L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
_$
Ps
--
YZ
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
```

```
**FILE**ID**IOSUBRAMS                                             C 10
```

```
            IIIIII      000000      SSSSSSSS  UU      UU  BBBBBBBB    RRRRRRRR      AAAAAA    MM      MM    SSSSSSSS
            IIIIII      000000      SSSSSSSS  UU      UU  BBBBBBBB    RRRRRRRR      AAAAAA    MM      MM    SSSSSSSS
              II      00      00  SS          UU      UU  BB      BB  RR      RR  AA      AA  MMMM  MMMM  SS
              II      00      00  SS          UU      UU  BB      BB  RR      RR  AA      AA  MMMM  MMMM  SS
              II      00      00  SS          UU      UU  BB      BB  RR      RR  AA      AA  MM  MM  MM  SS
              II      00      00  SS          UU      UU  BB      BB  RR      RR  AA      AA  MM  MM  MM  SS
              II      00      00    SSSSSS    UU      UU  BBBBBBBB    RRRRRRRR    AA      AA  MM      MM    SSSSSS
              II      00      00    SSSSSS    UU      UU  BBBBBBBB    RRRRRRRR    AA      AA  MM      MM    SSSSSS
              II      00      00          SS  UU      UU  BB      BB  RR  RR      AAAAAAAAAA  MM      MM          SS
              II      00      00          SS  UU      UU  BB      BB  RR  RR      AAAAAAAAAA  MM      MM          SS
              II      00      00          SS  UU      UU  BB      BB  RR    RR    AA      AA  MM      MM          SS      ....
              II      00      00          SS  UU      UU  BB      BB  RR    RR    AA      AA  MM      MM          SS      ....
            IIIIII      000000    SSSSSSSS    UUUUUUUUUU  BBBBBBBB    RR      RR  AA      AA  MM      MM  SSSSSSSS        ....
            IIIIII      000000    SSSSSSSS    UUUUUUUUUU  BBBBBBBB    RR      RR  AA      AA  MM      MM  SSSSSSSS        ....

          LL            IIIIII      SSSSSSSS
          LL            IIIIII      SSSSSSSS
          LL              II      SS
          LL              II      SS
          LL              II      SS
          LL              II      SS
          LL              II        SSSSSS
          LL              II        SSSSSS
          LL              II              SS
          LL              II              SS
          LL              II              SS
          LL              II              SS
          LLLLLLLLLL    IIIIII    SSSSSSSS
          LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
IO
Ps

WI



Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
74
Th
40
23



Ma
--
_$
-$
TO

15

Th

MA
```

Table of contents

```
0000        .TITLE  IOSUBRAMS - NONPAGED RANDOM ACCESS MASS STORAGE I/O RELATED ROUTINES
0000        .IDENT  'V04-000'
0000
0000  ;
0000  ;****************************************************************************
0000  ;*                                                                          *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000  ;*  ALL RIGHTS RESERVED.                                                    *
0000  ;*                                                                          *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000  ;*  TRANSFERRED.                                                            *
0000  ;*                                                                          *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000  ;*  CORPORATION.                                                            *
0000  ;*                                                                          *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000  ;*                                                                          *
0000  ;*                                                                          *
0000  ;****************************************************************************
0000  ;
0000  ; AUTHOR: D. N. CUTLER 16-MAR-1977
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ;       V02-005 ACG0237         Andrew C. Goldstein,    9-Dec-1981  11:39
0000  ;               Add cathedral window support; check mapping against
0000  ;               file size in FCB
0000  ;
0000  ;       V02-004 GRR2004         Greg R. Robert,         14-Jun-1981
0000  ;               Added alternate entry point to IOC$CVTLOGPHY
0000  ;
0000  ;       V02-003 ACG0176         Andrew C. Goldstein,    6-Jun-1980  13:54
0000  ;               Redirect UCB to file's UCB on unsuccessful virtual map
0000  ;
0000  ;**
0000  ;
0000  ; NONPAGED RANDOM ACCESS MASS STORAGE I/O RELATED ROUTINES
0000  ;
0000  ; MACRO LIBRARY CALLS
0000  ;
0000
0000        $DEVDEF                         ;DEFINE DEVICE CHARACTERISTIC BITS
0000        $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
0000        $FCBDEF                         ;DEFINE FCB OFFSETS
0000        $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
0000        $IRPDEF                         ;DEFINE IRP OFFSETS
0000        $PRDEF                          ;DEFINE PROCESSOR REGISTERS
0000        $PTEDEF                         ;DEFINE PAGE TABLE ENTRY FIELDS
0000        $RVTDEF                         ;DEFINE RVT OFFSETS
```

```
        0000    58        $SSDEF                               ;DEFINE I/O STATUS CODES
        0000    59        $UCBDEF                              ;DEFINE UCB OFFSETS
        0000    60        $VADEF                               ;DEFINE VIRTUAL ADDRESS FIELDS
        0000    61        $WCBDEF                              ;DEFINE WCB OFFSETS
```

```
                                 0000          .SBTTL  APPLY ECC CORRECTION
                                 0000     ;+
                                 0000     ; IOC$APPLYECC - APPLY ECC CORRECTION
                                 0000     ;
                                 0000     ; THIS ROUTINE IS CALLED TO APPLY AN ECC CORRECTION TO DATA THAT HAS BEEN
                                 0000     ; TRANSFERED INTO MEMORY FROM A DISK DEVICE.
                                 0000     ;
                                 0000     ; INPUTS:
                                 0000     ;
                                 0000     ;        R0 = NUMBER OF BYTES OF DATA THAT WERE TRANSFERED UP TO, BUT NOT
                                 0000     ;                INCLUDING, BLOCK TO BE CORRECTED (MUST BE A MULTIPLE OF 512
                                 0000     ;                BYTES).
                                 0000     ;        R5 = DEVICE UNIT UCB ADDRESS.
                                 0000     ;
                                 0000     ;        UCB$W_BCNT(R5) = LENGTH OF TRANSFER IN BYTES.
                                 0000     ;        UCB$W_EC1(R5) = STARTING BIT NUMBER OF ERROR BURST.
                                 0000     ;        UCB$W_EC2(R5) = EXCLUSIVE OR CORRECTION PATTERN.
                                 0000     ;        UCB$L_SVAPTE(R5) = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE THAT MAPS
                                 0000     ;                THE TRANSFER.
                                 0000     ;
                                 0000     ; OUTPUTS:
                                 0000     ;
                                 0000     ;        THE CORRECTION PATTERN IS EXCLUSIVE OR'ED WITH THE DATA IN MEMORY
                                 0000     ;        PROVIDING THE NECESSARY CORRECTION.
                                 0000     ;
                                 0000     ;        R3 IS PRESERVED ACROSS CALL.
                                 0000     ;-
                                 0000
                             00000000          .PSECT  WIONONPAGED
                                 0000     IOC$APPLYECC::                              ;APPLY ECC CORRECTION
                     18      BB  0000          PUSHR   #^M<R3,R4>                     ;SAVE REGISTERS
         52    00C4  C5      3C  0002          MOVZWL  UCB$W_EC1(R5),R2               ;GET STARTING BIT NUMBER OF ERROR BURST
                     52      D7  0007          DECL    R2                             ;CONVERT TO RELATIVE BIT NUMBER
     51    52    F8  8F      8B  0009          BICB3   #^XF8,R2,R1                    ;ISOLATE PATTERN SHIFT COUNT
               52    08      C6  000E          DIVL    #8,R2                          ;CALCULATE RELATIVE BYTE OFFSET IN BLOCK
               52    50      C0  0011          ADDL    R0,R2                          ;CALCULATE RELATIVE OFFSET IN BUFFER
         50    00C6  C5      3C  0014          MOVZWL  UCB$W_EC2(R5),R0               ;GET EXCLUSIVE OR CORRECTION PATTERN
         50    50    51      78  0019          ASHL    R1,R0,R0                       ;SHIFT PATTERN TO PROPER POSITION
               54    03      D0  001D          MOVL    #3,R4                          ;SET LOOP COUNT
          7E A5      52      B1  0020     10$: CMPW    R2,UCB$W_BCNT(R5)              ;BYTE OFFSET WITHIN RANGE?
                     47      1E  0024          BGEQU   40$                            ;IF GEQU NO
           51    7C  A5      3C  0026          MOVZWL  UCB$W_BOFF(R5),R1              ;GET BYTE OFFSET IN PAGE
               51    52      C0  002A          ADDL    R2,R1                          ;CALCULATE BYTE OFFSET OF TRANFER PTE
     51    51    F7  8F      78  002D          ASHL    #-VA$S_BYTE,R1,R1              ;CALCULATE LONGWORD OFFSET TO TRANSFER PTE
         53    78 B541       D0  0032          MOVL    @UCB$L_SVAPTE(R5)[R1],R3       ;GET TRANSFER PTE
                     03      19  0037          BLSS    20$                            ;IF LSS VALID PTE
                  FFC4'      30  0039          BSBW    IOC$PTETOPFN                   ;CONVERT TO VALID PTE
     51    74  A5    04      C5  003C     20$: MULL3   #4,UCB$L_SVPN(R5),R1           ;CALCULATE BYTE OFFSET TO SYSTEM PTE
               00    53      F0  0041          INSV    R3,#PTE$V_PFN,-                ;MOVE TRANSFER PTE INTO SYSTEM PAGE TABLE
   00000000'FF41     15          0044                  #PTE$S_PFN,@MMG$GL_SPTBASE[R1]  ;
         51    51    07      78  004B          ASHL    #VA$S_BYTE-2,R1,R1             ;CONVERT SVPN TO SYSTEM VIRTUAL ADDRESS
            00 51    1F      E2  004F          BBSS    #VA$V_SYSTEM,R1,30$            ;SET SYSTEM VIRTUAL ADDRESS BIT
                                 0053     30$: INVALID R1                             ;INVALIDATE TRANSLATION BUFFER
     53    52    7C  A5      C1  0056          ADDL3   UCB$W_BOFF(R5),R2,R3           ;CALCULATE BYTE OFFSET IN BLOCK
   51    09    00    53      F0  005B          INSV    R3,#VA$V_BYTE,#VA$S_BYTE,R1 ;INSERT BYTE OFFSET IN BLOCK
               61    50      8C  0060          XORB    R0,(R1)                        ;CORRECT MEMORY BYTE
     50    50    F8  8F      78  0063          ASHL    #-8,R0,R0                      ;SHIFT NEXT CORRECTION BYTE INTO PLACE
```

```
             52   D6  0068   120          INCL     R2                          ;UPDATE OFFSET IN BUFFER
          B3 54   F5  006A   121          SOBGTR   R4,10$                      ;ANY MORE CORRECTIONS TO MAKE?
             18   BA  006D   122 40$:     POPR     #^M<R3,R4>                  ;RESTORE REGISTERS
    68 A5    01   A8  006F   123          BISW     #UCB$M_ECC,UCB$W_DEVSTS(R5) ;SET ECC CORRECTION MADE
                  05  0073   124          RSB                                  ;
```

```
                              0074         .SBTTL  CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
                              0074  ;+
                              0074  ; IOC$CVTLOGPHY - CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
                              0074  ;
                              0074  ; THIS ROUTINE IS CALLED TO CONDITIONALLY CONVERT A LOGICAL BLOCK NUMBER
                              0074  ; TO A PHYSICAL DISK ADDRESS AND STORE THE RESULT IN THE I/O PACKET.
                              0074  ;
                              0074  ; INPUTS:
                              0074  ;
                              0074  ;        R0 = LOGICAL BLOCK NUMBER TO BE CONVERTED.
                              0074  ;        R3 = I/O PACKET ADDRESS.
                              0074  ;        R5 = DEVICE UNIT UCB ADDRESS.
                              0074  ;
                              0074  ; OUTPUTS:
                              0074  ;
                              0074  ;        IF UCB$V_NOCNVRT IS CLEAR IN UCB$W_DEVSTS, THE LOGICAL BLOCK NUMBER
                              0074  ;        IS CONVERTED TO A PHYSICAL DISK ADDRESS USING THE DISK GEOMETRY PARA-
                              0074  ;        METERS IN THE UCB. THE RESULT IS STORED IN THE MEDIA ADDRESS LONGWORD
                              0074  ;        OF THE I/O PACKET.
                              0074  ;
                              0074  ;        IF UCB$V_NOCNVRT IS SET, THE BLOCK NUMBER IS STORED IN THE MEDIA ADDRESS
                              0074  ;        LONGWORD WITHOUT CONVERSION.
                              0074  ;
                              0074  ;        IF THE ROUTINE IS ENTERED AT IOC$CVTLOGPHYU, THEN UCB$V_NOCNVRT IS
                              0074  ;        IGNORED.
                              0074  ;
                              0074  ;        R3 IS PRESERVED ACROSS CALL.
                              0074  ;-
                              0074
                              0074         .ENABLE LOCAL_BLOCK
                              0074
                              0074  IOC$CVTLOGPHY::                                 ;CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
            38 A3     50   D0 0074         MOVL    R0,IRP$L_MEDIA(R3)              ;ASSUME NO CONVERSION
         1D 68 A5     02   E0 0078         BBS     #UCB$V_NOCNVRT,UCB$W_DEVSTS(R5),10$ ;BYPASS CONVERSION IF SET
                              007D  IOC$CVTLOGPHYU::                                ;UNCONDITIONAL ENTRY POINT
            52    44 A5    9A 007D         MOVZBL  UCB$L_DEVDEPEND(R5),R2          ;GET NUMBER OF SECTORS PER TRACK
                      51   D4 0081         CLRL    R1                              ;CLEAR HIGH PART OF DIVIDEND
   38 A3   50    50   52   7B 0083         EDIV    R2,R0,R0,IRP$L_MEDIA(R3)        ;CALCULATE SECTOR NUMBER AND STORE
            52    45 A5    9A 0089         MOVZBL  UCB$L_DEVDEPEND+1(R5),R2        ;GET NUMBER OF TRACKS PER CYLINDER
      51   50    50   52   7B 008D         EDIV    R2,R0,R0,R1                     ;CALCULATE TRACK AND CYLINDER
            39 A3     51   90 0092         MOVB    R1,IRP$L_MEDIA+1(R3)            ;STORE TRACK NUMBER
            3A A3     50   B0 0096         MOVW    R0,IRP$L_MEDIA+2(R3)            ;STORE CYLINDER NUMBER
                           05 009A  10$:   RSB                                     ;
                              009B
                              009B         .DISABLE LOCAL_BLOCK
                              009B
```

```
                     009B            .SBTTL  MAP VIRTUAL TO LOGICAL BLOCK
                     009B   ;+
                     009B   ; IOC$MAPVBLK - MAP VIRTUAL TO LOGICAL BLOCK
                     009B   ;
                     009B   ; THIS ROUTINE IS CALLED TO MAP A VIRTUAL BLOCK TO A LOGICAL BLOCK USING A
                     009B   ; MAPPING WINDOW.
                     009B   ;
                     009B   ; INPUTS:
                     009B   ;
                     009B   ;       R0 = VIRTUAL BLOCK NUMBER.
                     009B   ;       R1 = NUMBER OF BYTES TO MAP.
                     009B   ;       R2 = ADDRESS OF WINDOW MAPPING BLOCK.
                     009B   ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                     009B   ;
                     009B   ; OUTPUTS:
                     009B   ;
                     009B   ;       R0 LOW BIT CLEAR INDICATES A TOTAL MAPPING FAILURE.
                     009B   ;
                     009B   ;               R2 = NUMBER OF UNMAPPED BYTES.
                     009B   ;
                     009B   ;       R0 LOW BIT SET INDICATES PARTIAL MAP WITH:
                     009B   ;
                     009B   ;               R1 = LOGICAL BLOCK NUMBER OF FIRST BLOCK.
                     009B   ;               R2 = NUMBER OF UNMAPPED BYTES.
                     009B   ;               R5 = UCB ADDRESS OF DEVICE UNIT (POSSIBLY MODIFIED).
                     009B   ;
                     009B   ;       R3 IS PRESERVED ACROSS CALL.
                     009B   ;-
                     009B
                     009B           .ENABLE LOCAL_BLOCK
                     009B
                     009B   10$:    BUG_CHECK STRNOTWCB,FATAL       ;STRUCTURE IS NOT A WINDOW BLOCK
                     009F
                     009F   IOC$MAPVBLK::                           ;MAP VIRTUAL TO LOGICAL BLOCK
                     009F   20$:    DSBINT  UCB$B_FIPL(R5)          ;SYNCHRONIZE ACCESS TO SYSTEM DATABASE
    01DA 8F      BB  00A6           PUSHR   #^M<RT,R3,R4,R6,R7,R8>  ;SAVE REGISTERS
 57    FF A0     9E  00AA           MOVAB   -1(R0),R7               ;SAVE START VBN -1
    53    52     D0  00AE           MOVL    R2,R3                   ;GET COPY OF WINDOW ADDRESS
                     00B1   ;
                     00B1   ; THE WINDOW MAY CONSIST OF A CHAIN OF WCB SEGMENTS. SEARCH THROUGH THE
                     00B1   ; CHAIN UNTIL WE FIND ONE WHICH IS BEYOND THE DESIRED VBN OR WE REACH
                     00B1   ; THE END OF THE CHAIN.
                     00B1   ;
 12    0A A3     91  00B1   30$:    CMPB    WCB$B_TYPE(R3),#DYN$C_WCB ;SEE IF THIS IS REALLY A WINDOW
          E4     12  00B5           BNEQ    10$                     ;IF NEQ NO
 2C A3    50     D1  00B7           CMPL    R0,WCB$L_STVBN(R3)      ;CHECK VBN AGAINST START VBN OF WINDOW
          09     1F  00BB           BLSSU   40$                     ;BRANCH IF VBN PRECEDES WINDOW
    52    53     D0  00BD           MOVL    R3,R2                   ;ELSE ADVANCE TO THIS WINDOW SEGMENT
 53    20 A2     D0  00C0           MOVL    WCB$L_LINK(R2),R3       ;LOOK AT NEXT WINDOW SEGMENT
          EB     12  00C4           BNEQ    30$                     ;BRANCH TO LOOK AT IF IT EXISTS
                     00C6
 53    16 A2     3C  00C6   40$:    MOVZWL  WCB$W_NMAP(R2),R3       ;GET COUNT OF RETRIEVAL POINTERS
          1B     13  00CA           BEQL    60$                     ;BRANCH IF EMPTY WINDOW
 54    2C A2     DE  00CC           MOVAL   WCB$L_STVBN(R2),R4      ;POINT TO STARTING VBN
    50    84     C2  00D0           SUBL    (R4)+,R0                ;SUBTRACT STARTING VBN FROM DESIRED
17 38 A5  05     E0  00D3           BBS     #DEV$V_SQD,UCB$L_DEVCHAR(R5),70$ ;IF SET, SEQUENTIAL DEVICE
          0D     1F  00D8           BLSSU   60$                     ;BRANCH IF VBN PRECEDES WINDOW
```

```
                        00DA
                        00DA      ;
                        00DA      ; SCAN THE WINDOW, SUBTRACTING THE COUNT FIELD OF EACH POINTER FROM THE
                        00DA      ; CURRENT RELATIVE BLOCK NUMBER.
                        00DA      ;
                        00DA
          51   84   3C  00DA      50$:    MOVZWL  (R4)+,R1                  ;GET COUNT FIELD OF RETRIEVAL POINTER
          50   51   C2  00DD              SUBL    R1,R0                     ;SUBTRACT FROM RELATIVE BLOCK NUMBER
               12   1F  00E0              BLSSU   80$                       ;BRANCH IF VBN LOCATED IN THIS POINTER
               84   D5  00E2              TSTL    (R4)+                     ;SKIP LBN FIELD OF POINTER
            F3 53   F5  00E4              SOBGTR  R3,50$                    ;LOOP THRU WINDOW
               50   D4  00E7      60$:    CLRL    R0                        ;VBN IS BEYOND WINDOW
     55     10 A2   D0  00E9              MOVL    WCB$L_ORGUCB(R2),R5       ;REDIRECT UCB TO VOLUME CONTAINING THE FILE
               5F   11  00ED              BRB     140$                      ;RETURN FAILURE
                        00EF
                        00EF      ;
                        00EF      ; DEVICE IS A SEQUENTIAL DEVICE. FIRST MAPPING POINTER CONTAINS THE UCB ADDRESS
                        00EF      ; OF THE CURRENT VOLUME THAT IS BEING PROCESSED. ALL BYTES ALWAYS MAP.
                        00EF      ;
                        00EF
          55   64   D0  00EF      70$:    MOVL    (R4),R5                   ;GET CURRENT VOLUME UCB ADDRESS
               55   11  00F2              BRB     120$                      ;
                        00F4
                        00F4      ;
                        00F4      ; FOUND THE RETRIEVAL POINTER CONTAINING THE STARTING VBN. R0 NOW
                        00F4      ; CONTAINS A NEGATIVE VALUE WHICH IS THE NUMBER OF BLOCKS BETWEEN
                        00F4      ; THE STARTING VBN AND THE END OF THE POINTER.
                        00F4      ;
                        00F4
          58   50   D0  00F4      80$:    MOVL    R0,R8                     ;SAVE # BLOCKS MAPPED PAST START VBN
          51   84   C0  00F7              ADDL    (R4)+,R1                  ;FIRST LBN BEYOND THIS POINTER
          50   51   C0  00FA              ADDL    R1,R0                     ;COMPUTE STARTING LBN
                        00FD
                        00FD      ;
                        00FD      ; IF THE NEXT RETRIEVAL POINTER IS CONTIGUOUS WITH THE ONE FOUND, ADD
                        00FD      ; IN ITS COUNT TO HANDLE THE CASE WHERE A TRANSFER SPANS TWO POINTERS.
                        00FD      ; NOTE THAT THE GREATEST NUMBER OF CONTIGUOUS POINTERS A TRANSFER CAN
                        00FD      ; SPAN IS TWO.
                        00FD      ;
                        00FD
               53   D7  00FD              DECL    R3                        ;SEE IF THERE IS ANOTHER POINTER
               0B   15  00FF              BLEQ    90$                       ;BRANCH IF NONE
          53   84   3C  0101              MOVZWL  (R4)+,R3                  ;GET COUNT OF NEXT RETRIEVAL POINTER
          64   51   D1  0104              CMPL    R1,(R4)                   ;SEE IF THE NEXT POINTER IS CONTIGUOUS
               03   12  0107              BNEQ    90$                       ;BRANCH IF NOT
          58   53   C2  0109              SUBL    R3,R8                     ;ADD TO # BLOCKS MAPPED (NEGATIVE)
                        010C
                        010C      ;
                        010C      ; EXTRACT THE LBN AND RVN COMPONENTS OF THE STARTING "LBN" AND SWITCH
                        010C      ; TO THE RIGHT UCB IF THIS IS A MULTI-VOLUME SET.
                        010C      ;
                        010C
51   50   18   00   EF  010C      90$:    EXTZV   #0,#24,R0,R1              ;EXTRACT LBN PART
50   50   08   18   EF  0111              EXTZV   #24,#8,R0,R0              ;EXTRACT RVN
               09   13  0116              BEQL    100$                      ;BRANCH IF NOT VOLUME SET
       56   1C A2   D0  0118              MOVL    WCB$L_RVT(R2),R6          ;GET RELATIVE VOLUME TABLE ADDR
     55   40 A640   D0  011C              MOVL    RVT$L_UCBLST-4(R6)[R0],R5 ;GET THE RIGHT UCB ADDRESS
```

```
                              0121
                              0121     ;
                              0121     ; CHECK THE RANGE OF VBN'S PROVIDED BY THE MAP POINTER AGAINST THE
                              0121     ; FILE SIZE RECORDED IN THE FCB. REDUCE IT IF THE FCB INDICATES A
                              0121     ; SMALLER FILE SIZE THAN THE WINDOW.
                              0121     ;
                              0121
       12 0B A2     02   E0   0121     100$:   BBS     #WCB$V_NOTFCP,WCB$B_ACCESS(R2),110$ ;SKIP CHECK IF NO FCB
         56    18 A2     D0   0126             MOVL    WCB$L_FCB(R2),R6            ;GET FCB ADDRESS
         57    38 A6     C2   012A             SUBL    FCB$L_FILESIZE(R6),R7       ;COMPUTE NEG BLOCKS PAST DESIRED VBN
                  B7     1E   012E             BGEQU   60$                         ;BRANCH IF VBN PAST END OF FILE
            58    57     D1   0130             CMPL    R7,R8                       ;COMPARE AGAINST BLOCKS MAPPED
                  03     1F   0133             BLSSU   110$                        ;BRANCH IF LESS MAPPED BY WINDOW
            58    57     D0   0135             MOVL    R7,R8                       ;ELSE LIMIT TO FILE SIZE
                              0138
                              0138     ;
                              0138     ; SEE IF THE ENTIRE TRANSFER IS MAPPED CONTIGUOUSLY.
                              0138     ;
                              0138
        00000000'EF      D6   0138     110$:   INCL    PMS$GL_HIT                  ;COUNT A WINDOW HIT
       58     58  09     78   013E             ASHL    #9,R8,R8                    ;CONVERT TO # BYTES MAPPED
                  05     1D   0142             BVS     120$                        ;BRANCH IF COUNT IS HUGE
            6E    58     C0   0144             ADDL    R8,(SP)                     ;SUBTRACT FROM BYTES DESIRED
                  02     18   0147             BGEQ    130$                        ;BRANCH IF NOT TOTAL MAP
                  6E     D4   0149     120$:   CLRL    (SP)                        ;ZERO INDICATES COMPLETE MAP
            50    01     D0   014B     130$:   MOVL    #SS$_NORMAL,R0              ;INDICATE SUCCESS
          01DC    8F     BA   014E     140$:   POPR    #^M<R2,R3,R4,R6,R7,R8>      ;RESTORE REGISTERS
                              0152             ENBINT                              ;ALLOW INTERRUPTS
                         05   0155             RSB                                 ;
                              0156
                              0156             .DISABLE LOCAL_BLOCK
```

```
                             0156          .SBTTL  UPDATE TRANSFER PARAMETERS
                             0156  ;+
                             0156  ; IOC$UPDATRANSP - UPDATE TRANSFER PARAMETERS
                             0156  ;
                             0156  ; THIS ROUTINE IS CALLED TO UPDATE THE TRANSFER PARAMETERS AFTER A DISK ERROR
                             0156  ; HAS BEEN DISCOVERED BUT GOOD DATA WAS TRANSFERED.
                             0156  ;
                             0156  ; INPUTS:
                             0156  ;
                             0156  ;       R0 = NUMBER OF BYTES OF DATA THAT WERE TRANSFERED (MUST BE A MULTIPLE
                             0156  ;               OF 512 BYTES).
                             0156  ;       R5 = DEVICE UNIT UCB ADDRESS.
                             0156  ;
                             0156  ;       UCB$W_BCNT(R5) = LENGTH OF TRANSFER IN BYTES.
                             0156  ;       UCB$W_DA(R5) = CURRENT SECTOR AND TRACK ADDRESS.
                             0156  ;       UCB$W_DC(R5) = CURRENT CYLINDER ADDRESS.
                             0156  ;       UCB$L_SVAPTE(R5) = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE THAT MAPS
                             0156  ;               THE TRANSFER.
                             0156  ;
                             0156  ; OUTPUTS:
                             0156  ;
                             0156  ;       THE NUMBER OF BYTES REMAINING TO BE TRANSFERED, THE SYSTEM VIRTUAL
                             0156  ;       ADDRESS OF THE NEXT PTE, AND THE CURRENT DISK ADDRESS OF THE TRANSFER
                             0156  ;       ARE UPDATED.
                             0156  ;
                             0156  ;       R3 IS PRESERVED ACROSS CALL.
                             0156  ;-
                             0156
                             0156  IOC$UPDATRANSP::                                 ;UPDATE TRANSFER PARAMETERS
      7E A5     50      A2   0156          SUBW    R0,UCB$W_BCNT(R5)       ;CALCULATE REMAINING BYTES TO TRANSFER
50    50    F9 8F       78   015A          ASHL    #-7,R0,R0               ;CALCULATE PTE LONGWORDS TO SKIP OVER
      78 A5     50      C0   015F          ADDL    R0,UCB$L_SVAPTE(R5)     ;UPDATE SYSTEM VIRTUAL ADDRESS OF NEXT PTE
         50     04      C6   0163          DIVL    #4,R0                   ;CALCULATE NUMBER OF SECTORS TRANSFERED
    00BC C5     50      80   0166          ADDB    R0,UCB$W_DA(R5)         ;UPDATE SECTOR ADDRESS
                             016B
                             016B  ;
                             016B  ; RIPPLE CARRY FROM SECTOR TO TRACK AND FROM TRACK TO CYLINDER
                             016B  ;
                             016B
00BC C5      44 A5      91   016B  10$:    CMPB    UCB$L_DEVDEPEND(R5),UCB$W_DA(R5) ;SECTOR OVERFLOW?
                1E      1A   0171          BGTRU   20$                     ;IF GTRU NO
00BC C5      44 A5      82   0173          SUBB    UCB$L_DEVDEPEND(R5),UCB$W_DA(R5) ;SUBTRACT OUT A TRACK
           00BD C5      96   0179          INCB    UCB$W_DA+1(R5)          ;INCREMENT TRACK ADDRESS
00BD C5      45 A5      91   017D          CMPB    UCB$L_DEVDEPEND+1(R5),UCB$W_DA+1(R5) ;TRACK OVERFLOW?
                E6      1A   0183          BGTRU   10$                     ;IF GTRU NO
00BD C5      45 A5      82   0185          SUBB    UCB$L_DEVDEPEND+1(R5),UCB$W_DA+1(R5) ;SUBTRACT OUT A CYLINDER
           00BE C5      B6   018B          INCW    UCB$W_DC(R5)            ;UPDATE CYLINDER ADDRESS
                DA      11   018F          BRB     10$                     ;
                        05   0191  20$:    RSB                             ;
                        0192          .SBTTL  SENSE DISK'S SIZE FDT ROUTINE
                        0192   ;+
                        0192   ; IOC$SENSEDISK - SENSE DISK'S SIZE FDT ROUTINE
                        0192   ;
                        0192   ; THIS ROUTINE IS THE STANDARD SENSEMODE/SENSECHAR FDT ROUTINE FOR
                        0192   ; DISK DEVICES.  IT OBTAINS THE DISK'S SIZE, IN LOGICAL BLOCKS, FROM THE
                        0192   ; UCB (UCB$L_MAXBLOCK) AND IMMEDIATELY COMPLETES THE I/O REQUEST WITH
                        0192   ; THE SECOND LONGWORD OF THE FINAL I/O STATUS EQUAL TO THE DISK'S SIZE.
                        0192   ;
                        0192   ; INPUTS:
                        0192   ;
                        0192   ;       R0 = SCRATCH.
                        0192   ;       R1 = SCRATCH.
                        0192   ;       R2 = SCRATCH.
                        0192   ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                        0192   ;       R4 = CURRENT PROCESS PCB ADDRESS.
                        0192   ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                        0192   ;       R6 = ADDRESS OF CCB.
                        0192   ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                        0192   ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                        0192   ;       R9 = SCRATCH.
                        0192   ;       R10 = SCRATCH.
                        0192   ;       R11 = SCRATCH.
                        0192   ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                        0192   ;
                        0192   ; OUTPUTS:
                        0192   ;
                        0192   ;       THE DISK'S SIZE, IN LOGICAL BLOCKS, IS OPTAINED FROM THE UCB
                        0192   ;       AND THE I/O IS COMPLETED WITH THE SECOND I/O STATUS LONGWORD
                        0192   ;       EQUAL TO THE DISK'S SIZE.
                        0192   ;-
                        0192
                        0192   IOC$SENSEDISK::                          ;SENSE DISK'S SIZE
51    00B0 C5    D0     0192          MOVL    UCB$L_MAXBLOCK(R5),R1   ;GET DISK'S SIZE IN LOGICAL BLOCKS
        50   01    3C     0197          MOVZWL  S^#SS$_NORMAL,R0        ;SET NORMAL COMPLETION STATUS
          FE63'    31     019A          BRW     EXE$FINISHIO            ;FINISH I/O OPERATION
                        019D
                        019D          .END
```

```
BUG$_STRNOTWCB                   ********   X    02
DEV$V_SQD                      = 00000005
DYN$C_WCB                      = 00000012
EXE$FINISHIO                     ********   X    02
FCB$L_FILESIZE                 = 00000038
IOC$APPLYECC                     00000000 RG     02
IOC$CVTLOGPHY                    00000074 RG     02
IOC$CVTLOGPHYU                   0000007D RG     02
IOC$MAPVBLK                      0000009F RG     02
IOC$PTETOPFN                     ********   X    02
IOC$SENSEDISK                    00000192 RG     02
IOC$UPDATRANSP                   00000156 RG     02
IRP$L_MEDIA                    = 00000038
MMG$GL_SPTBASE                   ********   X    02
PMS$GL_HIT                       ********   X    02
PR$_IPL                        = 00000012
PR$_TBIS                       = 0000003A
PTE$S_PFN                      = 00000015
PTE$V_PFN                      = 00000000
RVT$L_UCBLST                   = 00000044
SS$_NORMAL                     = 00000001
UCB$B_FIPL                     = 0000000B
UCB$L_DEVCHAR                  = 00000038
UCB$L_DEVDEPEND                = 00000044
UCB$L_MAXBLOCK                 = 000000B0
UCB$L_SVAPTE                   = 00000078
UCB$L_SVPN                     = 00000074
UCB$M_ECC                      = 00000001
UCB$V_NOCNVRT                  = 00000002
UCB$W_BCNT                     = 0000007E
UCB$W_BOFF                     = 0000007C
UCB$W_DA                       = 000000BC
UCB$W_DC                       = 000000BE
UCB$W_DEVSTS                   = 00000068
UCB$W_EC1                      = 000000C4
UCB$W_EC2                      = 000000C6
VA$S_BYTE                      = 00000009
VA$V_BYTE                      = 00000000
VA$V_SYSTEM                    = 0000001F
WCB$B_ACCESS                   = 0000000B
WCB$B_TYPE                     = 0000000A
WCB$L_FCB                      = 00000018
WCB$L_LINK                     = 00000020
WCB$L_ORGUCB                   = 00000010
WCB$L_RVT                      = 0000001C
WCB$L_STVBN                    = 0000002C
WCB$V_NOTFCP                   = 00000002
WCB$W_NMAP                     = 00000016

                                                +----------------+
                                                ! 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
```

```
WIONONPAGED                    0000019D  (  413.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE


                                         +------------------------+
                                         ! Performance indicators !
                                         +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     29    00:00:00.07    00:00:00.96
Command processing                120    0C:00:00.53    00:00:07.69
Pass 1                            372    00:00:12.88    00:00:54.55
Symbol table sort                   0    00:00:02.17    00:00:08.04
Pass 2                             91    00:00:02.37    00:00:08.01
Symbol table output                 7    00:00:00.08    00:00:00.08
Psect synopsis output               2    00:00:00.03    00:00:00.04
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              623    00:00:18.13    00:01:19.46

The working set limit was 1500 pages.
74962 bytes (147 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1452 non-local and 21 local symbols.
406 source lines were read in Pass 1, producing 14 object records in Pass 2.
23 pages of virtual memory were used to define 22 macros.

                                        +--------------------------+
                                        ! Macro library statistics !
                                        +--------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                        13
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      6
TOTALS (all libraries)                                19

1583 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

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

0376 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

IOSUBRAMS
LIS

IPCONTROL
LIS

IOSUBPAGD
LIS

LNMSUB
LIS

IOPERFORM
LIS

IOSUBNPAG
LIS

LINKVEC
LIS