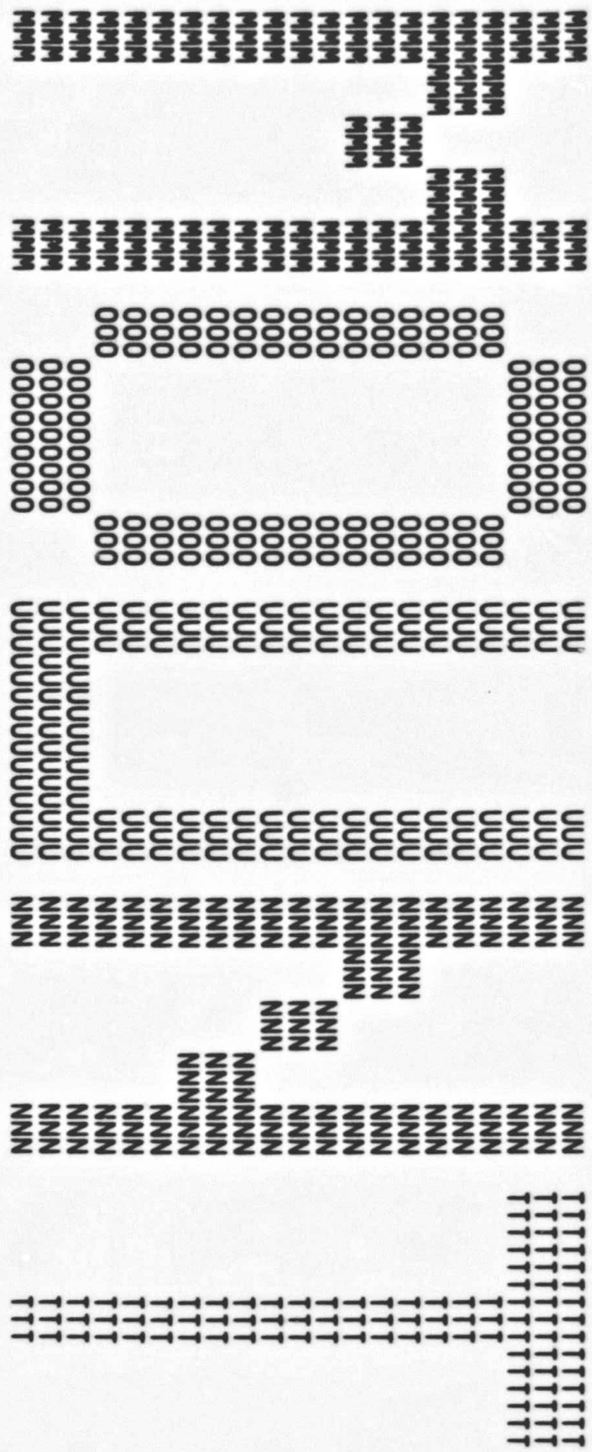

FILEID**MOUDK2

```
MM      MM    OOOOOO    UU      UU  DDDDDDDD    KK      KK    222222
MM      MM    OOOOOO    UU      UU  DDDDDDDD    KK      KK    222222
MMMM  MMMM  OO      OO  UU      UU  DD      DD  KK      KK  22      22
MMMM  MMMM  OO      OO  UU      UU  DD      DD  KK      KK  22      22
MM  MM  MM  OO      OO  UU      UU  DD      DD  KK    KK            22
MM  MM  MM  OO      OO  UU      UU  DD      DD  KK    KK            22
MM      MM  OO      OO  UU      UU  DD      DD  KKKKKK            22
MM      MM  OO      OO  UU      UU  DD      DD  KKKKKK          22
MM      MM  OO      OO  UU      UU  DD      DD  KK    KK      22
MM      MM  OO      OO  UU      UU  DD      DD  KK    KK      22
MM      MM  OO      OO  UU      UU  DD      DD  KK      KK    22          ....
MM      MM  OO      OO  UU      UU  DD      DD  KK      KK    22          ....
MM      MM    OOOOOO    UUUUUUUUUU  DDDDDDDD    KK      KK  2222222222  ....
MM      MM    OOOOOO    UUUUUUUUUU  DDDDDDDD    KK      KK  2222222222  ....

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
```

```
;     1          0001  0 MODULE MOUDK2 (
;     2          0002  0                 LANGUAGE (BLISS32),
;     3          0003  0                 IDENT = 'V04-002'
;     4          0004  0                 ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !**********************************************************************************
;     9          0009  1 !*                                                                                *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                       *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                         *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                           *
;    13          0013  1 !*                                                                                *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED          *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE         *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER         *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY         *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY         *
;    19          0019  1 !*  TRANSFERRED.                                                                   *
;    20          0020  1 !*                                                                                *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE         *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT         *
;    23          0023  1 !*  CORPORATION.                                                                   *
;    24          0024  1 !*                                                                                *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS         *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                        *
;    27          0027  1 !*                                                                                *
;    28          0028  1 !*                                                                                *
;    29          0029  1 !**********************************************************************************
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 !
;    33          0033  1 ! FACILITY:  MOUNT Utility Structure Level 2
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This routine performs all of the mechanics of mounting a disk,
;    38          0038  1 !       given as input the parsed and partially validated command line.
;    39          0039  1 !
;    40          0040  1 ! ENVIRONMENT:
;    41          0041  1 !
;    42          0042  1 !       STARLET operating system, including privileged system services
;    43          0043  1 !       and internal exec routines.
;    44          0044  1 !
;    45          0045  1 !--
;    46          0046  1 !
;    47          0047  1 !
;    48          0048  1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  17-Oct-1977  17:41
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;    52          0052  1 !       V04-002 HH0057          Hai Huang               12-Sep-1984
;    53          0053  1 !               Clear DEV$V_MNT bit along with UCB$L_VCB on error
;    54          0054  1 !               path.
;    55          0055  1 !
;    56          0056  1 !       V04-001 HH0056          Hai Huang               11-Sep-1984
;    57          0057  1 !               Return SS$_VOLINV status when appropriate to facilitate
```

```
;    58          0058  1 !                retry on volume invalid errors.
;    59          0059  1 !
;    60          0060  1 !        V03-030 CDS0011         Christian D. Saether    29-Aug-1984
;    61          0061  1 !                Only set up QUO_CACHE for RVN 1.  Ignore quodirty
;    62          0062  1 !                flag for other than RVN 1.
;    63          0063  1 !
;    64          0064  1 !        V03-029 HH0045          Hai Huang               10-Aug-1984
;    65          0065  1 !                Take out the volume lock for shared foreign mounts.
;    66          0066  1 !
;    67          0067  1 !        V03-028 ACG0438         Andrew C. Goldstein,    25-Jul-1984  11:48
;    68          0068  1 !                Initialize cache flusher ACB's in the VCA
;    69          0069  1 !
;    70          0070  1 !        V03-027 HH0041          Hai Huang               24-Jul-1984
;    71          0071  1 !                Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
;    72          0072  1 !
;    73          0073  1 !        V03-026 HH0039          Hai Huang               20-Jul-1984
;    74          0074  1 !                Return SS$_NOHOMEBLK if index file header checks fail.
;    75          0075  1 !
;    76          0076  1 !        V03-025 LMP0275         L. Mark Pilant,         12-Jul-1984  21:38
;    77          0077  1 !                Initialize the ACL info in the ORB to be a null descriptor
;    78          0078  1 !                list rather than an empty queue.  This avoids the overhead
;    79          0079  1 !                of locking and unlocking the ACL mutex, only to find out
;    80          0080  1 !                that the ACL was empty.
;    81          0081  1 !
;    82          0082  1 !        V03-024 CDS0010         Christian D. Saether    12-Jul-1984
;    83          0083  1 !                Clean up handling of quodirty flag and rebuilding
;    84          0084  1 !                decisions thereof.  Don't output REBLDREQD informational
;    85          0085  1 !                until after the volume is mounted.
;    86          0086  1 !
;    87          0087  1 !        V03-023 CDS0009         Christian D. Saether    11-Jul-1984
;    88          0088  1 !                Don't call check_cluster_sanity until after we've
;    89          0089  1 !                determined whether disk is write-locked.
;    90          0090  1 !                Don't rundown locks in cleanup handler until
;    91          0091  1 !                after scb has been rewritten.
;    92          0092  1 !                Revive use of SCB$L_STATUS and SCB$L_STATUS2 flags.
;    93          0093  1 !
;    94          0094  1 !        V03-022 HH0031          Hai Huang               03-Jul-1984
;    95          0095  1 !                Set up FCB$L_LOCKBASIS correctly when creating a volume
;    96          0096  1 !                set.
;    97          0097  1 !
;    98          0098  1 !        V03-021 HH0029          Hai Huang               29-Jun-1984
;    99          0099  1 !                Ouput informational message about reduced cache when
;   100          0100  1 !                appropriate.
;   101          0101  1 !
;   102          0102  1 !        V03-020 CDS0008         Christian D. Saether    17-May-1984
;   103          0103  1 !                Remove reference to VC_NOALLOC.  That flag is no
;   104          0104  1 !                longer used by the file system.
;   105          0105  1 !
;   106          0106  1 !        V03-019 CDS0007         Christian D. Saether    26-Apr-1984
;   107          0107  1 !                Bump FCB$W_REFCNT for index file fcb.
;   108          0108  1 !
;   109          0109  1 !        V03-018 LMP0221         L. Mark Pilant,         28-Mar-1984  9:57
;   110          0110  1 !                Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
;   111          0111  1 !                ORB$W_PROT.
;   112          0112  1 !
;   113          0113  1 !        V03-017 ROW0326         Ralph O. Weber          20-MAR-1984
;   114          0114  1 !                Add setup of VCB$Q_MOUNTTIME for later testing by mount
```

```
;  115          0115  1 !                          verification.
;  116          0116  1 !
;  117          0117  1 !                    V03-016 HH0004          Hai Huang                   11-Mar-1984
;  118          0118  1 !                          Fix truncation errors introduced by cluster-wide
;  119          0119  1 !                          mount support.
;  120          0120  1 !
;  121          0121  1 !                    V03-015 ACG0400         Andrew C. Goldstein,    10-Mar-1984  1:29
;  122          0122  1 !                          Turn quota cache back on
;  123          0123  1 !
;  124          0124  1 !                    V03-014 HH0002          Hai Huang                   01-Feb-1984
;  125          0125  1 !                          Add job-wide mount support, i.e. always deallocate
;  126          0126  1 !                          mount list entries to paged-pool in condition handler.
;  127          0127  1 !
;  128          0128  1 !                    V03-013 CDS0006         Christian D. Saether    18-Oct-1983
;  129          0129  1 !                          Move STORE_CONTEXT call to before quota file activation.
;  130          0130  1 !
;  131          0131  1 !                    V03-012 CDS0005         Christian D. Saether     1-Sep-1983
;  132          0132  1 !                          Only make duplicate volume set name test for first
;  133          0133  1 !                          volume mounted.  Clear UCB pointer in RVT in kernel
;  134          0134  1 !                          mode handler on error paths.
;  135          0135  1 !
;  136          0136  1 !                    V03-011 CDS0004         Christian D. Saether    30-Aug-1983
;  137          0137  1 !                          Use different local for rvt ucb scan.
;  138          0138  1 !
;  139          0139  1 !                    V03-010 CDS0003         Christian D. Saether     3-Aug-1983
;  140          0140  1 !                          Cluster consistency checks added.
;  141          0141  1 !                          Remove the earlier crude mount serialization now
;  142          0142  1 !                          that MOUNT_VOLUME does it based on device.
;  143          0143  1 !                          Delay increment of device refcount in UCB so that
;  144          0144  1 !                          it will not be left incorrectly biased on certain
;  145          0145  1 !                          error paths.
;  146          0146  1 !
;  147          0147  1 !                    V03-009 CDS0002         Christian D. Saether     3-Aug-1983
;  148          0148  1 !                          Remove reference to RVT$L_RVX (obsolete).
;  149          0149  1 !
;  150          0150  1 !                    V03-008 TCM0001         Trudy C. Matthews       21-Jun-1983
;  151          0151  1 !                          Increment device refcount in UCB on mount.
;  152          0152  1 !
;  153          0153  1 !                    V03-007 DMW4044         DMWalp                      7-Jun-1983
;  154          0154  1 !                          Remove (S)LOG_ENTRY
;  155          0155  1 !
;  156          0156  1 !                    V03-006 STJ3095         Steven T. Jeffreys,     28-Apr-1983
;  157          0157  1 !                          Propagate ERASE and NOHIGHWATER throughout the volume set.
;  158          0158  1 !
;  159          0159  1 !                    V03-005 STJ50311        Steven T. Jeffreys,     11-Feb-1983
;  160          0160  1 !                          Make all uses of PHYS_NAME indexed by DEVICE_INDEX.
;  161          0161  1 !
;  162          0162  1 !                    V03-004 CDS0001         Christian D. Saether     6-Jan-1983
;  163          0163  1 !                          Make test for xqp here and take out mount interlock lock
;  164          0164  1 !                          for duration of MOUNT_DISK2 if using xqp.
;  165          0165  1 !                          Temporarily disable write back caching and quota caching
;  166          0166  1 !                          when running with xqp, as well as rebuild until mount
;  167          0167  1 !                          can figure out if other mounters are present.
;  168          0168  1 !
;  169          0169  1 !                    V03-003 LMP0036         L. Mark Pilant,         6-Aug-1982  15:30
;  170          0170  1 !                          Add support for ACL's.
;  171          0171  1 !
```

```
;  172     0172  1 !       V03-002 STJ0301         Steven T. Jeffreys,     18-May-1982
;  173     0173  1 !               Add support for /NOUNLOAD qualifier.
;  174     0174  1 !
;  175     0175  1 !       V03-001 STJ0243         Steven T. Jeffreys,     03-Apr-1982
;  176     0176  1 !               - Use common I/O routines.
;  177     0177  1 !               - Remove code that sets device allocation access mode.
;  178     0178  1 !                 The device will be manually deallocated in VMOUNT.
;  179     0179  1 !               - Ensure that we back out a 'dirty' SCB in case the
;  180     0180  1 !                 specified ACP cannot be found.
;  181     0181  1 !
;  182     0182  1 !       V02-020 STJ0193         Steven T. Jeffreys,     02-Feb-1982
;  183     0183  1 !               Rearrange storage so that different modules can share
;  184     0184  1 !               the statically allocated buffers.
;  185     0185  1 !
;  186     0186  1 !       V02-019 STJ0179         Steven T. Jeffreys,     07-Jan-1982
;  187     0187  1 !               Add support for the VCB$V_MOUNTVER bit.
;  188     0188  1 !
;  189     0189  1 !       V02-018 ACG0246         Andrew C. Goldstein,    4-Jan-1982  14:27
;  190     0190  1 !               Add /OVER:LOCK support, add NOCACHE bit to VCB;
;  191     0191  1 !               Remove primary exception handler code.
;  192     0192  1 !
;  193     0193  1 !       V02-017 ACG0230         Andrew C. Goldstein,    29-Dec-1981  19:21
;  194     0194  1 !               Add file expiration support
;  195     0195  1 !
;  196     0196  1 !       V02-016 ACG0234         Andrew C. Goldstein,    4-Dec-1981  17:03
;  197     0197  1 !               Limit index file EOF to allocated space
;  198     0198  1 !
;  199     0199  1 !       V02-015 STJ0045         Steven T. Jeffreys,     31-May-1981
;  200     0200  1 !               Initialize a BLISS local variable to prevent a KERNEL mode
;  201     0201  1 !               access violation in MAKE_DISK_MOUNT for /FOREIGN mounts.
;  202     0202  1 !
;  203     0203  1 !       V02-014 STJ0040         Steven T. Jeffreys,     21-May-1981
;  204     0204  1 !               Copy volume serial number from homeblock to VCB.
;  205     0205  1 !
;  206     0206  1 !       V02-013 ACG35282        Andrew C. Goldstein,    23-Jan-1981  14:13
;  207     0207  1 !               Clean up SCB after ACP startup failure
;  208     0208  1 !
;  209     0209  1 !       V02-012 ACG0169         Andrew C. Goldstein,    18-Apr-1980  13:48
;  210     0210  1 !               Bug check on internal errors
;  211     0211  1 !
;  212     0212  1 !       V02-011 ACG0167         Andrew C. Goldstein,    18-Apr-1980  13:38
;  213     0213  1 !               Previous revision history moved to MOUNT.REV
;  214     0214  1 !**
;  215     0215  1
;  216     0216  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;  217     0217  1 REQUIRE 'SRC$:MOUDEF.B32';
;  218     0749  1
;  219     0750  1
;  220     0751  1 FORWARD ROUTINE
;  221     0752  1         MOUNT_DISK2     : NOVALUE,      ! main disk mounting routine
;  222     0753  1         MOUNT_HANDLER,                  ! condition handler for main mount code
;  223     0754  1         MAKE_DISK_MOUNT,                ! kernel mode mount routine
;  224     0755  1         SET_DATACHECK   : NOVALUE,      ! set volume data check attributes
;  225     0756  1         KERNEL_HANDLER  : NOVALUE;      ! kernel mode condition handler
```

```
;  227     0757  1 !+
;  228     0758  1 !
;  229     0759  1 ! Own storage for this module.
;  230     0760  1 !
;  231     0761  1 !-
;  232     0762  1
;  233     0763  1 LITERAL
;  234     0764  1         WINDOW_SIZE     = 30*6;         ! maximum index file window size
;  235     0765  1
;  236     0766  1 GLOBAL
;  237     0767  1         !
;  238     0768  1         ! Declare a one block buffer to be used by MOUDK1, MOUDK2, and BINDVL.
;  239     0769  1         ! Previously, each module declared the buffer as OWN storage.  Since
;  240     0770  1         ! the buffer is always written to before it is used, there is no need
;  241     0771  1         ! to zero it before hand.
;  242     0772  1         !
;  243     0773  1         BUFFER          : BBLOCK [512],         ! buffer for disk blocks
;  244     0774  1         !
;  245     0775  1         ! Likewise, MOUDK1 and MOUDK2 make use of PROTO_VCB, PROTO_FCB,
;  246     0776  1         ! PROTO_WCB and VOLUME_UIC.  In addition, MOUTAP also uses PROTO_VCB.
;  247     0777  1         ! Each module is responsible for zeroing the blocks before using them.
;  248     0778  1         !
;  249     0779  1         PROTO_VCB       : BBLOCK [VCB$C_LENGTH], ! prototype VCB
;  250     0780  1         PROTO_FCB       : BBLOCK [FCB$C_LENGTH], ! prototype index file FCB
;  251     0781  1         PROTO_WCB       : BBLOCK [WCB$C_LENGTH+WINDOW_SIZE],
;  252     0782  1                                                  ! prototype index file window
;  253     0783  1         VOLUME_UIC      : LONG,                  ! owner UIC of volume
;  254     0784  1         CACHE_STATUS;                            ! status of block cache allocation
;  255     0785  1
;  256     0786  1
;  257     0787  1 OWN
;  258     0788  1         IO_STATUS       : VECTOR [4, WORD];     ! I/O status block.
```

```
;  260     0789  1 GLOBAL ROUTINE MOUNT_DISK2 : NOVALUE =
;  261     0790  1
;  262     0791  1 !++
;  263     0792  1 !
;  264     0793  1 ! FUNCTIONAL DESCRIPTION:
;  265     0794  1 !
;  266     0795  1 !       This routine performs all of the mechanics of mounting a structure
;  267     0796  1 !       level 2 disk, given as input the parsed and partially validated
;  268     0797  1 !       command line.
;  269     0798  1 !
;  270     0799  1 !
;  271     0800  1 ! CALLING SEQUENCE:
;  272     0801  1 !       MOUNT_DISK ()
;  273     0802  1 !
;  274     0803  1 ! INPUT PARAMETERS:
;  275     0804  1 !       NONE
;  276     0805  1 !
;  277     0806  1 ! IMPLICIT INPUTS:
;  278     0807  1 !       MOUNT parser data base
;  279     0808  1 !       CHANNEL: channel number for I/O
;  280     0809  1 !       HOME_BLOCK: buffer containing volume home block
;  281     0810  1 !       HOMEBLOCK_LBN: LBN of home block
;  282     0811  1 !
;  283     0812  1 ! OUTPUT PARAMETERS:
;  284     0813  1 !       NONE
;  285     0814  1 !
;  286     0815  1 ! IMPLICIT OUTPUTS:
;  287     0816  1 !       NONE
;  288     0817  1 !
;  289     0818  1 ! ROUTINE VALUE:
;  290     0819  1 !       NONE
;  291     0820  1 !
;  292     0821  1 ! SIDE EFFECTS:
;  293     0822  1 !       volume mounted: VCB, etc., created, ACP started
;  294     0823  1 !
;  295     0824  1 !--
;  296     0825  1
;  297     0826  2 BEGIN
;  298     0827  2
;  299     0828  2 BUILTIN
;  300     0829  2         ROT,
;  301     0830  2         FFS,
;  302     0831  2         FFC,
;  303     0832  2         TESTBITSC;
;  304     0833  2
;  305     0834  2 LINKAGE
;  306     0835  2         L_MAP_POINTER   = JSB :
;  307     0836  2                           GLOBAL (COUNT = 6, LBN = 7, MAP_POINTER = 8);
;  308     0837  2
;  309     0838  2 LABEL
;  310     0839  2         IDX_SCAN;                       ! index file bitmap scan loop
;  311     0840  2
;  312     0841  2 GLOBAL REGISTER
;  313     0842  2         COUNT           = 6,            ! number of blocks in storage map
;  314     0843  2         LBN             = 7,            ! current LBN in use
;  315     0844  2         MAP_POINTER     = 8 : REF BBLOCK; ! pointer to scan map pointers
;  316     0845  2
```

```
;  317     0846  2 LOCAL
;  318     0847  2         PROCESS_UIC,                    ! UIC of this process
;  319     0848  2         PRIVILEGE_MASK  : REF BBLOCK,   ! address of process privilege mask
;  320     0849  2         P,                              ! random pointer
;  321     0850  2         C,                              ! string count
;  322     0851  2         STATUS          : BBLOCK [4],   ! utility status word
;  323     0852  2         IDX_EOF,                        ! end of file on index file
;  324     0853  2         FREE,                           ! number of free blocks on volume
;  325     0854  2         X,                              ! longword of bitmap
;  326     0855  2         B1,                             ! start point of bit scan
;  327     0856  2         B2;                             ! end point of bit scan
;  328     0857  2
;  329     0858  2 EXTERNAL
;  330     0859  2         DEV_CTX         : BBLOCK FIELD (DC), ! device lock value block context
;  331     0860  2         VOL_CTX         : BBLOCK FIELD (VC), ! volume lock value block context
;  332     0861  2         VOLOCK_COUNT,                   ! count of volume locks
;  333     0862  2         STORED_CONTEXT  : BITVECTOR,    ! XQP flag is used
;  334     0863  2         MOUNT_OPTIONS   : BITVECTOR,    ! command option flags
;  335     0864  2         CLEANUP_FLAGS   : BITVECTOR,    ! cleanup action flags
;  336     0865  2         DEVICE_CHAR     : BBLOCK,       ! device characteristics
;  337     0866  2         USER_STATUS,                    ! status return from some routines
;  338     0867  2         LABEL_STRING    : VECTOR,       ! volume label string in command
;  339     0868  2         DEVICE_INDEX    : VECTOR,       ! index into PHYS_NAME vector
;  340     0869  2         PHYS_NAME       : VECTOR,       ! descriptor of physical device name
;  341     0870  2         STRUCT_NAME     : VECTOR,       ! descriptor of volume set name
;  342     0871  2         DRIVE_COUNT     : VECTOR,       ! number of drives per device
;  343     0872  2         WINDOW,                         ! command specified window size
;  344     0873  2         ACCESSED,                       ! command specified LRU limit
;  345     0874  2         EXTENSION,                      ! command specified default file extend
;  346     0875  2         EXT_CACHE,                      ! size of extent cache to allocate
;  347     0876  2         FID_CACHE,                      ! size of file ID cache to allocate
;  348     0877  2         QUO_CACHE,                      ! size of quota file cache to allocate
;  349     0878  2         EXT_LIMIT,                      ! limit of volume space to cache
;  350     0879  2         HOME_BLOCK      : BBLOCK,       ! buffer containing volume home block
;  351     0880  2         HOMEBLOCK_LBN,                  ! LBN of home block
;  352     0881  2         HEADER_LBN,                     ! LBN of current file header
;  353     0882  2         CURRENT_RVN,                    ! RVN of disk being mounted
;  354     0883  2         CURRENT_VCB     : REF BBLOCK,   ! address of VCB being built
;  355     0884  2         CTL$GL_PHD      : REF BBLOCK ADDRESSING_MODE (ABSOLUTE),
;  356     0885  2                                         ! vector page pointer to process header
;  357     0886  2         ACP$GW_EXTCACHE : WORD ADDRESSING_MODE (ABSOLUTE),
;  358     0887  2                                         ! default space for extent cache
;  359     0888  2         ACP$GW_FIDCACHE : WORD ADDRESSING_MODE (ABSOLUTE),
;  360     0889  2                                         ! default space for file ID cache
;  361     0890  2         ACP$GW_QUOCACHE : WORD ADDRESSING_MODE (ABSOLUTE),
;  362     0891  2                                         ! default space for quota file cache
;  363     0892  2         ACP$GW_EXTLIMIT : WORD ADDRESSING_MODE (ABSOLUTE),
;  364     0893  2                                         ! max space to keep in extent cache
;  365     0894  2         ACP$GB_WRITBACK : BYTE ADDRESSING_MODE (ABSOLUTE),
;  366     0895  2                                         ! ACP write-back cache enable flag
;  367     0896  2         ACP$GB_WINDOW   : BYTE ADDRESSING_MODE (ABSOLUTE),
;  368     0897  2                                         ! default window size for /SYSTEM
;  369     0898  2         ACP$GW_SYSACC   : WORD  ADDRESSING_MODE (ABSOLUTE);
;  370     0899  2                                         ! default LRU limit for /SYSTEM
;  371     0900  2
;  372     0901  2 EXTERNAL ROUTINE
;  373     0902  2         CHECK_CLUSTER_SANITY : NOVALUE, ! check cluster mount consistency
```

```
;  374          0903  2          GET_VOLUME_LOCK,                    ! take out volume lock
;  375          0904  2          GET_VOLUME_LOCK_NAME,               ! generate volume lock name
;  376          0905  2          GET_UIC,                            ! get UIC of process
;  377          0906  2          CHECK_HEADER2,                      ! verify file header
;  378          0907  2          CHECKSUM,                           ! compute block checksum
;  379          0908  2          READ_BLOCK,                         ! read a block from the disk
;  380          0909  2          WRITE_BLOCK,                        ! write a block to the disk
;  381          0910  2          INIT_FCB2,                          ! initialize FCB
;  382          0911  2          TURN_WINDOW2,                       ! initialize window
;  383          0912  2          LEFT_ONE,                           ! leftmost one bit of value
;  384          0913  2          GET_MAP_POINTER : L_MAP_POINTER,    ! get value of file map pointer
;  385          0914  2          BIND_VOLUME;                        ! update volume set list
;  386          0915  2
;  387          0916  2
;  388          0917  2 ENABLE MOUNT_HANDLER;
;  389          0918  2
;  390          0919  2 CURRENT_VCB = PROTO_VCB;                     ! pointer used by CHECK_HEADER2
;  391          0920  2 CH$FILL (0, VCB$C_LENGTH, PROTO_VCB);       ! init to zero
;  392          0921  2 CACHE_STATUS = 1;                           ! init status of block cache allocation
;  393          0922  2
;  394          0923  2 ! For maximum safety, we do as much setup work in user mode as possible. We
;  395          0924  2 ! read all of the disk blocks (index file and storage map headers and the
;  396          0925  2 ! storage map) in user mode so that the program is abortable in case something
;  397          0926  2 ! hangs. Prototype control blocks are built in local storage and are copied
;  398          0927  2 ! into the system pool by the kernel mode routine.
;  399          0928  2 ! Get the process UIC and the volume owner UIC. Make the privilege checks
;  400          0929  2 ! for overriding volume protection and options requiring operator privilege.
;  401          0930  2 !
;  402          0931  2
;  403          0932  2 PROCESS_UIC = KERNEL_CALL (GET_UIC);
;  404          0933  2 PRIVILEGE_MASK = CTL$GL_PHD[PHD$Q_PRIVMSK];
;  405          0934  2 VOLUME_UIC = 0;
;  406          0935  2 IF .MOUNT_OPTIONS[OPT_IS_FILES11]
;  407          0936  2 THEN VOLUME_UIC = .HOME_BLOCK[HM2$L_VOLOWNER];
;  408          0937  2
;  409          0938  3 IF  (
;  410          0939  3      .MOUNT_OPTIONS[OPT_OVR_PRO]
;  411          0940  4      AND NOT (.PRIVILEGE_MASK[PRV$V_VOLPRO]
;  412          0941  4                  OR .VOLUME_UIC EQL 0
;  413          0942  4                  OR .VOLUME_UIC EQL .PROCESS_UIC)
;  414          0943  3      )
;  415          0944  3
;  416          0945  3 OR  (
;  417          0946  4      (  .MOUNT_OPTIONS[OPT_WINDOW]
;  418          0947  4      OR .MOUNT_OPTIONS[OPT_ACCESSED]
;  419          0948  4      OR .MOUNT_OPTIONS[OPT_UNIQUEACP]
;  420          0949  4      OR .MOUNT_OPTIONS[OPT_SAMEACP]
;  421          0950  4      OR .MOUNT_OPTIONS[OPT_FILEACP]
;  422          0951  4      OR .MOUNT_OPTIONS[OPT_CACHE]
;  423          0952  4      )
;  424          0953  3      AND NOT .PRIVILEGE_MASK[PRV$V_OPER]
;  425          0954  3      )
;  426          0955  3
;  427          0956  3 OR  (
;  428          0957  3      .MOUNT_OPTIONS[OPT_GROUP]
;  429          0958  3      AND NOT .PRIVILEGE_MASK [PRV$V_GRPNAM]
;  430          0959  3      )
```

```
;  431          0960  3
;  432          0961  3 OR  (
;  433          0962  3      .MOUNT_OPTIONS[OPT_SYSTEM]
;  434          0963  3      AND NOT .PRIVILEGE_MASK [PRV$V_SYSNAM]
;  435          0964  3      )
;  436          0965  3
;  437          0966  2 THEN ERR_EXIT (SS$_NOPRIV);
;  438          0967  2
;  439          0968  2 IF .MOUNT_OPTIONS[OPT_FOREIGN]
;  440          0969  2 THEN VOLUME_UIC = .PROCESS_UIC;
;  441          0970  2
;  442          0971  2 ! Unless the file= option was used to specify and acp for this
;  443          0972  2 ! volume always use the xqp.
;  444          0973  2 !
;  445          0974  2
;  446          0975  2 IF NOT .MOUNT_OPTIONS [OPT_FILEACP]
;  447          0976  2 THEN
;  448          0977  2     STORED_CONTEXT [XQP] = 1;
;  449          0978  2
;  450          0979  2 ! Establish the volume set name, if any. It comes from the /BIND switch,
;  451          0980  2 ! or from the home block. If both, they must match.
;  452          0981  2 !
;  453          0982  2
;  454          0983  2 IF .MOUNT_OPTIONS[OPT_BIND]
;  455          0984  2 THEN
;  456          0985  3     BEGIN
;  457          0986  3     IF .HOME_BLOCK[HM2$W_RVN] NEQ 0
;  458          0987  3     THEN
;  459          0988  4         BEGIN
;  460          0989  4         IF CH$NEQ (.STRUCT_NAME[0], .STRUCT_NAME[1],
;  461          0990  4                    HM2$S_STRUCNAME, HOME_BLOCK[HM2$T_STRUCNAME], ' ')
;  462          0991  4         THEN ERR_EXIT (MOUN$_VOLINSET);
;  463          0992  4         END
;  464          0993  4
;  465          0994  3     ELSE
;  466          0995  4         BEGIN
;  467          0996  4         CH$COPY (.STRUCT_NAME[0], .STRUCT_NAME[1], ' ',
;  468          0997  4                  HM2$S_STRUCNAME, HOME_BLOCK[HM2$T_STRUCNAME]);
;  469          0998  4         MOUNT_OPTIONS[OPT_DO_BIND] = 1;
;  470          0999  3         END;
;  471          1000  3     END
;  472          1001  3
;  473          1002  2 ELSE
;  474          1003  3     BEGIN
;  475          1004  3     IF .HOME_BLOCK[HM2$W_RVN] NEQ 0
;  476          1005  3     THEN
;  477          1006  4         BEGIN
;  478          1007  4         STRUCT_NAME[0] = HM2$S_STRUCNAME;
;  479          1008  4         STRUCT_NAME[1] = HOME_BLOCK[HM2$T_STRUCNAME];
;  480          1009  3         END;
;  481          1010  2     END;
;  482          1011  2
;  483          1012  2 ! Default the cache parameters to the system defaults.
;  484          1013  2 !
;  485          1014  2
;  486          1015  2 IF .EXT_CACHE EQL 0
;  487          1016  2 THEN EXT_CACHE = .ACP$GW_EXTCACHE;
```

```
;  488          1017  2 IF .MOUNT_OPTIONS[OPT_NOEXT_C]
;  489          1018  2 THEN EXT_CACHE = 0;
;  490          1019  2
;  491          1020  2 IF .FID_CACHE EQL 0
;  492          1021  2 THEN FID_CACHE = .ACP$GW_FIDCACHE;
;  493          1022  2 IF .MOUNT_OPTIONS[OPT_NOFID_C]
;  494          1023  2 OR .FID_CACHE EQL 0
;  495          1024  2 THEN FID_CACHE = 1;
;  496          1025  2
;  497          1026  2 IF .QUO_CACHE EQL 0
;  498          1027  2 THEN QUO_CACHE = .ACP$GW_QUOCACHE;
;  499          1028  2 IF .MOUNT_OPTIONS[OPT_NOQUO_C]
;  500          1029  2 THEN QUO_CACHE = 0;
;  501          1030  2
;  502          1031  2 IF .EXT_LIMIT EQL 0
;  503          1032  2 THEN EXT_LIMIT = .ACP$GW_EXTLIMIT;
;  504          1033  2
;  505          1034  2 ! First fill in the prototype VCB from the data in the home block.
;  506          1035  2 !
;  507          1036  2 .
;  508          1037  2 PROTO_VCB[VCB$W_TRANS] = 1;                ! transaction count
;  509          1038  2 PROTO_VCB[VCB$W_MCOUNT] = 1;               ! mount count
;  510          1039  2
;  511          1040  2 PROTO_VCB[VCB$V_ERASE] = .HOME_BLOCK[HM2$V_ERASE];
;  512          1041  2 PROTO_VCB[VCB$V_NOHIGHWATER] = .HOME_BLOCK[HM2$V_NOHIGHWATER];
;  513          1042  2
;  514          1043  2 IF .MOUNT_OPTIONS[OPT_GROUP]
;  515          1044  2 THEN PROTO_VCB[VCB$V_GROUP] = 1;
;  516          1045  2 IF .MOUNT_OPTIONS[OPT_SYSTEM]
;  517          1046  2 THEN PROTO_VCB[VCB$V_SYSTEM] = 1;
;  518          1047  2
;  519          1048  2 !
;  520          1049  2 ! Copy volume serial number from home block to VCB.
;  521          1050  2 !
;  522          1051  2 PROTO_VCB [VCB$L_SERIALNUM] = .HOME_BLOCK [HM2$L_SERIALNUM];
;  523          1052  2
;  524          1053  2 IF .MOUNT_OPTIONS[OPT_IS_FILES11]
;  525          1054  3 AND NOT (.MOUNT_OPTIONS[OPT_FOREIGN] AND .MOUNT_OPTIONS[OPT_LABEL])
;  526          1055  2 THEN
;  527          1056  2                                           ! volume label, blank filled
;  528          1057  2     CH$MOVE (HM2$S_VOLNAME, HOME_BLOCK[HM2$T_VOLNAME], PROTO_VCB[VCB$T_VOLNAME])
;  529          1058  2 ELSE
;  530          1059  2     CH$COPY (.LABEL_STRING[0], .LABEL_STRING[1], ' ',
;  531          1060  2              VCB$S_VOLNAME, PROTO_VCB[VCB$T_VOLNAME]);
;  532          1061  2
;  533          1062  2 IF NOT .MOUNT_OPTIONS[OPT_FOREIGN]
;  534          1063  2 THEN
;  535          1064  3     BEGIN
;  536          1065  3                                           ! relative volume number
;  537          1066  3     IF .HOME_BLOCK[HM2$W_RVN] GEQU 256
;  538          1067  3     OR .HOME_BLOCK[HM2$W_SETCOUNT] GEQU 256
;  539          1068  3     THEN ERR_EXIT (SS$_FILESTRUCT);
;  540          1069  3     PROTO_VCB[VCB$W_RVN] = .HOME_BLOCK[HM2$W_RVN];
;  541          1070  3     CURRENT_RVN = .HOME_BLOCK[HM2$W_RVN];
;  542          1071  3
;  543          1072  3     PROTO_VCB[VCB$L_HOMELBN] = .HOMEBLOCK_LBN; ! home block LBN
;  544          1073  3     PROTO_VCB[VCB$L_HOME2LBN] = .HOME_BLOCK[HM2$L_ALHOMELBN];
```

```
;  545          1074  3
;  546          1075  3        IF .PROTO_VCB[VCB$L_HOMELBN] EQL .PROTO_VCB[VCB$L_HOME2LBN]
;  547          1076  3        THEN
;  548          1077  3 4          BEGIN
;  549          1078  3 4          PROTO_VCB[VCB$V_HOMBLKBAD] = 1;
;  550          1079  3 4          ERR_MESSAGE (MOUN$_HOMBLKBAD);
;  551          1080  3 4          END;
;  552          1081  3
;  553          1082  3                                          ! index file bitmap LBN
;  554          1083  3        PROTO_VCB[VCB$L_IBMAPLBN] = .HOME_BLOCK[HM2$L_IBMAPLBN];
;  555          1084  3        PROTO_VCB[VCB$L_IXHDR2LBN] = .HOME_BLOCK[HM2$L_ALTIDXLBN];
;  556          1085  3                                          ! volume cluster factor
;  557          1086  3        PROTO_VCB[VCB$W_CLUSTER] = .HOME_BLOCK[HM2$W_CLUSTER];
;  558          1087  3                                          ! device blocking factor
;  559          1088  3        PROTO_VCB[VCB$B_BLOCKFACT] = (.DEVICE_CHAR[DIB$B_SECTORS]
;  560          1089  3                                      * .DEVICE_CHAR[DIB$B_TRACKS]
;  561          1090  3                                      * .DEVICE_CHAR[DIB$W_CYLINDERS])
;  562          1091  3                                      / .DEVICE_CHAR[DIB$L_MAXBLOCK];
;  563          1092  3                                          ! default window size
;  564          1093  3        PROTO_VCB[VCB$B_WINDOW] = .HOME_BLOCK[HM2$B_WINDOW];
;  565          1094  3        IF .PROTO_VCB[VCB$B_WINDOW] EQL 0
;  566          1095  3        THEN PROTO_VCB[VCB$B_WINDOW] = 7;
;  567          1096  3        IF .MOUNT_OPTIONS[OPT_SYSTEM]
;  568          1097  3        THEN PROTO_VCB[VCB$B_WINDOW] = .ACP$GB_WINDOW;
;  569          1098  3        IF .MOUNT_OPTIONS[OPT_WINDOW]
;  570          1099  3        THEN PROTO_VCB[VCB$B_WINDOW] = .WINDOW;
;  571          1100  3                                          ! directory LRU limit
;  572          1101  3        PROTO_VCB[VCB$B_LRU_LIM] = .HOME_BLOCK[HM2$B_LRU_LIM];
;  573          1102  3        IF .MOUNT_OPTIONS[OPT_SYSTEM]
;  574          1103  3        THEN PROTO_VCB[VCB$B_LRU_LIM] = .ACP$GW_SYSACC;
;  575          1104  3        IF .MOUNT_OPTIONS[OPT_ACCESSED]
;  576          1105  3        THEN PROTO_VCB[VCB$B_LRU_LIM] = .ACCESSED;
;  577          1106  3        IF .MOUNT_OPTIONS[OPT_NOCACHE]
;  578          1107  3            OR .STORED_CONTEXT [XQP]          !****TEMP****
;  579          1108  3        THEN PROTO_VCB[VCB$B_LRU_LIM] = 0;
;  580          1109  3                                          ! default file extend
;  581          1110  3        PROTO_VCB[VCB$W_EXTEND] = .HOME_BLOCK[HM2$W_EXTEND];
;  582          1111  3        IF .PROTO_VCB[VCB$W_EXTEND] EQL 0
;  583          1112  3        THEN PROTO_VCB[VCB$W_EXTEND] = 5;
;  584          1113  3        IF .MOUNT_OPTIONS[OPT_EXTENSION]
;  585          1114  3        THEN PROTO_VCB[VCB$W_EXTEND] = .EXTENSION;
;  586          1115  3                                          ! index file bitmap size
;  587          1116  3        PROTO_VCB[VCB$B_IBMAPSIZE] = .HOME_BLOCK[HM2$W_IBMAPSIZE];
;  588          1117  3        IF .HOME_BLOCK[HM2$W_IBMAPSIZE] GTRU 255
;  589          1118  3        THEN ERR_EXIT (SS$_FILESTRUCT);
;  590          1119  3                                          ! maximum number of files
;  591          1120  3        PROTO_VCB[VCB$L_MAXFILES] = .HOME_BLOCK[HM2$L_MAXFILES];
;  592          1121  3        IF .HOME_BLOCK[HM2$L_MAXFILES] GTRU 255^12
;  593          1122  3        THEN ERR_EXIT (SS$_FILESTRUCT);
;  594          1123  3        PROTO_VCB[VCB$V_EXTFID] = 1;
;  595          1124  3
;  596          1125  3        PROTO_VCB[VCB$B_RESFILES] = .HOME_BLOCK[HM2$W_RESFILES];
;  597          1126  3        IF .HOME_BLOCK[HM2$W_RESFILES] GTRU 255
;  598          1127  3        THEN ERR_EXIT (SS$_FILESTRUCT);
;  599          1128  3
;  600          1129  3        IF .MOUNT_OPTIONS[OPT_WTHRU]
;  601          1130  3            OR .STORED_CONTEXT [XQP]          !****TEMP*****
```

```
;  602          1131  3      THEN PROTO_VCB[VCB$V_WRITETHRU] = 1;
;  603          1132  3
;  604          1133  3      IF .MOUNT_OPTIONS[OPT_NOCACHE]
;  605          1134  3      THEN PROTO_VCB[VCB$V_NOCACHE] = 1;
;  606          1135  3
;  607          1136  3  ! Quota file is always on RVN 1.
;  608          1137  3  !
;  609          1138  3
;  610          1139  3      IF .CURRENT_RVN LEQU 1
;  611          1140  3      THEN
;  612          1141  3          PROTO_VCB[VCB$W_QUOSIZE] = .QUO_CACHE
;  613          1142  3      ELSE
;  614          1143  3          QUO_CACHE = 0;
;  615          1144  3
;  616          1145  3      CH$MOVE (HM2$S_RETAINMIN, HOME_BLOCK[HM2$Q_RETAINMIN], PROTO_VCB[VCB$Q_RETAINMIN]);
;  617          1146  3      CH$MOVE (HM2$S_RETAINMAX, HOME_BLOCK[HM2$Q_RETAINMAX], PROTO_VCB[VCB$Q_RETAINMAX]);
;  618          1147  3
;  619          1148  3  ! Now read the index file header, verify it, and initialize the prototype
;  620          1149  3  ! index file FCB. If the primary header is no good, try for the secondary.
;  621          1150  3  !
;  622          1151  3
;  623          1152  3      HEADER_LBN = .PROTO_VCB[VCB$L_IBMAPLBN] + .PROTO_VCB[VCB$B_IBMAPSIZE];
;  624          1153  3      STATUS = READ_BLOCK (.HEADER_LBN, BUFFER);
;  625          1154  3      IF NOT .STATUS OR NOT CHECK_HEADER2 (BUFFER, UPLIT WORD (1, 1, 0))
;  626          1155  3      THEN
;  627          1156  4          BEGIN
;  628          1157  4          USER_STATUS = 1;
;  629          1158  4          PROTO_VCB[VCB$V_IDXHDRBAD] = 1;
;  630          1159  4          PROTO_VCB[VCB$V_NOALLOC] = 1;
;  631          1160  4          ERR_MESSAGE (MOUN$_IDXHDRBAD);
;  632          1161  4          HEADER_LBN = .PROTO_VCB[VCB$L_IXHDR2LBN];
;  633          1162  4          STATUS = READ_BLOCK (.HEADER_LBN, BUFFER);
;  634          1163  3          END;
;  635          1164  3      IF NOT .STATUS THEN ERR_EXIT (.STATUS);
;  636          1165  3      IF NOT CHECK_HEADER2 (BUFFER, UPLIT WORD (1, 1, 0))
;  637          1166  3      THEN
;  638          1167  3          ERR_EXIT (SS$_NOHOMEBLK);
;  639          1168  3
;  640          1169  3      CH$FILL (0, FCB$C_LENGTH, PROTO_FCB);
;  641          1170  3      PROTO_FCB[FCB$L_STVBN] = 1;
;  642          1171  3      INIT_FCB2 (PROTO_FCB, BUFFER);
;  643          1172  3      PROTO_FCB[FCB$W_ACNT] = 1;
;  644          1173  3      PROTO_FCB[FCB$W_REFCNT] = 1;
;  645          1174  3
;  646          1175  3  ! Build the prototype index file window.
;  647          1176  3  !
;  648          1177  3
;  649          1178  3      CH$FILL (0, WCB$C_LENGTH, PROTO_WCB);
;  650          1179  3      PROTO_WCB[WCB$W_SIZE] = WCB$C_LENGTH + WINDOW_SIZE;
;  651          1180  3      PROTO_WCB[WCB$V_READ] = 1;
;  652          1181  3      TURN_WINDOW2 (PROTO_WCB, BUFFER, 3, 1, .PROTO_VCB[VCB$W_RVN]);
;  653          1182  3
;  654          1183  3  ! Now read the storage map file header and find the starting LBN of the
;  655          1184  3  ! storage map. Note that the storage map size is computed from the volume
;  656          1185  3  ! size and cluster factor, since the storage map file is rounded up to the
;  657          1186  3  ! next cluster boundary.
;  658          1187  3  !
```

```
;  659          1188  3
;  660          1189  3      STATUS = READ_BLOCK (.PROTO_VCB[VCB$L_IBMAPLBN] + .PROTO_VCB[VCB$B_IBMAPSIZE] + 1, BUFFER);
;  661          1190  3      IF NOT .STATUS OR NOT CHECK_HEADER2 (BUFFER, UPLIT WORD (2, 2, 0))
;  662          1191  3      THEN
;  663          1192  4          BEGIN
;  664          1193  4
;  665          1194  4  ! The shared file system cannot tolerate failure to read the storage
;  666          1195  4  ! control block, because that is where the volume label used for
;  667          1196  4  ! locking is stored.
;  668          1197  4  !
;  669          1198  4  ! If NOALLOC could be believed clusterwide such that we were guaranteed
;  670          1199  4  ! it was safe to proceed without doing any locking, failure to get
;  671          1200  4  ! SCB$T_VOLOCKNAME could be tolerated.  However, being able to issue
;  672          1201  4  ! an unlock control function makes that difficult.
;  673          1202  4  !
;  674          1203  4
;  675          1204  4          IF .STORED_CONTEXT [XQP]
;  676          1205  4          THEN
;  677          1206  4              IF .STATUS EQL SS$_VOLINV
;  678          1207  4              THEN
;  679          1208  4                  ERR_EXIT (SS$_VOLINV)
;  680          1209  4              ELSE
;  681          1210  4                  ERR_EXIT (MOUN$_MAPHDRBAD);
;  682          1211  4
;  683          1212  4          ERR_MESSAGE (MOUN$_MAPHDRBAD);
;  684          1213  4          PROTO_VCB[VCB$V_NOALLOC] = 1;
;  685          1214  4          END
;  686          1215  4
;  687          1216  3      ELSE
;  688          1217  4          BEGIN
;  689          1218  4          MAP_POINTER = BUFFER + .BUFFER[FH2$B_MPOFFSET]*2;
;  690          1219  4          GET_MAP_POINTER ();
;  691          1220  6          COUNT = ((.DEVICE_CHAR[DIB$L_MAXBLOCK] + .PROTO_VCB[VCB$W_CLUSTER] - 1)
;  692          1221  4                  / .PROTO_VCB[VCB$W_CLUSTER] + 4095) / 4096;
;  693          1222  4          IF .COUNT GTRU 255
;  694          1223  4          THEN ERR_EXIT (SS$_FILESTRUCT);
;  695          1224  4
;  696          1225  4          PROTO_VCB[VCB$L_SBMAPLBN] = .LBN + 1;
;  697          1226  4          PROTO_VCB[VCB$B_SBMAPSIZE] = .COUNT;
;  698          1227  4
;  699          1228  4  ! Now read the storage control block and check the various dirty bits, and
;  700          1229  4  ! issue messages if the volume was not properly dismounted.  Then set the
;  701          1230  4  ! appropriate bits and rewrite the storage control block.  If the write fails,
;  702          1231  4  ! write-lock the volume.
;  703          1232  4  !
;  704          1233  4
;  705          1234  4          STATUS = READ_BLOCK (.LBN, BUFFER);
;  706          1235  4          IF NOT .STATUS
;  707          1236  4          THEN
;  708          1237  4
;  709          1238  4  ! See comment above on failure to read sbm header.
;  710          1239  4  !
;  711          1240  4              IF .STATUS EQL SS$_VOLINV
;  712          1241  4              THEN
;  713          1242  4                  ERR_EXIT (SS$_VOLINV)
;  714          1243  4              ELSE
;  715          1244  4                  ERR_EXIT (MOUN$_BITMAPERR, 0, .STATUS);
```

```
;  716          1245  4
;  717          1246  4          IF .BUFFER[SCB$V_MAPDIRTY]
;  718          1247  4          THEN
;  719          1248  5              BEGIN
;  720          1249  5              ERR_MESSAGE (MOUN$_BITMAPINV);
;  721          1250  5              PROTO_VCB[VCB$V_NOALLOC] = 1;
;  722          1251  4              END;
;  723          1252  4
;  724          1253  4      ! Get volume lock and establish volume lock name.
;  725          1254  4      !
;  726          1255  4
;  727          1256  4          GET_VOLUME_LOCK_NAME ();
;  728          1257  4
;  729          1258  4          VOLOCK_COUNT = 0;
;  730          1259  4
;  731          1260  4          IF .STORED_CONTEXT [XQP]
;  732          1261  4          THEN
;  733          1262  5              BEGIN
;  734          1263  6              IF NOT (STATUS = KERNEL_CALL (GET_VOLUME_LOCK))
;  735          1264  5              THEN
;  736          1265  5                  ERR_EXIT (.STATUS);
;  737          1266  5
;  738          1267  5              VOLOCK_COUNT = .VOLOCK_COUNT - 1;           ! Don't count ourself.
;  739          1268  5
;  740          1269  5              IF .DEV_CTX [DC_NOTFIRST_MNT] NEQ .VOL_CTX [VC_NOTFIRST_MNT]
;  741          1270  5              THEN
;  742          1271  5                  ERR_EXIT (MOUN$_VOLALRMNT);
;  743          1272  5
;  744          1273  4              END;
;  745          1274  4
;  746          1275  4          CH$MOVE (8, BUFFER [SCB$Q_MOUNTTIME], PROTO_VCB [VCB$Q_MOUNTTIME]);
;  747          1276  4
;  748          1277  4          IF NOT .PROTO_VCB[VCB$V_NOALLOC]
;  749          1278  4          AND .MOUNT_OPTIONS[OPT_WRITE]
;  750          1279  4          THEN
;  751          1280  5              BEGIN
;  752          1281  5
;  753          1282  5              IF NOT .DEV_CTX [DC_NOTFIRST_MNT]   ! i.e., first
;  754          1283  5              THEN
;  755          1284  6                  BEGIN
;  756          1285  6                  CH$MOVE (12, PROTO_VCB [VCB$T_VOLCKNAM], BUFFER [SCB$T_VOLOCKNAME]);
;  757          1286  6                  $GETTIM (TIMADR = BUFFER [SCB$Q_MOUNTTIME]);
;  758          1287  5                  END;
;  759          1288  5
;  760          1289  5              IF .BUFFER [SCB$W_WRITECNT] NEQ .VOLOCK_COUNT
;  761          1290  5              THEN
;  762          1291  5
;  763          1292  5      ! If the count of volume locks does not match the count in the
;  764          1293  5      ! storage control block, someone that once mounted this volume
;  765          1294  5      ! did not dismount it.
;  766          1295  5      !
;  767          1296  5      ! Set the count straight now, but also note in status2 which caches
;  768          1297  5      ! need rebuilding by ORing the STATUS flags into it.
;  769          1298  5      ! The STATUS2 flags are only cleared upon successful completion
;  770          1299  5      ! of a rebuild, so we will continue to attempt a rebuild until
;  771          1300  5      ! the volume is actually rebuilt.
;  772          1301  5      !
```

```
;  773          1302  5
;  774          1303  6                  BEGIN
;  775          1304  6                  BUFFER [SCB$L_STATUS2] = .BUFFER [SCB$L_STATUS2]
;  776          1305  6                                          OR .BUFFER [SCB$L_STATUS];
;  777          1306  6                  BUFFER [SCB$W_WRITECNT] = .VOLOCK_COUNT;
;  778          1307  5                  END;
;  779          1308  5
;  780          1309  5              IF .BUFFER [SCB$V_MAPALLOC2] OR .BUFFER [SCB$V_FILALLOC2]
;  781          1310  5              THEN
;  782          1311  5                  CLEANUP_FLAGS [CLF_REBUILD] = 1;
;  783          1312  5
;  784          1313  5              IF .BUFFER [SCB$V_QUODIRTY2]
;  785          1314  5                  AND .CURRENT_RVN LEQU 1
;  786          1315  5              THEN
;  787          1316  6                  BEGIN
;  788          1317  6                  IF NOT .MOUNT_OPTIONS [OPT_NOQUOTA]
;  789          1318  6                  THEN
;  790          1319  6                      CLEANUP_FLAGS [CLF_REBUILDQUO] = 1;
;  791          1320  6                  END
;  792          1321  5              ELSE
;  793          1322  5                  BUFFER [SCB$V_QUODIRTY2] = 0;
;  794          1323  5
;  795          1324  5              BUFFER [SCB$W_WRITECNT] = .BUFFER [SCB$W_WRITECNT] + 1;
;  796          1325  5
;  797          1326  5  ! Note which caches we are enabling by setting the corresponding flag
;  798          1327  5  ! in the SCB.  These may already be set if
;  799          1328  5  ! the disk has been mounted elsewhere in the cluster with the
;  800          1329  5  ! same cache enabled.
;  801          1330  5  !
;  802          1331  5
;  803          1332  5              IF .EXT_CACHE NEQ 0
;  804          1333  5              THEN
;  805          1334  5                  BUFFER [SCB$V_MAPALLOC] = 1;
;  806          1335  5
;  807          1336  5              IF .FID_CACHE NEQ 1         ! 1 is no caching
;  808          1337  5              THEN
;  809          1338  5                  BUFFER [SCB$V_FILALLOC] = 1;
;  810          1339  5
;  811          1340  5  ! Note that we don't know yet whether quotas will be enabled until
;  812          1341  5  ! we actually try to turn them on in MAKE_DISK_MOUNT.  So far all
;  813          1342  5  ! we know is that we intend to turn them on if a quota.sys file is
;  814          1343  5  ! really there.
;  815          1344  5  !
;  816          1345  5
;  817          1346  6              IF (.QUO_CACHE NEQ 0 AND .CURRENT_RVN LEQU 1)
;  818          1347  5                  AND NOT .MOUNT_OPTIONS [OPT_NOQUOTA]
;  819          1348  5              THEN
;  820          1349  5                  BUFFER [SCB$V_QUODIRTY] = 1;
;  821          1350  5
;  822          1351  5  ! NOTE:  This read/write of the SCB is using the volume lock to
;  823          1352  5  ! serialize with the DISMOUNT subfunction (ACPCONTROL qio) which
;  824          1353  5  ! lowers the writecnt.  It does NOT correctly serialize with the
;  825          1354  5  ! REBUILD routine which only holds the volume blocking lock (LOCK_VOLUME).
;  826          1355  5  ! MOUNT does not respect the blocking lock because there isn't enough
;  827          1356  5  ! state yet to do it (volume sets are the problem).
;  828          1357  5  ! The correct solution here is probably to not rewrite the SCB at all
;  829          1358  5  ! in this leg of code (in mount), but rather have the file system do
```

```
;  830          1359  5 ! is as part of the MOUNT function QIO issued from START_ACP (called
;  831          1360  5 ! from the MAKE_DISK_MOUNT routine).  It can then respect the volume
;  832          1361  5 ! blocking lock (LOCK_VOL function) and interlock correctly with
;  833          1362  5 ! the bitmap rebuild, which does need to rewrite it.  The LOCK_VOL function
;  834          1363  5 ! should also make the lock count vs writecnt test and OR the STATUS flags
;  835          1364  5 ! into the STATUS2 flags if the counts mismatch, and rewrite the SCB
;  836          1365  5 ! (in an interlocked fashion) for the bitmap rebuilder to look at and
;  837          1366  5 ! correctly determine whether a rebuild is really necessary when it
;  838          1367  5 ! actually executes.
;  839          1368  5 ! This still leaves the problem of what block can mount read and attempt
;  840          1369  5 ! to write back to determine whether the volume is write-locked or not.
;  841          1370  5 ! All storage bitmap and index file bitmap blocks are really off limits
;  842          1371  5 ! because they are read and written by the bitmap rebuilder under the
;  843          1372  5 ! volume blocking lock (LOCK_VOL).  The home block, maybe?
;  844          1373  5 !
;  845          1374  5
;  846          1375  5                 CHECKSUM (BUFFER);
;  847          1376  5                 STATUS = WRITE_BLOCK (.LBN, BUFFER);
;  848          1377  5
;  849          1378  5 ! Bump storage bitmap sequence number in the volume lock value block
;  850          1379  5 ! to invalidate potential copies in file system caches elsewhere.
;  851          1380  5 !
;  852          1381  5
;  853          1382  5                 VOL_CTX [VC_SBMSEQNUM] = .VOL_CTX [VC_SBMSEQNUM] + 1;
;  854          1383  5
;  855          1384  5                 IF .STATUS
;  856          1385  5                 THEN
;  857          1386  5                     CH$MOVE (8, BUFFER [SCB$Q_MOUNTTIME],
;  858          1387  5                                                 PROTO_VCB [VCB$Q_MOUNTTIME])
;  859          1388  5                 ELSE
;  860          1389  6                     BEGIN
;  861          1390  6                     IF .STATUS EQL SS$_VOLINV
;  862          1391  6                     THEN
;  863          1392  6                         ERR_EXIT (SS$_VOLINV);
;  864          1393  6                     IF .STATUS EQL SS$_WRITLCK
;  865          1394  6                     THEN ERR_MESSAGE (MOUN$_WRITELOCK)
;  866          1395  6                     ELSE ERR_MESSAGE (MOUN$_WRITESCB, 0, .STATUS);
;  867          1396  6                     MOUNT_OPTIONS[OPT_WRITE] = 0;
;  868          1397  5                     END;
;  869          1398  5
;  870          1399  5                 CLEANUP_FLAGS[CLF_CLEANSCB] = 1;
;  871          1400  5
;  872          1401  4                 END;
;  873          1402  4
;  874          1403  4
;  875          1404  4 ! If this is not the first mount for this device, make sure
;  876          1405  4 ! essential mount parameters are consistent with other mounts
;  877          1406  4 ! elsewhere.
;  878          1407  4 ! For either the first mount of a cluster available device, or
;  879          1408  4 ! for mounts of local disks, the routine is not called.
;  880          1409  4 !
;  881          1410  4
;  882          1411  4           IF .DEV_CTX [DC_NOTFIRST_MNT]
;  883          1412  4           THEN
;  884          1413  4               CHECK_CLUSTER_SANITY();
;  885          1414  4
;  886          1415  4
```

```
;   887          1416  4 ! Scan the index file bitmap from the end backwards looking for the highest
;   888          1417  4 ! file number. Compute its index file VBN and check against the index file
;   889          1418  4 ! EOF. If the EOF is short, set the EOF delta high so that the first create
;   890          1419  4 ! will update the index file header.
;   891          1420  4 ! If this is not the initial mount of the volume, simply copy the index
;   892          1421  4 ! file eof from the value block.
;   893          1422  4 !
;   894          1423  4
;   895          1424  4         IF .VOL_CTX [VC_NOTFIRST_MNT]
;   896          1425  4         THEN
;   897          1426  4             PROTO_FCB [FCB$L_EFBLK] = .VOL_CTX [VC_IDXFILEOF]
;   898          1427  4         ELSE
;   899          1428  4         IDX_SCAN:
;   900          1429  5             BEGIN
;   901          1430  5             DECR J FROM .PROTO_VCB[VCB$B_IBMAPSIZE] - 1 TO 0
;   902          1431  5             DO
;   903          1432  6                 BEGIN
;   904          1433  6                 MAP BUFFER : VECTOR;
;   905          1434  6                 STATUS = READ_BLOCK (.PROTO_VCB[VCB$L_IBMAPLBN] + .J, BUFFER);
;   906          1435  6                 IF NOT .STATUS
;   907          1436  6                 THEN
;   908          1437  7                     BEGIN
;   909          1438  7                     IF .STATUS EQL SS$_VOLINV
;   910          1439  7                     THEN
;   911          1440  7                         ERR_EXIT (SS$_VOLINV)
;   912          1441  7                     ELSE
;   913          1442  7                         ERR_MESSAGE (MOUN$_IDXMAPERR, 0, .STATUS);
;   914          1443  7                     PROTO_VCB[VCB$V_NOALLOC] = 1;
;   915          1444  7                     IDX_EOF = 0;
;   916          1445  7                     LEAVE IDX_SCAN;
;   917          1446  6                     END;
;   918          1447  6
;   919          1448  6                 DECR I FROM 127 TO 0
;   920          1449  6                 DO
;   921          1450  7                     BEGIN
;   922          1451  7                     IF .BUFFER[.I] NEQ 0
;   923          1452  7                     THEN
;   924          1453  8                         BEGIN
;   925          1454  8                         IDX_EOF = .J*4096 + .I*32 + LEFT_ONE (.BUFFER[.I])
;   926          1455  8                             + .PROTO_VCB[VCB$B_IBMAPSIZE] + .PROTO_VCB[VCB$W_CLUSTER]*4;
;   927          1456  8                         LEAVE IDX_SCAN;
;   928          1457  7                         END;
;   929          1458  6                     END;
;   930          1459  5                 END;
;   931          1460  4             END;                            ! end of block IDX_SCAN
;   932          1461  4
;   933          1462  4         IDX_EOF = MINU (.IDX_EOF, .PROTO_FCB[FCB$L_FILESIZE]);
;   934          1463  4         IF .IDX_EOF GTRU .PROTO_FCB[FCB$L_EFBLK]
;   935          1464  4         THEN
;   936          1465  5             BEGIN
;   937          1466  5             PROTO_FCB[FCB$L_EFBLK] = .IDX_EOF;
;   938          1467  5             PROTO_VCB[VCB$B_EOFDELTA] = 250;
;   939          1468  4             END;
;   940          1469  4
;   941          1470  4         VOL_CTX [VC_IDXFILEOF] = .PROTO_FCB [FCB$L_EFBLK];
;   942          1471  4
;   943          1472  4 ! Scan the storage map to compute the number of free blocks on the volume.
```

```
;  944     1473  4 !
;  945     1474  4
;  946     1475  4          IF .VOL_CTX [VC_NOTFIRST_MNT]
;  947     1476  4          THEN
;  948     1477  4              PROTO_VCB [VCB$L_FREE] = .VOL_CTX [VC_VOLFREE]
;  949     1478  4          ELSE
;  950     1479  5              BEGIN
;  951     1480  5
;  952     1481  5              FREE = 0;
;  953     1482  5              DECR J FROM .COUNT TO 1 DO
;  954     1483  6                  BEGIN
;  955     1484  6                  MAP BUFFER : VECTOR;
;  956     1485  6
;  957     1486  6                  LBN = .LBN + 1;
;  958     1487  6                  STATUS = READ_BLOCK (.LBN, BUFFER);
;  959     1488  6                  IF NOT .STATUS
;  960     1489  6                  THEN
;  961     1490  7                      BEGIN
;  962     1491  7                      IF .STATUS EQL SS$_VOLINV
;  963     1492  7                      THEN
;  964     1493  7                          ERR_EXIT (SS$_VOLINV)
;  965     1494  7                      ELSE
;  966     1495  7                          ERR_MESSAGE (MOUN$_BITMAPERR, 0, .STATUS);
;  967     1496  7                      PROTO_VCB[VCB$V_NOALLOC] = 1;
;  968     1497  6                      END;
;  969     1498  6
;  970     1499  6                  INCR I FROM 0 TO 127 DO
;  971     1500  7                      BEGIN
;  972     1501  7                      X = .BUFFER[.I];
;  973     1502  7                      IF .X NEQ 0
;  974     1503  7                      THEN
;  975     1504  8                          BEGIN
;  976     1505  8                          B2 = 0;
;  977     1506  8                          WHILE 1 DO
;  978     1507  9                              BEGIN
;  979     1508  9                              IF FFS (B2, %REF (32-.B2), X, B1)
;  980     1509  9                              THEN EXITLOOP;
;  981     1510  9                              FFC (B1, %REF (32-.B1), X, B2);
;  982     1511  9                              FREE = .FREE + .B2 - .B1;
;  983     1512  9                              IF .B2 GEQ 32 THEN EXITLOOP;
;  984     1513  8                              END;
;  985     1514  7                          END;
;  986     1515  6                      END;
;  987     1516  5                  END;
;  988     1517  5
;  989     1518  5              PROTO_VCB[VCB$L_FREE] = .FREE * .PROTO_VCB[VCB$W_CLUSTER];
;  990     1519  5              VOL_CTX [VC_VOLFREE] = .PROTO_VCB [VCB$L_FREE];
;  991     1520  4              END;
;  992     1521  4
;  993     1522  3          END;                                ! end of storage bitmap hdr read success
;  994     1523  3
;  995     1524  3      END                                     ! end of Files-11 specific mount processing
;  996     1525  2  ELSE
;  997     1526  3      BEGIN
;  998     1527  3
;  999     1528  3  ! This is a foreign mount. If this is a shared foreign mount,
; 1000     1529  3  ! take out the volume lock.
```

```
;  1001          1530  3 !
;  1002          1531  3 ! If this is not the first mount for this device, make sure
;  1003          1532  3 ! essential mount parameters are consistent with other mounts
;  1004          1533  3 ! elsewhere.
;  1005          1534  3 ! For either the first mount of a cluster available device, or
;  1006          1535  3 ! for mounts of local disks, the routine is not called.
;  1007          1536  3 !
;  1008          1537  3
;  1009          1538  3     IF NOT .MOUNT_OPTIONS [OPT_NOSHARE]
;  1010          1539  3     THEN
;  1011          1540  4         BEGIN
;  1012          1541  4         GET_VOLUME_LOCK_NAME ();
;  1013          1542  5         IF NOT (STATUS = KERNEL_CALL (GET_VOLUME_LOCK))
;  1014          1543  4         THEN
;  1015          1544  4             ERR_EXIT (.STATUS);
;  1016          1545  4         IF .DEV_CTX [DC_NOTFIRST_MNT] NEQ .VOL_CTX [VC_NOTFIRST_MNT]
;  1017          1546  4         THEN
;  1018          1547  4             ERR_EXIT (MOUN$_VOLALRMNT);
;  1019          1548  3         END;
;  1020          1549  3
;  1021          1550  3     IF .DEV_CTX [DC_NOTFIRST_MNT]
;  1022          1551  3     THEN
;  1023          1552  3         CHECK_CLUSTER_SANITY();
;  1024          1553  3
;  1025          1554  2     END;                                    ! end of foreign-specific mount processing
;  1026          1555  2
;  1027          1556  2
;  1028          1557  2 ! Finally call the kernel mode routine to make it all real. Note that all the
;  1029          1558  2 ! hookups, including generating the mounted volume list entry, are done
;  1030          1559  2 ! within one kernel mode call so that they are uninterruptible by the user.
;  1031          1560  2 !
;  1032          1561  2
;  1033          1562  2 IF .MOUNT_OPTIONS[OPT_OVR_LOCK]
;  1034          1563  2 THEN PROTO_VCB[VCB$V_NOAL[0C] = 0;
;  1035          1564  2
;  1036          1565  2 STATUS = KERNEL_CALL (MAKE_DISK_MOUNT);
;  1037          1566  2 IF NOT .STATUS
;  1038          1567  2 THEN
;  1039          1568  3     BEGIN
;  1040          1569  3     IF .STATUS[STS$V_SEVERITY] EQL STS$K_SEVERE
;  1041          1570  3     THEN ERR_EXIT (.STATUS)
;  1042          1571  3     ELSE
;  1043          1572  4         BEGIN
;  1044          1573  4         IF .IO_STATUS
;  1045          1574  4         THEN ERR_MESSAGE (.STATUS)
;  1046          1575  4         ELSE ERR_MESSAGE (.STATUS, 0, .IO_STATUS<0,16>);
;  1047          1576  3         END;
;  1048          1577  2     END;
;  1049          1578  2
;  1050          1579  2 ! If this volume is being bound into a volume set, now do the on-disk
;  1051          1580  2 ! modifications.
;  1052          1581  2 !
;  1053          1582  2
;  1054          1583  2 CLEANUP_FLAGS[CLF_DISMOUNT] = 1;          ! cleanup from here requires a full dismount
;  1055          1584  2
;  1056          1585  2 IF TESTBITSC (MOUNT_OPTIONS[OPT_DO_BIND])
;  1057          1586  2 THEN BIND_VOLUME ();
```

```
;  1058          1587  2
;  1059          1588  2 ! Announce that the volume is mounted.
;  1060          1589  2 !
;  1061          1590  2
;  1062          1591  2 ERR_MESSAGE (MOUN$_MOUNTED, 3, VCB$S_VOLNAME, PROTO_VCB[VCB$T_VOLNAME], PHYS_NAME[.DEVICE_INDEX*2]);
;  1063          1592  2
;  1064          1593  2 !
;  1065          1594  2 ! If a FILES-11 volume is mounted with a reduced block cache, output the
;  1066          1595  2 ! appropriate informational message.
;  1067          1596  2 !
;  1068          1597  2
;  1069          1598  3 IF  ( NOT .CACHE_STATUS )
;  1070          1599  3 AND ( NOT .MOUNT_OPTIONS [OPT_FOREIGN] )
;  1071          1600  2 THEN
;  1072          1601  2     ERR_MESSAGE (MOUN$_REDCACHE);
;  1073          1602  2
;  1074          1603  2 ! Earlier in this routine, the CLF_REBUILD flag was set if either of
;  1075          1604  2 ! the bitmaps (storage and file number) needs rebuilding.  CLF_REBUILDQUO
;  1076          1605  2 ! was set if the quota file needs rebuilding and quota are in fact
;  1077          1606  2 ! enabled.  Check if anything needs rebuilding and, if so, whether
;  1078          1607  2 ! it should be done now.
;  1079          1608  2 !
;  1080          1609  2
;  1081          1610  2 IF .CLEANUP_FLAGS [CLF_REBUILD] OR .CLEANUP_FLAGS [CLF_REBUILDQUO]
;  1082          1611  2 THEN
;  1083          1612  2     IF .MOUNT_OPTIONS [OPT_NOREBUILD]
;  1084          1613  2     THEN
;  1085          1614  3         BEGIN
;  1086          1615  3         ERR_MESSAGE (MOUN$_REBLDREQD);
;  1087          1616  3         CLEANUP_FLAGS [CLF_REBUILD] = 0;
;  1088          1617  3         CLEANUP_FLAGS [CLF_REBUILDQUO] = 0;
;  1089          1618  3         END
;  1090          1619  2     ELSE
;  1091          1620  2         IF .CLEANUP_FLAGS [CLF_REBUILDQUO]
;  1092          1621  2         THEN
;  1093          1622  2             CLEANUP_FLAGS [CLF_REBUILD] = 1;
;  1094          1623  2
;  1095          1624  1 END;                                   ! end of routine MOUNT_DISK2

                                                    .TITLE  MOUDK2
                                                    .IDENT  \V04-002\

                                                    .PSECT  $PLIT$,NOWRT,NOEXE,2

                                0000  0001  0001  00000 P.AAA:  .WORD   1, 1, 0                 ;
                                0000  0001  0001  00006 P.AAB:  .WORD   1, 1, 0                 ;
                                0000  0002  0002  0000C P.AAC:  .WORD   2, 2, 0                 ;

                                                    .PSECT  $OWN$,NOEXE,2

                                              00000 IO_STATUS:
                                                    .BLKB   8

                                                    .PSECT  $GLOBAL$,NOEXE,2

                                              00000 BUFFER::.BLKB   512
```

```
00200 PROTO_VCB::
                .BLKB   236
002EC PROTO_FCB::
                .BLKB   180
003A0 PROTO_WCB::
                .BLKB   228
00484 VOLUME_UIC::
                .BLKB   4
00488 CACHE_STATUS::
                .BLKB   4

                .EXTRN  DEV_CTX, VOL_CTX
                .EXTRN  VOLOCK_COUNT, STORED_CONTEXT
                .EXTRN  MOUNT_OPTIONS, CLEANUP_FLAGS
                .EXTRN  DEVICE_CHAR, USER_STATUS
                .EXTRN  LABEL_STRING, DEVICE_INDEX
                .EXTRN  PHYS_NAME, STRUCT_NAME
                .EXTRN  DRIVE_COUNT, WINDOW
                .EXTRN  ACCESSED, EXTENSION
                .EXTRN  EXT_CACHE, FID_CACHE
                .EXTRN  QUO_CACHE, EXT_LIMIT
                .EXTRN  HOME_BLOCK, HOMEBLOCK_LBN
                .EXTRN  HEADER_LBN, CURRENT_RVN
                .EXTRN  CURRENT_VCB, CTL$GL_PHD
                .EXTRN  ACP$GW_EXTCACHE
                .EXTRN  ACP$GW_FIDCACHE
                .EXTRN  ACP$GW_QUOCACHE
                .EXTRN  ACP$GW_EXTLIMIT
                .EXTRN  ACP$GB_WRITBACK
                .EXTRN  ACP$GB_WINDOW, ACP$GW_SYSACC
                .EXTRN  CHECK_CLUSTER_SANITY
                .EXTRN  GET_VOLUME_LOCK
                .EXTRN  GET_VOLUME_LOCK_NAME
                .EXTRN  GET_UIC, CHECK_HEADER2
                .EXTRN  CHECKSUM, READ_BLOCK
                .EXTRN  WRITE_BLOCK, INIT_FCB2
                .EXTRN  TURN_WINDOW2, LEFT_ONE
                .EXTRN  GET_MAP_POINTER
                .EXTRN  BIND_VOLUME, SYS$CMKRNL
                .EXTRN  SYS$GETTIM

                .PSECT  $CODE$,NOWRT,2

                                        0FFC 00000      .ENTRY  MOUNT_DISK2, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 0789
                                                                R10,R11
                                5B      0000G   CF  9E  00002   MOVAB   MOUNT_OPTIONS, R11                  ;
                                5A      0000'   CF  9E  00007   MOVAB   BUFFER, R10                         ;
                                6D      08C4    CF  DE  0000C   MOVAL   1148, (FP)                          ; 0826
                        0000G   CF      0200    CA  9E  00011   MOVAB   PROTO_VCB, CURRENT_VCB              ; 0919
00EC    8F              00      6E              00  2C  00018   MOVC5   #0, (SP), #0, #236, PROTO_VCB        ; 0920
                                                0200    CA      0001F
                        0488    CA              01  D0  00022   MOVL    #1, CACHE_STATUS                  ; 0921
                                                7E  D4  00027   CLRL    -(SP)                             ; 0932
                                                5E  DD  00029   PUSHL   SP                                ;
                                                0000G   CF  9F  0002B   PUSHAB  GET_UIC                     ;
                    00000000G   9F              03  FB  0002F   CALLS   #3, a#SYS$CMKRNL                  ;
                                52              50  D0  00036   MOVL    R0, PROCESS_UIC                   ;
```

```
                                50 00000000G   9F   D0 00039          MOVL    @#CTL$GL_PHD, PRIVILEGE_MASK                    ;0933
                                     0484      CA   D4 00040          CLRL    VOLUME_UIC                                      ;0934
                     07      04   AB           01   E1 00044          BBC     #1, MOUNT_OPTIONS+4, 1$                         ;0935
                           0484   CA      0000G CF   D0 00049          MOVL    HOME_BLOCK+44, VOLUME_UIC                       ;0936
                                  0D        04  AB   E9 00050 1$:       BLBC    MOUNT_OPTIONS+4, 2$                             ;0939
                     09           60           15   E0 00054          BBS     #21, (PRIVILEGE_MASK), 2$                       ;0940
                                               07   13 00058          BEQL    2$                                              ;0941
                                  52      0484  CA   D1 0005A          CMPL    VOLUME_UIC, PROCESS_UIC                         ;0942
                                               31   12 0005F          BNEQ    6$
                                  19        03  AB   E8 00061 2$:       BLBS    MOUNT_OPTIONS+3, 3$                             ;0946
                     14      03   AB           01   E0 00065          BBS     #1, MOUNT_OPTIONS+3, 3$                         ;0947
                     0F      03   AB           02   E0 0006A          BBS     #2, MOUNT_OPTIONS+3, 3$                         ;0948
                     0A      03   AB           03   E0 0006F          BBS     #3, MOUNT_OPTIONS+3, 3$                         ;0949
                     05      03   AB           04   E0 00074          BBS     #4, MOUNT_OPTIONS+3, 3$                         ;0950
                     04      05   AB           05   E1 00079          BBC     #5, MOUNT_OPTIONS+5, 4$                         ;0951
                     10           60           12   E1 0007E 3$:       BBC     #18, (PRIVILEGE_MASK), 6$                       ;0953
                                               6B   95 00082 4$:       TSTB    MOUNT_OPTIONS                                   ;0957
                                               04   18 00084          BGEQ    5$
                     08           60           03   E1 00086          BBC     #3, (PRIVILEGE_MASK), 6$                        ;0958
                                  0D        01  AB   E9 0008A 5$:       BLBC    MOUNT_OPTIONS+1, 7$                             ;0962
                     09           60           02   E0 0008E          BBS     #2, (PRIVILEGE_MASK), 7$                        ;0963
                                               24   DD 00092 6$:       PUSHL   #36                                             ;0966
                00000000G      00              01   FB 00094          CALLS   #1, LIB$STOP
                     05      01   AB           03   E1 0009B 7$:       BBC     #3, MOUNT_OPTIONS+1, 8$                         ;0968
                           0484   CA           52   D0 000A0          MOVL    PROCESS_UIC, VOLUME_UIC                         ;0969
                     05      03   AB           04   E0 000A5 8$:       BBS     #4, MOUNT_OPTIONS+3, 9$                         ;0975
                           0000G  CF           04   88 000AA          BISB2   #4, STORED_CONTEXT                              ;0977
                                  50      0000G CF   3C 000AF 9$:       MOVZWL  HOME_BLOCK+38, R0                               ;0986
                                  31        05  AB   E9 000B4          BLBC    MOUNT_OPTIONS+5, 11$                            ;0983
                                               1D   13 000B8          BEQL    10$                                             ;0986
           0C      20      0000G  DF      0000G CF   2D 000BA          CMPC5   STRUCT_NAME, @STRUCT_NAME+4, #32, #12, -        ;0990
                                     0000G     CF      000C3                  HOME_BLOCK+460
                                               2F   13 000C6          BEQL    12$
                                00728194       8F   DD 000C8          PUSHL   #7504276                                        ;0991
                00000000G      00              01   FB 000CE          CALLS   #1, LIB$STOP
                                               20   11 000D5          BRB     12$                                             ;0986
           0C      20      0000G  DF      0000G CF   2C 000D7 10$:      MOVC5   STRUCT_NAME, @STRUCT_NAME+4, #32, #12, -        ;0997
                                     0000G     CF      000E0                  HOME_BLOCK+460
                             05   AB              02   88 000E3          BISB2   #2, MOUNT_OPTIONS+5                          ;0998
                                               0E   11 000E7          BRB     12$                                             ;0983
                                               0C   13 000E9 11$:      BEQL    12$                                             ;1004
                           0000G  CF              0C   D0 000EB          MOVL    #12, STRUCT_NAME                                ;1007
                           0000G  CF      0000G CF   9E 000F0          MOVAB   HOME_BLOCK+460, STRUCT_NAME+4                   ;1008
                                     0000G     CF   D5 000F7 12$:      TSTL    EXT_CACHE                                       ;1015
                                               09   12 000FB          BNEQ    13$
                           0000G  CF 00000000G 9F   3C 000FD          MOVZWL  @#ACP$GW_EXTCACHE, EXT_CACHE                    ;1016
                                        05     AB   95 00106 13$:      TSTB    MOUNT_OPTIONS+5                                 ;1017
                                               04   18 00109          BGEQ    14$
                                     0000G     CF   D4 0010B          CLRL    EXT_CACHE                                       ;1018
                                     0000G     CF   D5 0010F 14$:      TSTL    FID_CACHE                                       ;1020
                                               09   12 00113          BNEQ    15$
                           0000G  CF 00000000G 9F   3C 00115          MOVZWL  @#ACP$GW_FIDCACHE, FID_CACHE                    ;1021
                                        02     06  AB   E8 0011E 15$:      BLBS    MOUNT_OPTIONS+6, 16$                            ;1022
                                               05   12 00122          BNEQ    17$                                             ;1023
                           0000G  CF                  01   D0 00124 16$:      MOVL    #1, FID_CACHE                           ;1024
                                     0000G     CF   D5 00129 17$:      TSTL    QUO_CACHE                                       ;1026
                                               09   12 0012D          BNEQ    18$
```

```
                                0000G   CF 00000000G   9F   3C 0012F          MOVZWL  @#ACP$GW_QUOCACHE, QUO_CACHE                    ; 1027
                          04      06    AB             01   E1 00138 18$:     BBC     #1, MOUNT_OPTIONS+6, 19$                         ; 1028
                                                 0000G   CF   D4 0013D          CLRL    QUO_CACHE                                        ; 1029
                                                 0000G   CF   D5 00141 19$:     TSTL    EXT_LIMIT                                        ; 1031
                                                         09   12 00145          BNEQ    20$
                                0000G   CF 00000000G   9F   3C 00147          MOVZWL  @#ACP$GW_EXTLIMIT, EXT_LIMIT                     ; 1032
                                020C    CA             01   B0 00150 20$:     MOVW    #1, PROTO_VCB+12                                 ; 1037
                                024C    CA             01   B0 00155          MOVW    #1, PROTO_VCB+76                                 ; 1038
                    50          0000G   CF             01   02 EF 0015A       EXTZV   #2, #1, HOME_BLOCK+42, R0                        ; 1040
              0253  CA          01                     03   50 F0 00161       INSV    R0, #3, #1, PROTO_VCB+83
                    50          0000G   CF             01   03 EF 00168       EXTZV   #3, #1, HOME_BLOCK+42, R0                        ; 1041
              0253  CA          01                     04   50 F0 0016F       INSV    R0, #4, #1, PROTO_VCB+83
                                                         6B   95 00176          TSTB    MOUNT_OPTIONS                                    ; 1043
                                                         06   18 00178          BGEQ    21$
                                020B    CA               40   8F 88 0017A       BISB2   #64, PROTO_VCB+11                                ; 1044
                                        06               01   AB E9 00180 21$:     BLBC    MOUNT_OPTIONS+1, 22$                     ; 1045
                                020B    CA               80   8F 88 00184       BISB2   #128, PROTO_VCB+11                               ; 1046
                                0264    CA         0000G   CF   D0 0018A 22$:     MOVL    HOME_BLOCK+456, PROTO_VCB+100                    ; 1051
                          14      04    AB             01   E1 00191          BBC     #1, MOUNT_OPTIONS+4, 24$                         ; 1053
                          05      01    AB             03   E1 00196          BBC     #3, MOUNT_OPTIONS+1, 23$                         ; 1054
                                                   03    AB   95 0019B          TSTB    MOUNT_OPTIONS+3
                                                         0A   19 0019E          BLSS    24$
                    0214  CA        0000G   CF           0C   28 001A0 23$:     MOVC3   #12, HOME_BLOCK+472, PROTO_VCB+20                ; 1057
                                                         0C   11 001A8          BRB     25$
                OC              20      0000G   DF       0000G CF 2C 001AA 24$:     MOVC5   LABEL_STRING, @LABEL_STRING+4, #32, #12, -       ; 1060
                                                         0214  CA   001B3          PROTO_VCB+20
                          03      01    AB             03   E1 001B6 25$:     BBC     #3, MOUNT_OPTIONS+1, 26$                         ; 1062
                                                 0606    31   001BB          BRW     101$                                             ; 1066
                                0100    8F      0000G   CF   B1 001BE 26$:     CMPW    HOME_BLOCK+38, #256
                                                         09   1E 001C5          BGEQU   27$
                                0100    8F      0000G   CF   B1 001C7          CMPW    HOME_BLOCK+40, #256                              ; 1067
                                                         0C   1F 001CE          BLSSU   28$
                                        7E      08C0    8F   3C 001D0 27$:     MOVZWL  #2240, -(SP)                                     ; 1068
                          00000000G     00              01   FB 001D5          CALLS   #1, LIB$STOP
                                020E    CA      0000G   CF   B0 001DC 28$:     MOVW    HOME_BLOCK+38, PROTO_VCB+14                      ; 1069
                                0000G   CF      0000G   CF   3C 001E3          MOVZWL  HOME_BLOCK+38, CURRENT_RVN                       ; 1070
                                0224    CA      0000G   CF   D0 001EA          MOVL    HOMEBLOCK_LBN, PROTO_VCB+36                      ; 1072
                                0228    CA      0000G   CF   D0 001F1          MOVL    HOME_BLOCK+4, PROTO_VCB+40                       ; 1073
                                0228    CA      0224    CA   D1 001F8          CMPL    PROTO_VCB+36, PROTO_VCB+40                       ; 1075
                                                         12   12 001FF          BNEQ    29$
                                020B    CA                   04   88 00201       BISB2   #4, PROTO_VCB+11                                 ; 1078
                                        00729000             8F   DD 00206       PUSHL   #7507968                                         ; 1079
                          00000000G     00                   01   FB 0020C       CALLS   #1, LIB$SIGNAL
                                0230    CA      0000G   CF   D0 00213 29$:     MOVL    HOME_BLOCK+24, PROTO_VCB+48                      ; 1083
                                022C    CA      0000G   CF   D0 0021A          MOVL    HOME_BLOCK+8, PROTO_VCB+44                       ; 1084
                                023C    CA      0000G   CF   B0 00221          MOVW    HOME_BLOCK+14, PROTO_VCB+60                      ; 1086
                                        50      0000G   CF   9A 00228          MOVZBL  DEVICE_CHAR+8, R0                                ; 1089
                                        51      0000G   CF   9A 0022D          MOVZBL  DEVICE_CHAR+9, R1
                                        50              51   C4 00232          MULL2   R1, R0
                                        52      0000G   CF   3C 00235          MOVZWL  DEVICE_CHAR+10, R2                               ; 1090
                                        50              52   C4 0023A          MULL2   R2, R0
                    51                  50      0000G   CF   C7 0023D          DIVL3   DEVICE_CHAR+112, R0, R1                          ; 1091
                                0252    CA              51   90 00243          MOVB    R1, PROTO_VCB+82
                                0248    CA      0000G   CF   90 00248          MOVB    HOME_BLOCK+68, PROTO_VCB+72                      ; 1093
                                                         05   12 0024F          BNEQ    30$                                              ; 1094
                                0248    CA                   07   90 00251       MOVB    #7, PROTO_VCB+72                         ; 1095
                                        09      01      AB   E9 00256 30$:     BLBC    MOUNT_OPTIONS+1, 31$                             ; 1096
```

```
                0248   CA 00000000G   9F   90 00025A          MOVB    @#ACP$GB_WINDOW, PROTO_VCB+72                ; 1097
                07       03   AB      E9   00263 31$:        BLBC    MOUNT_OPTIONS+3, 32$                         ; 1098
                0248   CA     0000G   CF   90 00267           MOVB    WINDOW, PROTO_VCB+72                         ; 1099
                0249   CA     0000G   CF   90 0026E 32$:      MOVB    HOME_BLOCK+69, PROTO_VCB+73                  ; 1101
                09       01   AB      E9   00275             BLBC    MOUNT_OPTIONS+1, 33$                         ; 1102
                0249   CA 00000000G   9F   90 00279           MOVB    @#ACP$GW_SYSACC, PROTO_VCB+73                ; 1103
          07      03   AB             01   E1 00282 33$:      BBC     #1, MOUNT_OPTIONS+3, 34$                     ; 1104
                0249   CA     0000G   CF   90 00287           MOVB    ACCESSED, PROTO_VCB+73                       ; 1105
          06      06   AB             04   E0 0028E 34$:      BBS     #4, MOUNT_OPTIONS+6, 35$                     ; 1106
          04    0000G  CF             02   E1 00293           BBC     #2, STORED_CONTEXT, 36$                      ; 1107
                                0249   CA   94 00299 35$:      CLRB    PROTO_VCB+73                                 ; 1108
                023E   CA     0000G   CF   B0 0029D 36$:      MOVW    HOME_BLOCK+70, PROTO_VCB+62                  ; 1110
                                       05   12 002A4           BNEQ    37$                                          ; 1111
                023E   CA             05   B0 002A6           MOVW    #5, PROTO_VCB+62                             ; 1112
                              02       AB   95 002AB 37$:      TSTB    MOUNT_OPTIONS+2                              ; 1113
                                       07   18 002AE           BGEQ    38$                                          ; 1114
                023E   CA     0000G   CF   B0 002B0           MOVW    EXTENSION, PROTO_VCB+62                      ; 1116
                0238   CA     0000G   CF   90 002B7 38$:      MOVB    HOME_BLOCK+32, PROTO_VCB+56                  ; 1117
                00FF   8F     0000G   CF   B1 002BE           CMPW    HOME_BLOCK+32, #255                          ; 1118
                                       0C   1B 002C5           BLEQU   39$                                          ; 1120
                       7E     08C0    8F   3C 002C7           MOVZWL  #2240, -(SP)                                 ; 1121
          00000000G    00             01   FB 002CC           CALLS   #1, LIB$STOP                                 ; 1122
                0244   CA     0000G   CF   D0 002D3 39$:      MOVL    HOME_BLOCK+28, PROTO_VCB+68                  ; 1123
              000FF000 8F     0000G   CF   D1 002DA           CMPL    HOME_BLOCK+28, #1044480                      ; 1125
                                       0C   1B 002E3           BLEQU   40$                                          ; 1126
                       7E     08C0    8F   3C 002E5           MOVZWL  #2240, -(SP)                                 ; 1127
          00000000G    00             01   FB 002EA           CALLS   #1, LIB$STOP                                 ; 1129
                020B   CA             20   88 002F1 40$:      BISB2   #32, PROTO_VCB+11                            ; 1130
                024F   CA     0000G   CF   90 002F6           MOVB    HOME_BLOCK+34, PROTO_VCB+79                  ; 1131
                00FF   8F     0000G   CF   B1 002FD           CMPW    HOME_BLOCK+34, #255                          ; 1133
                                       0C   1B 00304           BLEQU   41$                                          ; 1134
                       7E     08C0    8F   3C 00306           MOVZWL  #2240, -(SP)                                 ; 1139
          00000000G    00             01   FB 0030B           CALLS   #1, LIB$STOP                                 ; 1141
          06      05   AB             06   E0 00312 41$:      BBS     #6, MOUNT_OPTIONS+5, 42$                     ; 1143
          05    0000G  CF             02   E1 00317           BBC     #2, STORED_CONTEXT, 43$                      ; 1145
                0253   CA             01   88 0031D 42$:      BISB2   #1, PROTO_VCB+83                             ; 1146
          05      06   AB             04   E1 00322 43$:      BBC     #4, MOUNT_OPTIONS+6, 44$                     ; 1152
                0253   CA             02   88 00327           BISB2   #2, PROTO_VCB+83                             ; 1153
                       01    0000G   CF   D1 0032C 44$:      CMPL    CURRENT_RVN, #1                              ; 1154
                                       09   1A 00331           BGTRU   45$                                          ; 1157
                0260   CA     0000G   CF   B0 00333           MOVW    QUO_CACHE, PROTO_VCB+96                      ; 1158
                                       04   11 0033A           BRB     46$                                          ; 1159
                                0000G  CF   D4 0033C 45$:      CLRL    QUO_CACHE                                    ; 1161
          026C  CA      0000G   CF         08   28 00340 46$:      MOVC3   #8, HOME_BLOCK+72, PROTO_VCB+108             ; 1163
          0274  CA      0000G   CF         08   28 00348           MOVC3   #8, HOME_BLOCK+80, PROTO_VCB+116             ; 1164
                       50     0238   CA     9A   00350           MOVZBL  PROTO_VCB+56, R0                             ; 1165
                0000G  CF     0230 DA40     9E   00355           MOVAB   @PROTO_VCB+48[R0], HEADER_LBN                ; 1166
                                       5A   DD 0035D           PUSHL   R10                                          ; 1167
                                0000G  CF   DD 0035F           PUSHL   HEADER_LBN                                   ; 1168
                0000G  CF             02   FB 00363           CALLS   #2, READ_BLOCK                               ; 1169
                       59             50   D0 00368           MOVL    R0, STATUS                                   ; 1170
                       0E             59   E9 0036B           BLBC    STATUS, 47$                                  ; 1171
                                0000'  CF   9F 0036E           PUSHAB  P.AAA                                        ; 1172
                                       5A   DD 00372           PUSHL   R10                                          ; 1173
                0000G  CF             02   FB 00374           CALLS   #2, CHECK_HEADER2                            ; 1174
                       2C             50   E8 00379           BLBS    R0, 48$                                      ; 1175
                0000G  CF             01   D0 0037C 47$:      MOVL    #1, USER_STATUS                              ; 1176
```

```
                    020B   CA                  18  88 00381          BISB2   #24, PROTO_VCB+11                          ; 1159
                                      00729008  8F  DD 00386          PUSHL   #7507976                                   ; 1160
                00000000G  00                   01  FB 0038C          CALLS   #1, LIB$SIGNAL
                    0000G  CF         022C      CA  D0 00393          MOVL    PROTO_VCB+44, HEADER_LBN                   ; 1161
                                                5A  DD 0039A          PUSHL   R10                                        ; 1162
                                        0000G   CF  DD 0039C          PUSHL   HEADER_LBN
                    0000G  CF                   02  FB 003A0          CALLS   #2, READ_BLOCK
                           59                   50  D0 003A5          MOVL    R0, STATUS
                           09                   59  E8 003A8  48$:     BLBS    STATUS, 49$                                ; 1164
                                                59  DD 003AB          PUSHL   STATUS
                00000000G  00                   01  FB 003AD          CALLS   #1, LIB$STOP
                                        0000'   CF  9F 003B4  49$:     PUSHAB  P.AAB                                      ; 1165
                                                5A  DD 003B8          PUSHL   R10
                    0000G  CF                   02  FB 003BA          CALLS   #2, CHECK_HEADER2
                           0C                   50  E8 003BF          BLBS    R0, 50$
                           7E         08E0     8F  3C 003C2          MOVZWL  #2272, -(SP)                               ; 1167
                00000000G  00                   01  FB 003C7          CALLS   #1, LIB$STOP
00B4    8F          00     6E                   00  2C 003CE  50$:     MOVC5   #0, (SP), #0, #180, PROTO_FCB              ; 1169
                                        02EC    CA      003D5
                    0318   CA                   01  D0 003D8          MOVL    #1, PROTO_FCB+44                           ; 1170
                                                5A  DD 003DD          PUSHL   R10                                        ; 1171
                                        02EC    CA  9F 003DF          PUSHAB  PROTO_FCB
                    0000G  CF                   02  FB 003E3          CALLS   #2, INIT_FCB2
                    0304   CA          00010001 8F  D0 003E8          MOVL    #65537, PROTO_FCB+24                       ; 1173
        30          00     6E                   00  2C 003F1          MOVC5   #0, (SP), #0, #48, PROTO_WCB               ; 1178
                                        03A0    CA      003F6
                    03A8   CA            E4     8F  9B 003F9          MOVZBW  #228, PROTO_WCB+8                          ; 1179
                    03AB   CA                   01  88 003FF          BISB2   #1, PROTO_WCB+11                           ; 1180
                           7E         020E     CA  3C 00404          MOVZWL  PROTO_VCB+14, -(SP)                        ; 1181
                                                01  DD 00409          PUSHL   #1
                                                03  DD 0040B          PUSHL   #3
                                                5A  DD 0040D          PUSHL   R10
                                        03A0    CA  9F 0040F          PUSHAB  PROTO_WCB
                        0000G  CF                   05  FB 00413          CALLS   #5, TURN_WINDOW2
                                                5A  DD 00418          PUSHL   R10                                        ; 1189
                               50       0238     CA  9A 0041A          MOVZBL  PROTO_VCB+56, R0
                               50       0230     CA  C0 0041F          ADDL2   PROTO_VCB+48, R0
                                          01     A0  9F 00424          PUSHAB  1(R0)
                        0000G  CF                   02  FB 00427          CALLS   #2, READ_BLOCK
                               59                   50  D0 0042C          MOVL    R0, STATUS
                               0E                   59  E9 0042F          BLBC    STATUS, 51$                                ; 1190
                                        0000'   CF  9F 00432          PUSHAB  P.AAC
                                                5A  DD 00436          PUSHL   R10
                        0000G  CF                   02  FB 00438          CALLS   #2, CHECK_HEADER2
                               38                   50  E8 0043D          BLBS    R0, 55$
1D              0000G  CF                           02  E1 00440  51$:     BBC     #2, STORED_CONTEXT, 54$                ; 1204
        00000254    8F                              59  D1 00446          CMPL    STATUS, #596                           ; 1206
                                                    07  12 0044D          BNEQ    52$
                               7E         0254     8F  3C 0044F          MOVZWL  #596, -(SP)                            ; 1208
                                                    06  11 00454          BRB     53$
                                      00729010      8F  DD 00456  52$:     PUSHL   #7507984                               ; 1210
                00000000G  00                       01  FB 0045C  53$:     CALLS   #1, LIB$STOP
                                      00729010      8F  DD 00463  54$:     PUSHL   #7507984                               ; 1212
                00000000G  00                       01  FB 00469          CALLS   #1, LIB$SIGNAL
                    020B   CA                       10  88 00470          BISB2   #16, PROTO_VCB+11                      ; 1213
                                              0395  31  00475          BRW     104$                                   ; 1190
                               50         01     AA  9A 00478  55$:     MOVZBL  BUFFER+1, R0                           ; 1218
```

```
                                58          6A40  3E 0047C          MOVAW   BUFFER[R0], MAP_POINTER                  ;
                                            0000G 30 00480          BSBW    GET_MAP_POINTER                         ; 1219
                                50      023C  CA 3C 00483          MOVZWL  PROTO_VCB+60, R0                        ; 1220
                                50      0000G CF C0 00488          ADDL2   DEVICE_CHAR+112, R0                     ;
                                              50 D7 0048D          DECL    R0                                      ;
                                51      023C  CA 3C 0048F          MOVZWL  PROTO_VCB+60, R1                        ; 1221
                                50            51 C6 00494          DIVL2   R1, R0                                  ;
                                50      0FFF  C0 9E 00497          MOVAB   4095(R0), R0                            ; 1220
                            56  50 00001000   8F C7 0049C          DIVL3   #4096, R0, COUNT                        ; 1221
                    000000FF    8F            56 D1 004A4          CMPL    COUNT, #255                             ; 1222
                                              0C 1B 004AB          BLEQU   56$                                     ;
                                7E      08C0  8F 3C 004AD          MOVZWL  #2240, -(SP)                            ; 1223
                    00000000G   00            01 FB 004B2          CALLS   #1, LIB$STOP                            ;
                        0234    CA          01  A7 9E 004B9 56$:    MOVAB   1(R7), PROTO_VCB+52                     ; 1225
                        0239    CA            56 90 004BF          MOVB    COUNT, PROTO_VCB+57                     ; 1226
                                        0480  8F BB 004C4          PUSHR   #^M<R7,R10>                             ; 1234
                        0000G   CF            02 FB 004C8          CALLS   #2, READ_BLOCK                          ;
                                59            50 D0 004CD          MOVL    R0, STATUS                              ;
                                28            59 E8 004D0          BLBS    STATUS, 58$                             ;
                    00000254    8F            59 D1 004D3          CMPL    STATUS, #596                            ; 1240
                                              0E 12 004DA          BNEQ    57$                                     ;
                                7E      0254  8F 3C 004DC          MOVZWL  #596, -(SP)                             ; 1242
                    00000000G   00            01 FB 004E1          CALLS   #1, LIB$STOP                            ;
                                              11 11 004E8          BRB     58$                                     ;
                                              59 DD 004EA 57$:     PUSHL   STATUS                                  ; 1244
                                7E            D4 004EC             CLRL    -(SP)                                   ;
                                    00729020  8F DD 004EE          PUSHL   #7508000                                ;
                    00000000G   00            03 FB 004F4          CALLS   #3, LIB$STOP                            ;
                                12      18    AA E9 004FB 58$:     BLBC    BUFFER+24, 59$                          ; 1246
                                    00729040  8F DD 004FF          PUSHL   #7508032                                ; 1249
                    00000000G   00            01 FB 00505          CALLS   #1, LIB$SIGNAL                          ;
                        020B    CA            10 88 0050C          BISB2   #16, PROTO_VCB+11                       ; 1250
                        0000G   CF            00 FB 00511 59$:     CALLS   #0, GET_VOLUME_LOCK_NAME                ; 1256
                                        0000G CF D4 00516          CLRL    VOLOCK_COUNT                            ; 1258
                            3A  0000G   CF      02 E1 0051A          BBC     #2, STORED_CONTEXT, 61$                 ; 1260
                                              7E D4 00520          CLRL    -(SP)                                   ; 1263
                                              5E DD 00522          PUSHL   SP                                      ;
                                        0000G CF 9F 00524          PUSHAB  GET_VOLUME_LOCK                         ;
                    00000000G   9F            03 FB 00528          CALLS   #3, @#SYS$CMKRNL                        ;
                                59            50 D0 0052F          MOVL    R0, STATUS                              ;
                                09            59 E8 00532          BLBS    STATUS, 60$                             ;
                                              59 DD 00535          PUSHL   STATUS                                  ; 1265
                    00000000G   00            01 FB 00537          CALLS   #1, LIB$STOP                            ;
                                        0000G CF D7 0053E 60$:     DECL    VOLOCK_COUNT                            ; 1267
                            50  0000G   CF  0000G CF 8D 00542      XORB3   DEV_CTX, VOL_CTX, R0                    ; 1269
                                0D            50 E9 0054A          BLBC    R0, 61$                                 ;
                                    007280B4  8F DD 0054D          PUSHL   #7504052                                ; 1271
                    00000000G   00            01 FB 00553          CALLS   #1, LIB$STOP                            ;
                0290    CA      2E          AA 08 28 0055A 61$:    MOVC3   #8, BUFFER+46, PROTO_VCB+144            ; 1275
                        03  020B  CA          04 E1 00561          BBC     #4, PROTO_VCB+11, 63$                   ; 1277
                                        00F3  31 00567 62$:        BRW     78$                                     ;
                            F8      01    AB  01 E1 0056A 63$:     BBC     #1, MOUNT_OPTIONS+1, 62$                ; 1278
                                          11  0000G CF E8 0056F    BLBS    DEV_CTX, 64$                            ; 1282
                        22  AA      0280  CA  0C 28 00574          MOVC3   #12, PROTO_VCB+128, BUFFER+34           ; 1285
                                        2E  AA 9F 0057B            PUSHAB  BUFFER+46                               ; 1286
                    00000000G   00            01 FB 0057E          CALLS   #1, SYS$GETTIM                          ;
0000G   CF          20  AA                  10  00 ED 00585 64$:   CMPZV   #0, #16, BUFFER+32, VOLOCK_COUNT       ; 1289
```

```
                                    0B  13 0058D        BEQL    65$                                   ;
             1C   AA          18    AA  C8 0058F        BISL2   BUFFER+24, BUFFER+28                  ; 1305
             20   AA       0000G    CF  B0 00594        MOVW    VOLOCK_COUNT, BUFFER+32               ; 1306
     05      1C   AA                01  E0 0059A 65$:   BBS     #1, BUFFER+28, 66$                    ; 1309
     05      1C   AA                02  E1 0059F        BBC     #2, BUFFER+28, 67$                    ;
          0000G   CF                02  88 005A4 66$:   BISB2   #2, CLEANUP_FLAGS+1                   ; 1311
     13      1C   AA                03  E1 005A9 67$:   BBC     #3, BUFFER+28, 68$                    ; 1313
                  01       0000G    CF  D1 005AE        CMPL    CURRENT_RVN, #1                       ; 1314
                                    0C  1A 005B3        BGTRU   68$                                   ;
     0B      05   AB                02  E0 005B5        BBS     #2, MOUNT_OPTIONS+5, 69$              ; 1317
          0000G   CF                04  88 005BA        BISB2   #4, CLEANUP_FLAGS+1                   ; 1319
                                    04  11 005BF        BRB     69$                                   ; 1313
             1C   AA                08  8A 005C1 68$:   BICB2   #8, BUFFER+28                         ; 1322
                              20    AA  B6 005C5 69$:   INCW    BUFFER+32                             ; 1324
                           0000G    CF  D5 005C8        TSTL    EXT_CACHE                             ; 1332
                                    04  13 005CC        BEQL    70$                                   ;
             18   AA                02  88 005CE        BISB2   #2, BUFFER+24                         ; 1334
                  01       0000G    CF  D1 005D2 70$:   CMPL    FID_CACHE, #1                         ; 1336
                                    04  13 005D7        BEQL    71$                                   ;
             18   AA                04  88 005D9        BISB2   #4, BUFFER+24                         ; 1338
                           0000G    CF  D5 005DD 71$:   TSTL    QUO_CACHE                             ; 1346
                                    10  13 005E1        BEQL    72$                                   ;
                  01       0000G    CF  D1 005E3        CMPL    CURRENT_RVN, #1                       ;
                                    09  1A 005E8        BGTRU   72$                                   ;
     04      05   AB                02  E0 005EA        BBS     #2, MOUNT_OPTIONS+5, 72$              ; 1347
             18   AA                08  88 005EF        BISB2   #8, BUFFER+24                         ; 1349
                                    5A  DD 005F3 72$:   PUSHL   R10                                   ; 1375
          0000G   CF                01  FB 005F5        CALLS   #1, CHECKSUM                          ;
                            0480    8F  BB 005FA        PUSHR   #^M<R7,R10>                           ; 1376
          0000G   CF                02  FB 005FE        CALLS   #2, WRITE_BLOCK                       ;
                  59                50  D0 00603        MOVL    R0, STATUS                            ;
                           0000G    CF  B6 00606        INCW    VOL_CTX+12                            ; 1382
                  09                59  E9 0060A        BLBC    STATUS, 73$                           ; 1384
0290   CA    2E   AA                08  28 0060D        MOVC3   #8, BUFFER+46, PROTO_VCB+144          ; 1387
                                    42  11 00614        BRB     77$                                   ;
       00000254   8F                59  D1 00616 73$:   CMPL    STATUS, #596                          ; 1390
                                    0C  12 0061D        BNEQ    74$                                   ;
                  7E        0254    8F  3C 0061F        MOVZWL  #596, -(SP)                           ; 1392
      00000000G   00                01  FB 00624        CALLS   #1, LIB$STOP                          ;
       0000025C   8F                59  D1 0062B 74$:   CMPL    STATUS, #604                          ; 1393
                                    0F  12 00632        BNEQ    75$                                   ;
                        0072A013    8F  DD 00634        PUSHL   #7512083                              ; 1394
      00000000G   00                01  FB 0063A        CALLS   #1, LIB$SIGNAL                        ;
                                    11  11 00641        BRB     76$                                   ;
                                    59  DD 00643 75$:   PUSHL   STATUS                                ; 1395
                                    7E  D4 00645        CLRL    -(SP)                                 ;
                        00729048    8F  DD 00647        PUSHL   #7508040                              ;
      00000000G   00                03  FB 0064D        CALLS   #3, LIB$SIGNAL                        ;
             01   AB                02  8A 00654 76$:   BICB2   #2, MOUNT_OPTIONS+1                   ; 1396
          0000G   CF                10  88 00658 77$:   BISB2   #16, CLEANUP_FLAGS+1                  ; 1399
                  05       0000G    CF  E9 0065D 78$:   BLBC    DEV_CTX, 79$                          ; 1411
          0000G   CF                00  FB 00662        CALLS   #0, CHECK_CLUSTER_SANITY              ; 1413
                  09       0000G    CF  E9 00667 79$:   BLBC    VOL_CTX, 80$                          ; 1424
           0328   CA       0000G    CF  D0 0066C        MOVL    VOL_CTX+8, PROTO_FCB+60               ; 1426
                                    7A  11 00673        BRB     86$                                   ;
                  53        0238    CA  9A 00675 80$:   MOVZBL  PROTO_VCB+56, J                       ; 1430
                                    78  11 0067A        BRB     88$                                   ;
```

```
                                5A   DD 0067C 81$:    PUSHL   R10                                     ; 1434
                          0230 DA43  9F 0067E         PUSHAB  @PROTO_VCB+48[J]                        ;
                0000G   CF         02  FB 00683         CALLS   #2, READ_BLOCK                      ;
                        59         50  D0 00688         MOVL    R0, STATUS                          ;
                        31         59  E8 0068B         BLBS    STATUS, 84$                         ; 1435
                00000254   8F      59  D1 0068E         CMPL    STATUS, #596                        ; 1438
                           0E      12  00695             BNEQ    82$                                 ;
                        7E      0254   8F  3C 00697         MOVZWL  #596, -(SP)                     ; 1440
                00000000G   00     01  FB 0069C         CALLS   #1, LIB$STOP                        ;
                           11      11  006A3             BRB     83$                                 ;
                           59      DD  006A5 82$:        PUSHL   STATUS                              ; 1442
                           7E      D4  006A7             CLRL    -(SP)                               ;
                     00729018      8F  DD 006A9         PUSHL   #7507992                            ;
                00000000G   00     03  FB 006AF         CALLS   #3, LIB$SIGNAL                      ;
                    020B   CA      10  88 006B6 83$:     BISB2   #16, PROTO_VCB+11                   ; 1443
                           54      D4  006BB             CLRL    IDX_EOF                             ; 1444
                           38      11  006BD             BRB     89$                                 ; 1445
                        52      7F     8F  9A 006BF 84$:     MOVZBL  #127, I                         ; 1448
                        51          6A42   D0 006C3 85$:     MOVL    BUFFER[I], R1                   ; 1451
                                   28      13 006C7         BEQL    87$                             ;
                50      53         0C      78 006C9         ASHL    #12, J, R0                      ; 1454
                55      52         05      78 006CD         ASHL    #5, I, R5                       ;
                        55         50      C0 006D1         ADDL2   R0, R5                          ;
                                   51      DD 006D4         PUSHL   R1                              ;
                0000G   CF         01      FB 006D6         CALLS   #1, LEFT_ONE                    ;
                        50         55      C0 006DB         ADDL2   R5, R0                          ;
                        51      0238   CA  9A 006DE         MOVZBL  PROTO_VCB+56, R1                ; 1455
                        51         50      C0 006E3         ADDL2   R0, R1                          ;
                        50      023C   CA  3C 006E6         MOVZWL  PROTO_VCB+60, R0                ;
                        54          6140   DE 006EB         MOVAL   (R1)[R0], IDX_EOF               ;
                                   06      11 006EF 86$:     BRB     89$                             ; 1456
                        CF         52      F4 006F1 87$:     SOBGEQ  I, 85$                          ; 1448
                        85         53      F4 006F4 88$:     SOBGEQ  J, 81$                          ; 1430
                        50         54      D0 006F7 89$:     MOVL    IDX_EOF, R0                     ; 1462
                0324    CA         50      D1 006FA         CMPL    R0, PROTO_FCB+56                ;
                                   05      1B 006FF         BLEQU   90$                             ;
                        50      0324   CA  D0 00701         MOVL    PROTO_FCB+56, R0                ;
                        54         50      D0 00706 90$:     MOVL    R0, IDX_EOF                     ;
                0328    CA         54      D1 00709         CMPL    IDX_EOF, PROTO_FCB+60           ; 1463
                                   0A      1B 0070E         BLEQU   91$                             ;
                0328    CA         54      D0 00710         MOVL    IDX_EOF, PROTO_FCB+60           ; 1466
                024E    CA         06      8E 00715         MNEGB   #6, PROTO_VCB+78                ; 1467
                0000G   CF      0328   CA  D0 0071A 91$:     MOVL    PROTO_FCB+60, VOL_CTX+8         ; 1470
                        0A      0000G  CF  E9 00721         BLBC    VOL_CTX, 92$                    ; 1475
                0240    CA      0000G  CF  D0 00726         MOVL    VOL_CTX+4, PROTO_VCB+64         ; 1477
                                   00DD    31 0072D         BRW     104$                            ;
                                   53      D4 00730 92$:     CLRL    FREE                            ; 1481
                                   56      D6 00732         INCL    J                               ; 1482
                                   77      11 00734         BRB     100$                            ;
                                   57      D6 00736 93$:     INCL    LBN                             ; 1486
                                0480   8F  BB 00738         PUSHR   #^M<R7,R10>                     ; 1487
                0000G   CF         02      FB 0073C         CALLS   #2, READ_BLOCK                  ;
                        59         50      D0 00741         MOVL    R0, STATUS                      ;
                        2D         59      E8 00744         BLBS    STATUS, 96$                     ; 1488
                00000254   8F      59      D1 00747         CMPL    STATUS, #596                    ; 1491
                           0E      12  00750             BNEQ    94$                                 ;
                        7E      0254   8F  3C 00750         MOVZWL  #596, -(SP)                     ; 1493
```

```
                                00000000G  00              01  FB 00755          CALLS   #1, LIB$STOP                                    ;
                                                           11  11 0075C          BRB     95$                                             ;
                                                           59  DD 0075E 94$:     PUSHL   STATUS                                          ; 1495
                                                           7E  D4 00760          CLRL    -(SP)                                           ;
                                        00729020           8F  DD 00762          PUSHL   #7508000                                        ;
                                00000000G  00              03  FB 00768          CALLS   #3, LIB$SIGNAL                                  ;
                                    020B   CA              10  88 0076F 95$:     BISB2   #16, PROTO_VCB+11                               ; 1496
                                                           50  D4 00774 96$:     CLRL    I                                               ; 1499
                                           55          6A40  D0 00776 97$:     MOVL    BUFFER[I], X                                    ; 1501
                                                           29  13 0077A          BEQL    99$                                             ; 1502
                                                           52  D4 0077C          CLRL    B2                                              ; 1505
                                           51       E0     A2  9E 0077E 98$:     MOVAB   -32(B2), R1                                     ; 1508
                                           51              51  CE 00782          MNEGL   R1, R1                                          ;
                    54          55         51              52  EA 00785          FFS     B2, R1, X, B1                                   ;
                                                           19  13 0078A          BEQL    99$                                             ;
                                           51       E0     A4  9E 0078C          MOVAB   -32(B1), R1                                     ; 1510
                                           51              51  CE 00790          MNEGL   R1, R1                                          ;
                    52          55         51              54  EB 00793          FFC     B1, R1, X, B2                                   ;
                                51         53              52  C1 00798          ADDL3   B2, FREE, R1                                    ; 1511
                                53         51              54  C3 0079C          SUBL3   B1, R1, FREE                                    ;
                                           20              52  D1 007A0          CMPL    B2, #32                                         ; 1512
                                                           D9  19 007A3          BLSS    98$                                             ;
                                C9         50 0000007F     8F  F3 007A5 99$:     AOBLEQ  #127, I, 97$                                    ; 1499
                                           86              56  F5 007AD 100$:    SOBGTR  J, 93$                                          ; 1482
                                           50       023C   CA  3C 007B0          MOVZWL  PROTO_VCB+60, R0                                ; 1518
                        0240    CA         53              50  C5 007B5          MULL3   R0, FREE, PROTO_VCB+64                          ;
                                     0000G CF       0240   CA  D0 007BB          MOVL    PROTO_VCB+64, VOL_CTX+4                         ; 1519
                                                           49  11 007C2          BRB     104$                                            ; 1062
                                3B         6B              04  E0 007C4 101$:    BBS     #4, MOUNT_OPTIONS, 103$                         ; 1538
                                     0000G CF              00  FB 007C8          CALLS   #0, GET_VOLUME_LOCK_NAME                        ; 1541
                                                           7E  D4 007CD          CLRL    -(SP)                                           ; 1542
                                                           5E  DD 007CF          PUSHL   SP                                              ;
                                                  0000G    CF  9F 007D1          PUSHAB  GET_VOLUME_LOCK                                 ;
                                00000000G  9F              03  FB 007D5          CALLS   #3, @#SYS$CMKRNL                                ;
                                           59              50  D0 007DC          MOVL    R0, STATUS                                      ;
                                           09              59  E8 007DF          BLBS    STATUS, 102$                                    ;
                                                           59  DD 007E2          PUSHL   STATUS                                          ; 1544
                                00000000G  00              01  FB 007E4          CALLS   #1, LIB$STOP                                    ;
                    50          0000G  CF       0000G  CF  8D 007EB 102$:     XORB3   DEV_CTX, VOL_CTX, R0                            ; 1545
                                           0D              50  E9 007F3          BLBC    R0, 103$                                        ;
                                        007280B4           8F  DD 007F6          PUSHL   #7504052                                        ; 1547
                                00000000G  00              01  FB 007FC          CALLS   #1, LIB$STOP                                    ;
                                           05       0000G  CF  E9 00803 103$:     BLBC    DEV_CTX, 104$                                   ; 1550
                                     0000G CF              00  FB 00808          CALLS   #0, CHECK_CLUSTER_SANITY                        ; 1552
                    05                06   AB              05  E1 0080D 104$:     BBC     #5, MOUNT_OPTIONS+6, 105$                       ; 1562
                                    020B   CA              10  8A 00812          BICB2   #16, PROTO_VCB+11                               ; 1563
                                                           7E  D4 00817 105$:     CLRL    -(SP)                                           ; 1565
                                                           5E  DD 00819          PUSHL   SP                                              ;
                                                  0000V    CF  9F 0081B          PUSHAB  MAKE_DISK_MOUNT                                 ;
                                00000000G  9F              03  FB 0081F          CALLS   #3, @#SYS$CMKRNL                                ;
                                           59              50  D0 00826          MOVL    R0, STATUS                                      ;
                                           32              59  E8 00829          BLBS    STATUS, 108$                                    ; 1566
                    04          59         03              00  ED 0082C          CMPZV   #0, #3, STATUS, #4                              ; 1569
                                                           0B  12 00831          BNEQ    106$                                            ;
                                                           59  DD 00833          PUSHL   STATUS                                          ; 1570
                                00000000G  00              01  FB 00835          CALLS   #1, LIB$STOP                                    ;
                                                           20  11 0083C          BRB     108$                                            ;
```

```
                    0B     0000'  CF   E9 0083E 106$:  BLBC    IO_STATUS, 107$                      ; 1573
                                  59   DD 00843        PUSHL   STATUS                               ; 1574
          00000000G 00            01   FB 00845        CALLS   #1, LIB$SIGNAL                       ;
                                  10   11 0084C        BRB     108$                                 ;
                    7E     0000'  CF   3C 0084E 107$:  MOVZWL  IO_STATUS, -(SP)                    ; 1575
                                  7E   D4 00853        CLRL    -(SP)                                ;
                                  59   DD 00855        PUSHL   STATUS                               ;
          00000000G 00            03   FB 00857        CALLS   #3, LIB$SIGNAL                       ;
               0000G CF      40   8F   88 0085E 108$:  BISB2   #64, CLEANUP_FLAGS                  ; 1583
05                  6B            29   E5 00864        BBCC    #41, MOUNT_OPTIONS, 109$             ; 1585
               0000G CF           00   FB 00868        CALLS   #0, BIND_VOLUME                      ; 1586
50             0000G CF           01   78 0086D 109$:  ASHL    #1, DEVICE_INDEX, R0                 ; 1591
                          0000GCF 40   DF 00873        PUSHAL  PHYS_NAME[R0]                        ;
                             0214 CA   9F 00878        PUSHAB  PROTO_VCB+20                         ;
                                  0C   DD 0087C        PUSHL   #12                                  ;
                                  03   DD 0087E        PUSHL   #3                                   ;
                       0072A003   8F   DD 00880        PUSHL   #7512067                             ;
          00000000G 00            05   FB 00886        CALLS   #5, LIB$SIGNAL                       ;
                    12     0488   CA   E8 0088D        BLBS    CACHE_STATUS, 110$                   ; 1598
0D                01    AB      03   E0 00892        BBS     #3, MOUNT_OPTIONS+1, 110$            ; 1599
                       0072A08B   8F   DD 00897        PUSHL   #7512203                             ; 1601
          00000000G 00            01   FB 0089D        CALLS   #1, LIB$SIGNAL                       ;
06             0000G CF           01   E0 008A4 110$:  BBS     #1, CLEANUP_FLAGS+1, 111$            ; 1610
23             0000G CF           02   E1 008AA        BBC     #2, CLEANUP_FLAGS+1, 113$            ;
                          07    AB   95   08B0 111$:  TSTB    MOUNT_OPTIONS+7                      ; 1612
                                  13   18 008B3        BGEQ    112$                                 ;
                       0072A093   8F   DD 008B5        PUSHL   #7512211                             ; 1615
          00000000G 00            01   FB 008BB        CALLS   #1, LIB$SIGNAL                       ;
               0000G CF           06   8A 008C2        BICB2   #6, CLEANUP_FLAGS+1                  ; 1617
                                  04   008C7          RET                                          ; 1612
05             0000G CF           02   E1 008C8 112$:  BBC     #2, CLEANUP_FLAGS+1, 113$            ; 1620
               0000G CF           02   88 008CE        BISB2   #2, CLEANUP_FLAGS+1                  ; 1622
                                  04   008D3 113$:     RET                                          ; 1624
                                0000   008D4 114$:     .WORD   Save nothing                         ; 0826
                                  7E   D4 008D6        CLRL    -(SP)                                ;
                                  5E   DD 008D8        PUSHL   SP                                   ;
                    7E     04     AC   7D 008DA        MOVQ    4(AP), -(SP)                         ;
          0000V     CF           03   FB 008DE        CALLS   #3, MOUNT_HANDLER                    ;
                                  04   008E3          RET                                          ;

; Routine Size:  2276 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  1097     1625  1 ROUTINE MOUNT_HANDLER (SIGNAL, MECHANISM) =
;  1098     1626  1
;  1099     1627  1 !++
;  1100     1628  1 !
;  1101     1629  1 ! FUNCTIONAL DESCRIPTION:
;  1102     1630  1 !
;  1103     1631  1 !       This routine is the condition handler for the main disk mount
;  1104     1632  1 !       code. It undoes any damage done so far and returns the error
;  1105     1633  1 !       status to the user mode caller.
;  1106     1634  1 !
;  1107     1635  1 !
;  1108     1636  1 ! CALLING SEQUENCE:
;  1109     1637  1 !       MOUNT_HANDLER (ARG1, ARG2)
;  1110     1638  1 !
;  1111     1639  1 ! INPUT PARAMETERS:
;  1112     1640  1 !       ARG1: address of signal vector
;  1113     1641  1 !       ARG2: address of mechanism vector
;  1114     1642  1 !
;  1115     1643  1 ! IMPLICIT INPUTS:
;  1116     1644  1 !       global pointers to blocks allocated
;  1117     1645  1 !
;  1118     1646  1 ! OUTPUT PARAMETERS:
;  1119     1647  1 !       NONE
;  1120     1648  1 !
;  1121     1649  1 ! IMPLICIT OUTPUTS:
;  1122     1650  1 !       NONE
;  1123     1651  1 !
;  1124     1652  1 ! ROUTINE VALUE:
;  1125     1653  1 !       SS$_RESIGNAL
;  1126     1654  1 !
;  1127     1655  1 ! SIDE EFFECTS:
;  1128     1656  1 !       necessary cleanups done
;  1129     1657  1 !
;  1130     1658  1 !--
;  1131     1659  1
;  1132     1660  2 BEGIN
;  1133     1661  2
;  1134     1662  2 MAP
;  1135     1663  2         SIGNAL          : REF BBLOCK,   ! signal vector
;  1136     1664  2         MECHANISM       : REF BBLOCK;   ! mechanism vector
;  1137     1665  2
;  1138     1666  2 EXTERNAL
;  1139     1667  2         MOUNT_OPTIONS   : BITVECTOR,    ! command parser options
;  1140     1668  2         CLEANUP_FLAGS   : BITVECTOR;    ! cleanup action flags
;  1141     1669  2
;  1142     1670  2 EXTERNAL ROUTINE
;  1143     1671  2         CHECKSUM,                       ! compute block checksum
;  1144     1672  2         LOCK_CLEANUP : NOVALUE,         ! cleanup dev and vol locks.
;  1145     1673  2         READ_BLOCK,                     ! read a disk block
;  1146     1674  2         WRITE_BLOCK;                    ! write a disk block
;  1147     1675  2
;  1148     1676  2
;  1149     1677  2 ! Note that cleanup is done if we are unwinding, which occurrs when
;  1150     1678  2 ! we take an error exit.
;  1151     1679  2 !
;  1152     1680  2
;  1153     1681  3 IF  (.SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND)
```

```
;  1154          1682  3 AND ((.BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_SEVERE) OR
;  1155          1683  3       (.BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_ERROR))
;  1156          1684  2 THEN
;  1157          1685  2     BEGIN
;  1158          1686  2
;  1159          1687  2 ! Clear the dirty bits in the storage control block if they were set.
;  1160          1688  2 !
;  1161          1689  2
;  1162          1690  3     IF .CLEANUP_FLAGS[CLF_CLEANSCB] AND .MOUNT_OPTIONS[OPT_WRITE]
;  1163          1691  3     THEN
;  1164          1692  4         BEGIN
;  1165          1693  4         IF READ_BLOCK (.PROTO_VCB[VCB$L_SBMAPLBN]-1, BUFFER)
;  1166          1694  4         THEN
;  1167          1695  4             IF BUFFER [SCB$W_WRITECNT] NEQ 0
;  1168          1696  4             THEN
;  1169          1697  5                 BEGIN
;  1170          1698  6                 IF (BUFFER [SCB$W_WRITECNT] = .BUFFER [SCB$W_WRITECNT] - 1)
;  1171          1699  5                     EQL 0
;  1172          1700  5                 THEN
;  1173          1701  5                     BUFFER [SCB$L_STATUS] = 0;
;  1174          1702  5
;  1175          1703  5                 CHECKSUM (BUFFER);
;  1176          1704  5                 WRITE_BLOCK (.PROTO_VCB[VCB$L_SBMAPLBN]-1, BUFFER);
;  1177          1705  4                 END;
;  1178          1706  3         END;
;  1179          1707  3
;  1180          1708  3     LOCK_CLEANUP ();
;  1181          1709  3
;  1182          1710  2     END;
;  1183          1711  2
;  1184          1712  2 SS$_RESIGNAL
;  1185          1713  1 END;                                    ! end of routine MOUNT_HANDLER


                                                        .EXTRN  LOCK_CLEANUP

                                        0004 00000 MOUNT_HANDLER:
                                                        .WORD   Save R2                                 ;  1625
                                    52  0000'  CF  9E 00002       MOVAB   BUFFER, R2                  ;
                                    50    04  AC  D0 00007       MOVL    SIGNAL, R0                  ;  1681
                          00000920  8F    04  A0  D1 0000B       CMPL    4(R0), #2336                ;
                                                  5C  13 00013   BEQL    4$                          ;
        04      04  A0            03            00  ED 00015   CMPZV   #0, #3, 4(R0), #4            ;  1682
                                                  08  13 0001B   BEQL    1$                          ;
        02      04  A0            03            00  ED 0001D   CMPZV   #0, #3, 4(R0), #2            ;  1683
                                                  4C  12 00023   BNEQ    4$                          ;
                41      0000G     CF            04  E1 00025 1$:BBC     #4, CLEANUP_FLAGS+1, 3$     ;  1690
                3B      0000G     CF            01  E1 0002B   BBC     #1, MOUNT_OPTIONS+1, 3$     ;
                                                  52  DD 00031   PUSHL   R2                          ;  1693
                7E      0234      C2            01  C3 00033   SUBL3   #1, PROTO_VCB+52, -(SP)      ;
                        0000G     CF            02  FB 00039   CALLS   #2, READ_BLOCK               ;
                                  2B            50  E9 0003E   BLBC    R0, 3$                      ;
                                  50      20    A2  9E 00041   MOVAB   BUFFER+32, R0                ;  1695
                                                  25  13 00045   BEQL    3$                          ;
                                  50      20    A2  3C 00047   MOVZWL  BUFFER+32, R0                ;  1698
                                                  50  D7 0004B   DECL    R0                          ;
```

```
                      20   A2              50  B0 0004D          MOVW    R0, BUFFER+32                          ;
                                           50  D5 00051          TSTL    R0                                     ; 1699
                                           03  12 00053          BNEQ    2$                                     ;
                                     18    A2  D4 00055          CLRL    BUFFER+24                              ; 1701
                                           52  DD 00058 2$:      PUSHL   R2                                     ; 1703
                    0000G   CF             01  FB 0005A          CALLS   #1, CHECKSUM                           ;
                                           52  DD 0005F          PUSHL   R2                                     ; 1704
                 7E 0234    C2             01  C3 00061          SUBL3   #1, PROTO_VCB+52, -(SP)                ;
                    0000G   CF             02  FB 00067          CALLS   #2, WRITE_BLOCK                        ;
                    0000G   CF             00  FB 0006C 3$:      CALLS   #0, LOCK_CLEANUP                       ; 1708
                            50     0918    8F  3C 00071 4$:      MOVZWL  #2328, R0                              ; 1713
                                           04  00076              RET                                            ;

; Routine Size:  119 bytes,    Routine Base:  $CODE$ + 08E4
```

```
;  1187     1714  1 ROUTINE MAKE_DISK_MOUNT =
;  1188     1715  1
;  1189     1716  1 !++
;  1190     1717  1 !
;  1191     1718  1 ! FUNCTIONAL DESCRIPTION:
;  1192     1719  1 !
;  1193     1720  1 !       This routine does all of the data base manipulation needed to get
;  1194     1721  1 !       a volume actually mounted. It allocates the real VCB, FCB, and
;  1195     1722  1 !       window, and hooks then all together. It also starts up the ACP
;  1196     1723  1 !       gets the mounted volume list entry made.
;  1197     1724  1 !
;  1198     1725  1 !
;  1199     1726  1 ! CALLING SEQUENCE:
;  1200     1727  1 !       MAKE_DISK_MOUNT ()
;  1201     1728  1 !
;  1202     1729  1 ! INPUT PARAMETERS:
;  1203     1730  1 !       NONE
;  1204     1731  1 !
;  1205     1732  1 ! IMPLICIT INPUTS:
;  1206     1733  1 !       MOUNT parser data base
;  1207     1734  1 !       own storage of this module
;  1208     1735  1 !
;  1209     1736  1 ! OUTPUT PARAMETERS:
;  1210     1737  1 !       NONE
;  1211     1738  1 !
;  1212     1739  1 ! IMPLICIT OUTPUTS:
;  1213     1740  1 !       NONE
;  1214     1741  1 !
;  1215     1742  1 ! ROUTINE VALUE:
;  1216     1743  1 !       1 if successful
;  1217     1744  1 !       status values if not
;  1218     1745  1 !
;  1219     1746  1 ! SIDE EFFECTS:
;  1220     1747  1 !       volume mounted
;  1221     1748  1 !
;  1222     1749  1 !--
;  1223     1750  1
;  1224     1751  2 BEGIN
;  1225     1752  2
;  1226     1753  2 BUILTIN
;  1227     1754  2         INSQUE;
;  1228     1755  2
;  1229     1756  2 LOCAL
;  1230     1757  2         WINDOW_SIZE,                    ! size in bytes needed for window
;  1231     1758  2         UCB             : REF BBLOCK,   ! pointer to volume UCB
;  1232     1759  2         ORB             : REF BBLOCK,   ! Pointer to device ORB
;  1233     1760  2         VCB             : REF BBLOCK,   ! pointer to volume VCB
;  1234     1761  2         RVT             : REF BBLOCK,   ! pointer to volume set RVT
;  1235     1762  2         SYS_STATUS,                     ! system service status
;  1236     1763  2         STATUS,                         ! general status value
;  1237     1764  2         NOWRITE,                        ! state of volume set write lock
;  1238     1765  2         ERASE,                          ! state of volume erase-on-delete
;  1239     1766  2         NOHIGHWATER,                    ! state of volume file-highwater-marking
;  1240     1767  2         MOUNTVER,                       ! state of volume set mount verification
;  1241     1768  2         LOCKED;                         ! state of volume set allocation lock
;  1242     1769  2
;  1243     1770  2 EXTERNAL
```

```
;  1244     1771  2          DEV_CTX          : BBLOCK FIELD (DC), ! device context
;  1245     1772  2          VLSETLCK_CTX     : BBLOCK FIELD (VC), ! volume set lock context
;  1246     1773  2          VOL_CTX          : BBLOCK FIELD (VC), ! volume lock context
;  1247     1774  2          MOUNT_OPTIONS    : BITVECTOR,         ! command parser options
;  1248     1775  2          STORED_CONTEXT   : BITVECTOR,         ! looks at xqp flag
;  1249     1776  2          CLEANUP_FLAGS    : BITVECTOR,         ! cleanup action flags
;  1250     1777  2          DEVICE_COUNT,                         ! number of devices specified
;  1251     1778  2          CHANNEL,                              ! channel assigned to device
;  1252     1779  2          STRUCT_NAME      : VECTOR,            ! descriptor of volume set name
;  1253     1780  2          HOME_BLOCK       : BBLOCK,            ! buffer containing home block
;  1254     1781  2          OWNER_UIC,                            ! owner UIC from command
;  1255     1782  2          PROTECTION,                           ! volume protection from command
;  1256     1783  2          EXT_CACHE,                            ! size of extent cache to allocate
;  1257     1784  2          FID_CACHE,                            ! size of file ID cache to allocate
;  1258     1785  2          QUO_CACHE,                            ! size of quota file cache to allocate
;  1259     1786  2          EXT_LIMIT,                            ! limit of volume space to cache
;  1260     1787  2          CURRENT_RVN,                          ! RVN of disk being mounted
;  1261     1788  2          REAL_VCB         : REF BBLOCK,        ! address of VCB allocated
;  1262     1789  2          REAL_VCA         : REF BBLOCK,        ! address of volume cache allocated
;  1263     1790  2          REAL_FCB         : REF BBLOCK,        ! address of FCB allocated
;  1264     1791  2          REAL_WCB         : REF BBLOCK,        ! address of window allocated
;  1265     1792  2          MTL_ENTRY        : REF BBLOCK,        ! address of mount list entry
;  1266     1793  2          SMTL_ENTRY       : REF BBLOCK,        ! address of mount list entry for volume set
;  1267     1794  2          CTL$GL_VOLUMES   : ADDRESSING_MODE (ABSOLUTE);
;  1268     1795  2                                               ! count of volumes mounted by process
;  1269     1796  2
;  1270     1797  2      EXTERNAL ROUTINE
;  1271     1798  2              GET_VOLSET_LOCK,                 ! get cluster lock for volume set.
;  1272     1799  2              STORE_CONTEXT,                   ! write appropriate value blocks.
;  1273     1800  2              GET_CHANNELUCB,                  ! get UCB assigned to channel
;  1274     1801  2              ALLOCATE_MEM,                    ! allocate system dynamic memory
;  1275     1802  2              START_ACP,                       ! start and connect ACP to device
;  1276     1803  2              LOCK_IODB        : ADDRESSING_MODE (GENERAL), ! lock I/O database mutex
;  1277     1804  2              UNLOCK_IODB      : ADDRESSING_MODE (GENERAL), ! unlock I/O database mutex
;  1278     1805  2              ENTER_RVT,                       ! attach to relative volume table
;  1279     1806  2              ALLOC_LOGNAME,                   ! create logical name and MTL blocks
;  1280     1807  2              ENTER_LOGNAME,                   ! enter logical name and MTL in lists
;  1281     1808  2              SEND_ERRLOG;                     ! send message to error logger
;  1282     1809  2
;  1283     1810  2
;  1284     1811  2      ! Allocate all of the required control blocks. We allocate them in
;  1285     1812  2      ! advance to avoid having to back out of some awkward situations later on.
;  1286     1813  2      ! The one exception is the AQB, which is either found or allocated by
;  1287     1814  2      ! START_ACP.
;  1288     1815  2      !
;  1289     1816  2
;  1290     1817  2      ENABLE KERNEL_HANDLER;
;  1291     1818  2
;  1292     1819  2      REAL_VCB = ALLOCATE_MEM (VCB$C_LENGTH, 0);
;  1293     1820  2      REAL_VCB[VCB$B_TYPE] = DYN$C_VCB;
;  1294     1821  2      CH$MOVE (VCB$C_LENGTH-11, PROTO_VCB+11, .REAL_VCB+11);
;  1295     1822  2      UCB = GET_CHANNELUCB (.CHANNEL);
;  1296     1823  2      ORB = .UCB[UCB$L_ORB];
;  1297     1824  2      RVT = 0;
;  1298     1825  2
;  1299     1826  2      IF NOT .MOUNT_OPTIONS[OPT_FOREIGN]
;  1300     1827  2      THEN
```

```
;  1301     1828  4      BEGIN
;  1302     1829  4      LOCAL       FCB_ORB : REF BBLOCK;
;  1303     1830  4
;  1304     1831  4      REAL_VCB[VCB$L_FCBFL] = REAL_VCB[VCB$L_FCBFL];
;  1305     1832  4      REAL_VCB[VCB$L_FCBBL] = REAL_VCB[VCB$L_FCBFL];
;  1306     1833  4
;  1307     1834  4      REAL_FCB = ALLOCATE_MEM (FCB$C_LENGTH, 0);
;  1308     1835  4      REAL_FCB[FCB$B_TYPE] = DYN$C_FCB;
;  1309     1836  4      CH$MOVE (FCB$C_LENGTH-11, PROTO_FCB+11, .REAL_FCB+11);
;  1310     1837  4      REAL_FCB[FCB$L_WLFL] = REAL_FCB[FCB$L_WLFL];
;  1311     1838  4      REAL_FCB[FCB$L_WLBL] = REAL_FCB[FCB$L_WLFL];
;  1312     1839  4
;  1313     1840  4      FCB_ORB = REAL_FCB[FCB$R_ORB];
;  1314     1841  4      FCB_ORB[ORB$V_ACL_QUEUE] = 0;
;  1315     1842  4      FCB_ORB[ORB$L_ACL_COUNT] = 0;
;  1316     1843  4      FCB_ORB[ORB$L_ACL_DESC] = 0;
;  1317     1844  4      INSQUE (.REAL_FCB, REAL_VCB[VCB$L_FCBFL]);
;  1318     1845  4
;  1319     1846  4      WINDOW_SIZE = WCB$C_LENGTH + MAXU (.PROTO_WCB[WCB$W_NMAP] + 2, 6) * 6;
;  1320     1847  4      REAL_WCB = ALLOCATE_MEM (.WINDOW_SIZE, 0);
;  1321     1848  4      REAL_WCB[WCB$B_TYPE] = DYN$C_WCB;
;  1322     1849  4      CH$MOVE (.WINDOW_SIZE-11, PROTO_WCB+11, .REAL_WCB+11);
;  1323     1850  4      REAL_WCB[WCB$L_FCB] = .REAL_FCB;
;  1324     1851  4      INSQUE (.REAL_WCB, REAL_FCB[FCB$L_WLFL]);
;  1325     1852  4
;  1326     1853  4  ! Allocate the cache block for the volume, computing the size from the cache
;  1327     1854  4  ! parameters.
;  1328     1855  4  !
;  1329     1856  4
;  1330     1857  4      REAL_VCA = ALLOCATE_MEM (VCA$C_LENGTH
;  1331     1858  4                               + $BYTEOFFSET (VCA$L_FIDLIST) + .FID_CACHE * 4
;  1332     1859  4                               + $BYTEOFFSET (VCA$Q_EXTLIST) + .EXT_CACHE * 8,
;  1333     1860  4                               0);
;  1334     1861  4      REAL_VCB[VCB$L_CACHE] = .REAL_VCA;
;  1335     1862  4      REAL_VCA[VCA$B_TYPE] = DYN$C_VCA;
;  1336     1863  4      REAL_VCA[VCA$L_FIDCACHE] = .REAL_VCA + VCA$C_LENGTH;
;  1337     1864  4      REAL_VCA[VCA$L_EXTCACHE] = .REAL_VCA + VCA$C_LENGTH
;  1338     1865  4                               + $BYTEOFFSET (VCA$L_FIDLIST) + .FID_CACHE * 4;
;  1339     1866  4      BBLOCK [.REAL_VCA[VCA$L_FIDCACHE], VCA$W_FIDSIZE] = .FID_CACHE;
;  1340     1867  4      BBLOCK [.REAL_VCA[VCA$L_EXTCACHE], VCA$W_EXTSIZE] = .EXT_CACHE;
;  1341     1868  4      BBLOCK [.REAL_VCA[VCA$L_EXTCACHE], VCA$W_EXTLIMIT] = .EXT_LIMIT;
;  1342     1869  4      BBLOCK [BBLOCK [.REAL_VCA[VCA$L_FIDCACHE], VCA$B_FIDCACB], ACB$B_RMOD] =
;  1343     1870  4          PSL$C_KERNEL + ACB$M_NODELETE;
;  1344     1871  4      BBLOCK [BBLOCK [.REAL_VCA[VCA$L_EXTCACHE], VCA$B_EXTCACB], ACB$B_RMOD] =
;  1345     1872  4          PSL$C_KERNEL + ACB$M_NODELETE;
;  1346     1873  4      REAL_VCB[VCB$W_QUOSIZE] = .QUO_CACHE;
;  1347     1874  4
;  1348     1875  4  ! If this volume is part of a volume set, attach it to the RVT for the set,
;  1349     1876  4  ! creating one if it doesn't exist.
;  1350     1877  4  !
;  1351     1878  4
;  1352     1879  4      REAL_VCB[VCB$L_RVT] = .UCB;
;  1353     1880  4
;  1354     1881  4      IF .HOME_BLOCK[HM2$W_RVN] NEQ 0 OR .MOUNT_OPTIONS[OPT_BIND]
;  1355     1882  4      THEN
;  1356     1883  4          BEGIN
;  1357     1884  4          RVT = ENTER_RVT (STRUCT_NAME[0], .UCB);
```

```
;  1358          1885  4          REAL_VCB[VCB$L_RVT] = .RVT;
;  1359          1886  4          REAL_WCB[WCB$L_RVT] = .RVT;
;  1360          1887  4          CURRENT_RVN = .HOME_BLOCK[HM2$W_RVN];
;  1361          1888  4          REAL_FCB[FCB$W_FID_RVN] = .HOME_BLOCK[HM2$W_RVN];
;  1362          1889  4          (REAL_FCB[FCB$L_LOCKBASIS])<24,8> = .REAL_FCB[FCB$B_FID_RVN];
;  1363          1890  4          REAL_VCB[VCB$W_RVN] = .HOME_BLOCK[HM2$W_RVN];
;  1364          1891  4
;  1365          1892  4      ! Take out the volume set lock.  Also check for cluster uniqueness
;  1366          1893  4      ! of the volume set structure name.  Note that this
;  1367          1894  4      ! test is based on whether or not this is (or is not) the first instance
;  1368          1895  4      ! of this device being mounted and the lock for the volume set being created.
;  1369          1896  4      ! A given volume set must always be mounted in the same order on
;  1370          1897  4      ! different nodes in the cluster.  If, for example, RVN 2 was mounted
;  1371          1898  4      ! first on node A, then if node B mounts RVN 1 next, it will fail because
;  1372          1899  4      ! the volume set lock already exists, even though it is the first mount
;  1373          1900  4      ! on the RVN 1 device.
;  1374          1901  4      !
;  1375          1902  4
;  1376          1903  4          IF .RVT [RVT$L_STRUCLKID] EQL 0
;  1377          1904  4          THEN
;  1378          1905  5              BEGIN
;  1379          1906  5              GET_VOLSET_LOCK();
;  1380          1907  5
;  1381          1908  5              IF .STORED_CONTEXT [XQP]
;  1382          1909  5              THEN
;  1383          1910  5                  IF .DEV_CTX [DC_NOTFIRST_MNT] NEQ .VLSETLCK_CTX [VC_NOTFIRST_MNT]
;  1384          1911  5                  THEN
;  1385          1912  5                      ERR_EXIT (MOUN$_VOLINSET);
;  1386          1913  4              END;
;  1387          1914  4
;  1388          1915  3          END;
;  1389          1916  2      END;
;  1390          1917  2
;  1391          1918  2      ! Now allocate space for logical name and mounted volume list entries.
;  1392          1919  2      ! If this is volume 1 of a set, we allocate 2 - one for the volume as usual
;  1393          1920  2      ! and one for the set. If a logical name is given in the command, it is assigned
;  1394          1921  2      ! to volume 1 of the set, or if only one volume is being mounted, to it.
;  1395          1922  2      ! Otherwise, the logical name is constructed from the volume label.
;  1396          1923  2      !
;  1397          1924  2
;  1398          1925  2      IF NOT .MOUNT_OPTIONS[OPT_FOREIGN] AND .HOME_BLOCK[HM2$W_RVN] EQL 1
;  1399          1926  2      THEN
;  1400          1927  2          BEGIN
;  1401          1928  2          ALLOC_LOGNAME (0);
;  1402          1929  2          SMTL_ENTRY = .MTL_ENTRY;            ! copy reserved entry to entry for set
;  1403          1930  2          MTL_ENTRY = 0;
;  1404          1931  2          ALLOC_LOGNAME (1);
;  1405          1932  2          END
;  1406          1933  2
;  1407          1934  2      ELSE
;  1408          1935  2          BEGIN
;  1409          1936  2          IF .DEVICE_COUNT EQL 1
;  1410          1937  2          THEN ALLOC_LOGNAME (0)
;  1411          1938  2          ELSE ALLOC_LOGNAME (1);
;  1412          1939  2          END;
;  1413          1940  2
;  1414          1941  2      ! All data blocks except the AQB are now allocated. First set up the
```

```
;  1415          1942  2 ! volume ownership and protection in the VCB. Now hook up the blocks
;  1416          1943  2 ! to the device data base and start the ACP.
;  1417          1944  2 !
;  1418          1945  2
;  1419          1946  2 UCB[UCB$V_UNLOAD] = NOT .MOUNT_OPTIONS [OPT_NOUNLOAD];
;  1420          1947  2 ORB[ORB$L_OWNER] = .VOLUME_UIC;
;  1421          1948  2 IF .MOUNT_OPTIONS[OPT_OWNER_UIC]
;  1422          1949  2 THEN ORB[ORB$L_OWNER] = .OWNER_UIC;
;  1423          1950  2
;  1424          1951  2 ORB[ORB$V_PROT_16] = 1;                          ! SOGW protection word
;  1425          1952  2 IF .MOUNT_OPTIONS[OPT_FOREIGN]
;  1426          1953  2 THEN ORB[ORB$W_PROT] = %X'FF00'
;  1427          1954  2 ELSE ORB[ORB$W_PROT] = .HOME_BLOCK[HM2$W_PROTECT];
;  1428          1955  2 IF .MOUNT_OPTIONS[OPT_PROTECTION]
;  1429          1956  2 THEN ORB[ORB$W_PROT] = .PROTECTION;
;  1430          1957  2
;  1431          1958  2 STATUS = 1;
;  1432          1959  2 IF NOT .MOUNT_OPTIONS[OPT_FOREIGN]
;  1433          1960  2 THEN
;  1434          1961  2     BEGIN
;  1435          1962  2     REAL_VCB [VCB$V_MOUNTVER] = .MOUNT_OPTIONS [OPT_MOUNTVER];
;  1436          1963  2     REAL_VCB[VCB$L_ORGUCB] = .UCB;
;  1437          1964  2     START_ACP (.UCB, .REAL_VCB, AQB$K_F11V2);
;  1438          1965  2
;  1439          1966  2 ! Store value blocks of device and volume locks, as appropriate.
;  1440          1967  2 !
;  1441          1968  2
;  1442          1969  2     STORE_CONTEXT ();
;  1443          1970  2
;  1444          1971  2 ! Unless the disk is being mounted /NOQUOTA or is write locked, attempt
;  1445          1972  2 ! to connect the quota file if the RVN is 0 or 1. If it fails with no such
;  1446          1973  2 ! file, then proceed; else lock the volume.
;  1447          1974  2 !
;  1448          1975  2
;  1449          1976  3     IF NOT .MOUNT_OPTIONS[OPT_NOQUOTA]
;  1450          1977  3     AND .REAL_VCB[VCB$W_RVN] [EQU 1
;  1451          1978  3     AND NOT .REAL_VCB[VCB$V_NOALLOC]
;  1452          1979  3     AND .MOUNT_OPTIONS[OPT_WRITE]
;  1453          1980  3     THEN
;  1454          1981  4         BEGIN
;  1455          1982  4         PSECT   PLIT = $OWN$;              ! ACP argument blocks must be writable
;  1456          1983  4
;  1457        P 1984  4         SYS_STATUS = DO_IO (
;  1458        P 1985  4                 EFN  = MOUNT_EFN,
;  1459        P 1986  4                 CHAN = .CHANNEL,
;  1460        P 1987  4                 FUNC = IO$_ACPCONTROL,
;  1461        P 1988  4                 IOSB = IO_STATUS[0],
;  1462        P 1989  4                 P1   = UPLIT (FIB$C_MTALEN,
;  1463        P 1990  4                               UPLIT (0, WORD (0, 0, 0), WORD (4, 4, 0), 0,
;  1464        P 1991  4                                      WORD (0, FIB$C_ENA_QUOTA), 0)),
;  1465        P 1992  4                 P2   = DESCRIPTOR ('QUOTA.SYS;1')
;  1466          1993  4                 );
;  1467          1994  4         IF NOT .SYS_STATUS THEN IO_STATUS = .SYS_STATUS;
;  1468          1995  4
;  1469          1996  4         IF NOT .IO_STATUS[0]
;  1470          1997  4         THEN
;  1471          1998  5             BEGIN
```

```
;  1472     1999  5
;  1473     2000  5 ! CLF_REBUILDQUO was set in MOUNT_DISK2 if the QUODIRTY2 flag was
;  1474     2001  5 ! set in the SCB, indicating that the disk was improperly dismounted
;  1475     2002  5 ! sometime in the past when quota caching was enabled.  However, if
;  1476     2003  5 ! we failed to enable quotas here for whatever reason (normally just
;  1477     2004  5 ! failure to find a quota file), clear the flag now so that quota
;  1478     2005  5 ! rebuild is not attempted.
;  1479     2006  5 !
;  1480     2007  5
;  1481     2008  5             CLEANUP_FLAGS [CLF_REBUILDQUO] = 0;
;  1482     2009  5
;  1483     2010  5             IF .IO_STATUS[0] NEQ SS$_NOSUCHFILE
;  1484     2011  5             THEN
;  1485     2012  6                 BEGIN
;  1486     2013  6                 REAL_VCB[VCB$V_NOALLOC] = 1;
;  1487     2014  6                 STATUS = MOUN$_QUOTAFAIL;
;  1488     2015  5                 END;
;  1489     2016  4             END;
;  1490     2017  3         END;
;  1491     2018  2     END
;  1492     2019  1
;  1493     2020  1 ! Do /FOREIGN processing if requested.
;  1494     2021  1 !
;  1495     2022  1
;  1496     2023  1 ELSE
;  1497     2024  1     BEGIN
;  1498     2025  1
;  1499     2026  1 ! Store value blocks of device and volume locks, as appropriate.
;  1500     2027  1 !
;  1501     2028  1
;  1502     2029  1     STORE_CONTEXT ();
;  1503     2030  1
;  1504     2031  1     LOCK_IODB ();
;  1505     2032  1     UCB[UCB$L_VCB] = .REAL_VCB;
;  1506     2033  1     UCB[UCB$L_DEVCHAR] = .UCB[UCB$L_DEVCHAR]
;  1507     2034  1                          OR (DEV$M_MNT OR DEV$M_DIR OR DEV$M_FOR);
;  1508     2035  1     SET_DATACHECK (.UCB, 0);
;  1509     2036  1     UNLOCK_IODB ();
;  1510     2037  1     END;
;  1511     2038  1
;  1512     2039  1 IF .MOUNT_OPTIONS[OPT_NOSHARE] AND .CLEANUP_FLAGS[CLF_DEALLOCATE]
;  1513     2040  1 THEN UCB[UCB$V_DEADMO] = 1;
;  1514     2041  1
;  1515     2042  1 IF NOT .MOUNT_OPTIONS[OPT_WRITE]
;  1516     2043  1 THEN BBLOCK [UCB[UCB$L_DEVCHAR], DEV$V_SWL] = 1;
;  1517     2044  1
;  1518     2045  1 ! Enter the logical name for the volume; bump the user's volume mount count,
;  1519     2046  1 ! and make the error log entry for the mount.
;  1520     2047  1 !
;  1521     2048  1
;  1522     2049  1 ENTER_LOGNAME (.UCB, .REAL_VCB);
;  1523     2050  1 CTL$GL_VOLUMES = .CTL$GL_VOLUMES + 1;
;  1524     2051  1 SEND_ERRLOG (1, .UCB);
;  1525     2052  1
;  1526     2053  1 ! If any volume in the set is mounted /NOWRITE or is locked due to an error,
;  1527     2054  1 ! the entire volume set must be similarly locked to prevent random behavior.
;  1528     2055  1 ! Scan the RVT and process all volumes in the set. Also inhibit disk rebuild
```

```
;  1529          2056  2 ! if the volumes are locked.
;  1530          2057  2 !
;  1531          2058  2
;  1532          2059  2 NOWRITE = .BBLOCK [UCB[UCB$L_DEVCHAR], DEV$V_SWL];
;  1533          2060  2 LOCKED = .REAL_VCB[VCB$V_NOA[LOC];
;  1534          2061  2 MOUNTVER = .REAL_VCB[VCB$V_MOUNTVER];
;  1535          2062  2 ERASE = .REAL_VCB[VCB$V_ERASE];
;  1536          2063  2 NOHIGHWATER = .REAL_VCB[VCB$V_NOHIGHWATER];
;  1537          2064  2
;  1538          2065  2 IF .RVT NEQ 0
;  1539          2066  2 THEN
;  1540          2067  3     BEGIN
;  1541          2068  3     LOCK_IODB ();
;  1542          2069  3
;  1543          2070  3     INCR J FROM 1 TO .RVT[RVT$B_NVOLS]
;  1544          2071  3     DO
;  1545          2072  4         BEGIN
;  1546          2073  4
;  1547          2074  4         LOCAL
;  1548          2075  4             RVUCB       : REF BBLOCK;
;  1549          2076  4
;  1550          2077  4         RVUCB = .VECTOR [RVT[RVT$L_UCBLST], .J-1];
;  1551          2078  4         IF .RVUCB NEQ 0
;  1552          2079  4         THEN
;  1553          2080  5             BEGIN
;  1554          2081  5             IF .NOWRITE
;  1555          2082  5             THEN BBLOCK [RVUCB[UCB$L_DEVCHAR], DEV$V_SWL] = 1;
;  1556          2083  5             NOWRITE = .BBLOCK [RVUCB[UCB$L_DEVCHAR], DEV$V_SWL];
;  1557          2084  5
;  1558          2085  5             VCB = .RVUCB[UCB$L_VCB];
;  1559          2086  5             IF .LOCKED
;  1560          2087  5             THEN VCB[VCB$V_NOALLOC] = 1;
;  1561          2088  5             LOCKED = .VCB[VCB$V_NOALLOC];
;  1562          2089  5
;  1563          2090  5             IF .MOUNTVER
;  1564          2091  5             THEN VCB[VCB$V_MOUNTVER] = 1;
;  1565          2092  5             MOUNTVER = .VCB[VCB$V_MOUNTVER];
;  1566          2093  5
;  1567          2094  5             IF .ERASE
;  1568          2095  5             THEN VCB[VCB$V_ERASE] = 1;
;  1569          2096  5             ERASE = .VCB[VCB$V_ERASE];
;  1570          2097  5
;  1571          2098  5             IF .NOHIGHWATER
;  1572          2099  5             THEN VCB[VCB$V_NOHIGHWATER] = 1;
;  1573          2100  5             NOHIGHWATER = .VCB[VCB$V_NOHIGHWATER];
;  1574          2101  4             END;
;  1575          2102  3         END;
;  1576          2103  3     UNLOCK_IODB ();
;  1577          2104  2     END;
;  1578          2105  2
;  1579          2106  2 IF .LOCKED OR .NOWRITE
;  1580          2107  2 THEN CLEANUP_FLAGS[CLF_REBUILD] = 0;
;  1581          2108  2
;  1582          2109  2 ! Increment the refcount, so that it never goes to zero while the device
;  1583          2110  2 ! is mounted.
;  1584          2111  2 ! All subsequent error paths from this point must do a full dismount to
;  1585          2112  2 ! correctly remove the refcount bias.
```

```
;  1586          2113  2 !
;  1587          2114  2
;  1588          2115  2 UCB[UCB$W_REFC] = .UCB[UCB$W_REFC] + 1;
;  1589          2116  2
;  1590          2117  2 RETURN .STATUS;
;  1591          2118  2
;  1592          2119  1 END;                                    ! end of routine MAKE_DISK_MOUNT


                                                                .PSECT  $OWN$,NOEXE,2

                                        00000000  00008 P.AAE:  .LONG   0                                                ;
                                  0000  0000  0000  0000C       .WORD   0, 0, 0                                          ;
                                  0000  0004  0004  00012       .WORD   4, 4, 0                                          ;
                                        00000000  00018       .LONG   0                                                ;
                                        0009  0000  0001C       .WORD   0, 9                                             ;
                                        00000000  00020       .LONG   0                                                ;
                                        0000001C  00024 P.AAD:  .LONG   28                                               ;
                                        00000000' 00028       .ADDRESS P.AAE                                           ;
31  3B  53  59  53  2E  41  54  4F  55  51  0002C P.AAG:  .ASCII  \QUOTA.SYS;1\                                           ;
                                            00037       .BLKB   1                                                        ;
                                        0000000B  00038 P.AAF:  .LONG   11                                               ;
                                        00000000' 0003C       .ADDRESS P.AAG                                           ;

                                                                .EXTRN  VLSETLCK_CTX, DEVICE_COUNT
                                                                .EXTRN  CHANNEL, OWNER_UIC
                                                                .EXTRN  PROTECTION, REAL_VCB
                                                                .EXTRN  REAL_VCA, REAL_FCB
                                                                .EXTRN  REAL_WCB, MTL_ENTRY
                                                                .EXTRN  SMTL_ENTRY, CTL$GL_VOLUMES
                                                                .EXTRN  GET_VOLSET_LOCK
                                                                .EXTRN  STORE_CONTEXT, GET_CHANNELUCB
                                                                .EXTRN  ALLOCATE_MEM, START_ACP
                                                                .EXTRN  LOCK_IODB, UNLOCK_IODB
                                                                .EXTRN  ENTER_RVT, ALLOC_LOGNAME
                                                                .EXTRN  ENTER_LOGNAME, SEND_ERRLOG
                                                                .EXTRN  COMMON_IO

                                                                .PSECT  $CODE$,NOWRT,2

                                            OFFC 00000 MAKE_DISK_MOUNT:
                                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11             ; 1714
                                    6D      03B2    CF  DE 00002        MOVAL   29$, (FP)                               ; 1751
                                            7E      D4 00007            CLRL    -(SP)                                   ; 1819
                                    7E        EC    8F  9A 00009        MOVZBL  #236, -(SP)                             ;
                          0000G     CF              02  FB 0000D        CALLS   #2, ALLOCATE_MEM                        ;
                          0000G     CF              50  D0 00012        MOVL    R0, REAL_VCB                            ;
                            0A      A0              11  90 00017        MOVB    #17, 10(R0)                             ; 1820
0B  A0                    0000'     CF      00E1    8F  28 0001B        MOVC3   #225, PROTO_VCB+11, 11(R0)              ; 1821
                          0000G     CF      0000G   CF  DD 00024        PUSHL   CHANNEL                                 ; 1822
                          0000G     CF              01  FB 00028        CALLS   #1, GET_CHANNELUCB                      ;
                                    57              50  D0 0002D        MOVL    R0, UCB                                 ;
                                    58        1C    A7  D0 00030        MOVL    28(UCB), ORB                            ; 1823
                                              59    D4  00034           CLRL    RVT                                     ; 1824
                              03    0000G     CF              03  E1 00036        BBC     #3, MOUNT_OPTIONS+1, 1$         ; 1826
                                                      017D  31  0003C        BRW     5$                                      ;
```

```
                        50     0000G   CF   D0 0003F 1$:     MOVL    REAL_VCB, R0                                    ; 1831
                        60             50   D0 00044         MOVL    R0, (R0)
                     04 A0             50   D0 00047         MOVL    R0, 4(R0)                                       ; 1832
                        7E             D4   D4 0004B         CLRL    -(SP)                                           ; 1834
                        7E     B4      8F   9A 0004D         MOVZBL  #180, -(SP)
                 0000G  CF             02   FB 00051         CALLS   #2, ALLOCATE_MEM
                 0000G  CF             50   D0 00056         MOVL    R0, REAL_FCB
                        56     0000G   CF   D0 0005B         MOVL    REAL_FCB, R6                                    ; 1835
                     0A A6             07   90 00060         MOVB    #7, 10(R6)
                0B   A6  0000'  CF     00A9 8F 28 00064      MOVC3   #169, PROTO_FCB+11, 11(R6)                      ; 1836
                     10 A6             10 A6 9E 0006D        MOVAB   16(R6), 16(R6)                                  ; 1837
                     14 A6             10 A6 9E 00072        MOVAB   16(R6), 20(R6)                                  ; 1838
                        50             58 A6 9E 00077        MOVAB   88(R6), FCB_ORB                                 ; 1840
                     0B A0             02   8A 0007B         BICB2   #2, 11(FCB_ORB)                                 ; 1841
                     28 A0             7C   0007F            CLRQ    40(FCB_ORB)                                     ; 1842
                 0000G  DF             66   0E 00082         INSQUE  (R6), @REAL_VCB                                 ; 1844
                        52     0000'   CF   3C 00087         MOVZWL  PROTO_WCB+22, R2                                ; 1846
                        52             02   C0 0008C         ADDL2   #2, R2
                        06             52   D1 0008F         CMPL    R2, #6
                                       03   1E 00092         BGEQU   2$
                        52             06   D0 00094         MOVL    #6, R2
                        52             06   C4 00097 2$:     MULL2   #6, R2
                        52             30   C0 0009A         ADDL2   #48, WINDOW_SIZE
                                       7E   D4 0009D         CLRL    -(SP)                                           ; 1847
                                       52   DD 0009F         PUSHL   WINDOW_SIZE
                 0000G  CF             02   FB 000A1         CALLS   #2, ALLOCATE_MEM
                 0000G  CF             50   D0 000A6         MOVL    R0, REAL_WCB
                        56     0000G   CF   D0 000AB         MOVL    REAL_WCB, R6                                    ; 1848
                     0A A6             12   90 000B0         MOVB    #18, 10(R6)
                        52             0B   C2 000B4         SUBL2   #11, R2                                         ; 1849
                0B   A6  0000'  CF           52 28 000B7     MOVC3   R2, PROTO_WCB+11, 11(R6)
                     18 A6     0000G   CF   D0 000BE         MOVL    REAL_FCB, 24(R6)                                ; 1850
                  50    0000G   CF         10 C1 000C4      ADDL3   #16, REAL_FCB, R0                               ; 1851
                        60             66   0E 000CA         INSQUE  (R6), (R0)
                                       7E   D4 000CD         CLRL    -(SP)                                           ; 1857
                        50     0000G   CF   D0 000CF         MOVL    FID_CACHE, R0                                   ; 1858
                  51    0000G   CF         03 78 000D4      ASHL    #3, EXT_CACHE, R1                               ; 1859
                                  5C A140   DF   000DA         PUSHAL  92(R1)[R0]
                 0000G  CF             02   FB 000DE         CALLS   #2, ALLOCATE_MEM
                 0000G  CF             50   D0 000E3         MOVL    R0, REAL_VCA
                        52     0000G   CF   D0 000E8         MOVL    REAL_VCB, R2                                    ; 1861
                        50     0000G   CF   D0 000ED         MOVL    REAL_VCA, R0
                     58 A2             50   D0 000F2         MOVL    R0, 88(R2)
                     0A A0             32   90 000F6         MOVB    #50, 10(R0)                                     ; 1862
                        60         0C  A0   9E 000FA         MOVAB   12(R0), (R0)                                    ; 1863
                        51     0000G   CF   D0 000FE         MOVL    FID_CACHE, R1                                   ; 1865
                     04 A0          30 A041  DE 00103        MOVAL   48(R0)[R1], 4(R0)
                     00 B0             51   B0 00109         MOVW    R1, @0(R0)                                      ; 1866
                        50             60   7D 0010D         MOVQ    (R0), R0                                        ; 1869
                        61     0000G   CF   B0 00110         MOVW    EXT_CACHE, (R1)                                 ; 1867
                     08 A1     0000G   CF   B0 00115         MOVW    EXT_LIMIT, 8(R1)                                ; 1868
                     13 A0             20   90 0011B         MOVB    #32, 19(R0)                                     ; 1870
                     1B A1             20   90 0011F         MOVB    #32, 27(R1)                                     ; 1872
                     60 A2     0000G   CF   B0 00123         MOVW    QUO_CACHE, 96(R2)                               ; 1873
                     20 A2             57   D0 00129         MOVL    UCB, 32(R2)                                     ; 1879
                                0000G  CF   B5 0012D         TSTW    HOME_BLOCK+38                                   ; 1881
                                       05   12 00131         BNEQ    3$                                              ;
```

```
                63     0000G   CF   E9 00133          BLBC    MOUNT_OPTIONS+5, 4$                                    ;
                                CF   DD 00138 3$:      PUSHL   UCB                                                    ; 1884
                       0000G    CF   9F 0013A          PUSHAB  STRUCT_NAME
             0000G     CF            02   FB 0013E    CALLS   #2, ENTER_RVT
                59                  50   D0 00143    MOVL    R0, RVT                                                ;
                51     0000G    CF   D0 00146          MOVL    REAL_VCB, R1                                           ; 1885
              20       A1           59   D0 0014B    MOVL    RVT, 32(R1)                                            ;
                50     0000G    CF   D0 0014F          MOVL    REAL_WCB, R0                                           ; 1886
              1C       A0           59   D0 00154    MOVL    RVT, 28(R0)                                            ;
                52     0000G    CF   3C 00158          MOVZWL  HOME_BLOCK+38, R2                                      ; 1887
             0000G     CF           52   D0 0015D    MOVL    R2, CURRENT_RVN                                        ;
                50     0000G    CF   D0 00162          MOVL    REAL_FCB, R0                                           ; 1888
              28       A0           52   B0 00167    MOVW    R2, 40(R0)                                             ;
              4F       A0     28    A0   90 0016B    MOVB    40(R0), 79(R0)                                         ; 1889
              0E       A1           52   B0 00170    MOVW    R2, 14(R1)                                             ; 1890
                                    69   D5 00174    TSTL    (RVT)                                                  ; 1903
                                    23   12 00176    BNEQ    4$                                                     ;
             0000G     CF           00   FB 00178    CALLS   #0, GET_VOLSET_LOCK                                    ; 1906
          18 0000G     CF           02   E1 0017D    BBC     #2, STORED_CONTEXT, 4$                                 ; 1908
          50 0000G     CF    0000G  CF   8D 00183    XORB3   DEV_CTX, VCSETLCK_CTX, R0                              ; 1910
                       0D           50   E9 0018B    BLBC    R0, 4$                                                 ;
                          00728194  8F   DD 0018E    PUSHL   #7504276                                               ; 1912
                00000000G  00            01   FB 00194    CALLS   #1, LIB$STOP
          1B    0000G     CF           03   E0 0019B 4$:    BBS     #3, MOUNT_OPTIONS+1, 5$                        ; 1925
                       01    0000G    CF   B1 001A1          CMPW    HOME_BLOCK+38, #1
                                           14   12 001A6    BNEQ    5$                                             ;
                                           7E   D4 001A8    CLRL    -(SP)                                          ; 1928
                             0000G    CF   01   FB 001AA    CALLS   #1, ALLOC_LOGNAME
                             0000G    CF   0000G CF   D0 001AF    MOVL    MTL_ENTRY, SMTL_ENTRY                      ; 1929
                                           0000G CF   D4 001B6    CLRL    MTL_ENTRY                                  ; 1930
                                           0B   11   001BA    BRB     6$                                             ; 1931
                                    01     0000G    CF   D1 001BC 5$:    CMPL    DEVICE_COUNT, #1                   ; 1936
                                           04   12   001C1    BNEQ    6$                                             ;
                                           7E   D4   001C3    CLRL    -(SP)                                          ; 1937
                                           02   11   001C5    BRB     7$                                             ;
                                           01   DD   001C7 6$:    PUSHL   #1                                         ; 1938
                             0000G    CF   01   FB 001C9 7$:    CALLS   #1, ALLOC_LOGNAME
             50              0000G    CF   01   EF 001CE    EXTZV   #2, #1, MOUNT_OPTIONS+1, R0                    ; 1946
                                           50   D2 001D5    MCOML   R0, R0                                         ;
          65 A7                    01     04   F0 001D8    INSV    R0, #4, #1, 101(UCB)                           ;
                                    68     0000'    CF   D0 001DE    MOVL    VOLUME_UIC, (ORB)                     ; 1947
                                    05     0000G    CF   E1 001E3    BBC     #2, MOUNT_OPTIONS+2, 8$               ; 1948
                                    68     0000G    CF   D0 001E9    MOVL    OWNER_UIC, (ORB)                      ; 1949
                                      0B   A8   01   88 001EE 8$:    BISB2   #1, 1T(ORB)                          ; 1951
                                    08     0000G    CF   E1 001F2    BBC     #3, MOUNT_OPTIONS+1, 9$               ; 1952
                                      18   A8   FF00   8F   B0 001F8    MOVW    #-256, 24(ORB)                        ; 1953
                                           06   11   001FE    BRB     10$                                            ;
                                      18   A8   0000G    CF   B0 00200 9$:    MOVW    HOME_BLOCK+52, 24(ORB)        ; 1954
                                    06     0000G    CF   E1 00206 10$:   BBC     #1, MOUNT_OPTIONS+2, 11$              ; 1955
                                      18   A8   0000G    CF   B0 0020C    MOVW    PROTECTION, 24(ORB)                   ; 1956
                                           5B   01   D0 00212 11$:   MOVL    #1, STATUS                            ; 1958
                                    03     0000G    CF   E1 00215    BBC     #3, MOUNT_OPTIONS+1, 12$              ; 1959
                                           009A 31   0021B    BRW     15$                                            ;
                                    51     0000G    CF   D0 0021E 12$:   MOVL    REAL_VCB, R1                          ; 1962
             50              0000G    CF   01   EF 00223    EXTZV   #6, #1, MOUNT_OPTIONS+6, R0                    ;
          53 A1                    01     02   F0 0022A    INSV    R0, #2, #1, 83(R1)                             ;
                                    50     0000G    CF   D0 00230    MOVL    REAL_WCB, R0                          ; 1963
```

```
                    10   A0              57  DO 00235          MOVL    UCB, 16(R0)                                    ;
                                         02  DD 00239          PUSHL   #2                                             ; 1964
                                         51  DD 0023B          PUSHL   R1                                             ;
                                         57  DD 0023D          PUSHL   UCB                                            ;
                 0000G  CF               03  FB 0023F          CALLS   #3, START_ACP                                  ;
                 0000G  CF               00  FB 00244          CALLS   #0, STORE_CONTEXT                              ; 1969
             67  0000G  CF               02  E0 00249          BBS     #2, MOUNT_OPTIONS+5, 14$                       ; 1976
                       50       0000G    CF  D0 0024F          MOVL    REAL_VCB, R0                                   ; 1977
                       01         0E     A0  B1 00254          CMPW    14(R0), #1                                     ;
                                         5C  1A 00258          BGTRU   14$                                            ;
                       50       0000G    CF  D0 0025A          MOVL    REAL_VCB, R0                                   ; 1978
             7E          0B     A0          04  E0 0025F          BBS     #4, T1(R0), 16$                                ;
             78  0000G  CF                   01  E1 00264          BBC     #1, MOUNT_OPTIONS+1, 16$                       ; 1979
                                         7E  7C 0026A          CLRQ    -(SP)                                          ; 1993
                                         7E  7C 0026C          CLRQ    -(SP)                                          ;
                                 0000'   CF  9F 0026E          PUSHAB  P.AAF                                          ;
                                 0000'   CF  9F 00272          PUSHAB  P.AAD                                          ;
                                         7E  7C 00276          CLRQ    -(SP)                                          ;
                                 0000'   CF  9F 00278          PUSHAB  IO_STATUS                                      ;
                                         38  DD 0027C          PUSHL   #56                                            ;
                                 0000G   CF  DD 0027E          PUSHL   CHANNEL                                        ;
                                         1A  DD 00282          PUSHL   #26                                            ;
                00000000G        00         0C  FB 00284          CALLS   #12, COMMON_IO                                 ;
                                            05  50 E8 0028B      BLBS    SYS_STATUS, 13$                                ; 1994
                          0000'  CF          50  D0 0028E      MOVL    SYS_STATUS, IO_STATUS                          ;
                                 4A   0000'  CF  E8 00293 13$: BLBS    IO_STATUS, 16$                                 ; 1996
                          0000G  CF          04  8A 00298      BICB2   #4, CLEANUP_FLAGS+1                            ; 2008
                          0910   8F   0000'  CF  B1 0029D      CMPW    IO_STATUS, #2320                               ; 2010
                                            3C  13 002A4      BEQL    16$                                            ;
                                 50       0000G  CF  D0 002A6      MOVL    REAL_VCB, R0                               ; 2013
                                    0B    A0     10  88 002AB      BISB2   #16, 11(R0)                                ;
                                 5B 00729038      8F  D0 002AF      MOVL    #7508024, STATUS                           ; 2014
                                                  2A  11 002B6 14$: BRB     16$                                        ; 1959
                          0000G  CF               00  FB 002B8 15$: CALLS   #0, STORE_CONTEXT                          ; 2029
                00000000G        00               00  FB 002BD      CALLS   #0, LOCK_IODB                              ; 2031
                             34  A7       0000G  CF  D0 002C4      MOVL    REAL_VCB, 52(UCB)                          ; 2032
                             38  A7 01080008      8F  C8 002CA      BISL2   #17301512, 56(UCB)                         ; 2034
                                                  7E  D4 002D2      CLRL    -(SP)                                      ; 2035
                                                  57  DD 002D4      PUSHL   UCB                                        ;
                          0000V  CF               02  FB 002D6      CALLS   #2, SET_DATACHECK                          ;
                00000000G        00               00  FB 002DB      CALLS   #0, UNLOCK_IODB                            ; 2036
                       0A 0000G  CF               04  E1 002E2 16$: BBC     #4, MOUNT_OPTIONS, 17$                     ; 2039
                       04 0000G  CF               01  E1 002E8      BBC     #1, CLEANUP_FLAGS, 17$                     ;
                             65  A7               04  88 002EE      BISB2   #4, 101(UCB)                               ; 2040
                       04 0000G  CF               01  E0 002F2 17$: BBS     #1, MOUNT_OPTIONS+1, 18$                   ; 2042
                             3B  A7               02  88 002F8      BISB2   #2, 59(UCB)                                ; 2043
                                       0000G  CF  DD 002FC 18$: PUSHL   REAL_VCB                                       ; 2049
                                              57  DD 00300      PUSHL   UCB                                            ;
                          0000G  CF           02  FB 00302      CALLS   #2, ENTER_LOGNAME                              ;
                                    00000000G 9F  D6 00307      INCL    a#CTL$GL_VOLUMES                               ; 2050
                                              57  DD 0030D      PUSHL   UCB                                            ; 2051
                                              01  DD 0030F      PUSHL   #1                                             ;
                          0000G  CF           02  FB 00311      CALLS   #2, SEND_ERRLOG                                ;
             55       3B  A7                01  EF 00316      EXTZV   #1, #1, 59(UCB), NOWRITE                       ; 2059
                                 50       0000G  CF  D0 0031C      MOVL    REAL_VCB, R0                               ; 2060
             54       0B  A0                01  EF 00321      EXTZV   #4, #1, 11(R0), LOCKED                         ;
             56       53  A0                01  EF 00327      EXTZV   #2, #1, 83(R0), MOUNTVER                       ; 2061
```

```
                52      53   A0          01          03  EF 0032D          EXTZV   #3, #1, 83(R0), ERASE                       ; 2062
                58      53   A0          01          04  EF 00333          EXTZV   #4, #1, 83(R0), NOHIGHWATER                 ; 2063
                                         59  D5 00339          TSTL    RVT                                                 ; 2065
                                         69  13 0033B          BEQL    26$                                                 ;
                                 00000000G   00  FB 0033D          CALLS   #0, LOCK_IODB                                       ; 2068
                                             5A      0B   A9  9A 00344          MOVZBL  11(RVT), R10                        ; 2070
                                             53      44   A9  9E 00348          MOVAB   68(RVT), R3                         ; 2077
                                                 59  D4 0034C          CLRL    J                                           ;
                                                 4B  11 0034E          BRB     25$                                         ;
                                             50      FC A349  D0 00350  19$:    MOVL    -4(R3)[J], RVUCB                    ;
                                                 44  13 00355          BEQL    25$                                         ; 2078
                                             04      55  E9 00357          BLBC    NOWRITE, 20$                            ; 2081
                                                 3B  A0  02  88 0035A          BISB2   #2, 59(RVUCB)                       ; 2082
                55      3B   A0                 01  01  EF 0035E  20$:    EXTZV   #1, #1, 59(RVUCB), NOWRITE                  ; 2083
                                             51      34   A0  D0 00364          MOVL    52(RVUCB), VCB                      ; 2085
                                             04      54  E9 00368          BLBC    LOCKED, 21$                             ; 2086
                                                 0B  A1  10  88 0036B          BISB2   #16, 11(VCB)                        ; 2087
                54      0B   A1                 01  04  EF 0036F  21$:    EXTZV   #4, #1, 11(VCB), LOCKED                     ; 2088
                                             04      56  E9 00375          BLBC    MOUNTVER, 22$                           ; 2090
                                                 53  A1  04  88 00378          BISB2   #4, 83(VCB)                         ; 2091
                                             50      53   A1  9E 0037C  22$:    MOVAB   83(VCB), R0                         ; 2092
                56              60               01  02  EF 00380          EXTZV   #2, #1, (R0), MOUNTVER                      ;
                                             03      52  E9 00385          BLBC    ERASE, 23$                              ; 2094
                                             60      08  88 00388          BISB2   #8, (R0)                                ; 2095
                52              60               01  03  EF 0038B  23$:    EXTZV   #3, #1, (R0), ERASE                         ; 2096
                                             03      58  E9 00390          BLBC    NOHIGHWATER, 24$                        ; 2098
                                             60      10  88 00393          BISB2   #16, (R0)                               ; 2099
                58              60               01  04  EF 00396  24$:    EXTZV   #4, #1, (R0), NOHIGHWATER                   ; 2100
                                 B1          59      5A  F3 0039B  25$:    AOBLEQ  R10, J, 19$                             ; 2070
                                     00000000G   00  00  FB 0039F          CALLS   #0, UNLOCK_IODB                             ; 2103
                                             03      54  E8 003A6  26$:    BLBS    LOCKED, 27$                             ; 2106
                                             05      55  E9 003A9          BLBC    NOWRITE, 28$                            ;
                                         0000G   CF      02  8A 003AC  27$:    BICB2   #2, CLEANUP_FLAGS+1                 ; 2107
                                                 5C  A7  B6 003B1  28$:    INCW    92(UCB)                                 ; 2115
                                             50      5B  D0 003B4          MOVL    STATUS, R0                              ; 2117
                                                     04 003B7          RET                                                 ; 2119
                                                 0000 003B8  29$:    .WORD   Save nothing                                ; 1751
                                                 7E  D4 003BA          CLRL    -(SP)                                       ;
                                                 5E  DD 003BC          PUSHL   SP                                          ;
                                             7E      04   AC  7D 003BE          MOVQ    4(AP), -(SP)                        ;
                                         0000V   CF      03  FB 003C2          CALLS   #3, KERNEL_HANDLER                  ;
                                                     04 003C7          RET                                                 ;

; Routine Size:  968 bytes,    Routine Base:  $CODE$ + 095B
```

```
;  1594          2120  1 GLOBAL ROUTINE SET_DATACHECK (UCB, HOME_BLOCK) : NOVALUE =
;  1595          2121  1
;  1596          2122  1 !++
;  1597          2123  1 !
;  1598          2124  1 ! FUNCTIONAL DESCRIPTION:
;  1599          2125  1 !
;  1600          2126  1 !       This routine sets the read and write check bits in the indicated UCB
;  1601          2127  1 !       according to the command switches and the volume characteristics.
;  1602          2128  1 !
;  1603          2129  1 !
;  1604          2130  1 ! CALLING SEQUENCE:
;  1605          2131  1 !       SET_DATACHECK (ARG1, ARG2)
;  1606          2132  1 !
;  1607          2133  1 ! INPUT PARAMETERS:
;  1608          2134  1 !       ARG2: address of home block or 0
;  1609          2135  1 !
;  1610          2136  1 ! IMPLICIT INPUTS:
;  1611          2137  1 !       MOUNT_OPTIONS: datacheck qualifier bits
;  1612          2138  1 !
;  1613          2139  1 ! OUTPUT PARAMETERS:
;  1614          2140  1 !       ARG1: address of UCB
;  1615          2141  1 !
;  1616          2142  1 ! IMPLICIT OUTPUTS:
;  1617          2143  1 !       NONE
;  1618          2144  1 !
;  1619          2145  1 ! ROUTINE VALUE:
;  1620          2146  1 !       NONE
;  1621          2147  1 !
;  1622          2148  1 ! SIDE EFFECTS:
;  1623          2149  1 !       NONE
;  1624          2150  1 !
;  1625          2151  1 !--
;  1626          2152  1
;  1627          2153  2 BEGIN
;  1628          2154  2
;  1629          2155  2 MAP
;  1630          2156  2         UCB             : REF BBLOCK,   ! UCB arg
;  1631          2157  2         HOME_BLOCK      : REF BBLOCK;   ! home block arg
;  1632          2158  2
;  1633          2159  2 EXTERNAL
;  1634          2160  2         MOUNT_OPTIONS   : BITVECTOR;    ! parser option flags
;  1635          2161  2
;  1636          2162  2 ! The read and write check attributes to be set are simply the inclusive
;  1637          2163  2 ! OR of the read and write check volume attributes and the command options.
;  1638          2164  2 !
;  1639          2165  2
;  1640          2166  2 BBLOCK [UCB[UCB$L_DEVCHAR], DEV$V_RCK] = .MOUNT_OPTIONS[OPT_READCHECK]
;  1641          2167  3     OR (IF .HOME_BLOCK NEQ 0
;  1642          2168  3         THEN .HOME_BLOCK[HM2$V_READCHECK]
;  1643          2169  3         ELSE 0
;  1644          2170  2         );
;  1645          2171  2
;  1646          2172  2 BBLOCK [UCB[UCB$L_DEVCHAR], DEV$V_WCK] = .MOUNT_OPTIONS[OPT_WRITECHECK]
;  1647          2173  3     OR (IF .HOME_BLOCK NEQ 0
;  1648          2174  3         THEN .HOME_BLOCK[HM2$V_WRITCHECK]
;  1649          2175  3         ELSE 0
;  1650          2176  2         );
```

```
; 1651          2177  2
; 1652          2178  1 END;                                    ! end of routine SET_DATACHECK



                                                    0004 00000          .ENTRY  SET_DATACHECK, Save R2                  ; 2120
                                        50      04  AC   7D 00002       MOVQ    UCB, R0                                 ; 2166
                                                    51   D5 00006       TSTL    R1                                      ; 2167
                                                    08   13 00008       BEQL    1$                                      ;
            51      2A      A1          01          00   EF 0000A       EXTZV   #0, #1, 42(R1), R1                      ; 2168
                                                    02   11 00010       BRB     2$                                      ;
                                                    51   D4 00012 1$:    CLRL    R1                                      ; 2167
            52      0000G   CF          01          03   EF 00014 2$:    EXTZV   #3, #1, MOUNT_OPTIONS+4, R2             ;
                                        52          51   88 0001B       BISB2   R1, R2                                  ;
      3B    A0              01          06          52   F0 0001E       INSV    R2, #6, #1, 59(R0)                      ;
                                        51      08  AC   D0 00024       MOVL    HOME_BLOCK, R1                          ; 2173
                                                    08   13 00028       BEQL    3$                                      ;
            51      2A      A1          01          01   EF 0002A       EXTZV   #1, #1, 42(R1), R1                      ; 2174
                                                    02   11 00030       BRB     4$                                      ;
                                                    51   D4 00032 3$:    CLRL    R1                                      ; 2173
            52      0000G   CF          01          04   EF 00034 4$:    EXTZV   #4, #1, MOUNT_OPTIONS+4, R2             ;
                                        52          51   88 0003B       BISB2   R1, R2                                  ;
      3B    A0              01          07          52   F0 0003E       INSV    R2, #7, #1, 59(R0)                      ;
                                                    04 00044            RET                                             ; 2178

; Routine Size:  69 bytes,    Routine Base:  $CODE$ + 0D23
```

```
;  1654     2179  1 ROUTINE KERNEL_HANDLER (SIGNAL, MECHANISM) : NOVALUE =
;  1655     2180  1
;  1656     2181  1 !++
;  1657     2182  1 !
;  1658     2183  1 ! FUNCTIONAL DESCRIPTION:
;  1659     2184  1 !
;  1660     2185  1 !       This routine is the condition handler for all of the kernel mode
;  1661     2186  1 !       code. It undoes any damage done so far and returns the error
;  1662     2187  1 !       status to the user mode caller.
;  1663     2188  1 !
;  1664     2189  1 !
;  1665     2190  1 ! CALLING SEQUENCE:
;  1666     2191  1 !       KERNEL_HANDLER (ARG1, ARG2)
;  1667     2192  1 !
;  1668     2193  1 ! INPUT PARAMETERS:
;  1669     2194  1 !       ARG1: address of signal vector
;  1670     2195  1 !       ARG2: address of mechanism vector
;  1671     2196  1 !
;  1672     2197  1 ! IMPLICIT INPUTS:
;  1673     2198  1 !       global pointers to blocks allocated
;  1674     2199  1 !
;  1675     2200  1 ! OUTPUT PARAMETERS:
;  1676     2201  1 !       NONE
;  1677     2202  1 !
;  1678     2203  1 ! IMPLICIT OUTPUTS:
;  1679     2204  1 !       NONE
;  1680     2205  1 !
;  1681     2206  1 ! ROUTINE VALUE:
;  1682     2207  1 !       NONE
;  1683     2208  1 !
;  1684     2209  1 ! SIDE EFFECTS:
;  1685     2210  1 !       stack unwound, allocations undone
;  1686     2211  1 !
;  1687     2212  1 !--
;  1688     2213  1
;  1689     2214  2 BEGIN
;  1690     2215  2
;  1691     2216  2 MAP
;  1692     2217  2         SIGNAL          : REF BBLOCK,   ! signal vector
;  1693     2218  2         MECHANISM       : REF BBLOCK;   ! mechanism vector
;  1694     2219  2
;  1695     2220  2 LOCAL
;  1696     2221  2         P               : REF BBLOCK,   ! pointer to scan system lists
;  1697     2222  2         UCB             : REF BBLOCK;   ! UCB being mounted
;  1698     2223  2
;  1699     2224  2 EXTERNAL
;  1700     2225  2         MOUNT_OPTIONS   : BITVECTOR,    ! command parser options
;  1701     2226  2         CLEANUP_FLAGS   : BITVECTOR,    ! cleanup action flags
;  1702     2227  2         CHANNEL,                        ! channel assigned to device
;  1703     2228  2         MAILBOX_CHANNEL,                ! channel number of ACP mailbox
;  1704     2229  2         REAL_VCB        : REF BBLOCK,   ! address of VCB allocated
;  1705     2230  2         REAL_VCA        : REF BBLOCK,   ! address of volume cache allocated
;  1706     2231  2         REAL_FCB        : REF BBLOCK,   ! address of FCB allocated
;  1707     2232  2         REAL_WCB        : REF BBLOCK,   ! address of window allocated
;  1708     2233  2         REAL_RVT        : REF BBLOCK,   ! address of disk RVT
;  1709     2234  2         REAL_AQB        : REF BBLOCK,   ! address of AQB allocated
;  1710     2235  2         MTL_ENTRY       : REF BBLOCK,   ! address of mounted volume list entry
```

```
;  1711          2236  2          SMTL_ENTRY      : REF BBLOCK,    ! address of volume set MTL
;  1712          2237  2          IOC$GL_AQBLIST  : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
;  1713          2238  2                                           ! system AQB list
;  1714          2239  2
;  1715          2240  2      EXTERNAL ROUTINE
;  1716          2241  2              GET_CHANNELUCB,              ! get UCB address of channel
;  1717          2242  2              LOCK_IODB       : ADDRESSING_MODE (GENERAL), ! interlock system I/O database
;  1718          2243  2              UNLOCK_IODB     : ADDRESSING_MODE (GENERAL), ! unlock system I/O database
;  1719          2244  2              DEALLOCATE_MEM;              ! deallocate system dynamic memory
;  1720          2245  2
;  1721          2246  2
;  1722          2247  2      ! Deallocate whatever control blocks exist to wherever they came from.
;  1723          2248  2      !
;  1724          2249  2
;  1725          2250  2      IF .SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND
;  1726          2251  2      THEN
;  1727          2252  2          BEGIN
;  1728          2253  2
;  1729          2254  2          IF .SIGNAL[CHF$L_SIG_ARGS] NEQ 3
;  1730          2255  2          THEN BUG_CHECK (UNXSIGNAL, FATAL, 'Unexpected signal in MOUNT');
;  1731          2256  2
;  1732          2257  2      ! If there is a mailbox in existence, deassign its channel, thereby
;  1733          2258  2      ! deleting the mailbox.
;  1734          2259  2      !
;  1735          2260  2
;  1736          2261  2          IF .CLEANUP_FLAGS[CLF_DEASSMBX]
;  1737          2262  2          THEN
;  1738          2263  2              $DASSGN (CHAN = .MAILBOX_CHANNEL);
;  1739          2264  2
;  1740          2265  2      ! Clean up the UCB.
;  1741          2266  2      !
;  1742          2267  2
;  1743          2268  2          UCB = GET_CHANNELUCB (.CHANNEL);
;  1744          2269  2          LOCK_IODB ();
;  1745          2270  2          BBLOCK [UCB [UCB$L_DEVCHAR], DEV$V_MNT] = 0;
;  1746          2271  2          UCB[UCB$L_VCB] = 0;
;  1747          2272  2          UNLOCK_IODB ();
;  1748          2273  2
;  1749          2274  2      ! If we have created an AQB but no ACP, we must remove the AQB from the
;  1750          2275  2      ! system list.
;  1751          2276  2      !
;  1752          2277  2
;  1753          2278  3          IF .CLEANUP_FLAGS[CLF_DELAQB]
;  1754          2279  3          THEN
;  1755          2280  4              BEGIN
;  1756          2281  4              LOCK_IODB ();
;  1757          2282  4              P = .IOC$GL_AQBLIST;
;  1758          2283  4              IF .P EQL .REAL_AQB
;  1759          2284  4              THEN
;  1760          2285  4                  IOC$GL_AQBLIST = .REAL_AQB[AQB$L_LINK]
;  1761          2286  4              ELSE
;  1762          2287  5                  BEGIN
;  1763          2288  5                  UNTIL .P[AQB$L_LINK] EQL .REAL_AQB
;  1764          2289  5                  DO P = .P[AQB$L_LINK];
;  1765          2290  5                  P[AQB$L_LINK] = .REAL_AQB[AQB$L_LINK];
;  1766          2291  4                  END;
;  1767          2292  4              DEALLOCATE_MEM (.REAL_AQB, 0);
```

```
;  1768     2293  4          UNLOCK_IODB ();
;  1769     2294  4          END;
;  1770     2295
;  1771     2296  ! If we have hooked up to an RVT, undo it. Note that this must be done under
;  1772     2297  ! interlock since others may be looking at the same RVT at the same time.
;  1773     2298  ! If the RVT is not disappearing entirely, remove knowledge of this volume
;  1774     2299  ! from it by zeroing the UCB entry in its list of UCB's.
;  1775     2300  !
;  1776     2301
;  1777     2302      IF .REAL_RVT NEQ 0
;  1778     2303      THEN
;  1779     2304  4       BEGIN
;  1780     2305  4       LOCK_IODB ();
;  1781     2306  4       REAL_RVT[RVT$W_REFC] = .REAL_RVT[RVT$W_REFC] - 1;
;  1782     2307  4       IF .REAL_RVT[RVT$W_REFC] EQL 0
;  1783     2308  4       THEN
;  1784     2309  4           DEALLOCATE_MEM (.REAL_RVT, 0)
;  1785     2310  4       ELSE
;  1786     2311  4           VECTOR [REAL_RVT [RVT$L_UCBLST], .REAL_VCB [VCB$W_RVN]-1] = 0;
;  1787     2312  4
;  1788     2313  4       UNLOCK_IODB ();
;  1789     2314  4       END;
;  1790     2315
;  1791     2316      IF .REAL_VCB NEQ 0
;  1792     2317      THEN DEALLOCATE_MEM (.REAL_VCB, 0);
;  1793     2318
;  1794     2319      IF .REAL_VCA NEQ 0
;  1795     2320      THEN DEALLOCATE_MEM (.REAL_VCA, 0);
;  1796     2321
;  1797     2322      IF .REAL_FCB NEQ 0
;  1798     2323      THEN DEALLOCATE_MEM (.REAL_FCB, 0);
;  1799     2324
;  1800     2325      IF .REAL_WCB NEQ 0
;  1801     2326      THEN DEALLOCATE_MEM (.REAL_WCB, 0);
;  1802     2327
;  1803     2328      IF .MTL_ENTRY NEQ 0
;  1804     2329      THEN DEALLOCATE_MEM (.MTL_ENTRY, 1);
;  1805     2330
;  1806     2331      IF .SMTL_ENTRY NEQ 0
;  1807     2332      THEN DEALLOCATE_MEM (.SMTL_ENTRY, 1);
;  1808     2333
;  1809     2334  ! Return the condition code in R0.
;  1810     2335  !
;  1811     2336
;  1812     2337      MECHANISM[CHF$L_MCH_SAVR0] = .SIGNAL[CHF$L_SIG_NAME];
;  1813     2338      $UNWIND ();
;  1814     2339
;  1815     2340      END;
;  1816     2341  1 END;                                    ! end of routine KERNEL_HANDLER


                                                .EXTRN  MAILBOX_CHANNEL
                                                .EXTRN  REAL_RVT, REAL_AQB
                                                .EXTRN  IOC$GL_AQBLIST, DEALLOCATE_MEM
                                                .EXTRN  BUG$_UNXSIGNAL, SYS$DASSGN
                                                .EXTRN  SYS$UNWIND
```

```
                                007C 00000 KERNEL_HANDLER:
                                           .WORD   Save R2,R3,R4,R5,R6                                     ; 2179
                 56 00000000G   9F   9E 00002       MOVAB   @#IOC$GL_AQBLIST, R6
                 55 00000000G   00   9E 00009       MOVAB   UNLOCK_IODB, R5
                 54 00000000G   00   9E 00010       MOVAB   LOCK_IODB, R4
                 53      0000G  CF   9E 00017       MOVAB   DEALLOCATE_MEM, R3
                 50         04  AC   D0 0001C       MOVL    SIGNAL, R0                                      ; 2250
     00000920    8F         04  A0   D1 00020       CMPL    4(R0), #2336
                                01   12 00028       BNEQ    1$
                                   04 0002A       RET
                 03               60   D1 0002B 1$:  CMPL    (R0), #3                                        ; 2254
                                  04   13 0002E       BEQL    2$
                                FEFF   00030       BUGW                                                    ; 2255
                                0000*  00032       .WORD   <BUG$_UNXSIGNAL!4>
     0B      0000G   CF               03   E1 00034 2$:  BBC     #3, CLEANUP_FLAGS, 3$                           ; 2261
                        0000G   CF      DD 00039       PUSHL   MAILBOX_CHANNEL                                 ; 2263
     00000000G   00             01   FB 0003E       CALLS   #1, SYS$DASSGN
                        0000G   CF      DD 00045 3$:  PUSHL   CHANNEL                                         ; 2268
           0000G   CF           01   FB 00049       CALLS   #1, GET_CHANNELUCB
                   52           50   D0 0004E       MOVL    R0, UCB
                   64           00   FB 00051       CALLS   #0, LOCK_IODB                                   ; 2269
                 3A   A2        08   8A 00054       BICB2   #8, 58(UCB)                                     ; 2270
                        34      A2   D4 00058       CLRL    52(UCB)                                         ; 2271
                   65           00   FB 0005B       CALLS   #0, UNLOCK_IODB                                 ; 2272
     31      0000G   CF           02   E1 0005E       BBC     #2, CLEANUP_FLAGS, 7$                           ; 2278
                   64           00   FB 00064       CALLS   #0, LOCK_IODB                                   ; 2281
                   50           66   D0 00067       MOVL    IOC$GL_AQBLIST, P                               ; 2282
                   51      0000G   CF   D0 0006A       MOVL    REAL_AQB, R1                                    ; 2283
                   51               50   D1 0006F       CMPL    P, R1
                                    06   12 00072       BNEQ    4$
                   66           10  A1   D0 00074       MOVL    16(R1), IOC$GL_AQBLIST                          ; 2285
                                11   11 00078       BRB     6$
                   51           10  A0   D1 0007A 4$:  CMPL    16(P), R1                                       ; 2288
                                06   13 0007E       BEQL    5$
                   50           10  A0   D0 00080       MOVL    16(P), P                                        ; 2289
                                F4   11 00084       BRB     4$
     10            A0           10  A1   D0 00086 5$:  MOVL    16(R1), 16(P)                                   ; 2290
                                7E   D4 0008B 6$:  CLRL    -(SP)                                           ; 2292
                                51   DD 0008D       PUSHL   R1
                   63               02   FB 0008F       CALLS   #2, DEALLOCATE_MEM
                   65               00   FB 00092       CALLS   #0, UNLOCK_IODB                                 ; 2293
                          0000G   CF   D5 00095 7$:  TSTL    REAL_RVT                                        ; 2302
                                26   13 00099       BEQL    10$
                   64           00   FB 0009B       CALLS   #0, LOCK_IODB                                   ; 2305
                   52      0000G   CF   D0 0009E       MOVL    REAL_RVT, R2                                    ; 2306
                             04    A2   B7 000A3       DECW    4(R2)
                                09   12 000A6       BNEQ    8$                                              ; 2307
                                7E   D4 000A8       CLRL    -(SP)                                           ; 2309
                                52   DD 000AA       PUSHL   R2
                   63               02   FB 000AC       CALLS   #2, DEALLOCATE_MEM
                                0D   11 000AF       BRB     9$
                   50      0000G   CF   D0 000B1 8$:  MOVL    REAL_VCB, R0                                    ; 2311
                   50         0E   A0   3C 000B6       MOVZWL  14(R0), R0
                                40 A240   D4 000BA       CLRL    64(R2)[R0]
                   65               00   FB 000BE 9$:  CALLS   #0, UNLOCK_IODB                                 ; 2313
                   50      0000G   CF   D0 000C1 10$:  MOVL    REAL_VCB, R0                                    ; 2316
```

```
                                07  13 000C6          BEQL    11$                                                     ;
                                7E  D4 000C8          CLRL    -(SP)                                                   ; 2317
                                50  DD 000CA          PUSHL   R0                                                      ;
                            63  02  FB 000CC          CALLS   #2, DEALLOCATE_MEM                                      ;
                            50  0000G CF DO 000CF 11$: MOVL    REAL_VCA, R0                                            ; 2319
                                07  13 000D4          BEQL    12$                                                     ;
                                7E  D4 000D6          CLRL    -(SP)                                                   ; 2320
                                50  DD 000D8          PUSHL   R0                                                      ;
                            63  02  FB 000DA          CALLS   #2, DEALLOCATE_MEM                                      ;
                            50  0000G CF DO 000DD 12$: MOVL    REAL_FCB, R0                                            ; 2322
                                07  13 000E2          BEQL    13$                                                     ;
                                7E  D4 000E4          CLRL    -(SP)                                                   ; 2323
                                50  DD 000E6          PUSHL   R0                                                      ;
                            63  02  FB 000E8          CALLS   #2, DEALLOCATE_MEM                                      ;
                            50  0000G CF DO 000EB 13$: MOVL    REAL_WCB, R0                                            ; 2325
                                07  13 000F0          BEQL    14$                                                     ;
                                7E  D4 000F2          CLRL    -(SP)                                                   ; 2326
                                50  DD 000F4          PUSHL   R0                                                      ;
                            63  02  FB 000F6          CALLS   #2, DEALLOCATE_MEM                                      ;
                            50  0000G CF DO 000F9 14$: MOVL    MTL_ENTRY, R0                                           ; 2328
                                07  13 000FE          BEQL    15$                                                     ;
                                01  DD 00100          PUSHL   #1                                                      ; 2329
                                50  DD 00102          PUSHL   R0                                                      ;
                            63  02  FB 00104          CALLS   #2, DEALLOCATE_MEM                                      ;
                            50  0000G CF DO 00107 15$: MOVL    SMTL_ENTRY, R0                                          ; 2331
                                07  13 0010C          BEQL    16$                                                     ;
                                01  DD 0010E          PUSHL   #1                                                      ; 2332
                                50  DD 00110          PUSHL   R0                                                      ;
                            63  02  FB 00112          CALLS   #2, DEALLOCATE_MEM                                      ;
                            50  04  AC 7D 00115 16$: MOVQ    SIGNAL, R0                                              ; 2337
                        0C  A1  04  A0 DO 00119        MOVL    4(R0), 12(R1)                                           ;
                                7E  7C 0011E          CLRQ    -(SP)                                                   ; 2338
                    00000000G   00  02 FB 00120        CALLS   #2, SYS$UNWIND                                          ;
                                    04 00127          RET                                                             ; 2341

; Routine Size:  296 bytes,    Routine Base:  $CODE$ + 0D68

; 1817          2342  1
; 1818          2343  1
; 1819          2344  1 END
; 1820          2345  0 ELUDOM


                                                                .EXTRN  LIB$SIGNAL, LIB$STOP

;                                PSECT SUMMARY
;
;      Name                     Bytes                          Attributes
;
;  $GLOBAL$                      1164  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                           64  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                          18  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        3728  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                               Library Statistics
;
;                                        --------- Symbols ---------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619       202         1        1000        00:01.9
;
;
;
;                                       COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:MOUDK2/OBJ=OBJ$:MOUDK2 MSRC$:MOUDK2/UPDATE=(ENH$:MOUDK2)
;
; Size:          3728 code + 1246 data bytes
; Run Time:        01:10.0
; Elapsed Time:    02:19.2
; Lines/CPU Min:    2010
; Lexemes/CPU-Min: 19769
; Memory Used:  546 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CHKHM2
LIS

CHNUCB
LIS

GETUIC
LIS

ERASE
LIS

LEFTONE
LIS

MOUDK2
LIS

CHKHM1
LIS

CHKSM2
LIS

CLUSTRMNT
LIS

INIFC2
LIS

MOUDK1
LIS

MAKRVT
LIS

MAKLOG
LIS

0245 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY