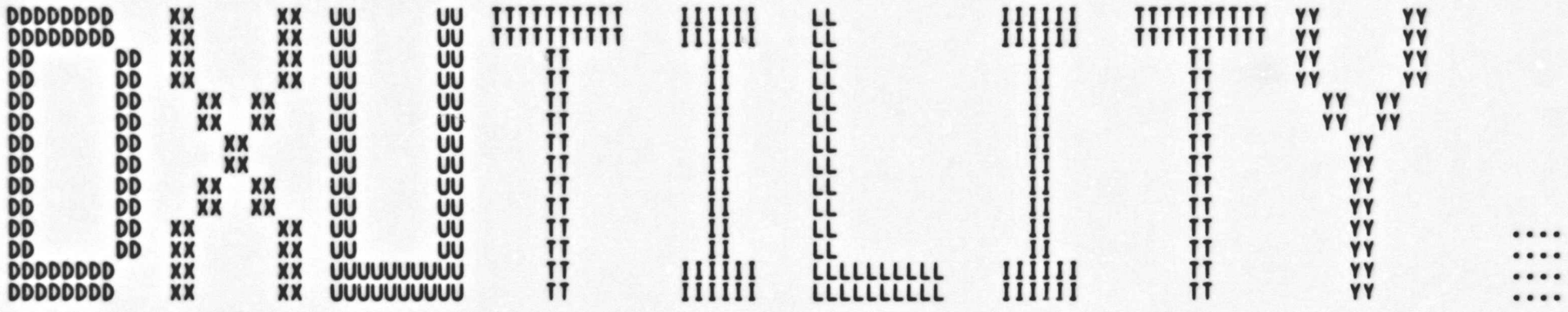
```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
```

FILEID**DXUTILITY



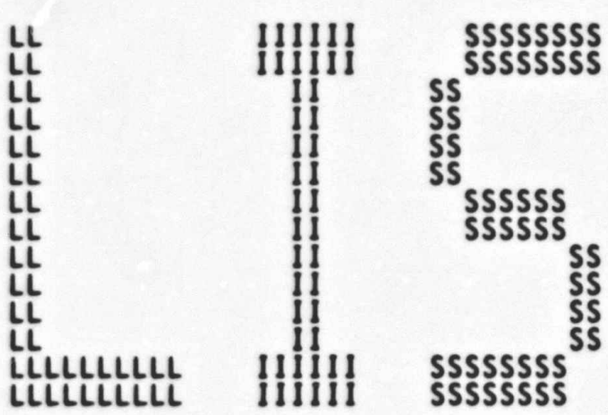


Table of contents

```
0000     1          .TITLE  DXUTILITY - FLOPPY DISK DRIVER UTILITY ROUTINES
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ; C. A. MONIA 25-JUN-77
0000    29 ;
0000    30 ; MODIFIED BY:
0000    31 ;
0000    32 ;       V03-001 PRD0021         Paul R. DeStefano       27-Apr-1983
0000    33 ;               Corrected computation of next media address to accept sector
0000    34 ;               addresses in the range of 1 to 26 when doing physical I/O.
0000    35 ;
0000    36 ;       V02-003 TCM0001         Trudy C. Matthews       19-Jun-1981
0000    37 ;               Clear R1 (which becomes the second longword of the IOSB)
0000    38 ;               before exiting.  This is because certain code paths through
0000    39 ;               I/O completion (notably for paging I/O and swapping I/O,
0000    40 ;               when this driver is being used for standalone sysgen systems)
0000    41 ;               expect the second longword of the IOSB to be zero.
0000    42 ;
0000    43 ;
0000    44 ; STAR FLOPPY DRIVER UTILITY ROUTINES
0000    45 ;
0000    46 ; MACRO LIBRARY CALLS
0000    47 ;
0000    48
0000    49          $ADPDEF                         ;DEFINE ADP OFFSETS
0000    50          $CRBDEF                         ;DEFINE CRB OFFSETS
0000    51          $EMBDEF                         ;DEFINE EMB OFFSETS
0000    52          $IDBDEF                         ;DEFINE IDB OFFSETS
0000    53          $IODEF                          ;DEFINE I/O FUNCTION CODES
0000    54          $IRPDEF                         ;DEFINE IRP OFFSETS
0000    55          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
0000    56          $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
0000    57          $UBADEF                         ;DEFINE UBA REGISTER OFFSETS
```

```
0000     58          $UCBDEF                         ;DEFINE UCB OFFSETS
0000     59          $VECDEF                         ;DEFINE INTERRUPT DISPATCH VECTOR OFFSETS
0000     60
0000     61 ;
0000     62 ; LOCAL MACROS
0000     63 ;
0000     64 ; REPORT DEVICE ERROR
0000     65 ;
0000     66
0000     67          .MACRO  RPTERR  CODE
0000     68          MOVZWL  CODE,R0
0000     69          BRB     FUNCXT
0000     70          .ENDM
```

```
                        .SBTTL  RX FUNCTION DECISION TABLE
                ;+
                ; RX FUNCTION DECISION TABLE
                ;-

00000000                .PSECT  $$$115_DRIVER,LONG
                DX$FUNCTABLE::                          ;FUNCTION DECISION TABLE
                        FUNCTAB ,-                      ;LEGAL FUNCTIONS
                                <SENSECHAR,-            ;SENSE CHARACTERISTICS
                                SETCHAR,-               ;SET CHARACTERISTICS
                                SENSEMODE,-             ;SENSE MODE
                                SETMODE,-               ;SET MODE
                                READLBLK,-              ;READ LOGICAL BLOCK
                                WRITELBLK,-             ;WRITE LOGICAL BLOCK
                                READPBLK,-              ;READ PHYSICAL BLOCK
                                WRITEPBLK,-             ;WRITE PHYSICAL BLOCK
                                READVBLK,-              ;READ VIRTUAL BLOCK
                                WRITEVBLK,-             ;WRITE VIRTUAL BLOCK
                                ACCESS,-                ;ACCESS FILE AND/OR FIND DIRECTORY ENTRY
                                ACPCONTROL,-            ;ACP CONTROL FUNCTION
                                CREATE,-                ;CREATE FILE AND/OR DIRECTORY ENTRY
                                DEACCESS,-              ;DEACCESS FILE
                                DELETE,-                ;DELETE FILE AND/OR DIRECTORY ENTRY
                                MODIFY,-                ;MODIFY FILE ATTRIBUTES
                                MOUNT>                  ;MOUNT VOLUME
                        FUNCTAB ,-                      ;BUFFERED I/O FUNCTIONS
                                <SENSECHAR,-            ;SENSE CHARACTERISTICS
                                SETCHAR,-               ;SET CHARACTERISTICS
                                SENSEMODE,-             ;SENSE MODE
                                SETMODE,-               ;SET MODE
                                ACCESS,-                ;ACCESS FILE AND/OR FIND DIRECTORY ENTRY
                                ACPCONTROL,-            ;ACP CONTROL FUNCTION
                                CREATE,-                ;CREATE FILE AND/OR DIRECTORY ENTRY
                                DEACCESS,-              ;DEACCESS FILE
                                DELETE,-                ;DELETE FILE AND/OR DIRECTORY ENTRY
                                MODIFY,-                ;MODIFY FILE ATTRIBUTES
                                MOUNT>                  ;MOUNT VOLUME
                FUNCTAB         +ACP$READBLK,-          ;READ FUNCTIONS
                                <READLBLK,-             ;READ LOGICAL BLOCK
                                READPBLK,-              ;READ PHYSICAL BLOCK
                                READVBLK>               ;READ VIRTUAL BLOCK
                FUNCTAB         +ACP$WRITEBLK,-         ;WRITE FUNCTIONS
                                <WRITELBLK,-            ;WRITE LOGICAL BLOCK
                                WRITEPBLK,-             ;WRITE PHYSICAL BLOCK
                                WRITEVBLK>              ;WRITE VIRTUAL BLOCK
                FUNCTAB         +ACP$ACCESS,<ACCESS,CREATE> ;ACCESS AND CREATE FILE OR DIRECTORY
                FUNCTAB         +ACP$DEACCESS,<DEACCESS>;DEACCESS FILE
                FUNCTAB         +ACP$MODIFY,-           ;
                                <ACPCONTROL,-           ;ACP CONTROL FUNCTION
                                DELETE,-                ;DELETE FILE OR DIRECTORY ENTRY
                                MODIFY>                 ;MODIFY FILE ATTRIBUTES
                FUNCTAB         +ACP$MOUNT,<MOUNT>      ;MOUNT VOLUME
                FUNCTAB         +EXE$SENSEMODE,-        ;
                                <SENSECHAR,-            ;SENSE CHARACTERISTICS
                                SENSEMODE>              ;SENSE MODE
                FUNCTAB         +EXE$SETCHAR,-          ;
                                <SETCHAR,-              ;SET CHARACTERISTICS
```

```
                0064    129                     SETMODE>                ;SET MODE
```

```
                                0070   131          .SBTTL  START I/O OPERATION
                                0070   132
                                0070   133 ;+
                                0070   134 ; DX$STARTIO - START I/O ON FLOPPY DEVICE UNIT
                                0070   135 ;
                                0070   136 ; THIS ROUTINE IS ENTERED VIA A 'BSBW' TO START I/O ON A DEVICE UNIT.
                                0070   137 ; CONTROL ALTERNATES BETWEEN THE FLOPPY DRIVER AND THIS CODE. THIS ROU-
                                0070   138 ; TINE IS CALLED TO PERFORM HARDWARE INDEPENDANT PROCESSING. ALL HARD-
                                0070   139 ; WARE SPECIFIC PROCESSING IS PERORMED BY DEVICE-SPECIFIC CODE IN THE
                                0070   140 ; DRIVER.
                                0070   141 ;
                                0070   142 ; THE PROTOCOL IS AS FOLLOWS:
                                0070   143 ;
                                0070   144 ;       1. THE DRIVER CALLS DX$STARTIO TO ESTABLISH INITIAL CONDITIONS.
                                0070   145 ;
                                0070   146 ;       2. DXUTILITY COMPUTES THE PHYSICAL MEDIA ADDRESS AND BUFFER ADDRESS
                                0070   147 ;          AND EXECUTES A CO-ROUTINE CALL TO THE DRIVER.
                                0070   148 ;
                                0070   149 ;       3. THE DRIVER POSITIONS THE MEDIA, PERFORMS ANY ADDITIONAL HARDWARE
                                0070   150 ;          FUNCTIONS AND RETURNS CONTROL TO DXUTILITY.
                                0070   151 ;
                                0070   152 ;       4. CONTROL ALTERNATES BETWEEN THE DRIVER AND DXUTILITY UNTIL ONE
                                0070   153 ;          SECTOR OF DATA HAS BEEN TRANSFERRED TO THE INTERNAL BUFFER.
                                0070   154 ;
                                0070   155 ;       5. DXUTILITY FLAGS COMPLETION OF A SECTOR TRANSFER AND CALLS THE DRI-
                                0070   156 ;          VER TO PERFORM END-OF-SECTOR PROCESSING AND DROP TO FORK LEVEL.
                                0070   157 ;
                                0070   158 ;       6. THE DRIVER CALLS DXUTILITY TO UPDATE THE MEDIA ADDRESS. IF MORE
                                0070   159 ;          DATA REMAINS TO BE TRANSFERRED, DXUTILITY TRANSFERS CONTROL TO
                                0070   160 ;          STEP 2 TO CONTINUE PROCESSING.
                                0070   161 ;
                                0070   162 ; EACH CO-ROUTINE CALL SITE CONTAINS A CONTINGENCY EXIT ADDRESS. IN THE
                                0070   163 ; EVENT OF A HARDWARE ERROR, CONTROL WILL BE PASSED TO THE ERROR HANDLER
                                0070   164 ; TO EFFECT A RE-TRY OR TERMINATE THE REQUEST.
                                0070   165 ;
                                0070   166 ; INPUTS:
                                0070   167 ;
                                0070   168 ;       R3 = ADDRESS OF I/O PACKET
                                0070   169 ;
                                0070   170 ;       R5 = UCB ADDRESS OF DEVICE UNIT
                                0070   171 ;
                                0070   172 ; OUTPUTS:
                                0070   173 ;
                                0070   174 ;       *******OUTPUTS*******
                                0070   175 ;
                                0070   176 ;-
                                0070   177
                                0070   178          .ENABL  LSB
                                0070   179
                                0070   180 DX$STARTIO::                                ;START I/O OPERATION
        21 2A A3    08    E0    0070   181          BBS     #IRP$V_PHYSIO,IRP$W_STS(R3),1$ ;BYPASS BLOCK COMPUTATION IF PHYSICAL
50   00    38 A3    04    7A    0075   182          EMUL    #4,IRP$L_MEDIA(R3),#0,R0 ;SCALE LBN, QUAD RESULT TO R0
            52    44 A5    9A    007B   183          MOVZBL  UCB$L_DEVDEPEND(R5),R2  ;GET NUMBER OF SECTORS PER TRACK
38 A3    50    50    52    7B    007F   184          EDIV    R2,R0,R0,IRP$L_MEDIA(R3) ;COMPUTE SECTOR
            52    45 A5    9A    0085   185          MOVZBL  UCB$L_DEVDEPEND+1(R5),R2 ;GET TRACKS PER CYLINDER
   51    50    50    52    7B    0089   186          EDIV    R2,R0,R0,R1             ;COMPUTE TRACKS (R1), CYL. (R0)
            39 A3    51    90    008E   187          MOVB    R1,IRP$L_MEDIA+1(R3)    ;SAVE TRACK ADDRESS
```

```
                3A A3   50   B0  0092   188                 MOVW    R0,IRP$L_MEDIA+2(R3)    ;SAVE CYLINDER ADDRESS
                                0096   189 1$:               ;
            00D8 C5   7E A5   B0  0096   190                 MOVW    UCB$W_BCNT(R5),UCB$W_DX_BCR(R5) ;STORE BYTE COUNT
              00 68 A5   03   E5  009C   191                 BBCC    #UCB$V_DX_WRITE,UCB$W_DEVSTS(R5),2$ ;CLEAR WRITE FLAG
                                00A1   192 2$:               ;
                      0126      30  00A1   193                 BSBW    SETBUF                  ;SETUP BUFFER PARAMETERS
        50    20 A3   06   00   EF  00A4   194                 EXTZV   #IRP$V_FCODE,#IRP$S_FCODE,IRP$W_FUNC(R3),R0 ;GET FUNCTION CODE
                        0C   50   91  00AA   195                 CMPB    R0,#IO$_READPBLK        ;READ PHYSICAL BLOCK?
                           08     13  00AD   196                 BEQL    5$                      ;IF EQL YES
              00 68 A5   03   E2  00AF   197                 BBSS    #UCB$V_DX_WRITE,UCB$W_DEVSTS(R5),3$ ;SET WRITE FLAG
                                00B4   198 3$:               ;
                      00EE      30  00B4   199                 BSBW    MOVFRUSER               ;GET INITIAL SECTOR FULL OF DATA
                                00B7   200 5$:               ;
          0080 C5   0081 C5   90  00B7   201                 MOVB    UCB$B_ERTMAX(R5),UCB$B_ERTCNT(R5) ;INITIALIZE ERROR RETRY COUNT
              04 20 A3   0F   E1  00BE   202                 BBC     #IO$V_INHRETRY,IRP$W_FUNC(R3),10$ ;IF BIT CLEAR, PERFORM NORMAL RETR
                      0080 C5   94  00C3   203                 CLRB    UCB$B_ERTCNT(R5)        ;INHIBIT RETRIES ON ERROR
                                00C7   204 10$:              ;
                      0081      30  00C7   205                 BSBW    TRKSEC                  ;COMPUTE MEDIA ADDRESS
                 46 A5   02 A2  91  00CA   206                 CMPB    2(R2),UCB$W_CYLINDERS(R5) ;LEGAL DISK ADDRESS?
                           46     1E  00CF   207                 BGEQU   70$                     ;IF GEQU NO
              64 A5   00E8 8F   AA  00D1   208                 BICW    #UCB$M_CANCEL!-         ;CLEAR CANCEL I/O,
                                00D7   209                 UCB$M_INTTYPE!-         ;INTERRUPT TYPE
                                00D7   210                 UCB$M_POWER!-           ;POWERFAIL,AND
                                00D7   211                 UCB$M_TIMOUT,UCB$W_STS(R5)      ;TIMEOUT STATUS BITS
                                00D7   212 20$:              ;
                           53     D6  00D7   213                 INCL    R3                      ;FLAG READY FOR TRANSFER
                           9E     16  00D9   214                 JSB     @(SP)+                  ;SEEK/TRANSFER ONE BYTE OF DATA
                           53     D4  00DB   215                 CLRL    R3                      ;ASSUME TRANSFERRED LAST BYTE
                      00D0 C5     D6  00DD   216                 INCL    UCB$L_DX_BFPNT(R5)      ;INCREMENT BUFFER POINTER
                      00DA C5     97  00E1   217                 DECB    UCB$B_DX_SCTCNT(R5)     ;DECREMENT SECTOR COUNT
                           F0     12  00E5   218                 BNEQ    20$                     ;IF NEQ TRANSFER ANOTHER BYTE
                           9E     16  00E7   219                 JSB     @(SP)+                  ;CALL THE CALLER
                        0095      30  00E9   220                 BSBW    XFER                    ;TRANSFER DATA TO/FROM USER
                     3F 50        E9  00EC   221                 BLBC    R0,IOSUCC               ;IF LBC DONE
                                00EF   222
                                00EF   223 ;
                                00EF   224 ; COMPUTE NEXT MEDIA ADDRESS
                                00EF   225 ;
                                00EF   226
            53    58 A5        D0  00EF   227                 MOVL    UCB$L_IRP(R5),R3        ; Get address of I/O packet.
                  38 A3        96  00F3   228                 INCB    IRP$L_MEDIA(R3)         ; Increment sector address.
                  38 A3        91  00F6   229                 CMPB    IRP$L_MEDIA(R3),-       ; Is the new sector address greater
                  44 A5                00F9   230                         UCB$B_SECTORS(R5)       ;  than the number of sectors per track?
                     08        E0  00FB   231                 BBS     #IRP$V_PHYSIO,-         ; Branch if performing physical I/O.
                07 2A A3              00FD   232                         IRP$W_STS(R3),50$       ;
                     C5        1F  0100   233                 BLSSU   10$                     ; Branch if new logical sector address
                                0102   234                                                 ;  is on the same track.
                  38 A3        94  0102   235                 CLRB    IRP$L_MEDIA(R3)         ; Otherwise, reset sector address to 0.
                     06        11  0105   236                 BRB     60$                     ; Jump to common code.
                     BE        1B  0107   237 50$:     BLEQU   10$                     ; Branch if new physical sector address
                                0109   238                                                 ;  is on the same track.
            38 A3    01        90  0109   239                 MOVB    #1,IRP$L_MEDIA(R3)      ; Otherwise, reset sector address to 1.
                  3A A3        96  010D   240 60$:     INCB    IRP$L_MEDIA+2(R3)       ; Increment track address.
                  3A A3        91  0110   241                 CMPB    IRP$L_MEDIA+2(R3),-     ; Is the new track address greater than
                  46 A5                0113   242                         UCB$W_CYLINDERS(R5)     ;  the number of tracks on the disk.
                     B0        1F  0115   243                 BLSSU   10$                     ; Branch if within range.
                                0117   244 70$:     RPTERR  #SS$_IVADDR             ; Otherwise, invalid disk address.
```

```
                     011E    245
                     011E    246
                     011E    247 ;+
                     011E    248 ; DX$ERR - CONTINGENCY EXIT HANDLER
                     011E    249 ;
                     011E    250 ; THIS ROUTINE IS ENTERED WHENEVER THE DRIVER DETECTS A POWER-FAIL, DEVICE
                     011E    251 ; TIMEOUT OR HARDWARE ERROR CONDITION.  IF THE ERROR WAS CAUSED BY DEVICE
                     011E    252 ; TIMEOUT OR POWER FAIL, THE TRANSFER IS RESTARTED. IF THE ERROR IS NON-
                     011E    253 ; FATAL, THE RETRY COUNT IS DECREMENTED AND THE TRANSFER IS REPEATED IF
                     011E    254 ; THE RESULT IS NONZERO. ALL OTHER CONDITIONS RESULT IN REQUEST TERMINA-
                     011E    255 ; TION WITH STATUS CONTAINED IN R0
                     011E    256 ;
                     011E    257 ; INPUTS:
                     011E    258 ;
                     011E    259 ;       R0 = ERROR STATUS CODE
                     011E    260 ;       R3 = ERROR SEVERITY INDICATION
                     011E    261 ;
                     011E    262 ;       R5 = ADDRESS OF UCB
                     011E    263 ;       (SP) = RETURN TO DRIVER RESTART CODE
                     011E    264 ;
                     011E    265 ;       R3 LBC = FATAL ERROR
                     011E    266 ;       R3 LBS = RETRIABLE ERROR
                     011E    267 ;
                     011E    268 ; OUTPUTS:
                     011E    269 ;
                     011E    270 ;       NONE
                     011E    271 ;-
                     011E    272
                     011E    273 DX$ERR::                                        ;
          10 53    E9 011E    274         BLBC    R3,FUNCXT                       ;IF LBC, FATAL HARDWARE ERROR
A1 64 A5   05    E0 0121    275         BBS     #UCB$V_POWER,UCB$W_STS(R5),10$ ;RETRY ON POWER FAIL
        0080 C5    97 0126    276         DECB    UCB$B_ERTCNT(R5)                ;DECREMENT RETRY COUNT
             9B    14 012A    277         BGTR    10$                             ;IF GTR, TRY AGAIN
             03    11 012C    278         BRB     FUNCXT                          ;TERMINATE REQUEST
                     012E    279
                     012E    280         .DSABL  LSB
                     012E    281
                     012E    282 ;
                     012E    283 ; TERMINATE REQUEST SUCCESFULLY
                     012E    284 ;
                     012E    285
                     012E    286 IOSUCC:                                         ;
          50 01    3C 012E    287         MOVZWL  #SS$_NORMAL,R0                  ;SET SUCCESS
                     0131    288
                     0131    289 ;
                     0131    290 ; FUNCTION COMPLETION COMMON EXIT
                     0131    291 ;
                     0131    292
                     0131    293 FUNCXT:                                         ;
             6E    D4 0131    294         CLRL    (SP)                            ;ZERO COROUTINE ADDRESS ON STACK
             50    DD 0133    295         PUSHL   R0                              ;SAVE REGISTER
   00000000'GF    16 0135    296         JSB     G^IOC$DIAGBUFILL                ;FILL DIAGNOSTIC BUFFER IF PRESENT
02 AE   7E A5 00D8 C5 A3 013B 297         SUBW3   UCB$W_DX_BCR(R5),UCB$W_BCNT(R5),2(SP) ;COMPUTE BYTES TRANSFERRED
             03    BA 0143    298         POPR    #^M<R0,R1>                      ;RESTORE R0 AND PUT ZERO IN R1
                     0145    299         REQCOM                                  ;TERMINATE REQUEST
                     014B    300
                     014B    301 ;+
```

```
                     014B    302 ; TRKSEC - CONVERT LOGICAL TO PHYSICAL DISK ADDRESS
                     014B    303 ;
                     014B    304 ; THIS ROUTINE IS ENTERED VIA A 'BSB' TO CONVERT A LOGICAL DISK
                     014B    305 ; ADDRESS TO A PHYSICAL ADDRESS. IF THE PHYSICAL I/O FLAG IS SET
                     014B    306 ; IN THE I/O REQUEST PACKET, THE CONVERSION CONSISTS OF SIMPLY
                     014B    307 ; MOVING THE LOGICAL TRACK, SECTOR AND CYLINDER ADDRESSES IN
                     014B    308 ; THE PACKET MEDIA LONGWORD TO THE MEDIA ADDRESS LONGWORD IN
                     014B    309 ; THE UCB. IF LOGICAL I/O IS BEING PERFORMED, THEN THE LOGICAL
                     014B    310 ; ADDRESS IN THE I/O PACKET IS CONVERTED TO A PHYSICAL ADDRESS
                     014B    311 ; BY APPLYING INTERLEAVE AND TRACK-TO-TRACK SKEW. THE RESULT IS
                     014B    312 ; PLACED IN THE UCB MEDIA ADDRESS LONGWORD.
                     014B    313 ;
                     014B    314 ; INPUTS:
                     014B    315 ;
                     014B    316 ;
                     014B    317 ;       R5 = ADDRESS OF UCB
                     014B    318 ;
                     014B    319 ; OUTPUTS:
                     014B    320 ;
                     014B    321 ;       R2 = POINTER TO PHYSICAL MEDIA ADDRESS
                     014B    322 ;
                     014B    323 ;       UCB$L_MEDIA CONTAINS THE PHYSICAL MEDIA ADDRESS
                     014B    324 ;
                     014B    325 ;
                     014B    326 ;-
                     014B    327
                     014B    328 TRKSEC:                                             ;
          53   58 A5   D0  014B    329         MOVL    UCB$L_IRP(R5),R3            ;GET ADDRESS OF REQUEST PACKET
        52   00BC C5   9E  014F    330         MOVAB   UCB$L_MEDIA(R5),R2          ;POINT TO PHYSICAL MEDIA ADDRESS
          62   38 A3   D0  0154    331         MOVL    IRP$L_MEDIA(R3),(R2)        ;COPY LOGICAL ADDRESS
        23 2A A3   08   E0  0158    332         BBS     #IRP$V_PHYSIO,IRP$W_STS(R3),10$ ;BYPASS CONVERSION IF PHYSICAL I/O
             51   62   9A  015D    333         MOVZBL  (R2),R1                     ;GET CURRENT LOGICAL SECTOR
             51   0C   91  0160    334         CMPB    #12,R1                      ;SET C IF 12 < SECTOR <= 26
             51   51   D8  0163    335         ADWC    R1,R1                       ;DOUBLE SECTOR NUMBER, ADD INTERLEAVE FACTOR
          50   02   A2   9A  0166    336         MOVZBL  2(R2),R0                    ;GET CYLINDER NUMBER
50   51   50   06   7A  016A    337         EMUL    #6,R0,R1,R0                 ;COMPUTE SKEW (6 * CYL + SECT)
        7E   44   A5   9A  016F    338         MOVZBL  UCB$B_SECTORS(R5),-(SP)     ;GET SECTORS/TRACK
51   50   50   8E   7B  0173    339         EDIV    (SP)+,R0,R0,R1              ;MODULO SECTOR INTO SECTORS PER TRACK
             51   D6  0178    340         INCL    R1                          ;OFFSET SECTOR NUMBER BY 1
             62   51   90  017A    341         MOVB    R1,(R2)                     ;SAVE SECTOR NUMBER
                02   A2   96  017D    342         INCB    2(R2)                       ;INCREMENT PAST UNUSED CYLINDER
                     0180    343 10$:                                            ;
                     05  0180    344         RSB                                 ;
                     0181    345
                     0181    346 ;+
                     0181    347 ; XFER - TRANSFER DATA TO OR FROM USER
                     0181    348 ;
                     0181    349 ; THIS ROUTINE IS ENTERED VIA A BSB TO TRANSFER ONE SECTOR'S WORTH OF DATA
                     0181    350 ; TO OR FROM THE USER'S PROCESS.
                     0181    351 ;
                     0181    352 ; INPUTS:
                     0181    353 ;
                     0181    354 ;       R5 = ADDRESS OF UCB
                     0181    355 ;       UCB$W_DX_BCR = BYTES LEFT TO TRANSFER
                     0181    356 ;
                     0181    357 ; OUTPUTS:
                     0181    358 ;
```

```
                          0181   359 ;       RO LSB SET = MORE DATA TO TRANSFER
                          0181   360 ;       UCB$W_DX_BCR = COUNT OF BYTES REMAINING
                          0181   361 ;       RO LSB CLEAR = NO MORE DATA TO TRANSFER
                          0181   362 ;
                          0181   363 ;-
                          0181   364
                          0181   365         .ENABL  LSB
                          0181   366
                          0181   367 XFER:                                          ;
           7E    D4       0181   368         CLRL    -(SP)                         ;ASSUME REQUEST COMPLETE
           45    10       0183   369         BSBB    SETBUF                        ;SETUP TRANSFER PARAMETERS
     00D8  C5    52   A2  0185   370         SUBW    R2,UCB$W_DX_BCR(R5)           ;UPDATE BYTE COUNT
  08 68  A5    03   E1  018A   371         BBC     #UCB$V_DX_WRITE,UCB$W_DEVSTS(R5),10$ ;BRANCH IF READ REQUEST
           10    13       018F   372         BEQL    30$                           ;IF EQL DONE
           37    10       0191   373         BSBB    SETBUF                        ;GET MORE DATA FROM USER
           10    10       0193   374         BSBB    MOVFRUSER                     ;...
           08    11       0195   375         BRB     20$                           ;EXIT SUCCESSFULLY
                          0197   376 10$:                                         ;
           17    10       0197   377         BSBB    MOVTOUSER                     ;MOVE DATA TO USER
     00D8  C5    B5       0199   378         TSTW    UCB$W_DX_BCR(R5)              ;TRANSFER COMPLETE NOW?
           02    13       019D   379         BEQL    30$                           ;IF EQL YES
                          019F   380 20$:                                         ;
           6E    D6       019F   381         INCL    (SP)                          ;SET SUCCESS
                          01A1   382 30$:                                         ;
     50    8E    D0       01A1   383         MOVL    (SP)+,RO                      ;SET SUCCESS
           05    01A4   384         RSB                                           ;
                          01A5   385
                          01A5   386         .DSABL  LSB
                          01A5   387
                          01A5   388 ;+
                          01A5   389 ; MOVFRUSER - MOVE DATA FROM USER TO FLOPPY BUFFER
                          01A5   390 ;
                          01A5   391 ; INPUTS:
                          01A5   392 ;
                          01A5   393 ;       R1 = ADDRESS OF FLOPPY BUFFER
                          01A5   394 ;       R2 = BYTE COUNT
                          01A5   395 ;       R5 = ADDRESS OF UCB
                          01A5   396 ;
                          01A5   397 ; OUTPUTS:
                          01A5   398 ;
                          01A5   399 ;       THE CONTENTS OF THE INTERNAL BUFFER ARE COPIED FROM THE USER'S
                          01A5   400 ;       ADDRESS SPACE.
                          01A5   401 ;
                          01A5   402 ;-
                          01A5   403
                          01A5   404         .ENABL  LSB
                          01A5   405
                          01A5   406 MOVFRUSER:                                    ;
     54    52    D0       01A5   407         MOVL    R2,R4                         ;SAVE BYTE COUNT
00000000'GF    16       01A8   408         JSB     G^IOC$MOVFRUSER               ;MOVE DATA
           09    11       01AE   409         BRB     10$                           ;UPDATE BUFFER ADDRESS
                          01B0   410
                          01B0   411 ;+
                          01B0   412 ; MOVTOUSER - MOVE CONTENTS OF FLOPPY INTERNAL BUFFER TO USER
                          01B0   413 ;
                          01B0   414 ; THIS ROUTINE IS CALLED TO TRANSFER THE CONTENTS OF THE FLOPPY DATA
                          01B0   415 ; BUFFER TO THE USER'S ADDRESS SPACE.
```

```
                        01B0   416 ;
                        01B0   417 ; INPUTS:
                        01B0   418 ;
                        01B0   419 ;       R1 = ADDRESS OF FLOPPY BUFFER
                        01B0   420 ;       R2 = BYTE COUNT
                        01B0   421 ;       R5 = ADDRESS OF UCB
                        01B0   422 ;
                        01B0   423 ; OUTPUTS:
                        01B0   424 ;
                        01B0   425 ;       THE FLOPPY BUFFER CONTENTS ARE COPIED TO THE USER'S ADDRESS SPACE
                        01B0   426 ;
                        01B0   427 ;-
                        01B0   428
                        01B0   429 MOVTOUSER:                                   ;
      54    52    D0    01B0   430         MOVL    R2,R4                        ;SAVE BYTE COUNT
 00000000'GF    16    01B3   431         JSB     G^IOC$MOVTOUSER              ;MOVE DATA TO USER'S BUFFER
                        01B9   432 10$:                                         ;
   7C A5    54    A0    01B9   433         ADDW    R4,UCB$W_BOFF(R5)            ;UPDATE PAGE OFFSET
7C A5    FE00 8F    AA    01BD   434         BICW    #^C<^X01FF>,UCB$W_BOFF(R5) ;MAKE MODULO PAGE SIZE
            04    12    01C3   435         BNEQ    20$                          ;IF NEQ PAGE BOUNDARY NOT CROSSED
   78 A5    04    C0    01C5   436         ADDL    #4,UCB$L_SVAPTE(R5)          ;UPDATE ADDRESS OF USER PTE
                        01C9   437 20$:                                         ;
            05    01C9   438         RSB                                      ;
                        01CA   439
                        01CA   440         .DSABL  LSB
                        01CA   441
                        01CA   442 ;+
                        01CA   443 ; SETBUF - SETUP PARAMETERS FOR TRANSFER TO OR FROM USER'S BUFFER
                        01CA   444 ;
                        01CA   445 ; THIS ROUTINE IS ENTERED VIA A BSB TO INITIALIZE ALL PARAMETERS REQUIRED
                        01CA   446 ; TO TRANSFER ONE SECTOR OF DATA TO OR FROM THE USER'S PROCESS.
                        01CA   447 ;
                        01CA   448 ; INPUTS:
                        01CA   449 ;
                        01CA   450 ;       R5 = ADDRESS OF UCB
                        01CA   451 ;
                        01CA   452 ; OUTPUTS:
                        01CA   453 ;
                        01CA   454 ;       R1 = ADDRESS OF SECTOR BUFFER
                        01CA   455 ;       R2 = NUMBER OF BYTES TO TRANSFER TO OR FROM USER
                        01CA   456 ;       UCB$B_SECTCNT = 128
                        01CA   457 ;       UCB$L_DXBFPNT = ADDRESS OF SECTOR BUFFER
                        01CA   458 ;-
                        01CA   459
                        01CA   460 SETBUF:                                      ;
50    00D8 C5    3C    01CA   461         MOVZWL  UCB$W_DX_BCR(R5),R0          ;GET COUNT OF BYTES REMAINING
  52     80 8F    9A    01CF   462         MOVZBL  #128,R2                      ;ASSUME FULL SECTOR TO TRANSFER
00DA C5    52    90    01D3   463         MOVB    R2,UCB$B_DX_SCTCNT(R5)       ;RESET SECTOR COUNT
51    00CC C5    D0    01D8   464         MOVL    UCB$L_DX_BUF(R5),R1          ;GET BUFFER ADDRESS
00D0 C5    51    D0    01DD   465         MOVL    R1,UCB$L_DX_BFPNT(R5)        ;SET ADDRESS
      50    52    B1    01E2   466         CMPW    R2,R0                        ;SECTOR EXCEED BYTES LEFT
            03    1F    01E5   467         BLSSU   10$                          ;IF LSSU NO
      52    50    D0    01E7   468         MOVL    R0,R2                        ;SET COUNT TO SMALLER AMOUNT
                        01EA   469 10$:                                         ;
            05    01EA   470         RSB                                      ;
                        01EB   471
                        01EB   472 DXP_END::
```

```
                                                L 12
01EB    473
01EB    474         .END
```

```
ACP$ACCESS                ********    X   02
ACP$DEACCESS              ********    X   02
ACP$MODIFY                ********    X   02
ACP$MOUNT                 ********    X   02
ACP$READBLK               ********    X   02
ACP$WRITEBLK              ********    X   02
DX$ERR                    0000011E RG     02
DX$FUNCTABLE              00000000 RG     02
DX$STARTIO                00000070 RG     02
DXP_END                   000001EB RG     02
EXE$SENSEMODE             ********    X   02
EXE$SETCHAR               ********    X   02
FUNCTAB_LEN             = 00000070
FUNCXT                    00000131 R      02
IO$V_INHRETRY           = 0000000F
IO$_ACCESS              = 00000032
IO$_ACPCONTROL          = 00000038
IO$_CREATE              = 00000033
IO$_DEACCESS            = 00000034
IO$_DELETE              = 00000035
IO$_MODIFY              = 00000036
IO$_MOUNT               = 00000039
IO$_READLBLK            = 00000021
IO$_READPBLK            = 0000000C
IO$_READVBLK            = 00000031
IO$_SENSECHAR           = 0000001B
IO$_SENSEMODE           = 00000027
IO$_SETCHAR             = 0000001A
IO$_SETMODE             = 00000023
IO$_VIRTUAL             = 0000003F
IO$_WRITELBLK           = 00000020
IO$_WRITEPBLK           = 0000000B
IO$_WRITEVBLK           = 00000030
IOC$DIAGBUFILL            ********    X   02
IOC$MOVFRUSER             ********    X   02
IOC$MOVTOUSER             ********    X   02
IOC$REQCOM                ********    X   02
IOSUCC                    0000012E R      02
IRP$L_MEDIA             = 00000038
IRP$S_FCODE             = 00000006
IRP$V_FCODE             = 00000000
IRP$V_PHYSIO            = 00000008
IRP$W_FUNC              = 00000020
IRP$W_STS               = 0000002A
MASKH                   = 00000008
MASKL                   = 04000000
MOVFRUSER                 000001A5 R      02
MOVTOUSER                 000001B0 R      02
SETBUF                    000001CA R      02
SS$_IVADDR              = 00000134
SS$_NORMAL              = 00000001
TRKSEC                    0000014B R      02
UCB$B_DX_SCTCNT         = 000000DA
UCB$B_ERTCNT            = 00000080
UCB$B_ERTMAX            = 00000081
UCB$B_SECTORS           = 00000044
UCB$L_DEVDEPEND         = 00000044
UCB$L_DX_BFPNT          = 000000D0
UCB$L_DX_BUF            = 000000CC
UCB$L_IRP               = 00000058
UCB$L_MEDIA             = 000000BC
UCB$L_SVAPTE            = 00000078
UCB$M_CANCEL            = 00000008
UCB$M_INTTYPE           = 00000080
UCB$M_POWER             = 00000020
UCB$M_TIMOUT            = 00000040
UCB$V_DX_WRITE          = 00000003
UCB$V_POWER             = 00000005
UCB$W_BCNT              = 0000007E
UCB$W_BOFF              = 0000007C
UCB$W_CYLINDERS         = 00000046
UCB$W_DEVSTS            = 00000068
UCB$W_DX_BCR            = 000000D8
UCB$W_STS               = 00000064
XFER                      00000181 R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	000001EB (491.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.04	00:00:02.07
Command processing	118	00:00:00.36	00:00:03.34
Pass 1	421	00:00:10.82	00:00:47.45
Symbol table sort	0	00:00:01.82	00:00:07.42
Pass 2	95	00:00:02.01	00:00:10.42
Symbol table output	10	00:00:00.09	00:00:00.63
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	682	00:00:15.16	00:01:11.35

The working set limit was 1650 pages.
91445 bytes (179 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1719 non-local and 16 local symbols.
474 source lines were read in Pass 1, producing 14 object records in Pass 2.
26 pages of virtual memory were used to define 25 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	15
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	21

1852 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:DXUTILITY/OBJ=OBJ$:DXUTILITY MSRC$:DXUTILITY/UPDATE=(ENH$:DXUTILITY)+EXECML$/LIB