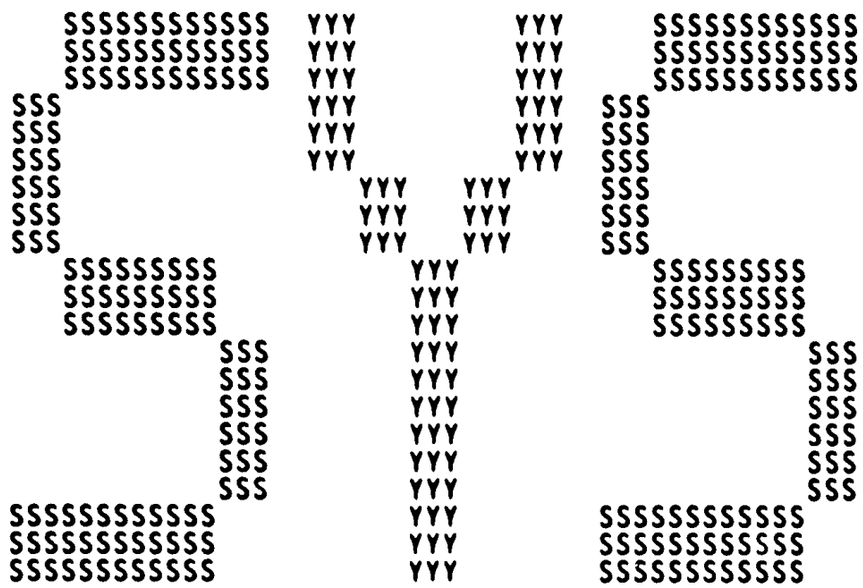

FILEID**SYSACPFDT

SYS
V04

```
  SSSSSSSS  YY      YY   SSSSSSSS    AAAAAA      CCCCCCCC  PPPPPPPP    FFFFFFFFFF  DDDDDDDD    TTTTTTTTTT
  SSSSSSSS  YY      YY   SSSSSSSS    AAAAAA      CCCCCCCC  PPPPPPPP    FFFFFFFFFF  DDDDDDDD    TTTTTTTTTT
SS          YY      YY  SS          AA    AA  CC          PP      PP  FF          DD      DD      TT
SS          YY      YY  SS          AA    AA  CC          PP      PP  FF          DD      DD      TT
SS            YY  YY    SS          AA    AA  CC          PP      PP  FF          DD      DD      TT
SS            YY  YY    SS          AA    AA  CC          PP      PP  FF          DD      DD      TT
  SSSSSS        YY        SSSSSS    AA    AA  CC          PPPPPPPP    FFFFFFFF    DD      DD      TT
  SSSSSS        YY        SSSSSS    AA    AA  CC          PPPPPPPP    FFFFFFFF    DD      DD      TT
        SS      YY              SS  AAAAAAAAAA  CC          PP          FF          DD      DD      TT
        SS      YY              SS  AAAAAAAAAA  CC          PP          FF          DD      DD      TT
        SS      YY              SS  AA    AA  CC          PP          FF          DD      DD      TT      ....
        SS      YY              SS  AA    AA  CC          PP          FF          DD      DD      TT      ....
SSSSSSSS        YY      SSSSSSSS    AA    AA    CCCCCCCC  PP          FF          DDDDDDDD        TT      ....
SSSSSSSS        YY      SSSSSSSS    AA    AA    CCCCCCCC  PP          FF          DDDDDDDD        TT      ....

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
```

Table of contents

```
0000     1          .TITLE  SYSACPFDT - ACP FUNCTION DECISION TABLE ACTION ROUTINES
0000     2          .IDENT  'V04-001'
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  ; D. N. CUTLER 22-NOV-76
0000    29  ;
0000    30  ; MODIFIED BY:
0000    31  ;
0000    32  ;       V04-001 ACG0467         Andrew C. Goldstein,    12-Sep-1984  21:45
0000    33  ;               Fix protection holes in QIO device protection check
0000    34  ;
0000    35  ;       V03-029 ACG0438         Andrew C. Goldstein,    2-Aug-1984  19:38
0000    36  ;               Fix field mask in XQP$UNLOCK_CACHE, remove code that
0000    37  ;               flushes caches when block lock is released. Add dismount
0000    38  ;               in progress check in mount check logic. Add error checks
0000    39  ;               on calls to SCH$QAST in XQP routines.
0000    40  ;
0000    41  ;       V03-028 ACG0438         Andrew C. Goldstein,    25-Jul-1984  11:09
0000    42  ;               Add XQP$UNLOCK_CACHE subroutine for cache flusher
0000    43  ;
0000    44  ;       V03-027 ACG0422         Andrew C. Goldstein,    23-Apr-1984  15:23
0000    45  ;               Further corrections to erase QIO processing with non-zero
0000    46  ;               erase patterns.
0000    47  ;
0000    48  ;       V03-026 CDS0003         Christian D. Saether     1-May-1984
0000    49  ;               Add blocking routine XQP$FCBSTALE.
0000    50  ;
0000    51  ;       V03-025 ACG0421         Andrew C. Goldstein,    20-Apr-1984  14:29
0000    52  ;               Fix segment byte count limiting in erase QIO's;
0000    53  ;               re-enable use of DSE function in MSCP disks.
0000    54  ;
0000    55  ;       V03-024 EMD0071         Ellen M. Dusseault      6-Apr-1984
0000    56  ;               Allocate and initialize a buffer for new parameter,
0000    57  ;               encryption key (P4), in common code path of ACP$WRITEBLK
```

```
0000     58 ;                  and ACP$READBLK.
0000     59 ;
0000     60 ;          V03-023 RKS0023         RICK SPITZ              6-APR-1984
0000     61 ;                  Allow new attribute access mode to always be probed
0000     62 ;                  at Callers access mode.
0000     63 ;
0000     64 ;          V03-022 RKS0022         RICK SPITZ              23-MAR-1984
0000     65 ;                  Add functionality in BUILDACPBUF to recognize a user supplied
0000     66 ;                  attribute specifying the access mode to probe the remaining
0000     67 ;                  attributes in the buffer.
0000     68 ;
0000     69 ;          V03-021 ACG0400         Andrew C. Goldstein,    10-Mar-1984  1:24
0000     70 ;                  Add routines to release quota cache locks for cluster
0000     71 ;                  wide quota cache.
0000     72 ;
0000     73 ;          V03-020 LMP0206         L. Mark Pilant,         9-Mar-1984  8:31
0000     74 ;                  Insure that the access check is only done once per channel
0000     75 ;                  for physical and logical read/write requests.
0000     76 ;
0000     77 ;          V03-019 CDS0002         Christian D. Saether     2-Mar-1984
0000     78 ;                  Add support for WRITE_TURN.
0000     79 ;
0000     80 ;          V03-018 WMC0001         Wayne Cardoza           01-Nov-1983
0000     81 ;                  XQP$DEQBLOCKER needs 4 arguments in call to EXE$DEQ.
0000     82 ;
0000     83 ;          V03-017 CDS0001         Christian D. Saether    13-Oct-1983
0000     84 ;                  Add routines to allow xqp file system blocking.
0000     85 ;                  XQP$BLOCK_ROUTINE and XQP$DEQBLOCKER.
0000     86 ;
0000     87 ;          V03-016 RPG0016         Bob Grosso              12-Sep-1983
0000     88 ;                  Back out branch around WCB refcnt checking so that
0000     89 ;                  deleted open known files will be closed with the
0000     90 ;                  WCB deallocated.
0000     91 ;
0000     92 ;          V03-015 ROW0218         Ralph O. Weber           7-SEP-1983
0000     93 ;                  Change maximum byte count, UCB$L_MAXBCNT, tests to be
0000     94 ;                  unsigned.
0000     95 ;
0000     96 ;          V03-014 ROW0212         Ralph O. Weber          20-AUG-1983
0000     97 ;                  Change mechanism used to disable use of the MSCP erase
0000     98 ;                  function to one which allow its use to be restored with a
0000     99 ;                  simple patch.  Also change MXDESCR from 5+17 to 5+30 at Andy's
0000    100 ;                  request.
0000    101 ;
0000    102 ;          V03-013 ROW0198         Ralph O. Weber          29-JUL-1983
0000    103 ;                  Temporarily disable use of the MSCP erase function for
0000    104 ;                  IO$M_ERASE requests because the HSC does not support MSCP
0000    105 ;                  erase.
0000    106 ;
0000    107 ;          V03-012 LJK0226         Lawrence J. Kenah       6-Jul-1983
0000    108 ;                  Temporarily disable decrement of REFCNT in shared windows.
0000    109 ;
0000    110 ;          V03-011 ROW0192         Ralph O. Weber          21-JUN-1983
0000    111 ;                  Fix ACP$WRITEBLK and ACP$READBLK to allow longword byte
0000    112 ;                  counts.  This should allow virtual disk transfers to exceed
0000    113 ;                  65K bytes.
0000    114 ;
```

```
0000    115 ;       V03-010 STJ3105         Steven T. Jeffreys,     17-Jun-1983
0000    116 ;               - Return SS$_ILLIOFUNC for virtual erase to tapes.
0000    117 ;
0000    118 ;       V03-009 STJ3099         Steven T. Jeffreys,     03-May-1983
0000    119 ;               - Enable erase I/O to tape devices.
0000    120 ;               - Greatly improve resource utilization for nonzero erase I/O.
0000    121 ;
0000    122 ;       V03-008 LJK0199         Lawrence J. Kenah       26-Apr-1983
0000    123 ;               Do not credit FILCNT quota when DEACCESS is performed
0000    124 ;               on a shared window. No charge was made on ACCESS path.
0000    125 ;
0000    126 ;       V03-007 MSH53232        Maryann Hinden          29-Apr-1983
0000    127 ;               Test count, not address, in CHKDESCR.
0000    128 ;
0000    129 ;       V03-006 STJ3086         Steven T. Jeffreys,     13-Apr-1983
0000    130 ;               - Performance enhancement to setup_erase made possible
0000    131 ;                 by change to IOCIOPOST (STJ3085).
0000    132 ;
0000    133 ;       V03-005 RLRMXBCNTa      Robert L. Rappaport     21-Mar-1983
0000    134 ;               Correct bug in original addition to allow segmentation
0000    135 ;               of Logical I/O.
0000    136 ;
0000    137 ;       V03-004 RLRMXBCNT       Robert L. Rappaport     11-Mar-1983
0000    138 ;               Allow for segmentation of Logical I/O (and Virtual)
0000    139 ;               based on the UCB$L_MAXBCNT field.
0000    140 ;
0000    141 ;       V03-003 STJ3059         Steven T. Jeffreys      02-Feb-1983
0000    142 ;               Temporary disable of ERASE to tapes to resolve I/O
0000    143 ;               function modifier collision.
0000    144 ;
0000    145 ;       V03-002 STJ3048         Steven T. Jeffreys      06-Jan-1983
0000    146 ;               Added support for erase qio.
0000    147 ;
0000    148 ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    149 ;               Added $SSDEF.
0000    150 ;
0000    151 ;**
0000    152 ;
0000    153 ; ACP FUNCTION DECISION TABLE ACTION ROUTINES
0000    154 ;
0000    155 ; MACRO LIBRARY CALLS
0000    156 ;
0000    157
0000    158         $ACBDEF                         ;DEFINE ACB SYMBOLS.
0000    159         $ATRDEF                         ;DEFINE ACP FILE ATTRIBUTES
0000    160         $CCBDEF                         ;DEFINE CCB OFFSETS
0000    161         $DCDEF                          ;DEFINE DEVICE CLASS CODES
0000    162         $DDTDEF                         ;DEFINE DDT OFFSETS
0000    163         $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS BITS
0000    164         $DYNDEF                         ;DEFINE STRUCTURE TYPE CODES
0000    165         $FCBDEF                         ;DEFINE FCB OFFSETS
0000    166         $FIBDEF                         ;DEFINE FIB FIELDS AND CODES
0000    167         $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
0000    168         $IODEF                          ;DEFINE I/O FUNCTION CODES
0000    169         $IRPDEF                         ;DEFINE IRP OFFSETS
0000    170         $JIBDEF                         ;DEFINE JIB OFFSETS
0000    171         $LCKDEF                         ;DEFINE LOCK MANAGER SYMBOLS
```

```
                     0000   172          $PCBDEF                          ;DEFINE PCB OFFSETS
                     0000   173          $PRDEF                           ;DEFINE PROCESSOR REGISTERS
                     0000   174          $PRTDEF                          ;DEFINE PAGE PROTECTION CODES
                     0000   175          $PRVDEF                          ;DEFINE PRIVILEGE BITS
                     0000   176          $PTEDEF                          ;DEFINE PTE FORMAT
                     0000   177          $RVTDEF                          ;DEFINE ROOT VOLUME TABLE OFFSETS
                     0000   178          $SSDEF                           ;DEFINE SYSTEM STATUS CODES
                     0000   179          $UCBDEF                          ;DEFINE UCB OFFSETS
                     0000   180          $VADEF                           ;DEFINE VIRTUAL ADDRESS FIELDS
                     0000   181          $VCADEF                          ;DEFINE VOLUME CACHE OFFSETS
                     0000   182          $VCBDEF                          ;DEFINE VCB OFFSETS
                     0000   183          $WCBDEF                          ;DEFINE WCB OFFSETS
                     0000   184
                     0000   185 ;
                     0000   186 ; LOCAL SYMBOLS
                     0000   187 ;
                     0000   188 ; ARGUMENT LIST OFFSET DEFINITIONS
                     0000   189 ;
                     0000   190
00000000             0000   191 P1=0                                     ;FIRST FUNCTION DEPENDENT PARAMETER
00000004             0000   192 P2=4                                     ;SECOND FUNCTION DEPENDENT PARAMETER
00000008             0000   193 P3=8                                     ;THIRD FUNCTION DEPENDENT PARAMETER
0000000C             0000   194 P4=12                                    ;FOURTH FUNCTION DEPENDENT PARAMETER
00000010             0000   195 P5=16                                    ;FIFTH FUNCTION DEPENDENT PARAMETER
00000014             0000   196 P6=20                                    ;SIXTH FUNCTION DEPENDENT PARAMETER
                     0000   197
                     0000   198 ;
                     0000   199 ; MAXIMUM NUMBER OF ACP DESCRIPTORS
                     0000   200 ;
                     0000   201
00000023             0000   202 MXDESCR=5+30                             ;MAXIMUM NUMBER OF ACP DESCRIPTORS
```

```
                     0000   204          .SBTTL  ACCESS AND CREATE ACP FUNCTION PROCESSING
                     0000   205 ;+
                     0000   206 ; ACP$ACCESS - ACCESS AND CREATE ACP FUNCTION PROCESSING
                     0000   207 ; ACP$ACCESSNET - ACCESS (CONNECT) TO NETWORK FUNCTION PROCESSING
                     0000   208 ;
                     0000   209 ; ***TBS***
                     0000   210 ;
                     0000   211 ; INPUTS:
                     0000   212 ;
                     0000   213 ;       R0 = SCRATCH.
                     0000   214 ;       R1 = SCRATCH.
                     0000   215 ;       R2 = SCRATCH.
                     0000   216 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     0000   217 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     0000   218 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                     0000   219 ;       R6 = ADDRESS OF CCB.
                     0000   220 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                     0000   221 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                     0000   222 ;       R9 = SCRATCH.
                     0000   223 ;       R10 = SCRATCH.
                     0000   224 ;       R11 = SCRATCH.
                     0000   225 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                     0000   226 ;
                     0000   227 ; OUTPUTS:
                     0000   228 ;
                     0000   229 ; ***TBS***
                     0000   230 ;-
                     0000   231
                     0000   232          .ENABL  LSB
                     0000   233 BRMODIFY:                                        ;
          0087    31 0000   234          BRW     ACP$MODIFY                      ;
                     0003   235 ACP$ACCESS::                                    ;ACCESS AND CREATE ACP FUNCTION PROCESSING
F8 20 A3   06    E1 0003   236          BBC     #IO$V_ACCESS,IRP$W_FUNC(R3),BRMODIFY ; IF CLR, NOT ACCESS
                     0008   237 ACP$ACCESSNET::                                ;ACCESS (CONNECT) TO NETWORK FUNCTION PROCES
 50    00A4 8F    3C 0008   238          MOVZWL  #SS$_FILALRACC,R0               ;ASSUME FILE ALREADY ACCESSED
       04 A6    D5 000D   239          TSTL    CCB$L_WIND(R6)                  ;FILE ALREADY ACCESSED ON CHANNEL?
          4E    14 0010   240          BGTR    20$                             ;IF GTR PROCESS SECTION OPEN ON CHANNEL
          51    12 0012   241          BNEQ    30$                             ;IF NEQ FILE OPEN ON CHANNEL
       50    1C 3C 0014   242          MOVZWL  #SS$_EXQUOTA,R0                 ;ASSUME FILE QUOTA EXCEEDED
 51    0080 C4    D0 0017   243          MOVL    PCB$L_JIB(R4),R1                ;GET JIB ADDRESS
       30 A1    B5 001C   244          TSTW    JIB$W_FILCNT(R1)                ;FILE QUOTA EXCEEDED?
          44    15 001F   245          BLEQ    30$                             ;IF LEQ YES
          03F3  30 0021   246          BSBW    BUILDACPBUF                     ;BUILD ACP BUFFER
          05AB  30 0024   247          BSBW    CHKDISMOUNT                     ;CHECK VOLUME AND UPDATE TRANSACTION COUNT
 51    0080 C4    D0 0027   248          MOVL    PCB$L_JIB(R4),R1                ;GET JIB ADDRESS
       30 A1    B7 002C   249          DECW    JIB$W_FILCNT(R1)                ;UPDATE FILE ACCESS COUNT
       04 A6    D6 002F   250          INCL    CCB$L_WIND(R6)                  ;SET CHANNEL INTERLOCK
          72    11 0032   251          BRB     70$                             ;
```

```
                                                                                .SBTTL  DEACCESS ACP FUNCTION PROCESSING
                                                                        ;+
                                                                        ; ACP$DEACCESS - DEACCESS ACP FUNCTION PROCESSING
                                                                        ;
                                                                        ; ***TBS***
                                                                        ;
                                                                        ; INPUTS:
                                                                        ;
                                                                        ;       R0 = SCRATCH.
                                                                        ;       R1 = SCRATCH.
                                                                        ;       R2 = SCRATCH.
                                                                        ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                                                                        ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                                                        ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                                                                        ;       R6 = ADDRESS OF CCB.
                                                                        ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                                                                        ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                                                                        ;       R9 = SCRATCH.
                                                                        ;       R10 = SCRATCH.
                                                                        ;       R11 = SCRATCH.
                                                                        ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                                                        ;
                                                                        ; OUTPUTS:
                                                                        ;
                                                                        ; ***TBS***
                                                                        ;-
                                                                        ;
                                                                        ACP$DEACCESS::                                  ;DEACCESS ACP FUNCTION PROCESSING
                50    00AC 8F    3C    0034    281                      MOVZWL  #SS$_FILNOTACC,R0                       ;ASSUME FILE NOT ACCESSED
                  51    04 A6    D0    0039    282                      MOVL    CCB$L_WIND(R6),R1                       ;FILE ACCESSED ON CHANNEL?
                           21    14    003D    283                      BGTR    20$                                     ;IF GTR PROCESS SECTION OPEN ON CHANNEL
                           24    13    003F    284                      BEQL    30$                                     ;IF EQL NO FILE OPEN ON CHANNEL
                  0B A1    0C    93    0041    285                      BITB    #WCB$M_NOTFCP!WCB$M_SHRWCB,WCB$B_ACCESS(R1) ;NORMAL WINDOW?
                           21    12    0045    286                      BNEQ    40$                                     ;IF NEQ NO
                        03CD    30    0047    287 10$:                 BSBW    BUILDACPBUF                             ;BUILD ACP BUFFER
                      04 A6    D6    004A    288                      INCL    CCB$L_WIND(R6)                          ;SET CHANNEL INTERLOCK
                        0596    30    004D    289                      BSBW    UPTRANSCNT                              ;UPDATE TRANSACTION COUNT
                  0A A6    01    B1    0050    290                      CMPW    #1,CCB$W_IOC(R6)                        ;ANY OTHER I/O ON CHANNEL?
                           50    13    0054    291                      BEQL    70$                                     ;IF EQL NO
                      0A A6    B7    0056    292                      DECW    CCB$W_IOC(R6)                           ;ADJUST COUNT FOR DEACCESS
                  0C A6    53    D0    0059    293                      MOVL    R3,CCB$L_DIRP(R6)                       ;SAVE ADDRESS OF DEACCESS PACKET
                        FFA0'   31    005D    294                      BRW     EXE$QIORETURN                           ;
               50       026C 8F    3C    0060    295 20$:                 MOVZWL  #SS$_IVCHNLSEC,R0                       ;SET INVALID SECTION CHANNEL STATUS
                        FF98'   31    0065    296 30$:                 BRW     EXE$ABORTIO                             ;
            0C 0B A1    03    E1    0068    297 40$:                 BBC     #WCB$V_SHRWCB,WCB$B_ACCESS(R1),50$ ;IF CLR, NOT SHARED WINDOW
                      0E A1    B7    006D    298                      DECW    WCB$W_REFCNT(R1)                        ;ANY SHARERS REMAINING?
                           0F    12    0070    299                      BNEQ    60$                                     ;IF NEQ YES
            D0 0B A1    02    E1    0072    300                      BBC     #WCB$V_NOTFCP,WCB$B_ACCESS(R1),10$ ;IF CLR, NOT FCP WINDOW
                           08    11    0077    301                      BRB     60$                                     ;REJOIN COMMON CODE
                                       0079    302
               50       0080 C4    D0    0079    303 50$:                 MOVL    PCB$L_JIB(R4),R0                        ;GET JIB ADDRESS
                        30 A0    B6    007E    304                      INCW    JIB$W_FILCNT(R0)                        ;INCREMENT FILE COUNT
                      04 A6    D4    0081    305 60$:                 CLRL    CCB$L_WIND(R6)                          ;CLEAR WINDOW ADDRESS
                        50    01    3C    0084    306                      MOVZWL  #SS$_NORMAL,R0                          ;SET NORMAL COMPLETION STATUS
                           FF76'   31    0087    307                      BRW     EXE$FINISHIOC                           ;
```

```
                     008A   309          .SBTTL  DELETE AND MODIFY ACP FUNCTION PROCESSING
                     008A   310 ;+
                     008A   311 ; ACP$MODIFY - DELETE AND MODIFY ACP FUNCTION PROCESSING
                     008A   312 ;
                     008A   313 ; ***TBS***
                     008A   314 ;
                     008A   315 ; INPUTS:
                     008A   316 ;
                     008A   317 ;       R0 = SCRATCH.
                     008A   318 ;       R1 = SCRATCH.
                     008A   319 ;       R2 = SCRATCH.
                     008A   320 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     008A   321 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     008A   322 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                     008A   323 ;       R6 = ADDRESS OF CCB.
                     008A   324 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                     008A   325 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                     008A   326 ;       R9 = SCRATCH.
                     008A   327 ;       R10 = SCRATCH.
                     008A   328 ;       R11 = SCRATCH.
                     008A   329 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                     008A   330 ;
                     008A   331 ; OUTPUTS:
                     008A   332 ;
                     008A   333 ; ***TBS***
                     008A   334 ;-
                     008A   335
                     008A   336 ACP$MODIFY::                                ;DELETE AND MODIFY ACP FUNCTION PROCESSING
038A   30            008A   337          BSBW    BUILDACPBUF               ;BUILD ACP BUFFER
0547   30            008D   338          BSBW    CHKMOUNT                  ;CHECK VOLUME AND UPDATE TRANSACTION COUNT
  14   11            0090   339          BRB     70$
```

```
                     0092   341          .SBTTL  MOUNT ACP FUNCTION PROCESSING
                     0092   342 ;+
                     0092   343 ; ACP$MOUNT - MOUNT ACP FUNCTION PROCESSING
                     0092   344 ;
                     0092   345 ; ***TBS***
                     0092   346 ;
                     0092   347 ; INPUTS:
                     0092   348 ;
                     0092   349 ;       R0 = SCRATCH.
                     0092   350 ;       R1 = SCRATCH.
                     0092   351 ;       R2 = SCRATCH.
                     0092   352 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     0092   353 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     0092   354 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                     0092   355 ;       R6 = ADDRESS OF CCB.
                     0092   356 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                     0092   357 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                     0092   358 ;       R9 = SCRATCH.
                     0092   359 ;       R10 = SCRATCH.
                     0092   360 ;       R11 = SCRATCH.
                     0092   361 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                     0092   362 ;
                     0092   363 ; OUTPUTS:
                     0092   364 ;
                     0092   365 ; ***TBS***
                     0092   366 ;-
                     0092   367
                     0092   368 ACP$MOUNT::                                 ;MOUNT ACP FUNCTION PROCESSING
           0382   30 0092   369          BSBW    BUILDACPBUF                ;BUILD ACP BUFFER
      50    24   3C 0095   370          MOVZWL  #SS$_NOPRIV,R0             ;ASSUME INSUFFICIENT PROCESS PRIVILEGE
                     0098   371          IFNPRIV MOUNT,90$                  ;PROCESS HAVE PRIVILEGE TO MOUNT VOLUME?
06 64 A5    09   E5 009E   372          BBCC    #UCB$V_MOUNTING,UCB$W_STS(R5),80$ ;IF CLR, NOT MOUNTING DEVICE
           0540   30 00A3   373          BSBW    UPTRANSCNT                 ;UPDATE VOLUME TRANSACTION COUNT
           FF57'  31 00A6   374 70$:     BRW     EXE$QIOACPPKT              ;QUEUE ACP PACKET
 50    007C 8F   3C 00A9   375 80$:     MOVZWL  #SS$_DEVNOTMOUNT,R0        ;SET DEVICE NOT MOUNTED STATUS
           FF4F'  31 00AE   376 90$:     BRW     EXE$ABORTIO                ;ABORT I/O OPERATION
                  00B1   377          .DSABL  LSB
```

```
                     00B1   379          .SBTTL  READ AND WRITE BLOCK ACP FUNCTION PROCESSING
                     00B1   380 ;+
                     00B1   381 ; ACP$READBLK - READ BLOCK ACP FUNCTION PROCESSING
                     00B1   382 ; ACP$WRITEBLK - WRITE BLOCK ACP FUNCTION PROCESSING
                     00B1   383 ;
                     00B1   384 ; ***TBS***
                     00B1   385 ;
                     00B1   386 ; INPUTS:
                     00B1   387 ;
                     00B1   388 ;       R0 = SCRATCH.
                     00B1   389 ;       R1 = SCRATCH.
                     00B1   390 ;       R2 = SCRATCH.
                     00B1   391 ;       R3 = ADDRESS OF I/O REQUEST PACKET.
                     00B1   392 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                     00B1   393 ;       R5 = ASSIGNED DEVICE UCB ADDRESS.
                     00B1   394 ;       R6 = ADDRESS OF CCB.
                     00B1   395 ;       R7 = I/O FUNCTION CODE BIT NUMBER.
                     00B1   396 ;       R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
                     00B1   397 ;       R9 = SCRATCH.
                     00B1   398 ;       R10 = SCRATCH.
                     00B1   399 ;       R11 = SCRATCH.
                     00B1   400 ;       AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                     00B1   401 ;
                     00B1   402 ; OUTPUTS:
                     00B1   403 ;
                     00B1   404 ; ***TBS***
                     00B1   405 ;-
                     00B1   406
                     00B1   407          .ENABL  LSB
                     00B1   408 ACP$READBLK::                                ;READ BLOCK ACP FUNCTION PROCESSING
          59   02    D0   00B1   409          MOVL    #CCB$V_RDCHKDON,R9     ;CCB BIT FOR READ PROT CHECK
     5A   0000'CF    9E   00B4   410          MOVAB   W^EXE$READLOCK,R10     ;SET ADDRESS OF READ CHECK AND LOCK ROUTINE
5B   00000000'EF    9E   00B9   411          MOVAB   EXE$CHKRDACCES,R11     ;SET ADDRESS OF ACCESS CHECK ROUTINE
   49 38 A5    1E    E1   00C0   412          BBC     S^#DEV$V_RCK,UCB$L_DEVCHAR(R5),20$ ;IF CLR, NO DEVICE READ CHECK
 20 A3    4000 8F    A8   00C5   413 10$:     BISW    #IO$M_DATACHECK,IRP$W_FUNC(R3) ;SET DATA CHECK ENABLE
             41    11   00CB   414          BRB     20$                    ;
                     00CD   415 ;
                     00CD   416 ; ZERO LENGTH TRANSFER
                     00CD   417 ;
                     00CD   418
          50   01    3C   00CD   419 13$:     MOVZWL  #SS$_NORMAL,R0         ;SET NORMAL COMPLETION
               FF2D'  31   00D0   420          BRW     EXE$FINISHIOC          ;FINISH I/O OPERATION
                     00D3   421 ;
                     00D3   422 ; DEVICE SOFTWARE WRITE LOCKED
                     00D3   423 ;
                     00D3   424
     50   025C 8F    3C   00D3   425 14$:     MOVZWL  #SS$_WRITLCK,R0        ;SET WRITE LOCK ERROR
               00F2  31   00D8   426          BRW     70$
                     00DB   427
                     00DB   428 ;
                     00DB   429 ; RECORD LENGTH TOO SHORT FOR MAGTAPE TRANSFER
                     00DB   430 ;
                     00DB   431
          50   14    3C   00DB   432 15$:     MOVZWL  #SS$_BADPARAM,R0       ;SET BAD PARAMETER STATUS
               00EC  31   00DE   433          BRW     70$                    ;
                     00E1   434
                     00E1   435
```

```
                                00E1   436 ;
                                00E1   437 ; SET DEVICE LEVEL WRITE CHECK
                                00E1   438 ;
                                00E1   439
                  4000 8F   A8  00E1   440 16$:    BISW    #IO$M_DATACHECK,-        ;SET DATA CHECK ENABLE
                    20 A3       00E5   441                 IRP$W_FUNC(R3)           ;
                        11  11  00E7   442          BRB     18$                      ;REJOIN COMMON CODE
                                00E9   443
                                00E9   444 ;
                                00E9   445 ; SET UP FOR CALL TO ERASE REQUEST PROCESSING.  IT'S DONE
                                00E9   446 ; THIS WAY TO MAKE THE LEAST IMPACT ON THE EXISTING CODE,
                                00E9   447 ; AND TO AVOID THE LENGTH CHECKS ON THE BYTE TRANSFER COUNT.
                                00E9   448 ;
                                00E9   449
                 5A   0302'CF  9E  00E9   450 17$:    MOVAB   W^SETUP_ERASE,R10        ;SET ADDRESS OF SERVICE ROUTINE
                        31  11  00EE   451          BRB     30$                      ;REJOIN COMMON CODE
                                00F0   452
                                00F0   453
                                00F0   454 ACP$WRITEBLK::                            ;WRITE BLOCK ACP FUNCTION PROCESSING
                DE 38 A5    19  E0  00F0   455          BBS     S^#DEV$V_SWL,UCB$L_DEVCHAR(R5),14$ ;IF SET, SOFTWARE WRITE LOCKED
                E7 38 A5    1F  E0  00F5   456          BBS     S^#DEV$V_WCK,UCB$L_DEVCHAR(R5),16$ ;IF SET, DEVICE LEVEL WRITE CHECK
                    59      03  D0  00FA   457 18$:    MOVL    #CCB$V_WRTCHKDON,R9      ;CCB BIT FOR WRITE PROT CHECK
             5B   00000000'EF  9E  00FD   458          MOVAB   EXE$CHKWRTACCES,R11      ;SET ADDRESS OF ACCESS CHECK ROUTINE
                        0A  E0  0104   459          BBS     #IO$V_ERASE,-            ;BRANCH IF ERASE REQUEST
                  E0 20 A3      0106   460                  IRP$W_FUNC(R3),17$       ;
                 5A   0000'CF  9E  0109   461          MOVAB   W^EXE$WRITELOCK,R10      ;SET ADDRESS OF WRITE CHECK AND LOCK ROUTINE
                    50      6C  D0  010E   462 20$:    MOVL    P1(AP),R0                ;GET STARTING ADDRESS OF TRANSFER
                  51    04  AC  D0  0111   463          MOVL    P2(AP),R1                ;GET NUMBER OF BYTES TO TRANSFER (LONGWORD)
                        B6  13  0115   464          BEQL    13$                      ;IF EQL ZERO LENGTH TRANSFER
                05 38 A5    05  E1  0117   465          BBC     S^#DEV$V_SQD,UCB$L_DEVCHAR(R5),30$ ;IF CLR, NOT SEQUENTIAL DEVICE
                    51      0E  D1  011C   466          CMPL    #14,R1                   ;RECORD TOO SHORT?
                        BA  14  011F   467          BGTR    15$                      ;IF GTR YES
                        6A  16  0121   468 30$:    JSB     (R10)                    ;CHECK BUFFER AND LOCK IN MEMORY
                50    08  AC  D0  0123   469          MOVL    P3(AP),R0                ;GET STARTING MED!A ADDRESS
                51    32  A3  D0  0127   470          MOVL    IRP$L_BCNT(R3),R1        ;RETRIEVE TRANSFER BYTE COUNT
                44 A3       51  D0  012B   471          MOVL    R1,IRP$L_OBCNT(R3)       ;SET ORIGNINAL BYTE COUNT
                  40 A3     D4  012F   472          CLRL    IRP$L_ABCNT(R3)          ;CLEAR ACCUMULATED BYTE COUNT
             31 20 A3       0D  E0  0132   473          BBS     #IO$V_ENCRYPT,IRP$W_FUNC(R3),36$    ; Set, key for encryption
                    06      00  ED  0137   474 33$:    CMPZV   #IRP$V_FCODE,#IRP$S_FCODE,- ;LOGICAL OR PHYSICAL I/O FUNCTION?
                2F    20 A3       013A   475                  IRP$W_FUNC(R3),#IO$_LOGICAL ;
                        7D  15  013D   476          BLEQ    39$                      ;IF LEQ YES
                52    18  A3  D0  013F   477          MOVL    IRP$L_WIND(R3),R2        ;FILE ACCESSED ON CHANNEL?
                        7E  13  0143   478          BEQL    50$                      ;IF EQL NO
             77 2A A3       01  E0  0145   479          BBS     #IRP$V_FUNC,IRP$W_STS(R3),41$ ;IF SET, READ FUNCTION
                                014A   480
                                014A   481 ;
                                014A   482 ; CHECK WRITE ACCESS
                                014A   483 ;
                                014A   484
             78 0B A2       01  E1  014A   485          BBC     #WCB$V_WRITE,WCB$B_ACCESS(R2),60$ ;IF CLR, NO WRITE ACCESS ALLOWED
             04 38 A5       05  E1  014F   486          BBC     S^#DEV$V_SQD,UCB$L_DEVCHAR(R5),35$ ;IF CLR, NOT SEQUENTIAL DEVICE
                0B A2       01  8A  0154   487          BICB    #WCB$M_READ,WCB$B_ACCESS(R2) ;DON'T ALLOW FURTHER READS
                      28 A2 D6  0158   488 35$:    INCL    WCB$L_QRITES(R2)         ;COUNT THE WRITE OPERATION
             5E 14 A2       0C  E0  015B   489          BBS     #WCB$V_WRITE_TURN, WCB$W_ACON (R2), 40$ ; FORCE WINDOW TURN
             7D 14 A2       05  E0  0160   490          BBS     #WCB$V_WRITECK,WCB$W_ACON(R2),90$ ;IF SET, WRITE DATA CHECK ENABLED
                      0080  31  0165   491          BRW     100$                     ;
                                0168   492 ;
```

```
                                0168   493 ; Encryption.  The key size must be less than or equal to 96 bytes.  This is
                                0168   494 ;              a security check to ensure that the user does not gobble up
                                0168   495 ;              pool.
                                0168   496 ;
                                0168   497 36$:
                        3B   BB 0168   498          PUSHR   #^M<R0,R1,R3,R4,R5>             ; Preserve contents of registers
                51   0C AC   D0 016A   499          MOVL    P4(AP), R1                      ; r1 contains address of descriptor
                     FEBF'   30 016E   500          BSBW    EXE$PROBER_DSC                  ; check read accessability
                    44   50   E9 0171   501          BLBC    R0,38$                          ; clear, error condition
            51   0060   8F   B1 0174   502          CMPW    #96, R1                         ; size must be leq 96 bytes
                        3A   19 0179   503          BLSS    37$                             ; less than, size to large
                51      51   3C 017B   504          MOVZWL  R1,R1                           ; zero extend the size
                7E      51   7D 017E   505          MOVQ    R1,-(SP)                        ; save descriptor
                51      08   C0 0181   506          ADDL    #8, R1                          ; Account for header on buffer
             00000000'GF     16 0184   507          JSB     G^EXE$ALLOCBUF                  ; Allocate nonpaged pool buffer
                53      8E   7D 018A   508          MOVQ    (SP)+,R3                        ; restore descriptor
                  28    50   E9 018D   509          BLBC    R0, 38$                         ; Clear, error condition
                53      04   C2 0190   510          SUBL2   #4, R3                          ; Subtract 4 because of flags
                51      0C   C2 0193   511          SUBL2   #12, R1                         ; R1 size of allocated buffer-header
                62      64   D0 0196   512          MOVL    (R4),(R2)                       ; Move flags field into buffer
                52      DD   0199   513          PUSHL   R2                              ; save address of buffer
OC A2   51   00   04 A4   53   2C 019B   514          MOVC5   R3,4(R4),#0,R1,12(R2)           ; Move key into buffer
                52 8ED0   01A3   515          POPL    R2                              ; restore address of buffer
                3B   BA   01A6   516          POPR    #^M<R0,R1,R3,R4,R5>             ; Restore register contents
             5C A3   52   D0 01A8   517          MOVL    R2,IRP$L_KEYDESC(R3)            ; store address of buffer in IRP
          2A A3   8000   8F   A8 01AC   518          BISW    #IRP$M_KEY, IRP$W_STS(R3)       ; Set flag in status field
                  FF82   31   01B2   519          BRW     33$
                         01B5   520
                50   14   D0 01B5   521 37$:     MOVL    #SS$_BADPARAM, R0                       ; set error status
                     3E   BA 01B8   522 38$:     POPR    #^M<R1,R2,R3,R4,R5>             ; Restore register contents
                     11   11 01BA   523          BRB     70$                             ; Abort this io
                         01BC   524
                         01BC   525 ;
                         01BC   526 ; EXTENDED BRANCH TO ACCESS CHECK.
                         01BC   527 ;
                         01BC   528
                   5E    11 01BC   529 39$:     BRB     120$
                         01BE   530
                         01BE   531 ;
                         01BE   532 ; EXTENDED BRANCH TO FORCED WINDOW TURN CODE.
                         01BE   533 ;
                         01BE   534
                 0123    31 01BE   535 40$:     BRW     220$
                         01C1   536
                         01C1   537 ;
                         01C1   538 ; EXTENDED BRANCH TO CHECK READ ACCESS
                         01C1   539 ;
                   12    11 01C1   540 41$:     BRB     80$
                         01C3   541
                         01C3   542 ;
                         01C3   543 ; FILE NOT ACCESSED ON CHANNEL
                         01C3   544 ;
                         01C3   545
50   00AC   8F   3C 01C3   546 50$:     MOVZWL  #SS$_FILNOTACC,R0               ;SET FILE NOT ACCESSED
                03   11 01C8   547          BRB     70$                             ;
                         01CA   548
                         01CA   549 ;
```

```
                                01CA   550 ; PRIVILEGE VIOLATION
                                01CA   551 ;
                                01CA   552
                     50    24   3C   01CA   553 60$:     MOVZWL  #SS$_NOPRIV,R0          ;SET NO PRIVILEGE
                        FE30'   31   01CD   554 70$:     BRW     EXE$ABORTIO             ;
                     5E    08   C0   01D0   555 75$:     ADDL    #8,SP                   ;CLEAN TEMPS OFF STACK
                           F8   11   01D3   556           BRB     70$                     ;AND EXIT
                                01D5   557
                                01D5   558 ;
                                01D5   559 ; CHECK READ ACCESS
                                01D5   560 ;
                                01D5   561
                  F0 0B A2      00   E1   01D5   562 80$:     BBC     #WCB$V_READ,WCB$B_ACCESS(R2),60$ ;IF CLR, READ ACCESS NOT ALLOWED
                           24 A2      D6   01DA   563           INCL    WCB$L_READS(R2)         ;COUNT THE READ OPERATION
                  06 14 A2      09   E1   01DD   564           BBC     #WCB$V_READCK,WCB$W_ACON(R2),100$ ;IF CLR, NO READ CHECK
               20 A3    4000 8F   A8   01E2   565 90$:     BISW    #IO$M_DATACHECK,IRP$W_FUNC(R3) ;SET DATA CHECK ENABLE
                  2A A3      10   A8   01E8   566 100$:    BISW    #IRP$M_VIRTUAL,IRP$W_STS(R3) ;SET VIRTUAL I/O FUNCTION
                  48 A3      50   D0   01EC   567           MOVL    R0,IRP$L_SEGVBN(R3)     ;SAVE STARTING VIRTUAL BLOCK NUMBER
                  4C A3    2C A3   D0   01F0   568           MOVL    IRP$L_SVAPTE(R3),IRP$L_DIAGBUF(R3) ;COPY STARTING PTE ADDRESS
                             25   A2   01F5   569           SUBW    #IO$_WRITEVBLK-IO$_WRITEPBLK,- ;CONVERT TO PHYSICAL I/O FUNCTION
                           20 A3        01F7   570                   IRP$W_FUNC(R3)          ;
                                01F9   571           SETIPL  UCB$B_FIPL(R5)          ;RAISE TO DRIVER FORK LEVEL
                        FE00'   30   01FD   572           BSBW    IOC$MAPVBLK             ;MAP VIRTUAL BLOCK TO LOGICAL BLOCK
                     1C A3      55   D0   0200   573           MOVL    R5,IRP$L_UCB(R3)        ;STORE MODIFIED UCB IN I/O PACKET
                        50      51   D0   0204   574           MOVL    R1,R0                   ;COPY STARTING LOGICAL BLOCK NUMBER
               51    44 A3      52   C3   0207   575           SUBL3   R2,IRP$L_OBCNT(R3),R1   ;CALCULATE LENGTH OF SEGMENT
                     32 A3      51   D0   020C   576           MOVL    R1,IRP$L_BCNT(R3)       ;STORE IN I/O PACKET
                                7D   13   0210   577           BEQL    160$                    ;IF EQL COMPLETE MAP FAILURE
                  6A 38 A5      05   E0   0212   578           BBS     S^#DEV$V_SQD,UCB$L_DEVCHAR(R5),140$ ;IF CLR, NOT SEQUENTIAL DEVICE
                        59      50   7D   0217   579           MOVQ    R0,R9                   ;COPY LBN AND COUNT FOR USE BELOW
                                33   11   021A   580           BRB     130$                    ;
                                021C   581
                     38 A3      50   D0   021C   582 120$:    MOVL    R0,IRP$L_MEDIA(R3)      ;SAVE MEDIA ADDRESS
                        7E      50   7D   0220   583           MOVQ    R0,-(SP)                ;SAVE TRANSFER PARAMETERS
               0A 08 A6       59   E0   0223   584           BBS     R9,CCB$B_STS(R6),125$   ;XFER IF CHECK DONE ALREADY
                                0228   585
                                0228   586 ; R4 - PCB ADDRESS
                                0228   587 ; R5 - UCB ADDRESS
                                0228   588
                                6B   16   0228   589           JSB     (R11)                   ;CHECK FOR PROPER ACCESS RIGHTS
                           A3 50      E9   022A   590           BLBC    R0,75$                  ;IF LBC PRIVILEGE VIOLATION
               00 08 A6       59   E2   022D   591           BBSS    R9,CCB$B_STS(R6),125$   ;MARK PROT CHECK DONE
                        59      8E   7D   0232   592 125$:    MOVQ    (SP)+,R9                ; GET BACK TRANSFER PARAMETERS
                        06      00   ED   0235   593           CMPZV   #IRP$V_FCODE,#IRP$S_FCODE,- ;PHYSICAL I/O FUNCTION?
                     1F    20 A3        0238   594                   IRP$W_FUNC(R3),#IO$_PHYSICAL ;
                                4A   15   023B   595           BLEQ    150$                    ;IF LEQ YES
                                15   A2   023D   596           SUBW    #IO$_WRITELBLK-IO$_WRITEPBLK,- ;CONVERT TO PHYSICAL I/O FUNCTION
                           20 A3        023F   597                   IRP$W_FUNC(R3)          ;
                     48 A3      59   D0   0241   598           MOVL    R9,IRP$L_SEGVBN(R3)     ; Prepare for possible segmentation of
                                0245   599                                                   ;  Logical I/O function by setting
                        2C A3   D0   0245   600           MOVL    IRP$L_SVAPTE(R3),-      ;  up Starting segmented LBN and
                        4C A3        0248   601                   IRP$L_DIAGBUF(R3)       ;  by copying the starting PTE address.
               32 38 A5       05   E0   024A   602           BBS     S^#DEV$V_SQD,UCB$L_DEVCHAR(R5),140$ ;IF SET, SEQUENTIAL DEVICE
                50    00B4 C5      D0   024F   603 130$:    MOVL    UCB$L_MAXBCNT(R5),R0    ; R0 = 0 or Max. permissible BCNT.
                                05   12   0254   604           BNEQ    133$                    ; NEQ implies Max. permissible BCNT in R0.
                50    FE00 8F      3C   0256   605           MOVZWL  #512*127,R0             ; If 0, use default Max. permissible.
                                025B   606 133$:
```

```
                32 A3    50   D1  025B   607          CMPL    R0,IRP$L_BCNT(R3)       ; See if BCNT too large.
                         07   1E  025F   608          BGEQU   137$                    ; GEQU implies we are OK.
                32 A3    50   D0  0261   609          MOVL    R0,IRP$L_BCNT(R3)       ; Else scale down to maximum allowed.
                   5A    50   D0  0265   610          MOVL    R0,R10                  ; Also update R10 copy of BCNT if changed.
                                 0268   611 137$:
                   5A    D7   0268   612          DECL    R10                     ;ROUND BYTE COUNT DOWN
5A   5A    F7 8F    78   026A   613          ASHL    #-VA$S_BYTE,R10,R10     ;CONVERT TO BLOCK COUNT
                   5A    59   C0   026F   614          ADDL    R9,R10                  ;CALCULATE ENDING BLOCK NUMBER
                         2B   1F   0272   615          BCS     170$                    ;IF CS ILLEGAL BLOCK NUMBER
            00B0 C5      5A   D1   0274   616          CMPL    R10,UCB$L_MAXBLOCK(R5)  ;LEGAL BLOCK NUMBER?
                         24   1E   0279   617          BGEQU   170$                    ;IF GEQU NO
                   50    59   D0   027B   618          MOVL    R9,R0                   ;RETRIEVE STARTING MEDIA ADDRESS
                      FD7F' 30   027E   619          BSBW    IOC$CVTLOGPHY           ;CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
                   1800 8F    AA   0281   620 140$:   BICW    #IO$M_INHERLOG!IO$M_INHSEEK,- ;CLEAR INHIBIT ERROR LOGGING
                     20 A3           0285   621                  IRP$W_FUNC(R3)          ;AND EXPLICIT SEEK
                         0A   E0   0287   622 150$:   BBS     #IO$V_ERASE,-           ;BRANCH IF ERASE REQUEST
                   22 20 A3        0289   623                  IRP$W_FUNC(R3),200$     ;
                      FD71' 31   028C   624 155$:   BRW     EXE$QIODRVPKT           ;QUEUE I/O PACKET TO DRIVER
                32 A3    52   D0   028F   625 160$:   MOVL    R2,IRP$L_BCNT(R3)       ;SET REQUESTED BYTE COUNT
                52    18 A3   D0   0293   626 165$:   MOVL    IRP$L_WIND(R3),R2       ;GET WINDOW ADDRESS
             0A 0B A2    02   E0   0297   627          BBS     #WCB$V_NOTFCP,WCB$B_ACCESS(R2),180$ ;IF SET, NOT FCP WINDOW
                      FD61' 31   029C   628          BRW     EXE$QIOACPPKT           ;QUEUE ACP PACKET
                50   00DC 8F    3C   029F   629 170$:   MOVZWL  #SS$_ILLBLKNUM,R0      ;SET ILLEGAL BLOCK NUMBER STATUS
                         05   11   02A4   630          BRB     190$
                50   0870 8F    3C   02A6   631 180$:   MOVZWL  #SS$_ENDOFFILE,R0      ;SET END OF FILE STATUS
                      FD52' 31   02AB   632 190$:   BRW     EXE$FINISHIOC           ;FINISH I/O OPERATION
                                 02AE   633          ;
                                 02AE   634          ;
                                 02AE   635          ; SEE IF THE FUNCTION CODE SHOULD BE CHANGED TO IO$_DSE.
                                 02AE   636          ; THIS IS DONE FOR ALL TAPE DEVICES, AND ALL OTHER DEVICES
                                 02AE   637          ; THAT HAVE SPECIAL HARDWARE SUPPORT (INDICATED BY A SVAPTE
                                 02AE   638          ; OF ZERO).  AFTER CHANGING THE CODE, CHECK THE NEW FUNCTION
                                 02AE   639          ; CODE AGAINST THE DRIVER'S FDT TABLE, SINCE WE'VE BYPASSED
                                 02AE   640          ; THE NORMAL CHECK BY CHANGING THE FUNCTION CODE AT THIS LATE
                                 02AE   641          ; DATE.
                                 02AE   642          ;
                                 02AE   643          ;
                      02   91    02AE   644 200$:   CMPB    #DC$_TAPE,-             ;CHECK DEVICE CLASS
                   40 A5         02B0   645                  UCB$B_DEVCLASS(R5)      ;
                         18   13   02B2   646          BEQL    210$                    ;BRANCH IF TAPE
                   2C A3   D5   02B4   647          TSTL    IRP$L_SVAPTE(R3)        ;TEST SVAPTE ADDRESS
                         13   13   02B7   648          BEQL    210$                    ;BRANCH IF ZERO
50   0000FE00 8F    D0   02B9   649          MOVL    #127*512,R0             ;GET LIMIT FOR PSEUDO PAGE TABLE
                50   32 A3   D1   02C0   650          CMPL    IRP$L_BCNT(R3),R0       ;CHECK BYTE COUNT AGAINST LIMIT
                         C6   1B   02C4   651          BLEQU   155$                    ;BRANCH IF OK
                32 A3    50   D0   02C6   652          MOVL    R0,IRP$L_BCNT(R3)       ;SET COUNT TO LIMIT
                         C0   11   02CA   653          BRB     155$
                                 02CC   654          ;
                         15   F0   02CC   655 210$:   INSV    #IO$_DSE,-              ;CHANGE WRITE REQUEST TO ERASE
                         00        02CE   656                  #IRP$V_FCODE,-          ;
                         06        02CF   657                  #IRP$S_FCODE,-          ;
                     20 A3         02D0   658                  IRP$W_FUNC(R3)          ;
                50   0088 C5   D0   02D2   659          MOVL    UCB$L_DDT(R5),R0        ;GET THE DDT ADDRESS
                         15   E0   02D7   660          BBS     #IO$_DSE,-              ;BRANCH IF DSE SUPPORTED
                   B0 08 B0        02D9   661                  @DDT$L_FDT(R0),155$     ;
                50   00F4 8F    3C   02DC   662          MOVZWL  #SS$_ILLIOFUNC,R0       ;SET FAILURE STATUS
                      FD1C' 31   02E1   663          BRW     EXE$ABORTIO             ;ABORT THE I/O REQUEST
```

```
                                02E4    664
                                02E4    665 ;
                                02E4    666 ; FORCE WINDOW TURN.  THIS IS DONE TO ALLOW THE FILE SYSTEM TO INVALIDATE
                                02E4    667 ; FILE SYSTEM CACHE ENTRIES THAT CORRESPOND TO VIRTUAL WRITES ON FILE
                                02E4    668 ; SYSTEM FILES (INDEXF, BITMAP, DIRECTORIES).  THE WRITE_TURN FLAG WAS
                                02E4    669 ; SET BY THE FILE SYSTEM WHEN THE FILE WAS WRITE ACCESSED.
                                02E4    670 ; SOME CODE IS REPEATED HERE FROM ABOVE TO AVOID MUCKING UP THE NORMAL
                                02E4    671 ; (NON WRITE_TURN) PATH.
                                02E4    672 ;
                                02E4    673
  06 14 A2    05    E1    02E4    674 220$:   BBC     #WCB$V_WRITECK, WCB$W_ACON (R2), 230$ ; BR IF NO WRITE DATA CHK
20 A3    4000 8F    A8    02E9    675         BISW    #IO$M_DATACHECK, IRP$W_FUNC (R3) ; SET DATA CHECK
   2A A3    10    A8    02EF    676 230$:   BISW    #IRP$M_VIRTUAL, IRP$W_STS (R3) ; NOTE AS VIRTUAL I/O
   48 A3    50    D0    02F3    677         MOVL    R0, IRP$L_SEGVBN (R3)   ; SAVE STARTING VBN
 4C A3    2C A3    D0    02F7    678         MOVL    IRP$L_SVAPTE (R3), IRP$L_DIAGBUF (R3) ; SAVE STARTING PTE ADDR
   20 A3    0B    B0    02FC    679         MOVW    #IO$_WRITEPBLK, IRP$W_FUNC (R3) ; CONVERT TO PHYSICAL I/O FUNC
            91    11    0300    680         BRB     165$                    ; AND OFF TO QUEUE THE PACKET
                        0302    681
                        0302    682         .DISABL LSB
```

```
0302   684          .SBTTL  SET UP ERASE REQUEST
0302   685 ;++
0302   686 ; SETUP_ERASE
0302   687 ;
0302   688 ; THE ERASE I/O REQUEST IS IMPLEMENTED AS A MODIFIED IO$_WRITExBLK REQUEST.
0302   689 ; THIS SCHEME ALLOWS THE ERASE FUNCTION TO BE USED AS A VIRTUAL, LOGICAL,
0302   690 ; OR PHYSICAL I/O REQUEST, AND HAS THE ADDITIONAL ADVANTAGE OF NOT REQUIRING
0302   691 ; EXISTING DISK DRIVERS TO RECODE THEIR FDT TABLES TO USE THE NEW I/O FUNCTION.
0302   692 ;
0302   693 ; THERE ARE SEVERAL WAYS TO PROCESS THE ERASE REQUEST, DEPENDING ON 1) THE SIZE
0302   694 ; OF THE TRANSFER, 2) THE ERASE PATTERN (ZERO OR NONZERO), AND THE LEVEL OF
0302   695 ; HARDWARE SUPPORT.  THE VARIOUS CASES ARE PRESENTED BELOW.  NOTE THAT FOR DEVICES
0302   696 ; THAT HAVE HARDWARE SUPPORT, IT IS NOT NECESSARY, IN SOME CASES, TO TRANSFER DATA
0302   697 ; FROM THE HOST TO THE DEVICE.  WE CAN TAKE ADVANTAGE OF THIS, AND NOT HAVE TO
0302   698 ; SET UP AND MAP AN ERASE PATTERN BUFFER.  WHEN THE HARDWARE SUPPORT IS NOT
0302   699 ; AVAILABLE OR PRACTICAL, THE ERASE IS PROCESSED IN A MANNER SIMILAR TO A WRITE
0302   700 ; REQUEST.  HOWEVER, TO MINIMIZE OVERHEAD, THE ERASE BUFFER IS ONLY 1 PAGE LONG,
0302   701 ; AND A SPECIAL PSUEDO PAGE TABLE (PPT) IS CREATE TO MAP THAT BUFFER.  EACH PTE
0302   702 ; IN THE PPT MAPS THE ERASE BUFFER.  THIS ALLOWS NORMAL DMA TRANSFER FROM THE
0302   703 ; HOST TO THE DEVICE.  SINCE MOST ERASE REQUESTS USE A PATTERN OF ZERO, AND ARE
0302   704 ; LESS THAN OR EQUAL TO 127 PAGES LONG, THE SYSTEM PREALLOCATES A PAGE-ALIGNED
0302   705 ; 512 BYTE ERASE PATTERN BUFFER (EPB) AND A ONE PAGE PPT TO MAP IT.  NOTE THAT
0302   706 ; ONE PAGE OF PTE'S WILL ALLOW TRANSFERS OF UP TO 127 PAGES.
0302   707 ;
0302   708 ;       CASE 0: TARGET DEVICE IS A 'TAPE CLASS' DEVICE
0302   709 ;
0302   710 ;                               THE ERASE QIO IS LEGAL ONLY FOR THOSE TAPE DEVICES THAT
0302   711 ;                               HAVE SPECIAL HARDWARE SUPPORT, NAMELY A DATA SECURITY
0302   712 ;                               ERASE (DSE) FUNCTION.  THIS FUNCTION WILL START AT THE
0302   713 ;                               CURRENT TAPE POSITION, ERASE UNTIL TEN FEET PAST THE EOT
0302   714 ;                               MARK, AND REWIND THE TAPE TO THE STARTING POSITION.  THE
0302   715 ;                               HARDWARE DOES NOT ACCEPT A LENGTH OR ERASE PATTERN ARGUEMENT
0302   716 ;                               SO BOTH P1 AND P2 (IN R0 AND R1) ARE IGNORED.  MAP A
0302   717 ;                               VIRTUAL WRITE TO A LOGICAL WRITE, SO THAT IMPLICIT TRANSFER
0302   718 ;                               LENGTH CHECKS ARE AVOIDED.  THE FUNCTION CODE WILL BE
0302   719 ;                               MAPPED TO IO$_DSE BY ACP$WRITEBLK BEFORE QUEUING THE
0302   720 ;                               REQUEST TO THE DRIVER.
0302   721 ;
0302   722 ;       CASE 1: PATTERN IS ZERO, HARDWARE SUPPORT, TRANSFER LENGTH IRRELEVENT
0302   723 ;                               IRP$L_SVAPTE =  0
0302   724 ;
0302   725 ;                               TO SIGNAL THE DRIVER TO USE THE HARDWARE ERASE
0302   726 ;                               FEATURE, THE FUNCTION CODE IS CHANGED TO IO$_DSE.
0302   727 ;                               HOWEVER, SINCE THE CALLER (ACP$WRITEBLK) WOULD
0302   728 ;                               FAIL IF THE FUNCTION CODE IS CHANGED PREMATURELY,
0302   729 ;                               IT IS NOT CHANGED UNTIL JUST BEFORE THE REQUEST IS
0302   730 ;                               QUEUED TO THE DRIVER.  FOR THIS TO HAPPEN, THE
0302   731 ;                               IO$M_ERASE BIT MUST BE SET, AND IRP$L_SVAPTE MUST
0302   732 ;                               BE ZERO.
0302   733 ;
0302   734 ;       CASE 2: PATTERN IS ZERO, NO HARDWARE SUPPORT, TRANSFER <= 127 PAGES
0302   735 ;                               IRP$L_SVAPTE =  ADDRESS OF PREALLOCATED PPT
0302   736 ;
0302   737 ;       CASE 3: PATTERN IS ZERO, NO HARDWARE SUPPORT, TRANSFER > 127 PAGES
0302   738 ;                               IRP$L_SVAPTE =  ADDRESS OF FIRST PTE WITHIN THE PPT
0302   739 ;                                               THAT WAS CREATED FROM POOL
0302   740 ;                               *** NOTE ***
```

```
0302   741 ;                          CASE 3 NOW COLLAPSES INTO CASE 2 WITH THE AUTOMATIC
0302   742 ;                          SEGMENTATION OF LOGICAL I/O REQUESTS.  NOTE THAT
0302   743 ;                          THE SVAPTE IS NOT ADVANCED DURING SEGMENTATION, WHICH
0302   744 ;                          ALLOWS THE SYSTEM PPT TO HANDLE TRANSFERS OF ANY LENGTH.
0302   745 ;
0302   746 ;          CASE 4: PATTERN IS NONZERO
0302   747 ;                          IRP$L_SVAPTE =  ADDRESS OF FIRST PTE WITHIN THE PPT
0302   748 ;                                          THAT MAPS THE EPB.  BOTH THE PPT AND
0302   749 ;                                          EPB ARE BUILT OUT OF THE SAME PIECE
0302   750 ;                                          OF POOL, WITH THE PPT STARTING DIRECTLY
0302   751 ;                                          AFTER THE 12 BYTE HEADER, AND THE EPB
0302   752 ;                                          STARTING ON THE FIRST PAGE BOUNDARY
0302   753 ;                                          BELOW THE END OF THE PPT.
0302   754 ;
0302   755 ; THE SYSTEM RESOURCE USAGE:
0302   756 ;
0302   757 ;          IF THE ERASE PATTERN IS ZERO, THEN NO SYSTEM RESOURCES ARE USED.
0302   758 ;          OTHERWISE, AN EPB AND PPT TO MAP IT MUST BE CREATED FROM SYSTEM
0302   759 ;          NONPAGED POOL.  THE COST IN POOL IS
0302   760 ;
0302   761 ;              COST = POOL_HEADER  + EPB  +  PPT
0302   762 ;              COST = 12           + 1024 +  (4*MIN(XFER_SIZE/512),128)
0302   763 ;              COST = 1548 BYTES OF POOL, WORST CASE.
0302   764 ;
0302   765 ; NOTE THAT THE I/O POST PROCESSING ROUTINE IS RESPONSIBLE FOR CLEANING
0302   766 ; UP AFTER THE ERASE I/O COMPLETES, AND WILL RETURN ALL BORROWED RESOURCES
0302   767 ; BACK TO THE SYSTEM.  THE PREALLOCATED EPB AND PPT ARE SHARED RESOURCES.
0302   768 ;
0302   769 ; INPUTS:
0302   770 ;
0302   771 ;          R0 = SCRATCH
0302   772 ;          R1 = SCRATCH
0302   773 ;          R2 = SCRATCH
0302   774 ;          R3 = ADDRESS OF I/O REQUEST PACKET
0302   775 ;          R4 = CURRENT PROCESS PCB ADDRESS
0302   776 ;          R5 = ADDRESS OF DEVICE UCB
0302   777 ;          R6 = ADDRESS OF CCB
0302   778 ;          R7 = I/O FUNCTION CODE BIT NUMBER
0302   779 ;          R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS
0302   780 ;          R9 = SCRATCH
0302   781 ;          R10= SCRATCH
0302   782 ;          R11= ADDRESS OF EXE$CHKWRTACCES
0302   783 ;          AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER
0302   784 ;          P1(AP) = ADDRESS OF ERASE PATTERN BUFFER
0302   785 ;          P2(AP) = NUMBER OF BYTES TO ERASE
0302   786 ;
0302   787 ; OUTPUT:
0302   788 ;
0302   789 ;          NONE.
0302   790 ;
0302   791 ; IMPLICIT INPUT:
0302   792 ;
0302   793 ;          EXE$GL_ERASEPPT : CONTAINS THE ADDRESS OF THE PREALLOCATED PPT
0302   794 ;          EXE$GL_ERASEPB  : CONTAINS THE ADDRESS OF THE PREALLOCATED EPB
0302   795 ;
0302   796 ; IMPLICIT OUTPUT:
0302   797 ;
```

```
                                0302   798 ;       - ALL NON-SCRATCH REGISTERS ARE PRESERVED
                                0302   799 ;       - IRP$L_SVAPTE(R3) IS ALTERED
                                0302   800 ;       - IRP$L_BCNT(R3) CONTAINS NUMBER OF BYTES TO TRANSFER
                                0302   801 ;--
                                0302   802
                                0302   803 SETUP_ERASE:                                ; PROCESS ERASE REQUEST
          50    6C    D0        0302   804         MOVL    P1(AP),R0                  ; GET THE ERASE PATTERN ADDRESS
        51    04 AC    D0        0305   805         MOVL    P2(AP),R1                  ; GET THE TRANSFER BYTE COUNT
           30 A3    B4           0309   806         CLRW    IRP$W_BOFF(R3)             ; ASSUME EPB IS PAGE-ALIGNED
           2C A3    D4           030C   807         CLRL    IRP$L_SVAPTE(R3)           ; ASSUME DEVICE HAS HARDWARE SUPPORT
                                030F   808 ;
                                030F   809 ; PEFORM A SANITY CHECK TO ENSURE THIS REQUEST IS TO A DISK OR TAPE.
                                030F   810 ; IF NOT, TRANSFER CONTROL TO THE NORMAL BUFFER CHECK ROUTINE.
                                030F   811 ;
                01    91        030F   812         CMPB    #DC$_DISK,-                ; IS THIS A DISK DEVICE?
           40 A5                0311   813                 UCB$B_DEVCLASS(R5)         ;
                 21    13        0313   814         BEQL    30$                        ; BRANCH IF SO
                02    91        0315   815         CMPB    #DC$_TAPE,-                ; IS THIS A TAPE DEVICE?
           40 A5                0317   816                 UCB$B_DEVCLASS(R5)         ;
                 12    12        0319   817         BNEQ    10$                        ; BRANCH IF NOT
           32 A3    D4           031B   818         CLRL    IRP$L_BCNT(R3)             ; NO DATA TO TRANSFER
                00    ED        031E   819         CMPZV   #IRP$V_FCODE,-              ; IS THIS A VIRTUAL WRITE?
                06               0320   820                 #IRP$S_FCODE,-         ;
           20 A3                0321   821                 IRP$W_FUNC(R3),-       ;
                30               0323   822                 #IO$_WRITEVBLK         ;
                36    12        0324   823         BNEQ    50$                        ; BRANCH IF NOT (RETURN)
      50    00F4 8F    3C        0326   824         MOVZWL  #SS$_ILLIOFUNC,R0          ; DO NOT ALLOW VIRTUAL ERASE TO TAPES
                40    11        032B   825         BRB     80$                        ;  (ERASE FORGETS TAPE POSITION)
                   FCD0'  31     032D   826 10$:     BRW     EXE$WRITELOCK          ; OTHERWISE USE 'NORMAL' ROUTINE
           50    01    3C        0330   827 20$:     MOVZWL  #SS$_NORMAL,R0         ; SET SUCCESS STATUS
                   FCCA'  31     0333   828         BRW     EXE$FINISHIOC          ; COMPLETE THE I/O REQUEST
                                0336   829 ;
                                0336   830 ; ROUND THE BYTE TRANSFER COUNT UP TO THE NEAREST MULTIPLE OF 512.
                                0336   831 ; THIS ENSURES THAT NO PART OF A DISK BLOCK WILL BE UNTOUCHED.
                                0336   832 ;
                51    D5        0336   833 30$:     TSTL    R1                         ; CHECK BYTE COUNT
                F6    13        0338   834         BEQL    20$                        ; BRANCH IF NO BYTES TO TRANSFER
      51    01FF C1    9E        033A   835         MOVAB   ^X1FF(R1),R1               ; ROUND UP TO NEXT PAGE
      51    01FF 8F    AA        033F   836         BICW    #^X1FF,R1                  ; TRUNCATE TRANSFER COUNT TO PAGE #
        32 A3    51    D0        0344   837         MOVL    R1,IRP$L_BCNT(R3)          ; SAVE R1
                                0348   838 ;
                                0348   839 ; DETECT AND PROCESS 'CASE 1' ERASE REQUESTS. THIS CASE IS HANDLED
                                0348   840 ; FIRST BECAUSE IT IS THE SIMPLEST TO PROCESS AND OCCURS MOST FREQUENTLY.
                                0348   841 ;
                50    D5        0348   842         TSTL    R0                         ; CHECK ERASE PATTERN ADDRESS
                09    13        034A   843         BEQL    40$                        ; BRANCH IF ZERO - ASSUME PATTERN = 0
                                034C   844         IFNORD  #4,(R0),70$                ; BRANCH IF NO READ ACCESS
           50    60    D0        0352   845         MOVL    (R0),R0                    ; GET ERASE PATTERN
                06    12        0355   846 40$:     BNEQ    60$                        ; BRANCH IF PATTERN NONZERO
                08    E1        0357   847         BBC     #DEV$V_RCT,-               ; BRANCH IF NOT MSCP DEVICE
        01 38 A5                0359   848                 UCB$L_DEVCHAR(R5),60$      ;  OTHERWISE WE'RE DONE (CASE 1)
                05               035C   849 50$:     RSB                            ; RETURN
                                035D   850 ;
                                035D   851 ; IF WE GET THIS FAR, IT MEANS THIS IS NOT A 'CASE 1' ERASE REQUEST.
                                035D   852 ; THIS MEANS THAT THERE IS NO HARDWARE SUPPORT (VERY LIKELY), AND/OR
                                035D   853 ; THE ERASE PATTERN IS NONZERO (VERY UNLIKELY).  THE PREALLOCATED
                                035D   854 ; EPB AND PPT WILL BE USED WHENEVER POSSIBLE.
```

```
                        035D   855 ;
00000000'GF   D0        035D   856 60$:    MOVL    G^EXE$GL_ERASEPPT,-     ; USE SYSTEM PPT - PUT ADDRESS IN IRP
        2C A3           0363   857                 IRP$L_SVAPTE(R3)        ;   SVAPTE NONZERO FOR ROBUSTNESS
           50   D5      0365   858         TSTL    R0                      ; IS ERASE PATTERN ZERO?
           07   12      0367   859         BNEQ    90$                     ; BRANCH IF NOT - DO IT THE HARD WAY
                05      0369   860         RSB                             ; RETURN - 'CASE 2' DONE
        50   0C   3C    036A   861 70$:    MOVZWL  #SS$_ACCVIO,R0          ; INDICATE ACCESS VIOLATION
          FC90'  31     036D   862 80$:    BRW     EXE$ABORTIO             ; ABORT THE I/O
                        0370   863 ;
                        0370   864 ; 'CASE 4' ERASE REQUESTS ARE HANDLED HERE.
                        0370   865 ; ALLOCATE A BLOCK OF POOL LARGE ENOUGH TO CONTAIN A PAGE-ALIGNED
                        0370   866 ; ERASE PATTERN BUFFER AND A PPT LARGE ENOUGH TO MAP IT.  SINCE
                        0370   867 ; THE SVAPTE IS NOT ADVANCED FOR ERASE I/O, AN ARBITRARILY LARGE
                        0370   868 ; ERASE REQUEST CAN BE MAPPED BY A ONE PAGE PPT.
                        0370   869 ; THERE IS AN ADDITIONAL 12 BYTES OF OVERHEAD FOR THE USUAL HEADER.
                        0370   870 ;
59   51   F9 8F   7B    0370   871 90$:    ASHL    #-7,R1,R9               ; CALC BYTES OF PTE'S IN PPT
  52   0200 8F   3C    0375   872         MOVZWL  #512,R2                 ; LOAD VALUE INTO REGISTER
       52   59   D1    037A   873         CMPL    R9,R2                   ; IS THE PPT BIGGER THAN ONE PAGE?
            03   15    037D   874         BLEQ    100$                    ; BRANCH IF NOT
       59   52   D0    037F   875         MOVL    R2,R9                   ; TRUNCATE PPT TO ONE PAGE
51   020C C249   9E    0382   876 100$:    MOVAB   12+512(R2)[R9],R1       ; CALC NUMBER OF BYTES IN PPT
            53   DD    0388   877         PUSHL   R3                       ; SAVE IRP ADDRESS
       5A   50   D0    038A   878         MOVL    R0,R10                  ; SAVE ERASE PATTERN
          FC70'  30     038D   879         BSBW    EXE$ALLOCBUF            ; ALLOCATE THE EPB
            53 8ED0     0390   880         POPL    R3                      ; RESTORE IRP ADDRESS
         D7 50   E9     0393   881         BLBC    R0,80$                  ; EXIT IF ERROR
 2C A3   0C A2   9E     0396   882         MOVAB   12(R2),IRP$L_SVAPTE(R3) ; SET ADDRESS OF FIRST PTE
                        039B   883 ;
                        039B   884 ; THE CHUNK OF POOL FOR THE EPB AND PPT IS ALLOCATED.
                        039B   885 ; CALCULATE THE ADDRESS OF THE EPB WITHIN THE CHUNK AND FILL IT IN.
                        039B   886 ; THEN FETCH THE PTE OF THE EPB AND FILL IN PPT.
                        039B   887 ;
  52   020B C249   9E    039B   888         MOVAB   12+511(R2)[R9],R2       ; CALC ADDRESS OF END OF THE PPT
    52   01FF 8F   AA    03A1   889         BICW    #VA$M_BYTE,R2           ; PAGE ALIGN THE BUFFER
    51   0200 8F   3C    03A6   890         MOVZWL  #512,R1                 ; SET SIZE OF EPB
         50   5A   D0    03AB   891         MOVL    R10,R0                  ; RESTORE PATTERN TO R0
              1C   10    03AE   892         BSBB    FILL_BUFFER             ; FILL ERASE BUFFER
         15   09   EF    03B0   893         EXTZV   #VA$V_VPN,#VA$S_VPN,-   ; EXTRACT VIRTUAL PAGE NUMBER
         50   52         03B3   894                 R2,R0                   ;
50 00000000'FF40   D0    03B5   895         MOVL    @MMG$GL_SPTBASE[R0],R0  ; GET PTE THAT MAPS THE EPB
         1B   0F   F0    03BD   896         INSV    #PRT$C_OR,#PTE$V_PROT,- ; MAKE PTE USER READABLE
         50   04         03C0   897                 #PTE$S_PROT,R0
         51   59   D0    03C2   898         MOVL    R9,R1                   ; SIZE OF PAGE TABLE
    52   2C A3   D0    03C5   899         MOVL    IRP$L_SVAPTE(R3),R2     ; GET ADDRESS OF FIRST PTE IN PPT
              01   10    03C9   900         BSBB    FILL_BUFFER             ; PROPAGATE PTE THROUGHOUT PPT
                   05   03CB   901         RSB                             ; RETURN
```

```
                                03CC    903          .SBTTL  FILL BUFFER (ERASE QIO)
                                03CC    904 ;++
                                03CC    905 ; FILL_BUFFER
                                03CC    906 ;
                                03CC    907 ;       THIS IS LOCAL SUBROUTINE USED BY THE SETUP ERASE ROUTINE TO
                                03CC    908 ;       PROPAGATE A 4 BYTE PATTERN THROUGHOUT A CONTIGUOUS RANGE OF
                                03CC    909 ;       BYTES.  THE NUMBER OF BYTES IS ASSUMED TO BE AN INTEGER NUMBER
                                03CC    910 ;       OF LONGWORDS.
                                03CC    911 ;
                                03CC    912 ; INPUT:
                                03CC    913 ;
                                03CC    914 ;       R0 = 4 BYTE PATTERN
                                03CC    915 ;       R1 = LENGTH OF BUFFER
                                03CC    916 ;       R2 = ADDRESS OF BUFFER
                                03CC    917 ;
                                03CC    918 ; OUTPUT:
                                03CC    919 ;
                                03CC    920 ;       NONE.
                                03CC    921 ;
                                03CC    922 ; SIDE EFFECTS:
                                03CC    923 ;
                                03CC    924 ;       THE BUFFER IS FILLED.
                                03CC    925 ;       ALL REGISTERS ARE PRESERVED.
                                03CC    926 ;       NO ROUTINE VALUE IS RETURNED.
                                03CC    927 ;--
                                03CC    928
                                03CC    929 FILL_BUFFER:                                ; PROPAGATE A LONGWORD INTO A BUFFER
                          3F    BB  03CC    930          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; SAVE REGISTERS
                          50    D5  03CE    931          TSTL    R0                      ; CHECK FOR A PATTERN OF 0
                          09    12  03D0    932          BNEQ    100$                    ; BRANCH IF NOT ZERO
                62  51    50    62  00    2C  03D2    933 10$:     MOVC5   #0,(R2),R0,R1,(R2)      ; DO IT THE FAST WAY
                          3F    BA  03D8    934 69$:     POPR    #^M<R0,R1,R2,R3,R4,R5>  ; RESTORE REGISTERS
                          05    03DA    935          RSB
                                03DB    936 ;
                                03DB    937 ; IF THE PATTERN IS SIGNIFICANT TO ONE BYTE, USE THE MOVC5.
                                03DB    938 ;
                    01  AE    50    91  03DB    939 100$:    CMPB    R0,1(SP)                ; ARE THE FIRST TWO BYTES IDENTICAL?
                          06    12  03DF    940          BNEQ    200$                    ; BRANCH IF NOT
                    02  AE    50    B1  03E1    941          CMPW    R0,2(SP)                ; ARE THE FIRST TWO WORDS IDENTICAL?
                          EB    13  03E5    942          BEQL    10$                     ; YES - USE THE MOVC5
                                03E7    943 ;
                                03E7    944 ; THE PATTERN CANNOT BE PROPAGATED VIA A MOVC INSTRUCTION.
                                03E7    945 ; ATTEMPT TO MOVE 32 BYTES AT A TIME VIA A MOVQ LOOP.
                                03E7    946 ; ALL REMAINING BYTES WILL BE MOVED A LONGWORD AT A TIME.
                                03E7    947 ;
                      53    51  D0    03E7    948 200$:    MOVL    R1,R3                   ; COPY SIZE OF BUFFER
                            54  D4    03EA    949          CLRL    R4                      ; CLEAR FOR EDIV
              54    53    53  20    7B  03EC    950          EDIV    #32,R3,R3,R4            ; R3 = # OF 32 BYTE CHUNKS, R4 = REM.
                54    54    FE  8F    78  03F1    951          ASHL    #-2,R4,R4               ; CONVERT R4 TO # OF LONGWORDS
                      51    50  D0    03F6    952          MOVL    R0,R1                   ; SET UP FOR QUADWORD MOVE
                            53  D5    03F9    953          TSTL    R3                      ; MAKE SURE THE BUFFER IS BIG ENOUGH
                            0F  13    03FB    954          BEQL    220$                    ; BRANCH IF NOT
                      82    50  7D    03FD    955 210$:    MOVQ    R0,(R2)+                ; MOVE A QUADWORD
                      82    50  7D    0400    956          MOVQ    R0,(R2)+                ; MOVE A QUADWORD
                      82    50  7D    0403    957          MOVQ    R0,(R2)+                ; MOVE A QUADWORD
                      82    50  7D    0406    958          MOVQ    R0,(R2)+                ; MOVE A QUADWORD
                        F1  53  F5    0409    959          SOBGTR  R3,210$                 ; DECREMENT COUNTER AND LOOP IF MORE
```

```
                                                N 8
                54   D5  040C   960 220$:   TSTL    R4                      ; CHECK REMAINDER
                C8   13  040E   961 230$:   BEQL    69$                     ; BRANCH IF NONE
            82  50   D0  0410   962         MOVL    R0,(R2)+                ; MOVE A LONGWORD
                54   D7  0413   963         DECL    R4                      ; DECREMENT COUNTER
                F7   11  0415   964         BRB     230$                    ; LOOP
```

```
                     0417   966          .SBTTL  BUILD ACP BUFFER
                     0417   967 ;
                     0417   968 ; SUBROUTINE TO BUILD ACP BUFFER
                     0417   969 ;
                     0417   970
                     0417   971          .ENABL  LSB
                     0417   972 BUILDACPBUF:                                ;BUILD ACP BUFFER
        5E    FEE8 CE    9E  0417   973          MOVAB   -MXDESCR*8(SP),SP       ;ALLOCATE SPACE FOR MAXIMUM DESCRIPTORS
            5B    5E    D0  041C   974          MOVL    SP,R11                  ;SET ADDRESS TO STORE DESCRIPTORS
            5A    10    D0  041F   975          MOVL    #16,R10                 ;SET INITIAL BYTE COUNT
            8B    04    D0  0422   976          MOVL    #4,(R11)+               ;INSERT WINDOW ADDRESS LENGTH AND ACCESS MOD
          8B    04 A6    DE  0425   977          MOVAL   CCB$L_WIND(R6),(R11)+   ;INSERT WINDOW ADDRESS
              0139    30  0429   978          BSBW    CHKDESCR                ;INSERT FIB DESCRIPTOR
            57    34    91  042C   979          CMPB    #IO$_DEACCESS,R7        ;IS OPERATION A DEACCESS
                  17    12  042F   980          BNEQ    3$                      ;IF NEQ, NO
          1C A3   66    D1  0431   981          CMPL    CCB$L_UCB(R6),IRP$L_UCB(R3) ;IS OPERATION FOR IMPLICIT SPOOLING
                  11    13  0435   982          BEQL    3$                      ;IF EQL, NO
            50    14    D0  0437   983          MOVL    #12+8,R0                ;SIZE OF USER NAME PLUS ACCOUNT
    51  00000000'EF    9E  043A   984          MOVAB   CTL$T_USERNAME,R1       ;ADDRESS OF THOSE IN P1 SPACE
              014F    30  0441   985          BSBW    UPBYTCNT                ;INSERT DESCRIPTOR FOR NAME AND ACCOUNT
                 8C   D5  0444   986          TSTL    (AP)+                   ;IGNORE FILE NAME ARGUMENT
                 03   11  0446   987          BRB     7$                      ;
              011A    30  0448   988 3$:      BSBW    CHKDESCR                ;INSERT NAME STRING DESCRIPTOR
            50    8C    D0  044B   989 7$:      MOVL    (AP)+,R0                ;GET ADDRESS TO STORE RESULT STRING LENGTH
                  0C    13  044E   990          BEQL    10$                     ;IF EQL NONE SPECIFIED
            51    50    D0  0450   991          MOVL    R0,R1                   ;SET ADDRESS OF RESULT LENGTH
            50    02    D0  0453   992          MOVL    #2,R0                   ;GET LENGTH OF RESULT LENGTH
                        0456   993          IFNOWRT R0,(R1),ACCVIO          ;CAN RESULT LENGTH BE WRITTEN?
              0134    30  045C   994 10$:     BSBW    UPBYTCNT                ;INSERT DESCRIPTOR AND UPDATE BYTE ACCUMULAT
              0103    30  045F   995          BSBW    CHKDESCR                ;INSERT RESULT STRING DESCRIPTOR
            59    1E    D0  0462   996          MOVL    #MXDESCR-5,R9           ;SET MAXIMUM NUMBER OF ATTRIBUTE DESCRIPTORS
            5C    6C    D0  0465   997          MOVL    (AP),AP                 ;GET ADDRESS OF ATTRIBUTE LIST
                  40    13  0468   998          BEQL    30$                     ;IF EQL NONE SPECIFIED
                        046A   999
                  51    D4  046A  1000          CLRL    R1                      ;INIT CURRENT ACCESS MODE (KERNEL)
                        046C  1001 20$:     IFNORD  #2,(AP),ACCVIO          ;CAN ATTRIBUTE LENGTH BE READ?
            8B    8C    B0  0472  1002          MOVW    (AP)+,(R11)+            ;INSERT ATTRIBUTE LENGTH
                  33    13  0475  1003          BEQL    30$                     ;IF EQL END OF ATTRIBUTE LIST
                        0477  1004          IFNORD  #6,(AP),ACCVIO          ;CAN REST OF ATTRIBUTE DESCRIPTOR BE READ?
            8B    8C    B0  047D  1005          MOVW    (AP)+,(R11)+            ;INSERT ATTRIBUTE NUMBER
            8B    8C    D0  0480  1006          MOVL    (AP)+,(R11)+            ;INSERT ATTRIBUTE ADDRESS
                        0483  1007
                  2D    B1  0483  1008          CMPW    #ATR$C_ACCESS_MODE,-    ;CHECK FOR CHANGE ACCESS MODE ATTRIBUTE
                FA AB         0485  1009                  -6(R11)
                  4C    13  0487  1010          BEQL    NEWMODE                 ;IT IS, SO SPECIAL CASE
                        0489  1011
                  0A    BB  0489  1012          PUSHR   #^M<R1,R3>              ;SAVE IRP ADDRESS AND CURRENT ACCESS MODE
            53    51    D0  048B  1013          MOVL    R1,R3                   ;SPECIFY MODE
          51    F8 AB   3C  048E  1014          MOVZWL  -8(R11),R1              ;GET LENGTH
          50    FC AB   D0  0492  1015          MOVL    -4(R11),R0              ;GET ADDRESS OF BUFFER
    00000000'EF         16  0496  1016          JSB     EXE$PROBEW              ;PROBE BUFFER
                  0A    BA  049C  1017          POPR    #^M<R1,R3>              ;RESTORE REGISTERS
                2F 50   E9  049E  1018          BLBC    R0,ACCVIO               ;IF LBC, NO WRITE ACCESS
                        04A1  1019
        5A    F8 AB   A0  04A1  1020 25$:     ADDW    -8(R11),R10             ;UPDATE BYTE ACCUMULATION
                  0E    1F  04A5  1021          BCS     35$                     ;IF CS ACCUMULATION OVERFLOW
                C2 59   F5  04A7  1022          SOBGTR  R9,20$                  ;ANY MORE ATTRIBUTES TO PROCESS?
```

```
                59   23   59   C3   04AA  1023 30$:     SUBL3   R9,#MXDESCR,R9          ;CALCULATE TOTAL NUMBER OF DESCRIPTORS
                5C   59   09   C5   04AE  1024           MULL3   #9,R9,AP               ;CALCULATE BYTES REQUIRED BY DESCRIPTORS
                     5A   5C   A0   04B2  1025           ADDW    AP,R10                ;UPDATE BYTE ACCUMULATION
                          3F   1F   04B5  1026 35$:      BCS     BRXQUOTA              ;IF CS ACCUMULATION OVERFLOW
                          53   DD   04B7  1027           PUSHL   R3                    ;SAVE I/O PACKET ADDRESS
                     51   5A   D0   04B9  1028           MOVL    R10,R1                ;SET NUMBER OF BYTES REQUIRED
                        FB41'  30   04BC  1029           BSBW    EXE$BUFFRQUOTA        ;CHECK BUFFER BYTE COUNT QUOTA
                      09 50    E9   04BF  1030           BLBC    R0,40$                ;IF LBC QUOTA EXCEEDED
                     51   5A   D0   04C2  1031           MOVL    R10,R1                ;SET NUMBER OF BYTES REQUIRED
                        FB38'  30   04C5  1032           BSBW    EXE$ALLOCBUF          ;ALLOCATE BUFFERED I/O BUFFER
                      2D 50    E8   04C8  1033           BLBS    R0,50$                ;IF LBS SUCCESSFUL ALLOCATION
                               04CB  1034
                               04CB  1035 ;
                               04CB  1036 ; QUOTA CHECK OR BUFFER ALLOCATION FAILURE
                               04CB  1037 ;
                               04CB  1038
                          08   BA   04CB  1039 40$:      POPR    #^M<R3>               ;RETRIEVE I/O PACKET ADDRESS
                        FB30'  31   04CD  1040 42$:      BRW     EXE$ABORTIO
                               04D0  1041 ;
                               04D0  1042 ; ACCESS VIOLATION ON DESCRIPTOR OR INFORMATION STRING
                               04D0  1043 ;
                               04D0  1044
                     50   0C   3C   04D0  1045 ACCVIO:  MOVZWL  #SS$_ACCVIO,R0        ;SET ACCESS VIOLATION
                          F8   11   04D3  1046           BRB     42$
                               04D5  1047
                               04D5  1048 ;
                               04D5  1049 ; CHANGE ACCESS MODE ATTRIBUTE PROCESSING
                               04D5  1050 ;
                               04D5  1051 NEWMODE:
                          53   DD   04D5  1052           PUSHL   R3                    ;SAVE IRP
                          53   D4   04D7  1053           CLRL    R3                    ;SPECIFY MODE (MAXIMIZE CALLERS MODE)
                     51   F8 AB   3C   04D9  1054           MOVZWL  -8(R11),R1            ;GET LENGTH
                     50   FC AB   D0   04DD  1055           MOVL    -4(R11),R0            ;GET ADDRESS OF BUFFER
                00000000'EF     16   04E1  1056           JSB     EXE$PROBEW            ;PROBE BUFFER
                          53 8ED0   04E7  1057           POPL    R3
                        E3 50    E9   04EA  1058           BLBC    R0,ACCVIO             ;IF LBC, NO WRITE ACCESS
                     50   FC AB   D0   04ED  1059           MOVL    -4(R11),R0            ;GET ADDRESS OF USER BUFFER
                        51   60   9A   04F1  1060           MOVZBL  (R0),R1               ;LOAD NEW CURRENT ACCESS MODE
                          AB   11   04F4  1061           BRB     25$                   ;AND REJOIN MAINLINE PATH
                               04F6  1062
                          67   11   04F6  1063 BRXQUOTA:BRB    XQUOTA                ;
                               04F8  1064 ;
                               04F8  1065 ; ACP BUFFER ALLOCATED AND ALL DESCRIPTORS CHECKED
                               04F8  1066 ;
                               04F8  1067
                          08   BA   04F8  1068 50$:      POPR    #^M<R3>               ;RETRIEVE I/O PACKET ADDRESS
                     2C A3    52   D0   04FA  1069           MOVL    R2,IRP$L_SVAPTE(R3)   ;INSERT ADDRESS OF ACP BUFFER
                     32 A3    59   D0   04FE  1070           MOVL    R9,IRP$L_BCNT(R3)     ;INSERT NUMBER OF DESCRIPTORS
                     30 A3    5A   B0   0502  1071           MOVW    R10,IRP$W_BOFF(R3)    ;SET NUMBER OF BYTES CHARGED TO QUOTA
                2A A3   1018 8F   A8   0506  1072           BISW    #IRP$M_COMPLX!-       ;SET COMPLEX BUFFERED I/O AND
                               050C  1073           IRP$M_FILACP!IRP$M_VIRTUAL,IRP$W_STS(R3) ; VIRTUAL, COMPLEX FILE ACP I/O
                        3E A4   B7   050C  1074           DECW    PCB$W_DIOCNT(R4)      ;CHARGE A DIRECT I/O
                     82   0C A2   DE   050F  1075           MOVAL   12(R2),(R2)+          ;SET POINTER TO FIRST DESCRIPTOR
                          82   D4   0513  1076           CLRL    (R2)+                 ;CLEAR SPARE LONGWORD
                        82   5A   B0   0515  1077           MOVW    R10,(R2)+             ;SET SIZE OF BUFFER
                        82   13   9B   0518  1078           MOVZBW  #DYN$C_BUFIO,(R2)+    ;SET DATA STRUCTURE TYPE
                5C   0080 C4   D0   051B  1079           MOVL    PCB$L_JIB(R4),AP      ;GET JIB ADDRESS
```

```
                5A    5A   3C  0520  1080          MOVZWL  R10,R10                  ;CONVERT TO LONGWORD
              20 AC   5A   C2  0523  1081          SUBL    R10,JIB$L_BYTCNT(AP)    ;UPDATE CURRENT BUFFERED I/O USAGE
           5C 59      08   C5  0527  1082          MULL3   #8,R9,AP                ;CALCULATE OFFSET TO START OF DATA AREA
              5C      52   C0  052B  1083          ADDL    R2,AP                   ;CALCULATE ADDRESS OF DATA AREA
              5A      52   D0  052E  1084          MOVL    R2,R10                  ;SET ADDRESS OF FIRST OUTPUT DESCRIPTOR
              5B      5E   D0  0531  1085          MOVL    SP,R11                  ;SET ADDRESS OF FIRST INPUT DESCRIPTOR
                      38   BB  0534  1086          PUSHR   #^M<R3,R4,R5>           ;SAVE REGISTERS R3, R4, AND R5
                      77   10  0536  1087          BSBB    WRDESCR                 ;PROCESS WINDOW ADDRESS DESCRIPTOR
                      71   10  0538  1088          BSBB    RDDESCR                 ;PROCESS FIB DESCRIPTOR
                      6F   10  053A  1089          BSBB    RDDESCR                 ;PROCESS NAME STRING DESCRIPTOR
                      6D   10  053C  1090          BSBB    RDDESCR                 ;PROCESS RESULT STRING LENGTH DESCRIPTOR
                      6B   10  053E  1091          BSBB    RDDESCR                 ;PROCESS RESULT STRING DESCRIPTOR
                59    05   C2  0540  1092          SUBL    #5,R9                   ;CALCULATE NUMBER OF REMAINING DESCRIPTORS
                      05   13  0543  1093          BEQL    70$                     ;IF EQL NONE
                      64   10  0545  1094 60$:     BSBB    RDDESCR                 ;PROCESS ATTRIBUTE DESCRIPTOR
                   FB 59   F5  0547  1095          SOBGTR  R9,60$                  ;ANY MORE ATTRIBUTES TO PROCESS?
                      38   BA  054A  1096 70$:     POPR    #^M<R3,R4,R5>           ;RESTORE REGISTERS R3, R4, AND R5
           5E   0118  CE   9E  054C  1097          MOVAB   MXDESCR+8(SP),SP        ;REMOVE DESCRIPTOR STORAGE FROM STACK
             38 A3    66   D0  0551  1098          MOVL    CCB$L_UCB(R6),IRP$L_MEDIA(R3) ;INSERT ORIGINAL UCB ADDRESS
                           0555  1099          SETIPL  #IPL$_SYNCH             ;SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
                      05   0558  1100          RSB                             ;
                           0559  1101 BRACCVIO1:
                      50 8ED0   0559  1102          POPL    R0
                           055C  1103 BRACCVIO:                                 ;
                   FF71    31   055C  1104          BRW     ACCVIO                  ;
                           055F  1105
                           055F  1106 ;
                           055F  1107 ; BYTE ACCUMULATION OVERFLOW
                           055F  1108 ;
                           055F  1109
                50    1C   3C   055F  1110 XQUOTA: MOVZWL  #SS$_EXQUOTA,R0         ;SET QUOTA EXCEEDED
                   FA9B'    31   0562  1111          BRW     EXE$ABORTIO
                           0565  1112          .DSABL  LSB
```

```
                     0565  1114          .SBTTL  CHECK DESCRIPTOR AND UPDATE BYTE ACCUMULATION
                     0565  1115 ;
                     0565  1116 ; SUBROUTINE TO CHECK PARAMETER DESCRIPTOR AND UPDATE BYTE ACCUMULATION
                     0565  1117 ;
                     0565  1118
                     0565  1119 CHKDESCR:                                    ;CHECK PARARMETER DESCRIPTOR
     50    8C    D0  0565  1120          MOVL    (AP)+,R0                   ;GET ADDRESS OF DESCRIPTOR
           29    13  0568  1121          BEQL    UPBYTCNT                   ;IF EQL NONE SPECIFIED
                     056A  1122          IFNORD  #8,(R0),BRACCVIO           ;CAN DESCRIPTOR BE READ?
     51    60    3C  0570  1123          MOVZWL  (R0),R1                    ;GET LENGTH OF INFORMATION STRING
           2E    13  0573  1124          BEQL    UPBYTCT1                   ;IF EQL ZERO LENGTH
  50    04    A0    D0  0575  1125          MOVL    4(R0),R0                   ;GET ADDRESS OF INFORMATION STRING
           51    DD  0579  1126          PUSHL   R1                         ;SAVE REGISTERS
           50    DD  057B  1127          PUSHL   R0
           53    DD  057D  1128          PUSHL   R3
           53    D4  057F  1129          CLRL    R3                         ;SPECIFY MODE
00000000'  EF    16  0581  1130          JSB     EXE$PROBEW                 ;PROBE FOR WRITE ACCESS
           53  8ED0  0587  1131          POPL    R3                         ;RESTORE REGISTERS
           51  8ED0  058A  1132          POPL    R1                         ; (SWAP R0 & R1)
        C9    50    E9  058D  1133          BLBC    R0,BRACCVIO1               ;IF LBC, NO ACCESS
           50  8ED0  0590  1134          POPL    R0                         ;RESTORE R0
                     0593  1135 UPBYTCNT:                                   ;INSERT DESCRIPTOR AND UPDATE BYTE ACCUMULAT
     5A    50    A0  0593  1136          ADDW    R0,R10                     ;UPDATE BYTE ACCUMULATION
           C7    1F  0596  1137          BCS     XQUOTA                     ;IF CS ACCUMULATION OVERFLOW
     8B    50    B0  0598  1138          MOVW    R0,(R11)+                  ;INSERT LENGTH OF INFORMATION STRING
  8B    0B    A3    9B  059B  1139          MOVZBW  IRP$B_RMOD(R3),(R11)+      ;INSERT ACCESS MODE
     8B    51    D0  059F  1140          MOVL    R1,(R11)+                  ;INSERT INFORMATION STRING ADDRESS
                 05  05A2  1141          RSB                                ;
                     05A3  1142 UPBYTCT1:
  51    04    A0    D0  05A3  1143          MOVL    4(R0),R1                   ;GET ADDRESS
           50    D4  05A7  1144          CLRL    R0                         ;WE KNOW IT'S A ZERO COUNT
           E8    11  05A9  1145          BRB     UPBYTCNT                   ;JOIN COMMON CODE
```

```
                        05AB  1147          .SBTTL  BUILD INFORMATION DESCRIPTOR AND COPY DATA
                        05AB  1148 ;
                        05AB  1149 ; SUBROUTINE TO BUILD DESCRIPTOR FOR INFORMATION THAT IS BEING READ FROM THE
                        05AB  1150 ; CALLING PROCESS' ADDRESS SPACE.
                        05AB  1151 ;
                        05AB  1152
                        05AB  1153          .ENABL  LSB
                        05AB  1154 RDDESCR:                                     ;BUILD READ DESCRIPTOR
                  01 DD 05AB  1155          PUSHL   #1                          ;SET READ INDICATOR
                  02 11 05AD  1156          BRB     10$                         ;
                        05AF  1157
                        05AF  1158 ;
                        05AF  1159 ; SUBROUTINE TO BUILD DESCRIPTOR FOR INFORMATION THAT WILL BE WRITTEN TO THE
                        05AF  1160 ; CALLING PROCESS' ADDRESS SPACE.
                        05AF  1161 ;
                        05AF  1162
                        05AF  1163 WRDESCR:                                     ;BUILD WRITE DESCRIPTOR
                  00 DD 05AF  1164          PUSHL   #0                          ;SET WRITE INDICATOR
            8A    5C 5A C3  05B1  1165 10$:     SUBL3   R10,AP,(R10)+           ;CALCULATE OFFSET TO DATA AREA
                        05B5  1166
              FE  AA 8B B0  05B5  1167          MOVW    (R11)+,-2(R10)          ;INSERT LENGTH OF DATA AREA
                 8C 8B 90   05B9  1168          MOVB    (R11)+,(AP)+            ;INSERT ACCESS MODE OR ATTRIBUTE NUMBER
                    5B D6   05BC  1169          INCL    R11                     ;ADVANCE TO INFORMATION ADDRESS
                 8A 8B D0   05BE  1170          MOVL    (R11)+,(R10)+           ;INSERT ADDRESS OF INFORMATION
                   06 8E E9 05C1  1171          BLBC    (SP)+,20$               ;IF LBC INFORMATION BEING WRITTEN
                        05C4  1172
        6C  FC BA   FA AA 28  05C4  1173          MOVC    -6(R10),@-4(R10),(AP)   ;MOVE INFORMATION TO BUFFER
              50    FA AA 3C  05CA  1174 20$:     MOVZWL  -6(R10),R0              ;GET LENGTH OF DATA AREA
                 5C 50 C0   05CE  1175          ADDL    R0,AP                   ;POINT TO NEXT DATA AREA
                    05 05D1  1176          RSB                                 ;
                        05D2  1177          .DSABL  LSB
```

```
                                05D2  1179          .SBTTL  CHECK VOLUME AND UPDATE TRANSACTION COUNT
                                05D2  1180 ;
                                05D2  1181 ; SUBROUTINE TO CHECK IF VOLUME MARKED FOR DISMOUNT, VOLUME NOT MOUNTED,
                                05D2  1182 ; OR VOLUME MOUNTED FOREIGN. IF ALL CHECKS SUCCEED, THEN UPDATE VOLUME
                                05D2  1183 ; TRANSACTION COUNT. ELSE RETURN APPROPRIATE ERROR.
                                05D2  1184 ;
                                05D2  1185
                                05D2  1186          .ENABL  LSB
                                05D2  1187 CHKDISMOUNT:                                 ;CHECK IF VOLUME MARKED FOR DISMOUNT
3A 38 A5   15   E0              05D2  1188          BBS     S^#DEV$V_DMT,UCB$L_DEVCHAR(R5),10$ ;IF SET, VOLUME MARKED FOR DISMOU
                                05D7  1189
                                05D7  1190 ;
                                05D7  1191 ; SUBROUTINE TO CHECK IF VOLUME NOT MOUNTED OR MOUNTED FOREIGN. IF BOTH
                                05D7  1192 ; CHECKS SUCCEED, THEN UPDATE TRANSACTION COUNT. ELSE RETURN APPROPRIATE
                                05D7  1193 ; ERROR.
                                05D7  1194 ;
                                05D7  1195
                                05D7  1196 CHKMOUNT:                                    ;CHECK IF VOLUME MOUNTED AND NOT FOREIGN
35 38 A5   13   E1              05D7  1197          BBC     S^#DEV$V_MNT,UCB$L_DEVCHAR(R5),10$ ;IF CLR, VOLUME NOT MOUNTED
30 64 A5   14   E0              05DC  1198          BBS     S^#UCB$V_DISMOUNT,UCB$L_STS(R5),10$ ;IF SET, NOT REALLY MOUNTED
32 38 A5   18   E0              05E1  1199          BBS     S^#DEV$V_FOR,UCB$L_DEVCHAR(R5),20$ ;IF SET, VOLUME FOREIGN
                                05E6  1200
                                05E6  1201 ;
                                05E6  1202 ; SUBROUTINE TO UPDATE VOLUME TRANSACTION COUNT. IF THERE IS A FILE OPEN ON
                                05E6  1203 ; THE CHANNEL, REDIRECT THE I/O FUNCTION TO THE UCB ON WHICH THE FILE IS
                                05E6  1204 ; OPEN.
                                05E6  12.5 ;
                                05E6  1206
                                05E6  1207 UPTRANSCNT:                                  ;UPDATE VOLUME TRANSACTION COUNT
1E 38 A5    0E  E1              05E6  1208          BBC     #DEV$V_FOD,UCB$L_DEVCHAR(R5),5$ ;BRANCH IF DEVICE IS NOT FILE DEV
19 38 A5    05  E0              05EB  1209          BBS     #DEV$V_SQD,UCB$L_DEVCHAR(R5),5$ ;BRANCH IF DEVICE IS MAGTAPE
   50   18 A3  D0              05F0  1210          MOVL    IRP$L_WIND(R3),R0           ;GET WINDOW ADDRESS FROM IRP
             13  13              05F4  1211          BEQL    5$                          ;BRANCH IF NO FILE OPEN
   55   10 A0  D0              05F6  1212          MOVL    WCB$L_ORGUCB(R0),R5         ;GET UCB ADDRESS OF FILE
38 A3   1C A3  D1              05FA  1213          CMPL    IRP$L_UCB(R3),IRP$L_MEDIA(R3) ;SEE IF THIS IS A SPOOL OPERATION
             04  12              05FF  1214          BNEQ    4$                          ;BRANCH IF YES
   38 A3    55  D0              0601  1215          MOVL    R5,IRP$L_MEDIA(R3)          ;REDIRECT ALTERNATE UCB AS WELL
   1C A3    55  D0              0605  1216 4$:      MOVL    R5,IRP$L_UCB(R3)            ;REDIRECT UCB ADDRESS IN IRP
                                0609  1217
   50   34 A5  D0              0609  1218 5$:      MOVL    UCB$L_VCB(R5),R0            ;GET ADDRESS OF VCB
        0C A0  B6              060D  1219          INCW    VCB$W_TRANS(R0)             ;UPDATE VOLUME TRANSACTION COUNT
               05              0610  1220          RSB                                 ;
                                0611  1221
                                0611  1222 ;
                                0611  1223 ; VOLUME MARKED FOR DISMOUNT OR NOT MOUNTED
                                0611  1224 ;
                                0611  1225
 50    007C 8F   3C             0611  1226 10$:     MOVZWL  #SS$_DEVNOTMOUNT,R0         ;SET DEVICE NOT MOUNTED
             05  11              0616  1227          BRB     30$
                                0618  1228
                                0618  1229 ;
                                0618  1230 ; DEVICE MOUNTED FOREIGN
                                0618  1231 ;
                                0618  1232
 50    0064 8F   3C             0618  1233 20$:     MOVZWL  #SS$_DEVFOREIGN,R0          ;SET DEVICE FOREIGN
          F9E0' 31              061D  1234 30$:     BRW     EXE$ABORTIO
                                0620  1235
```

```
                                                H 9
0620  1236
0620  1237                .DSABL  LSB
```

```
                                                0620  1239          .SBTTL  XQP$UNLOCK_CACHE - Release Cache Contents and Unlock
                                                0620  1240 ;++
                                                0620  1241 ;
                                                0620  1242 ; This subroutine is entered as a system blocking AST when another file
                                                0620  1243 ; system in the cluster requests a flush of all caches of a particular
                                                0620  1244 ; type. It passes the AST on as a real process AST to the file system's
                                                0620  1245 ; cache server process.
                                                0620  1246 ;
                                                0620  1247 ; INPUTS:
                                                0620  1248 ;       R1 = AST parameter
                                                0620  1249 ;          = UCB address + cache type in low bits
                                                0620  1250 ;
                                                0620  1251 ;--
                                                0620  1252
                                                0620  1253 XQP$UNLOCK_CACHE::
          55    51    07    CB                  0620  1254          BICL3   #^X7,R1,R5               ; get UCB address
    50    51     FFFFFFF8 8F    CB              0624  1255          BICL3   #^C^X7,R1,R0             ; and cache type code
             4B 38 A5     13    E1              062C  1256          BBC     #DEV$V_MNT,UCB$L_DEVCHAR(R5),40$ ; if not mounted, don't do it
             46 38 A5     18    E0              0631  1257          BBS     #DEV$V_FOR,UCB$L_DEVCHAR(R5),40$ ; if mounted foreign, don't
                55    34 A5    D0              0636  1258          MOVL    UCB$L_VCB(R5),R5        ; get VCB address
                54    58 A5    D0              063A  1259          MOVL    VCB$L_CACHE(R5),R4      ; get cache block address
                                                063E  1260          ASSUME  FIB$C_FID_CACHE EQ 1
                                                063E  1261          ASSUME  FIB$C_EXTENT_CACHE EQ 2
                                                063E  1262          ASSUME  FIB$C_QUOTA_CACHE EQ 3
                      50    02    C2            063E  1263          SUBL    #2,R0                   ; check cache type
                            0F    14            0641  1264          BGTR    20$                     ; branch if quota cache
                            06    13            0643  1265          BEQL    10$                     ; branch if extent cache
                55    64    08    C1            0645  1266          ADDL3   #VCA$B_FIDCACB,VCA$L_FIDCACHE(R4),R5 ; get file ID cache ACB
                            10    11            0649  1267          BRB     30$
                                                064B  1268
          55    04 A4       10    C1            064B  1269 10$:     ADDL3   #VCA$B_EXTCACB,VCA$L_EXTCACHE(R4),R5 ; get extent cache ACB
                            09    11            0650  1270          BRB     30$
                                                0652  1271
                55    5C A5    D0              0652  1272 20$:     MOVL    VCB$L_QUOCACHE(R5),R5   ; find quota cache
                            24    13            0656  1273          BEQL    40$                     ; branch if none
                   55       28    C0           0658  1274          ADDL    #VCA$B_QUOFLUSHACB,R5   ; get quota cache ACB
                14 A5       51    D0           065B  1275 30$:     MOVL    R1,ACB$L_ASTPRM(R5)     ; set up AST parameter
    0C A5    00000000'EF    D0                 065F  1276          MOVL    XQP$GL_FILESERVER,ACB$L_PID(R5) ; and PID
                            14    13            0667  1277          BEQL    NOACP                   ; bad news if there isn't one
    10 A5    00000000'EF    D0                 0669  1278          MOVL    XQP$GL_FILESERV_ENTRY,ACB$L_AST(R5) ; and address
                            52    D4            0671  1279          CLRL    R2                      ; no priority boost
                00000000'EF 16    0673  1280          JSB     SCH$QAST                ; and queue the AST
                      01 50 E9    0679  1281          BLBC    R0,NOACP                ; bug check if error
                            05    067C  1282 40$:     RSB
                                  067D  1283
                                  067D  1284 ;
                                  067D  1285 ; To here on error returns from SCH$QAST - our server process has disappeared.
                                  067D  1286 ;
                                  067D  1287 NOACP:  BUG_CHECK NONEXSTACP,FATAL      ; server process disappeared
```

```
                     0681  1289          .SBTTL  XQP$BLOCK_ROUTINE - Block further XQP activity.
                     0681  1290 ;++
                     0681  1291 ; XQP$BLOCK_ROUTINE
                     0681  1292 ;
                     0681  1293 ; This routine is not an fdt routine, but needs a home in the non-paged exec.
                     0681  1294 ; It is invoked as a system blocking routine.  The lock itself is
                     0681  1295 ; armed by the FilBXQP before the first file system request is performed
                     0681  1296 ; for a given volume as part of mounting the volume.
                     0681  1297 ;
                     0681  1298 ; If the file system quiescent when this routine is called, a kernel
                     0681  1299 ; ast will be queued to execute in the context of the swapper.  That
                     0681  1300 ; routine follows this one.  Otherwise, a flag is set so that all
                     0681  1301 ; further file system requests for the volume are blocked, and the last
                     0681  1302 ; one underway will complete shutting this volume down.
                     0681  1303 ;
                     0681  1304 ; All of this is necessary so that volume rebuild can rummage around
                     0681  1305 ; the volume without other activity messing it up.
                     0681  1306 ;
                     0681  1307 ; The ACB used by this routine is part of the per volume file system
                     0681  1308 ; database in non-paged pool, and was set up by the file system when
                     0681  1309 ; the lock with this blocking routine was armed.
                     0681  1310 ;
                     0681  1311 ; Called at IPL_SYNCH.
                     0681  1312 ;
                     0681  1313 ; INPUTS:
                     0681  1314 ;
                     0681  1315 ;       R1 - blocking routine parameter = address of VCB.
                     0681  1316 ;
                     0681  1317 ; Registers R0 - R5 are available as scratch registers.
                     0681  1318 ;
                     0681  1319 ;--
                     0681  1320 XQP$BLOCK_ROUTINE::
        0E A1   B5   0681  1321          TSTW    VCB$W_RVN (R1)          ; is this a volume set?
          19    12   0684  1322          BNEQ    50$                     ; Neq it is.
      00A0 C1   B7   0686  1323          DECW    VCB$W_ACTIVITY (R1)     ; Is volume quiescent?
          01    13   068A  1324          BEQL    10$                     ; EQL it is.
                05   068C  1325          RSB                             ; Not quiet.  Exit.
                     068D  1326
      008C C1   D4   068D  1327 10$:     CLRL    VCB$L_BLOCKID (R1)      ; note lock is disarmed.
                     0691  1328          ASSUME  VCB$S_ACB EQ ACB$C_LENGTH
                     0691  1329 ;
                     0691  1330 ; There is an ACB block built into each VCB.  The size allocated
                     0691  1331 ; in the VCB MUST match the real VCB.
                     0691  1332 ;
 55   00A8 C1   9E   0691  1333          MOVAB   VCB$B_ACB (R1), R5      ; address of ACB
           52   D4   0696  1334 20$:     CLRL    R2                      ; no priority increment
         F965'  30   0698  1335          BSBW    SCH$QAST                ; queue ast to dequeue lock.
        DF 50   E9   069B  1336          BLBC    R0,NOACP                ; bug check if error
                05   069E  1337 30$:     RSB
                     069F  1338
 50     20 A1   D0   069F  1339 50$:     MOVL    VCB$L_RVT (R1), R0      ; get RVT address.
        06 A0   B7   06A3  1340          DECW    RVT$W_ACTIVITY (R0)     ; is volume set quiescent?
           F6   12   06A6  1341          BNEQ    30$                     ; NEQ not quiet.  Exit.
        24 A0   D4   06A8  1342          CLRL    RVT$L_BLOCKID (R0)
 55     28 A0   9E   06AB  1343          MOVAB   RVT$B_ACB (R0), R5
                     06AF  1344          ASSUME  RVT$S_ACB EQ ACB$C_LENGTH
                     06AF  1345 ;
```

```
                06AF  1346 ; There is an ACB built into each RVT.
                06AF  1347 ; The length provided in the RVT MUST match that of the real ACB.
                06AF  1348 ;
E5   11         06AF  1349          BRB     20$
```

```
                                06B1  1351          .SBTTL  XQP$DEQBLOCKER - dequeue blocking lock
                                06B1  1352
                                06B1  1353 ;++
                                06B1  1354 ; XQP$DEQBLOCKER
                                06B1  1355 ;
                                06B1  1356 ; This routine dequeues the per volume (or volume set) blocking lock.
                                06B1  1357 ; It executes as a kernel ast in the context of the swapper.
                                06B1  1358 ; It is initiated by the blocking routine above.
                                06B1  1359 ;
                                06B1  1360 ; The ast parameter is address of the lock id to be dequeued.
                                06B1  1361 ;
                                06B1  1362 ;--
                                06B1  1363
                           0000 06B1  1364          .ENTRY  XQP$DEQBLOCKER, 0       ; save no registers
                         7E     7C  06B3  1365          CLRQ    -(SP)                   ; null arguments
                         7E     D4  06B5  1366          CLRL    -(SP)
                      04 AC     DD  06B7  1367          PUSHL   4(AP)                   ; lock id to dequeue.
          80000000'GF    04     FB  06BA  1368          CALLS   #4,G^SYS$DEQ-P1SYSVECTORS+^X80000000
                      66 50     E9  06C1  1369          BLBC    R0,LOCKERR              ; bug check on any error
                                04  06C4  1370          RET                             ; exit.
```

```
                        06C5  1372          .SBTTL  XQP$REL_QUOTA - Release Quota Cache Entry
                        06C5  1373
                        06C5  1374 ;++
                        06C5  1375 ;
                        06C5  1376 ; XQP$REL_QUOTA
                        06C5  1377 ;
                        06C5  1378 ; This routine is entered by a fork-level blocking AST on a system owned
                        06C5  1379 ; quota cache lock. The event of the blocking AST indicates that the cache
                        06C5  1380 ; entry is being requested by another processor; therefore, we release
                        06C5  1381 ; the lock (and the cache entry contents). The actual releasing of the
                        06C5  1382 ; lock is done by an AST queued to the swapper, which executes the routine
                        06C5  1383 ; following this one.
                        06C5  1384 ;
                        06C5  1385 ; Called at IPL$_SYNCH
                        06C5  1386 ;
                        06C5  1387 ; INPUTS:
                        06C5  1388 ;
                        06C5  1389 ;       R1 = address of cache entry requested
                        06C5  1390 ;
                        06C5  1391 ;--
                        06C5  1392
                        06C5  1393 XQP$REL_QUOTA::
                        06C5  1394 ;
                        06C5  1395 ; There is one ACB for each volume, back in the header of the quota
                        06C5  1396 ; cache. We find the cache header by using the cache entry's index
                        06C5  1397 ; to subtract back.
                        06C5  1398 ;
            50  61  3C  06C5  1399          MOVZWL  VCA$W_QUOINDEX(R1),R0   ; get entry's index
            50  1C  C4  06C8  1400          MULL    #VCA$C_QUOLENGTH,R0     ; compute offset in bytes
            50  1C  C0  06CB  1401          ADDL    #VCA$L_QUOLIST-VCA$B_QUOACB-VCA$C_QUOLENGTH,R0
                        06CE  1402                                          ; add in header, point to ACB
        55  51  50  C3  06CE  1403          SUBL3   R0,R1,R5                ; point to ACB
          14 A5  51  D0  06D2  1404          MOVL    R1,ACB$L_ASTPRM(R5)     ; AST param is cache entry
                 52  D4  06D6  1405          CLRL    R2                      ; no priority increment
               F925'  30  06D8  1406          BSBW    SCH$QAST                ; rest of ACB is set up - queue it
             9F  50  E9  06DB  1407          BLBC    R0,NOACP
                     05  06DE  1408          RSB                             ; bug check if error
```

```
                     06DF  1410          .SBTTL  XQP$UNLOCK_QUOTA - Release Lock on Quota Cache Entry
                     06DF  1411
                     06DF  1412 ;++
                     06DF  1413 ;
                     06DF  1414 ; XQP$UNLOCK_QUOTA
                     06DF  1415 ;
                     06DF  1416 ; This routine executes in the context of the swapper as a kernel AST.
                     06DF  1417 ; It dequeues or demotes the lock on the specified quota cache entry
                     06DF  1418 ; according to its current status.
                     06DF  1419 ;
                     06DF  1420 ; INPUTS:
                     06DF  1421 ;
                     06DF  1422 ;       ASTPARAM = address of quota cache entry
                     06DF  1423 ;
                     06DF  1424 ;--
                     06DF  1425
           00000004  06DF  1426 ASTPARAM = 4
                     06DF  1427
                0004 06DF  1428          .ENTRY  XQP$UNLOCK_QUOTA,^M<R2> ; save R2
         52  04 AC   D0  06E1  1429          MOVL    ASTPARAM(AP),R2         ; get address of cache entry
                     06E5  1430          ASSUME  VCA$W_QUOINDEX+2 EQ VCA$W_QUOLRUX
            62   DD  06E5  1431          PUSHL   VCA$W_QUOINDEX(R2)      ; save cache index and LRU counter
                     06E7  1432          ASSUME  VCA$V_QUOVALID EQ 0
          16 0B A2   E8  06E7  1433          BLBS    VCA$B_QUOFLAGS(R2),10$  ; branch if valid - demote lock
                     06EB  1434
            7E   7C  06EB  1435          CLRQ    -(SP)                   ; null arguments
            7E   D4  06ED  1436          CLRL    -(SP)                   ; no value block
         04 A2   DD  06EF  1437          PUSHL   VCA$L_QUOLKID(R2)       ; lock id to dequeue.
80000000'GF   04   FB  06F2  1438          CALLS   #4, G^SYS$DEQ-P1SYSVECTORS+^X80000000
         2E 50   E9  06F9  1439          BLBC    R0,LOCKERR              ; bug check on any error
                     06FC  1440          ASSUME  VCA$L_QUORECNUM EQ VCA$L_QUOLKID+4
                     06FC  1441          ASSUME  VCA$B_QUOFLAGS EQ VCA$L_QUORECNUM+3
         04 A2   7C  06FC  1442          CLRQ    VCA$L_QUOLKID(R2)       ; mark cache entry vacant
            25   11  06FF  1443          BRB     20$                     ; exit
                     0701  1444 ;
                     0701  1445 ; To here if we are converting the lock down, instead of releasing it
                     0701  1446 ; entirely.
                     0701  1447 ;
            7E   7C  0701  1448 10$:     CLRQ    -(SP)                   ; null acmode and extra param
         BF AF   9F  0703  1449          PUSHAB  XQP$REL_QUOTA           ; re-arm same blocking AST
            52   DD  0706  1450          PUSHL   R2                      ; cache entry is AST param
            7E   7C  0708  1451          CLRQ    -(SP)                   ; null astadr and parent ID
            7E   D4  070A  1452          CLRL    -(SP)                   ; null resource name
   0000004F 8F   DD  070C  1453          PUSHL   #LCK$M_CONVERT!LCK$M_NOQUEUE!LCK$M_CVTSYS!LCK$M_VALBLK!LCK$M_SYNCSTS
            62   9F  0712  1454          PUSHAB  VCA$R_QUOLOCK(R2)       ; lock status block
            01   DD  0714  1455          PUSHL   #LCK$K_CRMODE           ; lock mode
            7E   D4  0716  1456          CLRL    -(SP)                   ; null EFN
80000000'GF   0B   FB  0718  1457          CALLS   #11,G^SYS$ENQ-P1SYSVECTORS+^X80000000
         08 50   E9  071F  1458          BLBC    R0,LOCKERR              ; bug check on any error
      0B A2   01   8A  0722  1459          BICB    #VCA$M_QUOVALID,VCA$B_QUOFLAGS(R2) ; mark entry not valid
         62   8E   D0  0726  1460 20$:     MOVL    (SP)+,VCA$W_QUOINDEX(R2) ; restore index and LRU counter
              04   0729  1461          RET
                     072A  1462 ;
                     072A  1463 ; Bug check on any lock manager errors.
                     072A  1464 ;
                     072A  1465 LOCKERR:
                     072A  1466          BUG_CHECK XQPERR,FATAL
```

```
                     072E  1468          .SBTTL  XQP$FCBSTALE - Blocking routine to mark FCB as stale.
                     072E  1469 ;++
                     072E  1470 ;
                     072E  1471 ; This routine is a system blocking routine called from the lock manager
                     072E  1472 ; with a jsb at ipl_syrch.  It will set the STALE flag in the FCB.
                     072E  1473 ; The blocking routine is associated with the access lock on the file.
                     072E  1474 ;
                     072E  1475 ; INPUTS:
                     072E  1476 ;       R1 = fcb address.
                     072E  1477 ;
                     072E  1478 ;--
                     072E  1479
                     072E  1480 XQP$FCBSTALE::
     07    0A A1    91   072E  1481          CMPB    FCB$B_TYPE (R1), #DYN$C_FCB     ; Is this an FCB?
                  F6   12   0732  1482          BNEQ    LOCKERR                         ; Unexpected answer.
 22 A1    0100 8F    A8   0734  1483          BISW2   #1@FCB$V_STALE, FCB$W_STATUS (R1) ; Set flag.
                  05   073A  1484          RSB                                     ; That's it.
                     073B  1485
                     073B  1486
                     073B  1487          .END
```

```
ACB$C_LENGTH            =  0000001C
ACB$L_AST               =  00000010
ACB$L_ASTPRM            =  00000014
ACB$L_PID               =  0000000C
ACCVIO                     000004D0  R      01
ACP$ACCESS                 00000003  RG     01
ACP$ACCESSNET              00000008  RG     01
ACP$DEACCESS               0C000034  RG     01
ACP$MODIFY                 0000008A  RG     01
ACP$MOUNT                  00000092  RG     01
ACP$READBLK                000000B1  RG     01
ACP$WRITEBLK               000000F0  RG     01
ASTPARAM                =  00000004
ATR$C_ACCESS_MODE       =  0000002D
BRACCVIO                   0000055C  R      01
BRACCVIO1                  00000559  R      01
BRMODIFY                   00000000  R      01
BRXQUOTA                   000004F6  R      01
BUG$_NONEXSTACP            ********    X    01
BUG$_XQPERR                ********    X    01
BUILDACPBUF                00000417  R      01
CCB$B_STS               =  00000008
CCB$L_DIRP              =  0000000C
CCB$L_UCB               =  00000000
CCB$L_WIND              =  00000004
CCB$V_RDCHKDON          =  00000002
CCB$V_WRTCHKDON         =  00000003
CCB$W_IOC               =  0000000A
CHKDESCR                   00000565  R      01
CHKDISMOUNT                000005D2  R      01
CHKMOUNT                   000005D7  R      01
CTL$T_USERNAME             ********    X    01
DC$_DISK                =  00000001
DC$_TAPE                =  00000002
DDT$L_FDT               =  00000008
DEV$V_DMT               =  00000015
DEV$V_FOD               =  0000000E
DEV$V_FOR               =  00000018
DEV$V_MNT               =  00000013
DEV$V_RCK               =  0000001E
DEV$V_RCT               =  00000008
DEV$V_SQD               =  00000005
DEV$V_SWL               =  00000019
DEV$V_WCK               =  0000001F
DYN$C_BUFIO             =  00000013
DYN$C_FCB               =  00000007
EXE$ABORTIO                ********    X    01
EXE$ALLOCBUF               ********    X    01
EXE$BUFFRQUOTA             ********    X    01
EXE$CHKRDACCES             ********    X    01
EXE$CHKWRTACCES            ********    X    01
EXE$FINISHIOC              ********    X    01
EXE$GL_ERASEPPT            ********    X    01
EXE$PROBER_DSC             ********    X    01
EXE$PROBEW                 ********    X    01
EXE$QIOACPPKT              ********    X    01
EXE$QIODRVPKT              ********    X    01
EXE$QIORETURN              ********    X    01
EXE$READLOCK               ********    X    01
EXE$WRITELOCK              ********    X    01
FCB$B_TYPE              =  0000000A
FCB$V_STALE             =  00000008
FCB$W_STATUS            =  00000022
FIB$C_EXTENT_CACHE      =  00000002
FIB$C_FID_CACHE         =  00000001
FIB$C_QUOTA_CACHE       =  00000003
FILL_BUFFER                00003CC  R      01
IO$M_DATACHECK          =  00004000
IO$M_INHERLOG           =  00000800
IO$M_INHSEEK            =  00001000
IO$V_ACCESS             =  00000006
IO$V_ENCRYPT            =  0000000D
IO$V_ERASE              =  0000000A
IO$_DEACCESS            =  00000034
IO$_DSE                 =  00000015
IO$_LOGICAL             =  0000002F
IO$_PHYSICAL            =  0000001F
IO$_WRITELBLK           =  00000020
IO$_WRITEPBLK           =  0000000B
IO$_WRITEVBLK           =  00000030
IOC$CVTLOGPHY              ********    X    01
IOC$MAPVBLK                ********    X    01
IPL$_SYNCH              =  00000008
IRP$B_RMOD              =  0000000B
IRP$L_ABCNT             =  00000040
IRP$L_BCNT              =  00000032
IRP$L_DIAGBUF           =  0000004C
IRP$L_KEYDESC           =  0000005C
IRP$L_MEDIA             =  00000038
IRP$L_OBCNT             =  00000044
IRP$L_SEGVBN            =  00000048
IRP$L_SVAPTE            =  0000002C
IRP$L_UCB               =  0000001C
IRP$L_WIND              =  00000018
IRP$M_COMPLX            =  00000008
IRP$M_FILACP            =  00001000
IRP$M_KEY               =  00008000
IRP$M_VIRTUAL           =  00000010
IRP$S_FCODE             =  00000006
IRP$V_FCODE             =  00000000
IRP$V_FUNC              =  00000001
IRP$W_BOFF              =  00000030
IRP$W_FUNC              =  00000020
IRP$W_STS               =  0000002A
JIB$L_BYTCNT            =  00000020
JIB$W_FILCNT            =  00000030
LCK$K_CRMODE            =  00000001
LCK$M_CONVERT           =  00000002
LCK$M_CVTSYS            =  00000040
LCK$M_NOQUEUE           =  00000004
LCK$M_SYNCSTS           =  00000008
LCK$M_VALBLK            =  00000001
LOCKERR                    0000072A  R      01
MMG$GL_SPTBASE             ********    X    01
```

```
MXDESCR                 = 00000023
NEWMODE                   000004D5 R      01
NOACP                     0000067D R      01
P1                      = 00000000
P1SYSVECTORS              ********    X   01
P2                      = 00000004
P3                      = 00000008
P4                      = 0000000C
P5                      = 00000010
P6                      = 00000014
PCB$L_JIB               = 00000080
PCB$Q_PRIV              = 00000084
PCB$W_DIOCNT            = 0000003E
PR$_IPL                 = 00000012
PRT$C_UR                = 0000000F
PRV$V_MOUNT             = 00000011
PTE$S_PROT              = 00000004
PTE$V_PROT              = 0000001B
RDDESCR                   000005AB R      01
RVT$B_ACB               = 00000028
RVT$L_BLOCKID           = 00000024
RVT$S_ACB               = 0000001C
RVT$W_ACTIVITY          = 00000006
SCH$QAST                  ********    X   01
SETUP_ERASE               00000302 R      01
SS$_ACCVIO              = 0000000C
SS$_BADPARAM            = 00000014
SS$_DEVFOREIGN          = 00000064
SS$_DEVNOTMOUNT         = 0000007C
SS$_ENDOFFILE           = 00000870
SS$_EXQUOTA             = 0000001C
SS$_FILALRACC           = 000000A4
SS$_FILNOTACC           = 000000AC
SS$_ILLBLKNUM           = 000000DC
SS$_ILLIOFUNC           = 000000F4
SS$_IVCHNLSEC           = 0000026C
SS$_NOPRIV              = 00000024
SS$_NORMAL              = 00000001
SS$_WRITLCK             = 0000025C
SYS$DEQ                   ********    X   01
SYS$ENQ                   ********    X   01
UCB$B_DEVCLASS          = 00000040
UCB$B_FIPL              = 0000000B
UCB$L_DDT               = 00000088
UCB$L_DEVCHAR           = 00000038
UCB$L_MAXBCNT           = 000000B4
UCB$L_MAXBLOCK          = 000000B0
UCB$L_STS               = 00000064
UCB$L_VCB               = 00000034
UCB$V_DISMOUNT          = 00000014
UCB$V_MOUNTING          = 00000009
UCB$W_STS               = 00000064
UPBYTCNT                  00000593 R      01
UPBYTCT1                  000005A3 R      01
UPTRANSCNT                000005E6 R      01
VA$M_BYTE               = 000001FF
VA$S_BYTE               = 00000009
VA$S_VPN                = 00000015
VA$V_VPN                = 00000009
VCA$B_EXTCACB           = 00000010
VCA$B_FIDCACB           = 00000008
VCA$B_QUOACB            = 0000000C
VCA$B_QUOFLAGS          = 0000000B
VCA$B_QUOFLUSHACB       = 00000028
VCA$C_QUOLENGTH         = 0000001C
VCA$L_EXTCACHE          = 00000004
VCA$L_FIDCACHE          = 00000000
VCA$L_QUOLIST           = 00000044
VCA$L_QUOLKID           = 00000004
VCA$L_QUORECNUM         = 00000008
VCA$M_QUOVALID          = 00000001
VCA$R_QUOLOCK           = 00000000
VCA$V_QUOVALID          = 00000000
VCA$W_QUOINDEX          = 00000000
VCA$W_QUOLRUX           = 00000002
VCB$B_ACB               = 000000A8
VCB$L_BLOCKID           = 0000008C
VCB$L_CACHE             = 00000058
VCB$L_QUOCACHE          = 0000005C
VCB$L_RVT               = 00000020
VCB$S_ACB               = 0000001C
VCB$W_ACTIVITY          = 000000A0
VCB$W_RVN               = 0000000E
VCB$W_TRANS             = 0000000C
WCB$B_ACCESS            = 0000000B
WCB$L_ORGUCB            = 00000010
WCB$L_READS             = 00000024
WCB$L_WRITES            = 00000028
WCB$M_NOTFCP            = 00000004
WCB$M_READ              = 00000001
WCB$M_SHRWCB            = 00000008
WCB$V_NOTFCP            = 00000002
WCB$V_READ              = 00000000
WCB$V_READCK            = 00000009
WCB$V_SHRWCB            = 00000003
WCB$V_WRITE             = 00000001
WCB$V_WRITECK           = 00000005
WCB$V_WRITE_TURN        = 0000000C
WCB$W_ACON              = 00000014
WCB$W_REFCNT            = 0000000E
WRDESCR                   000005AF R      01
XQP$BLOCK_ROUTINE         00000681 RG     01
XQP$DEQBLOCKER            000006B1 RG     01
XQP$FCBSTALE              0000072E RG     01
XQP$GL_FILESERVER         ********    X   01
XQP$GL_FILESERV_ENTRY     ********    X   01
XQP$REC_QUOTA             000006C5 RG     01
XQP$UNLOCK_CACHE          00000620 RG     01
XQP$UNLOCK_QUOTA          000006DF RG     01
XQUOTA                    0000055F R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	0000073B (1851.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.05	00:00:01.71
Command processing	123	00:00:00.57	00:00:04.15
Pass 1	650	00:00:27.96	00:01:40.35
Symbol table sort	0	00:00:04.70	00:00:14.26
Pass 2	275	00:00:05.93	00:00:14.78
Symbol table output	27	00:00:00.22	00:00:00.33
Psect synopsis output	2	00:00:00.02	00:00:00.11
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1115	00:00:39.47	00:02:15.83

The working set limit was 2250 pages.
164025 bytes (321 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 2959 non-local and 96 local symbols.
1487 source lines were read in Pass 1, producing 24 object records in Pass 2.
39 pages of virtual memory were used to define 38 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	21
_255DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	35

3093 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

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

0381 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SSDEF
LIS

SYSACPFDT
LIS

SYSADJSTK
LIS

SYSASCEFC
LIS

SYSBRKTHR
LIS

SYSASSIGN
LIS

SWAPPER
LIS

SYSASTCON
LIS

SVAPTE
LIS

SYSADJWSL
LIS