 4B       MSG_SIZ DB      ' 0.0 KiB; assigned Drive: '
                69 42 3B 20 61 73
                73 69 67 6E 65 64
                20 44 72 69 76 65
                3A 20
031B            3F 0D 0A 24             MSG_DRV DB      '?',13,10,'$'
031F            46 69 78 65 64 20       MSG_FAT DB      'Fixed disk track 0 is unformatted. T0Drive not installed.'
                64 69 73 6B 20 74
                72 61 63 6B 20 30
                20 69 73 20 75 6E
                66 6F 72 6D 61 74
                74 65 64 2E 20 54
                30 44 72 69 76 65
                20 6E 6F 74 20 69
                6E 73 74 61 6C 6C
                65 64 2E
0358            0D 0A 24                        DB      13,10,'$'
035B            4E 6F 20 66 69 78       MSG_BAD DB      'No fixed disk found or failure to install T0Drive.',13,10,'$'
                65 64 20 64 69 73
                6B 20 66 6F 75 6E
                64 20 6F 72 20 66
                61 69 6C 75 72 65
                20 74 6F 20 69 6E
                73 74 61 6C 6C 20
```

```
                54 30 44 72 69 76
                65 2E 0D 0A 24

                                        ; Space for our own stack
                                                EVEN
0390            80 [                            DW      80h dup (?)     ;minimum stack size
                        ????
                             ]

0490            48 [                            DW      (NXT_DEV-$)/2 AND 0FFh dup (?)        ;round up to page boundary
                        ????
                             ]

0520                                    MYSTACK LABEL   WORD
0520  0200 [                           FAT1    DB      512 dup (?)
                        ??
                           ]

0720  0200 [                           FAT2    DB      512 dup (?)
                        ??
                           ]

= 0920                                  RES_SIZ EQU     $
0920                                    T0DRIVE ENDS
                                                END
```