```
MMM         MMM     OOOOOOOOO     UUU         UUU  NNN         NNN  TTTTTTTTTTTTTTT
MMM         MMM     OOOOOOOOO     UUU         UUU  NNN         NNN  TTTTTTTTTTTTTTT
MMM         MMM     OOOOOOOOO     UUU         UUU  NNN         NNN  TTTTTTTTTTTTTTT
MMMMMM   MMMMMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMMMMM   MMMMMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMMMMM   MMMMMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM   MMM   MMM  OOO         OOO  UUU         UUU  NNNNNN      NNN        TTT
MMM   MMM   MMM  OOO         OOO  UUU         UUU  NNNNNN      NNN        TTT
MMM   MMM   MMM  OOO         OOO  UUU         UUU  NNNNNN      NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN   NNN   NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN   NNN   NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN   NNN   NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN      NNNNNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN      NNNNNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN      NNNNNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM         MMM     OOOOOOOOO     UUUUUUUUUUUUUUU  NNN         NNN        TTT
MMM         MMM     OOOOOOOOO     UUUUUUUUUUUUUUU  NNN         NNN        TTT
MMM         MMM     OOOOOOOOO     UUUUUUUUUUUUUUU  NNN         NNN        TTT
```

```
_S

Sy
--
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI

LI
LI


LI
LI
LI
LI
LI
LI
LI
LI
LN
LN
LO
LO

LO
LO
LO
LO
MA

MO
MO
MO
MO
MO
MO
MO


MO
MO
```

```
**FILE**ID**VMOUNT                        C  4




VV        VV  MM        MM    OOOOOO    UU        UU  NN        NN  TTTTTTTTTT
VV        VV  MM        MM    OOOOOO    UU        UU  NN        NN  TTTTTTTTTT
VV        VV  MMMM    MMMM  OO      OO  UU        UU  NN        NN      TT
VV        VV  MMMM    MMMM  OO      OO  UU        UU  NN        NN      TT
VV        VV  MM  MM  MM    OO      OO  UU        UU  NNNN      NN      TT
VV        VV  MM  MM  MM    OO      OO  UU        UU  NNNN      NN      TT
VV        VV  MM        MM  OO      OO  UU        UU  NN  NN    NN      TT
VV        VV  MM        MM  OO      OO  UU        UU  NN  NN    NN      TT
VV        VV  MM        MM  OO      OO  UU        UU  NN      NNNN      TT
VV        VV  MM        MM  OO      OO  UU        UU  NN      NNNN      TT
  VV    VV    MM        MM  OO      OO  UU        UU  NN        NN      TT        ....
  VV    VV    MM        MM  OO      OO  UU        UU  NN        NN      TT        ....
    VV        MM        MM    OOOOOO    UUUUUUUUUU    NN        NN      TT        ....
    VV        MM        MM    OOOOOO    UUUUUUUUUU    NN        NN      TT        ....

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

```
0 MODULE VMOUNT (
0               LANGUAGE (BLISS32),
0               ADDRESSING_MODE (NONEXTERNAL = LONG_RELATIVE),
0               IDENT = 'V04-002'
0               ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:  MOUNT Utility Structure Levels 1 & 2
1 !
1 ! ABSTRACT:
1 !
1 !       This is the main routine of the MOUNT utility. It provides the
1 !       general control flow of the MOUNT command and contains most of
1 !       the base data structures.
1 !
1 ! ENVIRONMENT:
1 !
1 !       STARLET operating system, including privileged system services
1 !       and internal exec routines.
1 !
1 !--
1 !
1 !
1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  5-Sep-1977  16:58
1 !
1 ! MODIFIED BY:
1 !
1 !       V04-002 HH0056          Hai Huang                   10-Sep-1984
1 !               Suppress outputting VOLINV error messages during
1 !               VOLINV retries.
1 !
```

```
1 !     V04-001 HH0055          Hai Huang                       06-Sep-1984
1 !             Send mount/cluster requests with operator assist
1 !             disabled.
1 !
1 !     V03-035 CDS0005         Christian D. Saether            29-Aug-1984
1 !             Call STAND_ALONE_REBUILD routine which will
1 !             only do rebuild if necessary at that time.
1 !
1 !     V03-034 HH0043          Hai Huang                       07-Aug-1984
1 !             Wait a while before retrying IOC$SEARCH.
1 !
1 !     V03-033 HH0042          Hai Huang                       27-Jul-1984
1 !             Clear the global lock storage area during run time.
1 !
1 !     V03-032 HH0041          Hai Huang                       24-Jul-1984
1 !             Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
1 !
1 !     V03-031 HH0037          Hai Huang                       12-Jul-1984
1 !             Make the label lock node-specific, i.e. make the CSID
1 !             part of the label lock.
1 !
1 !     V03-030 HH0036          Hai Huang                       11-Jul-1984
1 !             Send the mount request cluster wide even if the volume
1 !             is already mounted on the local node.
1 !
1 !     V03-029 HH0034          Hai Huang                       09-Jul-1984
1 !             Add yet another interlock to serialize shared mounts.
1 !
1 !     V03-028 HH0032          Hai Huang                       05-Jul-1984
1 !             For private mounts, transfer device ownership to the top
1 !             level process in the process tree.
1 !
1 !     V03-027 HH0024          Hai Huang                       18-Jun-1984
1 !             Do not call IOC$LOCK_DEV to test mode of the device lock,
1 !             as this routine could corrupt the lock value block.
1 !
1 !     V03-026 HH0021          Hai Huang                       14-May-1984
1 !             Refine HH0019 to mark the device as allocated after
1 !             IOC$SEARCH while holding the I/O database mutex. Also,
1 !             reject private mounts if IOC$SEARCH failed.
1 !
1 !     V03-025 HH0019          Hai Huang                       07-May-1984
1 !             Properly interlock simultaneous mounts in a cluster-
1 !             environment.
1 !
1 !     V03-024 HH0016          Hai Huang                       23-Apr-1984
1 !             Get the device name if IOC$SEARCH failed.
1 !
1 !     V03-023 HH0015          Hai Huang                       20-Apr-1984
1 !             Get IOC$SEARCH to return the lock value block of the
1 !             device lock.
1 !
1 !     V03-022 HH0010          Hai Huang                       30-Mar-1984
1 !             Fix generic mount.
1 !
1 !     V03-021 HH0004          Hai Huang                       09-Mar-1984
1 !             Add cluster-wide mount support.
```

```
1 !
1 !        V03-020 HH0002          Hai Huang               02-Feb-1984
1 !                Add job-wide mount support.
1 !
1 !        V03-019 ACG0369         Andrew C. Goldstein,    8-Nov-1983  11:24
1 !                Don't issue IO$_AVAILABLE on mount failure of mounted disk
1 !
1 !        V03-018 CDS0004         Christian D. Saether    13-Sep-1983
1 !                Only clear VALID for tapes in the main error handler.
1 !                Move the CLEAR_VALID routine here from RDHOME as it
1 !                is only referenced here now.
1 !
1 !        V03-017 TCM0003         Trudy C. Matthews       07-Sep-1983
1 !                When converting the exclusive device lock to a shared lock,
1 !                make sure it is still system-owned.
1 !
1 !        V03-016 TCM0002         Trudy C. Matthews       01-Sep-1983
1 !                Make allocating a device followed by mounting a shared
1 !                volume on that device work correctly (i.e. deallocate
1 !                the device and convert the lock to CR mode).
1 !
1 !        V03-015 CDS0003         Christian D. Saether    5-Aug-1983
1 !                Add cluster consistency checking routines.
1 !                Add status block to GETDVIW call so that wait
1 !                always works correctly.
1 !
1 !        V03-014 CDS0002         Christian D. Saether    3-Aug-1983
1 !                Remove the device ref count check prior to assigning
1 !                the channel (from tcm0001) as it was racy.
1 !
1 !        V03-013 STJ3015         Steven T. Jeffreys      30-Jul-1983
1 !                Fix link-time truncation error.
1 !
1 !        V03-012 TCM0001         Trudy C. Matthews       28-Jul-1983
1 !                Re-write the MOUNT_VOLUME routine so that it uses a
1 !                mount interlock rather than temporarily allocating the
1 !                volume.  Also ensure that cluster-wide locks are taken
1 !                out in the appropriate mode (EX for private mounts and
1 !                CR for shared mounts).
1 !
1 !        V03-011 STJ3113         Steven T. Jeffreys,     26-Jul-1983
1 !                Moved ACTIVATE_JOURNAL and helper routines to their own
1 !                module, RUJMAN.
1 !
1 !        V03-010 STJ3111         Steven T. Jeffreys,     18-Jul-1983
1 !                When the privileges are amplified, take pains to include
1 !                those privileges that are in the second longword of the
1 !                privilege mask, notably PRMJNL privilege.
1 !
1 !        V03-009 DMW4045         DMWalp                   7-Jun-1983
1 !                Remove (S)LOG_Entry
1 !
1 !        V03-008 CDS0001         Christian D. Saether    28-May-1983
1 !                Tolerate allocation failure for F11B mounts.
1 !
1 !        V03-007 STJ3102         Steven T. Jeffreys,     25-May-1983
1 !                - Add call to $CREJNL.
```

```
1 !
1 !     V03-006 MMD0115          Meg Dumont,        29-Mar-1983  0:39
1 !             Add OPT_OVR_VOLO to override options set
1 !
1 !     V03-005 STJ3061          Steven T. Jeffreys,     08-Mar-1983
1 !             - Grant user PSWAPM privilege.  Needed to create ACP.
1 !
1 !     V03-004 STJ50311         Steven T. Jeffreys,     11-Feb-1983
1 !             - Make all uses of PHYS_NAME indexed by DEVICE_INDEX
1 !             - Ensure DEVICE_INDEX is not reset on retry
1 !             - Remove references to  FIRST_CHANNEL.
1 !             - Make CALLERS_ACMOD a global cell containing the
1 !               caller's access mode.
1 !             - Changed device allocation/deallocation logic.
1 !               Moved routine DEALLOCATE_DEVICE to ASSIST.
1 !
1 !     V03-003 STJ3037          Steven T. Jeffreys,     14-Oct-1982
1 !             If the mount attempt fails, free up the drive(s) via
1 !             an IO$_AVAILABLE $qio.
1 !
1 !     V03-002 KTA0103          Kerbey T. Altmann       29-Jun-1982
1 !             Change a register to NOPRESERVE in DEALLOCATE_DEVICE.
1 !
1 !     V03-001 STJ0252          Steven T. Jeffreys,     03-Apr-1982
1 !             - Allocate devices in the access mode of the caller.
1 !             - Check allocation return status and terminate the mount
1 !               attempt if the specified device does not exist.
1 !             - Manually deallocate shared disk volumes after
1 !               they are mounted.  This is necessitated by a
1 !               change to $DALLOC such that mounted volumes may
1 !               no longer be deallocated.
1 !
1 !     V02-020 STJ0229          Steven T. Jeffreys,     01-Mar-1982
1 !             - Set inhibit message bit in the exit status code
1 !               if the message text was written via $PUTMSG.
1 !
1 !     V02-019 STJ0190          Steven T. Jeffreys,     02-Feb-1982
1 !             - Zero OWN and GLOBAL storage to guaranty restartablity.
1 !
1 !     V02-018 STJ0170          Steven T. Jeffreys,     13-Jan-1982
1 !             More work for $MOUNT support.
1 !
1 !     V02-017 RNG0001          Rod N. Gamache          05-Jan-1982
1 !             Declare MOUNT_OPTIONS to be external.
1 !
1 !     V02-016 STJ0161          Steven T. Jeffreys,     04-Jan-1982
1 !             Changed OPT_OVERLOCK to OPT_OVR_LOCK.  Do not print
1 !             messages if OPT_MESSAGE is not set.
1 !
1 !     V02-015 ACG0246          Andrew C. Goldstein,    4-Jan-1982  15:21
1 !             Add /OVER:LOCK
1 !
1 !     V02-014 STJ0149          Stevem T. Jeffreys      02-Jan-1981
1 !             Extensive rewrite to support the $MOUNT system service.
1 !
1 !     V02-013 STJ0089          Steven T. Jeffreys      09-Aug-1981
1 !             Reset mount options at the beginning of each attempt
```

```
!               to mount a volume.
!
!       V02-012 DMW0008         David Michael Walp      10-Jun-1981
!               Liberal re-write to facilitate operator assisted mount
!               work for tapes.
!
!       V0111   STJ0005         Steven T. Jeffreys,     9-Oct-1980
!               Liberal re-write to facilitate operator assisted mount.
!
!       V0110   ACG0125         Andrew C. Goldstein,    23-Jan-1980  14:57
!               Init USER_STATUS cell for correct header error reporting
!
!       V0109   ACG0123         Andrew C. Goldstein,    17-Jan-1980  20:33
!               Complete integration of disk rebuild
!
!       V0108   RIH0051         Richard I. Hustvedt,    13-Jan-1979 14:33
!               Add call to rebuild bitmaps and quota file on volume mount.
!
!       V0107   ACG0079         Andrew C. Goldstein,    5-Nov-1979  13:53
!               Structures for file ID and extent cacheing
!
!       V0106   ACG0072         Andrew C. Goldstein,    15-Oct-1979  16:12
!               Check primary and secondary device characteristics
!
!       V0105   ACG0069         Andrew C. Goldstein,    8-Oct-1979  18:32
!               Remove device data table
!
!       V0104   ACG0044         Andrew C. Goldstein, 18-Jun-1979  16:15
!       Add disk quota support
!
!       V0103   ACG21786        Andrew C. Goldstein, 2-Feb-1979  14:19
!       Fix home block scan loop limit conditional
!
!       V0102   ACG0013         Andrew C. Goldstein, 5-Jan-1979  13:52
!       Don't clear valid bit on failure on already mounted volume
!
!       V0101   ACG0003         Andrew C. Goldstein, 27-Nov-1978  17:48
!       Add multi-volume support for disk
!
!       V0100   ACG00001        Andrew C. Goldstein, 10-Oct-1978  19:56
!       Previous revision history moved to [MOUNT.SRC]MOUNT.REV
!**


LIBRARY 'SYS$LIBRARY:LIB.L32';
REQUIRE 'SRC$:MOUDEF.B32';
REQUIRE 'LIBD$:[VMSLIB.OBJ]INITMSG.REQ';


FORWARD ROUTINE
        SYS$VMOUNT,                             ! entry point (w/o operator assist)
        VMOUNT_ENVELOPE,                        ! base call frame for MOUNT_VOLUME
        REBUILD_ENVELOPE,                       ! base call frame for REBUILD
        INTERCEPT_SIGNAL,                       ! Intercept EXEC mode signal
        MOUNT_VOLUME,                           ! Mount a given volume
        MAIN_HANDLER,                           ! main condition handler
        FORCE_DISMOUNT,                         ! dismount a volume just mounted
```

```
;        1          CLEAR_VALID,                          ! Clear VALID flag in UCB.
;        1          DALLOC_SHR_DEV,                       ! deallocate device for shared mount
;        1          XFER_DEV_OWNER,                       ! transfer device ownership
;        1          MOUNT_CLUSTER,                        ! cluster-wide mount
;        1          MOUNT_ENCIPHER,                       ! create a cluster-mount packet
;        1          SEARCH_DEVICE,                        ! generic device search/allocate routine
;        1          DEQ_MOUNT_LOCK     : NOVALUE,         ! dequeue the mount lock
;        1          WAIT_DELTA         : NOVALUE;         ! wait before IOC$SEARCH retry
;        1
```

```
!+
!
! Own storage for general use in the MOUNT utility
! Note that DATA_BASE_READY and STORED_CONTEXT initialzed
! in the module ASSIST.
!
!-

GLOBAL
        VMOUNT_GBL_START: VECTOR [0],     ! Mark start of global storage
        STORED_CONTEXT  : BITVECTOR [32],! store the context of some 1 time only
        DATA_BASE_READY : LONG,           ! Boolean
        DEV_ALLOCATED   : BITVECTOR [DEVMAX] VOLATILE,  !Indicates which physical devices are allocated
        DEV_ACQUIRED    : BITVECTOR [DEVMAX] VOLATILE,
                                          ! Indicates which devices have been
                                          ! interlocked.
        LOCK_STATUS     : VECTOR [2],     ! Lock status block for $ENQ.
        CLEANUP_ALLOC   : BITVECTOR [DEVMAX] VOLATILE,  !Indicates which physical devices need to be dealloc
        CLEANUP_FLAGS   : BITVECTOR [32] VOLATILE, ! error cleanup status flags
        CHANNEL         : LONG VOLATILE,! channel number for I/O
        DEVICE_INDEX    : LONG VOLATILE,! Index into device list
        MAILBOX_CHANNEL,                  ! channel number of ACP termination mailbox
        CALLERS_ACMOD   : LONG,           ! caller's access mode
        PHYS_COUNT,                       ! number of physical devices in use
        PHYS_NAME       : VECTOR [DEVMAX*2],    ! descriptor of physical device name
        NAME_BUFFER     : VECTOR [NAMEBUF_LEN*DEVMAX, BYTE],
                                          ! string buffer for physical device name
        LOG_BUFFER      : VECTOR [63, BYTE],
                                          ! buffer to construct logical name
        HOME_BLOCK      : BBLOCK [512],   ! buffer for volume header label or home block
        DEVICE_CHAR     : BBLOCK [DIB$K_LENGTH],
                                          ! buffer for device characteristics
        DEVICE_CHAR2    : BBLOCK [DIB$K_LENGTH],
                                          ! buffer for sec. device characteristics
        HOMEBLOCK_LBN,                    ! LBN of home block read
        HEADER_LBN,                       ! LBN of file header read
        CURRENT_RVN,                      ! RVN of disk being mounted
        USER_STATUS     : VECTOR [2],     ! status return for various routines
        CURRENT_VCB     : REF BBLOCK,     ! address of VCB used by CHECK_HEADER2
        REAL_MVL        : REF BBLOCK,     ! address of MVL allocated for mag tape volume
        REAL_RVT        : REF BBLOCK,     ! address of RVT allocated for mag tape volume
        REAL_VCB        : REF BBLOCK,     ! address of VCB allocated for volume
        REAL_VCA        : REF BBLOCK,     ! address of cache block allocated for vol.
        REAL_FCB        : REF BBLOCK,     ! address of FCB allocated for volume
        REAL_WCB        : REF BBLOCK,     ! address of window allocated for volume
        REAL_AQB        : REF BBLOCK,     ! address of AQB allocated for volume
        MTL_ENTRY       : REF BBLOCK,     ! address of mounted volume list entry
        SMTL_ENTRY      : REF BBLOCK,     ! address of volume set MTL
        MOUNT_ITMLST,                     ! address of the mount item list
        LABLCK_STATUS   : VECTOR [2],     ! label lock status
        VMOUNT_GBL_END  : VECTOR [0];     ! Mark end of GLOBAL storage


GLOBAL BIND
        VOL1            = HOME_BLOCK;     ! BUFFER FOR VOL1 MAGNETIC TAPE LABEL

GLOBAL
```

```
ALLDEVNAM_BUF    : VECTOR [NAMEBUF_LEN, BYTE]
                   INITIAL (BYTE ('MOU$',
                            REP NAMEBUF_LEN-4 OF (' '))),
                            ! string buffer for alloc class devnam
ALLDEVNAM_DESC   : VECTOR [2] INITIAL (0, ALLDEVNAM_BUF),
                            ! descriptor for alloc class devnam
DEVCHAR_DESC     : VECTOR [2] INITIAL (DIB$K_LENGTH, DEVICE_CHAR),
                            ! descriptor for device characteristics
DEVCHAR_DESC2    : VECTOR [2] INITIAL (DIB$K_LENGTH, DEVICE_CHAR2),
                            ! descriptor for sec. device characteristics

LABLCKNAM_BUF    : VECTOR [NAMEBUF_LEN+4, BYTE]
                   INITIAL (BYTE ('MOU$',
                            REP NAMEBUF_LEN OF (' '))),
                            ! label lock name buffer
LABLCKNAM_DESC   : VECTOR [2, LONG]
                   INITIAL (0, LABLCKNAM_BUF);
                            ! label lock descriptor
```

```
1 GLOBAL ROUTINE SYS$VMOUNT (ITEM_LIST) =
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This is the main routine of the MOUNT utility.
1 !
1 ! CALLING SEQUENCE:
1 !       $MOUNT (arglist)
1 !
1 ! INPUT PARAMETERS:
1 !       ITEM_LIST         : Address of a $GETJPI-like item list
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       assorted status values
1 !
1 ! SIDE EFFECTS:
1 !       volume(s) mounted, device data base updated
1 !
1 !--
1
2 BEGIN
2
2 BUILTIN
2       MOVPSL,                                         ! Get current PSL
2       CALLG,                                          ! Used to call CHECK_PARAMS
2       AP;                                             ! Used to pass params to CHECK_PARAMS
2
2 EXTERNAL ROUTINE
2       ACTIVATE_JOURNAL: ADDRESSING_MODE (GENERAL),    ! activate RUJ
2       $DALLOC_DEVS$U  : ADDRESSING_MODE (GENERAL),
2       CHECK_PARAMS;                                   ! Process the user-supplied parameters
2
2 EXTERNAL
2       DEVICE_COUNT    : ADDRESSING_MODE (GENERAL)
2                         LONG,                         ! Number of devices specified
2       LCK_GLOBAL_START: ADDRESSING_MODE (GENERAL),
2                                                       ! Start of global lock area
2       LCK_GLOBAL_END  : ADDRESSING_MODE (GENERAL);
2                                                       ! End of global lock area
2
2 LOCAL
2       !
2       ! Declare the privileges that are necessary for MOUNT to work.
2       !
2       CURRENT_PSL     : BBLOCK [4],                   ! holds current PSL
2       MOUNT_PRIVS     : BBLOCK [8],                   ! Amplified privilege mask
2       USER_PRIVS      : BBLOCK [8],                   ! Temp storage for privilege mask
```

```
2           STATUS;                                        ! system service status
2
2
2 CHANNEL        = 0;
2 USER_STATUS    = 1;
2
3 MOUNT_PRIVS    =  (1^$BITPOSITION (PRV$V_ACNT)    OR                    ! Amplified privilege mask
3                    1^$BITPOSITION (PRV$V_ALTPRI)  OR
3                    1^$BITPOSITION (PRV$V_BUGCHK)  OR
3                    1^$BITPOSITION (PRV$V_BYPASS)  OR
3                    1^$BITPOSITION (PRV$V_DETACH)  OR
3                    1^$BITPOSITION (PRV$V_EXQUOTA)OR
3                    1^$BITPOSITION (PRV$V_GROUP)   OR
3                    1^$BITPOSITION (PRV$V_MOUNT)   OR
3                    1^$BITPOSITION (PRV$V_PHY_IO)  OR
3                    1^$BITPOSITION (PRV$V_PSWAPM)  OR
3                    1^$BITPOSITION (PRV$V_TMPMBX)  OR
3                    1^$BITPOSITION (PRV$V_SETPRV)  OR
3                    1^$BITPOSITION (PRV$V_SYSLCK)  OR
3                    1^$BITPOSITION (PRV$V_WORLD)
2                   );
2 MOUNT_PRIVS[PRV$V_PRMJNL] = 1;                           ! PRMJNL is in the 2nd longword
2
2 ! Process the user-supplied parameters, if
2 ! we haven't already.  The conditional call
2 ! is to save the overhead of having to do it
2 ! for each attempt at a mount, and to make
2 ! sure that it is done at least once.
2 !
2
2 IF NOT .DATA_BASE_READY
2 THEN
3     BEGIN
3     CH$FILL (0, VMOUNT_GBL_END-VMOUNT_GBL_START, VMOUNT_GBL_START);
3     CH$FILL (0, LCK_GLOBAL_END-LCK_GLOBAL_START, LCK_GLOBAL_START);
3     MOUNT_ITMLST  = .ITEM_LIST;
3     DATA_BASE_READY = 1;
4     IF NOT (STATUS = CALLG (.AP, CHECK_PARAMS))
3     THEN
3         RETURN (.STATUS);
3     MOVPSL (CURRENT_PSL);
3     CALLERS_ACMOD = .CURRENT_PSL [PSL$V_PRVMOD];
2     END;
2
2 ! Save the current privilege mask and grant the
2 ! caller the necessary privileges.
2 !
2
2 $SETPRV (ENBFLG=1, PRVADR=MOUNT_PRIVS, PRVPRV=USER_PRIVS);
2
2 ! Loop for all devices in the command line to mount multiple disks and tapes.
2 ! However, it is necessary to reset DEVICE_INDEX for tape mounts because tape
2 ! volumes are not mounted until every volume in the command line has been
2 ! processed, and an error condition on the Nth volume will force all the work
2 ! done on previous volumes to be undone.
2 !
2
```

```
2 IF (.DEVICE_INDEX GTR 0) AND .STORED_CONTEXT[TAPE_MOUNT]
2 THEN
2     DEVICE_INDEX = 0;
2
2 INCR I FROM .DEVICE_INDEX TO .DEVICE_COUNT-1
2 DO
3     BEGIN
3     !
3     ! Mount the volume.  If the attempt failed, abort the mount
3     ! and return the error status.  Always dequeue the mount interlock(s),
3     ! no matter if the mount attempt succeeded or failed.
3     !
3     STATUS = VMOUNT_ENVELOPE ();
3     KERNEL_CALL ( DEQ_MOUNT_LOCK );
3     IF .LABLCK_STATUS [1] NEQ 0                    ! Dequeue the label lock if it exists
3     THEN                                           !
3         $DEQ ( LKID = .LABLCK_STATUS [1] );
3
3     IF NOT .STATUS
3     THEN
4         BEGIN
4         $SETPRV (ENBFLG=0, PRVADR=MOUNT_PRIVS); ! Clear granted privileges
4         $SETPRV (ENBFLG=1, PRVADR=USER_PRIVS);  ! Restore old privileges
4         RETURN (.STATUS);
3         END;
3     DEVICE_INDEX = .DEVICE_INDEX+1;
2     END;
2
2 !
2 ! Deallocate all devices that are not mounted.
2 !
2 $DALLOC_DEVS$U (0);
2
2 !
2 ! Rebuild volume if mounting files-11 ODS-2 disk
2 !
2
2 IF .CLEANUP_FLAGS[CLF_REBUILD]
2 THEN
3     BEGIN
3     STATUS = REBUILD_ENVELOPE ();
3     $DASSGN (CHAN   = .CHANNEL);                ! Deassign channel used by REBUILD
2     END;
2
2 !
2 ! If the rebuild was successful, attempt to activate the RUJ.
2 !
2 IF .STATUS
2 THEN
2     STATUS = ACTIVATE_JOURNAL ();
2
2 !
2 ! If the mount was successful, sent this mount request cluster-wide
2 ! when appropriate.
2 !
2 IF .STATUS
2 THEN
```

```
 2      STATUS = MOUNT_CLUSTER (.ITEM_LIST);                ! Mount cluster-wide
 2
 2  ! Revoke any privileges that were granted.
 2  !
 2
 2  $SETPRV (ENBFLG=0, PRVADR=MOUNT_PRIVS);                 ! Clear granted privileges
 2  $SETPRV (ENBFLG=1, PRVADR=USER_PRIVS);                  ! Restore old privileges
 2
 3  RETURN (.STATUS)
 3
 1  END;                                                    ! end of routine MOUNT_COMMAND

                                                           .TITLE  VMOUNT
                                                           .IDENT  \V04-002\

                                                           .PSECT  $GLOBAL$,NOEXE,2

                                       00000 VMOUNT_GBL_START::
                                                           .BLKB   0
                                       00000 STORED_CONTEXT::
                                                           .BLKB   4
                                       00004 DATA_BASE_READY::
                                                           .BLKB   4
                                       00008 DEV_ALLOCATED::
                                                           .BLKB   2
                                       0000A               .BLKB   2
                                       0000C DEV_ACQUIRED::
                                                           .BLKB   2
                                       0000E               .BLKB   2
                                       00010 LOCK_STATUS::
                                                           .BLKB   8
                                       00018 CLEANUP_ALLOC::
                                                           .BLKB   2
                                       0001A               .BLKB   2
                                       0001C CLEANUP_FLAGS::
                                                           .BLKB   4
                                       00020 CHANNEL::
                                                           .BLKB   4
                                       00024 DEVICE_INDEX::
                                                           .BLKB   4
                                       00028 MAILBOX_CHANNEL::
                                                           .BLKB   4
                                       0002C CALLERS_ACMOD::
                                                           .BLKB   4
                                       00030 PHYS_COUNT::
                                                           .BLKB   4
                                       00034 PHYS_NAME::
                                                           .BLKB   128
                                       000B4 NAME_BUFFER::
                                                           .BLKB   512
                                       002B4 LOG_BUFFER::
                                                           .BLKB   63
                                       002F3               .BLKB   1
                                       002F4 HOME_BLOCK::
                                                           .BLKB   512
                                       004F4 DEVICE_CHAR::
```

```
                                                .BLKB   116
                          00568 DEVICE_CHAR2::
                                                .BLKB   116
                          005DC HOMEBLOCK_LBN::
                                                .BLKB   4
                          005E0 HEADER_LBN::
                                                .BLKB   4
                          005E4 CURRENT_RVN::
                                                .BLKB   4
                          005E8 USER_STATUS::
                                                .BLKB   8
                          005F0 CURRENT_VCB::
                                                .BLKB   4
                          005F4 REAL_MVL::
                                                .BLKB   4
                          005F8 REAL_RVT::
                                                .BLKB   4
                          005FC REAL_VCB::
                                                .BLKB   4
                          00600 REAL_VCA::
                                                .BLKB   4
                          00604 REAL_FCB::
                                                .BLKB   4
                          00608 REAL_WCB::
                                                .BLKB   4
                          0060C REAL_AQB::
                                                .BLKB   4
                          00610 MTL_ENTRY::
                                                .BLKB   4
                          00614 SMTL_ENTRY::
                                                .BLKB   4
                          00618 MOUNT_ITMLST::
                                                .BLKB   4
                          0061C LABLCK_STATUS::
                                                .BLKB   8
                          00624 VMOUNT_GBL_END::
                                                .BLKB   0
        24  55  4F  4D    00624 ALLDEVNAM_BUF::
                                                .ASCII  \MOU$\
                    20    00628                 .ASCII  \ \
                    20    00629                 .ASCII  \ \
                    20    0062A                 .ASCII  \ \
                    20    0062B                 .ASCII  \ \
                    20    0062C                 .ASCII  \ \
                    20    0062D                 .ASCII  \ \
                    20    0062E                 .ASCII  \ \
                    20    0062F                 .ASCII  \ \
                    20    00630                 .ASCII  \ \
                    20    00631                 .ASCII  \ \
                    20    00632                 .ASCII  \ \
                    20    00633                 .ASCII  \ \
                    20    00634                 .ASCII  \ \
                    20    00635                 .ASCII  \ \
                    20    00636                 .ASCII  \ \
                    20    00637                 .ASCII  \ \
                    20    00638                 .ASCII  \ \
                    20    00639                 .ASCII  \ \
                20  0063A           .ASCII  \ \                                   ;
                20  0063B           .ASCII  \ \                                   ;
                20  0063C           .ASCII  \ \                                   ;
                20  0063D           .ASCII  \ \                                   ;
                20  0063E           .ASCII  \ \                                   ;
                20  0063F           .ASCII  \ \                                   ;
                20  00640           .ASCII  \ \                                   ;
                20  00641           .ASCII  \ \                                   ;
                20  00642           .ASCII  \ \                                   ;
                20  00643           .ASCII  \ \                                   ;
          00000000  00644 ALLDEVNAM_DESC::
                                    .LONG   0                                     ;
         00000000'  00648           .ADDRESS ALLDEVNAM_BUF                        ;
          00000074  0064C DEVCHAR_DESC::
                                    .LONG   116                                   ;
         00000000'  00650           .ADDRESS DEVICE_CHAR                          ;
          00000074  00654 DEVCHAR_DESC2::
                                    .LONG   116                                   ;
         00000000'  00658           .ADDRESS DEVICE_CHAR2                         ;
       24  55  4F  4D  0065C LABLCKNAM_BUF::
                                    .ASCII  \MOU$\                                ;
                20  00660           .ASCII  \ \                                   ;
                20  00661           .ASCII  \ \                                   ;
                20  00662           .ASCII  \ \                                   ;
                20  00663           .ASCII  \ \                                   ;
                20  00664           .ASCII  \ \                                   ;
                20  00665           .ASCII  \ \                                   ;
                20  00666           .ASCII  \ \                                   ;
                20  00667           .ASCII  \ \                                   ;
                20  00668           .ASCII  \ \                                   ;
                20  00669           .ASCII  \ \                                   ;
                20  0066A           .ASCII  \ \                                   ;
                20  0066B           .ASCII  \ \                                   ;
                20  0066C           .ASCII  \ \                                   ;
                20  0066D           .ASCII  \ \                                   ;
                20  0066E           .ASCII  \ \                                   ;
                20  0066F           .ASCII  \ \                                   ;
                20  00670           .ASCII  \ \                                   ;
                20  00671           .ASCII  \ \                                   ;
                20  00672           .ASCII  \ \                                   ;
                20  00673           .ASCII  \ \                                   ;
                20  00674           .ASCII  \ \                                   ;
                20  00675           .ASCII  \ \                                   ;
                20  00676           .ASCII  \ \                                   ;
                20  00677           .ASCII  \ \                                   ;
                20  00678           .ASCII  \ \                                   ;
                20  00679           .ASCII  \ \                                   ;
                20  0067A           .ASCII  \ \                                   ;
                20  0067B           .ASCII  \ \                                   ;
                20  0067C           .ASCII  \ \                                   ;
                20  0067D           .ASCII  \ \                                   ;
                20  0067E           .ASCII  \ \                                   ;
                20  0067F           .ASCII  \ \                                   ;
          00000000  00680 LABLCKNAM_DESC::
                                    .LONG   0                                     ;
         00000000'  00684           .ADDRESS LABLCKNAM_BUF                        ;
```

```
                                              VOL1==            HOME_BLOCK
                                                       .EXTRN   ACTIVATE_JOURNAL
                                                       .EXTRN   $DALLOC_DEVS$U, CHECK_PARAMS
                                                       .EXTRN   DEVICE_COUNT, LCK_GLOBAL_START
                                                       .EXTRN   LCK_GLOBAL_END, SYS$SETPRV
                                                       .EXTRN   SYS$CMKRNL, SYS$DEQ
                                                       .EXTRN   SYS$DASSGN

                                                       .PSECT   $CODE$,NOWRT,2

                                   00FC 00000          .ENTRY   SYS$VMOUNT, Save R2,R3,R4,R5,R6,R7          ; 1033
                    57 00000000G    00  9E 00002       MOVAB    SYS$SETPRV, R7                              ;
                    56 00000000'    EF  9E 00009       MOVAB    DEVICE_INDEX, R6                            ;
                    5E              10  C2 00010       SUBL2    #16, SP                                     ;
                             FC     A6  D4 00013       CLRL     CHANNEL                                     ; 1093
                  05C4  C6          01  D0 00016       MOVL     #1, USER_STATUS                             ; 1094
                    08  AE 60CBF320 8F  D0 0001B       MOVL     #1623978784, MOUNT_PRIVS                    ; 1096
                    0C  AE          20  88 00023       BISB2    #32, MOUNT_PRIVS+4                          ; 1111
                        35   E0     A6  E8 00027       BLBS     DATA_BASE_READY, 2$                         ; 1120
   0624  8F     00      6E          00  2C 0002B       MOVC5    #0, (SP), #0, #1572, VMOUNT_GBL_START       ; 1123
                             DC     A6     00032
   0000* 8F     00      6E          00  2C 00034       MOVC5    #0, (SP), #0, #<LCK_GLOBAL_END--            ; 1124
                    00000000G       00     0003B                LCK_GLOBAL_START>, LCK_GLOBAL_START
                  05F4  C6   04     AC  D0 00040       MOVL     ITEM_LIST, MOUNT_ITMLST                     ; 1125
                    E0  A6          01  D0 00046       MOVL     #1, DATA_BASE_READY                         ; 1126
                  0000G CF          6C  FA 0004A       CALLG    (AP), CHECK_PARAMS                          ; 1127
                        52          50  D0 0004F       MOVL     R0, STATUS                                  ;
                        03          50  E8 00052       BLBS     R0, 1$                                      ;
                                  00B6  31 00055       BRW      9$                                          ;
                                    50  DC 00058 1$:   MOVPSL   CURRENT_PSL                                 ; 1130
      08  A6    50      02          16  EF 0005A       EXTZV    #22, #2, CURRENT_PSL, CALLERS_ACMOD         ; 1131
                                    5E  DD 00060 2$:   PUSHL    SP                                          ; 1138
                                    7E  D4 00062       CLRL     -(SP)                                       ;
                             10     AE  9F 00064       PUSHAB   MOUNT_PRIVS                                 ;
                                    01  DD 00067       PUSHL    #1                                          ;
                        67          04  FB 00069       CALLS    #4, SYS$SETPRV                              ;
                                    66  D5 0006C       TSTL     DEVICE_INDEX                                ; 1147
                                    06  15 0006E       BLEQ     3$                                          ;
                        02   DC     A6  E9 00070       BLBC     STORED_CONTEXT, 3$                          ;
                                    66  D4 00074       CLRL     DEVICE_INDEX                                ; 1149
                    54 00000000G    00  D0 00076 3$:   MOVL     DEVICE_COUNT, R4                            ; 1151
                53      66          01  C3 0007D       SUBL3    #1, DEVICE_INDEX, I                         ;
                                    34  11 00081       BRB      6$                                          ;
             00000000V  EF          00  FB 00083 4$:   CALLS    #0, VMOUNT_ENVELOPE                         ; 1159
                        52          50  D0 0008A       MOVL     R0, STATUS                                  ;
                                    7E  D4 0008D       CLRL     -(SP)                                       ; 1160
                                    5E  DD 0008F       PUSHL    SP                                          ;
                    00000000V       EF  9F 00091       PUSHAB   DEQ_MOUNT_LOCK                              ;
             00000000G  9F          03  FB 00097       CALLS    #3, @#SYS$CMKRNL                            ;
                        50   05FC   C6  D0 0009E       MOVL     LABLCK_STATUS+4, R0                         ; 1161
                                    0D  13 000A3       BEQL     5$                                          ;
                                    7E  7C 000A5       CLRQ     -(SP)                                       ; 1163
                                    7E  D4 000A7       CLRL     -(SP)                                       ;
                                    50  DD 000A9       PUSHL    R0                                          ;
             00000000G  00          04  FB 000AB       CALLS    #4, SYS$DEQ                                 ;
                        45          52  E9 000B2 5$:   BLBC     STATUS, 8$                                  ; 1165
                                    66  D6 000B5       INCL     DEVICE_INDEX                                ; 1172
```

```
                                                    F  5
VMOUNT                                              16-Sep-1984 01:00:56    VAX-11 Bliss-32 V4.0-742                     Page  16
V04-002                                             12-Sep-1984 11:14:53    DISK$VMSMASTER:[MOUNT.SRC]VMOUNT.B32;3         (3)

                    C8          53              54  F2 000B7 6$:    AOBLSS  R4, I, 4$                                      : 1151
                                                7E  D4 000BB        CLRL    -(SP)                                          : 1178
                     00000000G  00              01  FB 000BD        CALLS   #1, $DALLOC_DEVS$U
                    14      F9  A6              01  E1 000C4        BBC     #1, CLEANUP_FLAGS+1, 7$                        : 1184
                     00000000V  EF              00  FB 000C9        CALLS   #0, REBUILD_ENVELOPE                           : 1187
                                52              50  D0 000D0        MOVL    R0, STATUS
                                        FC      A6  DD 000D3        PUSHL   CHANNEL                                        : 1188
                     00000000G  00              01  FB 000D6        CALLS   #1, SYS$DASSGN
                                1A              52  E9 000DD 7$:    BLBC    STATUS, 8$                                     : 1194
                     00000000G  00              00  FB 000E0        CALLS   #0, ACTIVATE_JOURNAL                           : 1196
                                52              50  D0 000E7        MOVL    R0, STATUS
                                0D              52  E9 000EA        BLBC    STATUS, 8$                                     : 1202
                                        04      AC  DD 000ED        PUSHL   ITEM_LIST                                      : 1204
                     00000000V  EF              01  FB 000F0        CALLS   #1, MOUNT_CLUSTER
                                52              50  D0 000F7        MOVL    R0, STATUS
                                                7E  7C 000FA 8$:    CLRQ    -(SP)                                          : 1209
                                        10      AE  9F 000FC        PUSHAB  MOUNT_PRIVS
                                                7E  D4 000FF        CLRL    -(SP)
                                67              04  FB 00101        CALLS   #4, SYS$SETPRV
                                                7E  7C 00104        CLRQ    -(SP)                                          : 1210
                                        08      AE  9F 00106        PUSHAB  USER_PRIVS
                                                01  DD 00109        PUSHL   #1
                                67              04  FB 0010B        CALLS   #4, SYS$SETPRV
                                50              52  D0 0010E 9$:    MOVL    STATUS, R0                                     : 1212
                                                    04 00111        RET                                                    : 1214

; Routine Size:  274 bytes,    Routine Base:  $CODE$ + 0000
```

```
1215 1 ROUTINE VMOUNT_ENVELOPE =
1216 1
1217 1 !++
1218 1 !
1219 1 ! FUNCTIONAL DESCRIPTION:
1220 1 !
1221 1 !       This routine serves as the base call frame for all the EXEC
1222 1 !       mode code, and provides a convenient (and necessary) spot
1223 1 !       from which to intercept all EXEC mode conditions.
1224 1 !
1225 1 ! CALLING SEQUENCE:
1226 1 !
1227 1 !       This routine should be called in EXEC mode.
1228 1 !
1229 1 ! INPUT:
1230 1 !
1231 1 !       None.
1232 1 !
1233 1 ! OUTPUT:
1234 1 !
1235 1 !       None.
1236 1 !
1237 1 ! IMPLICIT INPUTS:
1238 1 !
1239 1 !       Current mode is EXEC, DEVICE_INDEX contains an integer value.
1240 1 !
1241 1 ! ROUTINE VALUE:
1242 1 !
1243 1 !       This routine returns the status returned by MOUNT_VOLUME.
1244 1 !
1245 1 !--
1246 1
1247 2 BEGIN
1248 2
1249 2 LOCAL
1250 2       STATUS;
1251 2
1252 2 !
1253 2 ! Establish the special EXEC mode condition handler.
1254 2 !
1255 2 ENABLE  INTERCEPT_SIGNAL;
1256 2
1257 2 !
1258 2 ! Attempt to mount the volume.
1259 2 !
1260 2 STATUS = MOUNT_VOLUME (.DEVICE_INDEX);
1261 2
1262 3 RETURN (.STATUS)
1263 3
1264 1 END;



                                 0000 00000 VMOUNT_ENVELOPE:
                                                 .WORD   Save nothing                              ; 1215
```

```
                     6D         000F    CF   DE 00002           MOVAL   1$, (FP)                          ; 1247
                            00000000'   EF   DD 00007           PUSHL   DEVICE_INDEX                      ; 1260
         00000000V   EF                 01   FB 0000D           CALLS   #1, MOUNT_VOLUME                  ;
                                             04 00014           RET                                       ; 1264
                                           0000 00015 1$:       .WORD   Save nothing                      ; 1247
                                        7E   D4 00017           CLRL    -(SP)                             ;
                                        5E   DD 00019           PUSHL   SP                                ;
                     7E           04    AC   7D 0001B           MOVQ    4(AP), -(SP)                      ;
         00000000V   EF                 03   FB 0001F           CALLS   #3, INTERCEPT_SIGNAL              ;
                                             04 00026           RET                                       ;

; Routine Size:  39 bytes,    Routine Base:  $CODE$ + 0112
```

```
  1 ROUTINE REBUILD_ENVELOPE =
  1
  1 !++
  1 !
  1 ! FUNCTIONAL DESCRIPTION:
  1 !
  1 !       This routine serves as the base call frame for all the EXEC
  1 !       mode code, and provides a convenient (and necessary) spot
  1 !       from which to intercept all EXEC mode conditions.
  1 !
  1 ! CALLING SEQUENCE:
  1 !
  1 !       This routine should be called in EXEC mode.
  1 !
  1 ! INPUT:
  1 !
  1 !       None.
  1 !
  1 ! OUTPUT:
  1 !
  1 !       None.
  1 !
  1 ! IMPLICIT INPUTS:
  1 !
  1 !       Current mode is EXEC, DEVICE_INDEX contains an integer value.
  1 !
  1 ! ROUTINE VALUE:
  1 !
  1 !       This routine returns the status returned by MOUNT_VOLUME.
  1 !
  1 !--
  1
  2 BEGIN
  2
  2 EXTERNAL ROUTINE
  2         STAND_ALONE_REBUILD;            ! Rebuild quota file and bitmaps (ODS2)
  2
  2 LOCAL
  2         STATUS;
  2
  2 !
  2 ! Establish the special EXEC mode condition handler.
  2 !
  2 ENABLE  INTERCEPT_SIGNAL;
  2
  2     !
  2     ! Rebuild the volume.
  2     !
  2     ERR_MESSAGE (MOUN$_REBUILD);
P 2     STATUS = $ASSIGN (DEVNAM = PHYS_NAME[0],
  2                       CHAN   = CHANNEL);
  2     IF NOT .STATUS THEN ERR_EXIT (.STATUS);
  2     STAND_ALONE_REBUILD (.CHANNEL);
  2
  2 RETURN 1
  2
  1 END;
```

```
                                                  .EXTRN  STAND_ALONE_REBUILD
                                                  .EXTRN  SYS$ASSIGN

                                 0004 00000 REBUILD_ENVELOPE:
                                                  .WORD   Save R2                                  ; 1265
                   52 00000000'  EF  9E 00002     MOVAB   CHANNEL, R2                              ;
                   6D     0033   CF  DE 00009     MOVAL   2$, (FP)                                 ; 1297
                      0072A01B   8F  DD 0000E     PUSHL   #7512091                                 ; 1313
        00000000G  00            01  FB 00014     CALLS   #1, LIB$SIGNAL                           ;
                                 7E  7C 0001B     CLRQ    -(SP)                                    ; 1315
                                 52  DD 0001D     PUSHL   R2                                       ;
                          14     A2  9F 0001F     PUSHAB  PHYS_NAME                                ;
        00000000G  00            04  FB 00022     CALLS   #4, SYS$ASSIGN                           ;
                   09            50  E8 00029     BLBS    STATUS, 1$                               ; 1316
                                 50  DD 0002C     PUSHL   STATUS                                   ;
        00000000G  00            01  FB 0002E     CALLS   #1, LIB$STOP                             ;
                                 62  DD 00035 1$: PUSHL   CHANNEL                                  ; 1317
            0000G  CF            01  FB 00037     CALLS   #1, STAND_ALONE_REBUILD                  ;
                   50            01  D0 0003C     MOVL    #1, R0                                   ; 1319
                                     04 0003F     RET                                              ; 1321
                                 0000 00040 2$:   .WORD   Save nothing                             ; 1297
                                 7E  D4 00042     CLRL    -(SP)                                    ;
                                 5E  DD 00044     PUSHL   SP                                       ;
                   7E     04     AC  7D 00046     MOVQ    4(AP), -(SP)                             ;
        00000000V  EF            03  FB 0004A     CALLS   #3, INTERCEPT_SIGNAL                     ;
                                     04 00051     RET                                              ;

; Routine Size:  82 bytes,    Routine Base:  $CODE$ + 0139
```

```
1 ROUTINE INTERCEPT_SIGNAL (SIGNAL, MECHANISM) =
1
1 !++
1 ! Functional Description:
1 !
1 !       This routine is a conditon handler whose sole
1 !       reason for existence is to force the primary
1 !       conditon code's facility-code to that of the
1 !       MOUNT facility.
1 !
1 ! Input:
1 !
1 !       SIGNAL    = Address of the signal array
1 !       MECHANISM = Address of the mechanism array
1 !
1 ! Output:
1 !
1 !       The condition facility code is equal to MOUN$_FACILITY
1 !--
1
2 BEGIN                                             ! Start of INTERCEPT_SIGNAL
2
2 MAP
2
2       SIGNAL          : REF BBLOCK,               ! Signal array
2       MECHANISM       : REF BBLOCK;               ! Mechanism array
2
2 EXTERNAL
2
2       MOUNT_OPTIONS   : ADDRESSING_MODE (GENERAL)
2                         BITVECTOR VOLATILE,       ! parser option flags
2       USER_STATUS     : VECTOR,                   ! Status return of some routines
2       VOLINV_COUNT    : ADDRESSING_MODE (GENERAL);
2                                                   ! VOLINV retry counter
2
2 EXTERNAL LITERAL
2       VOLINV_LIMIT;                               ! VOLINV retry limit
2
2
2 IF .SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND
2 THEN
3     BEGIN
3     !
3     ! Make the facility code MOUN$_FCILITY.
3     !
3     IF .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] EQL 0
3     OR .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] EQL INIT$_FACILITY
3     THEN
3         BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] = MOUN$_FACILITY;
3
3     IF .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_MSG_NO] EQL 0
3     THEN
3         BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_MSG_NO] = .USER_STATUS [0] ^ (-$BITPOSITION (STS$V_MSG_NO));
3
3     !
3     ! If the caller requested it, print the message text associated with the
3     ! message.  Also make sure that the particular error is not covered by
```

```
3         ! operator assisted mount.  If it is, do not print the message.
3         !
5         IF (.MOUNT_OPTIONS [OPT_MESSAGE] AND NOT (.MOUNT_OPTIONS [OPT_ASSIST]
6         AND (SELECTONEU (.SIGNAL [CHF$L_SIG_NAME] AND STS$M_MSG_NO) OF
6                 SET
6                 [SS$_DEVALLOC       AND STS$M_MSG_NO] : 1;
6                 [SS$_MEDOFL         AND STS$M_MSG_NO] : 1;
6                 [SS$_VOLINV         AND STS$M_MSG_NO] : 1;
6                 [SS$_NODEVAVL       AND STS$M_MSG_NO] : 1;
6                 [SS$_NOSUCHDEV      AND STS$M_MSG_NO] : 1;
6                 [SS$_INCVOLLABEL    AND STS$M_MSG_NO] : 1;
6                 [OTHERWISE]                         : 0;
4                 TES)))
4
4         !
4         ! If mounting with /NOASSIST and we are in VOLINV retry, supress outputting
4         ! the VOLINV error message unless this is the last retry attempt.
4         !
5         AND (.MOUNT_OPTIONS [OPT_MESSAGE] AND NOT (NOT .MOUNT_OPTIONS [OPT_ASSIST]
6         AND (SELECTONEU (.SIGNAL [CHF$L_SIG_NAME] AND STS$M_MSG_NO) OF
6                 SET
6                 [SS$_VOLINV AND STS$M_MSG_NO] : IF .VOLINV_COUNT LSS VOLINV_LIMIT-1
6                                                 THEN
6                                                     1
6                                                 ELSE
6                                                     0;
6
6                 [OTHERWISE]                     : 0;
4                 TES)))
3         THEN
4             BEGIN
4             SIGNAL [CHF$L_SIG_ARGS] = .SIGNAL [CHF$L_SIG_ARGS] - 2;
4             $PUTMSG (MSGVEC = SIGNAL [CHF$L_SIG_ARGS], ACTRTN=0, FACNAM=0);
4             SIGNAL [CHF$L_SIG_ARGS] = .SIGNAL [CHF$L_SIG_ARGS] + 2;
4             BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_INHIB_MSG] = 1;
3             END;
3
3         !
3         ! If the condition severity code is SEVERE or ERROR, then unwind the
3         ! stack back to the caller of the frame that established this handler.
3         ! Return the condition code in R0.
3         !
3         IF .BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_SEVERE
3         OR .BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_ERROR
3         THEN
4             BEGIN
4             MECHANISM [CHF$L_MCH_SAVR0] = .SIGNAL [CHF$L_SIG_NAME];
4             $UNWIND ();
3             END;
2         END;
2
2     !
2     ! Attempt to continue the operation.
2     !
2     RETURN (SS$_CONTINUE);
2
1     END;                                            ! End of INTERCEPT_SIGNAL
```

```
                                                                .EXTRN  MOUNT_OPTIONS, VOLINV_COUNT
                                                                .EXTRN  VOLINV_LIMIT, SYS$PUTMSG
                                                                .EXTRN  SYS$UNWIND

                                                  001C 00000 INTERCEPT_SIGNAL:
                                                                .WORD   Save R2,R3,R4                                ; 1322
                                   54 00000000G   00   9E 00002 MOVAB   MOUNT_OPTIONS+6, R4
                                   52        04   AC   D0 00009 MOVL    SIGNAL, R2                                   ; 1361
                                   53        04   A2   9E 0000D MOVAB   4(R2), R3
                        00000920   8F             63   D1 00011 CMPL    (R3), #2336
                                                  03   12 00018 BNEQ    1$
                                                00D6   31 0001A BRW     9$
                            0FFF   8F        02   A3   B3 0001D 1$:  BITW    2(R3), #4095                          ; 1367
                                                  0C   13 00023 BEQL    2$
00000075   8F        02   A3       0C             00   ED 00025 CMPZV   #0, #12, 2(R3), #117                         ; 1368
                                                  0A   12 0002F BNEQ    3$
      02   A3             0C             00 00000072   8F   F0 00031 2$:  INSV    #114, #0, #12, 2(R3)              ; 1370
                            FFF8   8F             63   B3 0003B 3$:  BITW    (R3), #65528                          ; 1372
                                                  0C   12 00040 BNEQ    4$
                      50   0000G   CF        FD   8F   78 00042 ASHL    #-3, USER_STATUS, R0                         ; 1374
                63   0D            03             50   F0 00049 INSV    R0, #3, #13, (R3)
                51   64            01             03   EF 0004E 4$:  EXTZV   #3, #1, MOUNT_OPTIONS+6, R1           ; 1381
                     7E                           51   E9 00053 BLBC    R1, 7$
                     3E            64             02   E1 00056 BBC     #2, MOUNT_OPTIONS+6, 5$
                     50            63 FFFF0007    8F   CB 0005A BICL3   #-65529, (R3), R0                            ; 1382
                        00000840   8F             50   D1 00062 CMPL    R0, #2112                                    ; 1384
                                                  69   13 00069 BEQL    7$
                        000001A0   8F             50   D1 0006B CMPL    R0, #416                                     ; 1385
                                                  60   13 00072 BEQL    7$
                        00000250   8F             50   D1 00074 CMPL    R0, #592                                     ; 1386
                                                  57   13 0007B BEQL    7$
                        000009B0   8F             50   D1 0007D CMPL    R0, #2480                                    ; 1387
                                                  4E   13 00084 BEQL    7$
                        00000908   8F             50   D1 00086 CMPL    R0, #2312                                    ; 1388
                                                  45   13 0008D BEQL    7$
                        00000108   8F             50   D1 0008F CMPL    R0, #264                                     ; 1389
                                                  3C   13 00096 BEQL    7$
                                   39             51   E9 00098 5$:  BLBC    R1, 7$                                ; 1397
                     1E            64             02   E0 0009B BBS     #2, MOUNT_OPTIONS+6, 6$
                     50            63 FFFF0007    8F   CB 0009F BICL3   #-65529, (R3), R0                            ; 1398
                        00000250   8F             50   D1 000A7 CMPL    R0, #592                                     ; 1400
                                                  0D   12 000AE BNEQ    6$
                        00000000G  8F 00000000G   00   D1 000B0 CMPL    VOLINV_COUNT, #VOLINV_LIMIT-1
                                                  17   19 000BB BLSS    7$
                                   62             02   C2 000BD 6$:  SUBL2   #2, (R2)                              ; 1410
                                                  7E   7C 000C0 CLRQ    -(SP)                                        ; 1411
                                                  7E   D4 000C2 CLRL    -(SP)
                                                  52   DD 000C4 PUSHL   R2
                        00000000G  00             04   FB 000C6 CALLS   #4, SYS$PUTMSG
                                                  62   02 C0 000CD ADDL2   #2, (R2)                              ; 1412
                              03   A3             10   88 000D0 BISB2   #16, 3(R3)                               ; 1413
      04             63            03             00   ED 000D4 7$:  CMPZV   #0, #3, (R3), #4                      ; 1421
                                                  07   13 000D9 BEQL    8$
      02             63            03             00   ED 000DB CMPZV   #0, #3, (R3), #2                             ; 1422
```

```
                                 11   12 000E0           BNEQ    9$                                             ;
                  50        08   AC   D0 000E2 8$:       MOVL    MECHANISM, R0                                  ; 1425
             0C   A0             63   D0 000E6           MOVL    (R3), 12(R0)                                   ;
                                 7E   7C 000EA           CLRQ    -(SP)                                          ; 1426
      00000000G   00             02   FB 000EC           CALLS   #2, SYS$UNWIND                                 ;
                  50             01   D0 000F3 9$:       MOVL    #1, R0                                         ; 1433
                                      04 000F6           RET                                                    ; 1435

; Routine Size:  247 bytes,    Routine Base:  $CODE$ + 018B
```

```
ROUTINE MOUNT_VOLUME (J) =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine will mount a single disk or tape volume.
!
! CALLING SEQUENCE:
!
!       mount_volume (.j)
!
! INPUT:
!
!       J : Index into device list.
!
! OUTPUT:
!
!       None.
!
! IMPLICIT INPUT:
!
!       Mount data base
!
! IMPLICIT OUTPUT:
!
!       None.
!
! ROUTINE VALUE:
!
!       Assorted status codes.
!
! SIDE EFFECTS:
!
!       Volume mounted, device data base updated.
!
!--

BEGIN

LOCAL
        DEVICE_ITMLST1  : BBLOCK [(1 * 12) + 4] INITIAL
                        !
                        ! item: allocation class plus device name
                        !
                        (WORD (NAMEBUF_LEN-4),
                         WORD (DVI$_ALLDEVNAM),
                         LONG (ALLDEVNAM_BUF+4),
                         LONG (ALLDEVNAM_DESC),
                        !
                        ! end of list
                        !
                         LONG (0)),
        P,                                      ! string scan pointer
        STATUS;                                 ! system service status

EXTERNAL
        DEV_CTX         : BBLOCK FIELD (DC),    ! device value block context fields
        MOUNT_FAILED    : ADDRESSING_MODE (GENERAL) LONG VOLATILE,      ! State of the current mount
        MOUNT_OPTIONS   : ADDRESSING_MODE (GENERAL) BITVECTOR VOLATILE, ! parser option flags
        DEVICE_COUNT    : ADDRESSING_MODE (GENERAL), ! number of devices specified
        LABEL_COUNT     : ADDRESSING_MODE (GENERAL), ! number of volume labels specified
        DEVICE_STRING   : ADDRESSING_MODE (GENERAL) VECTOR VOLATILE,    ! device name string descriptor
        LABEL_STRING    : ADDRESSING_MODE (GENERAL) VECTOR VOLATILE;    ! volume label string descriptor

EXTERNAL ROUTINE

        SEARCH_VOL,                             ! search I/O database for volume
        TRAN_LOGNAME,                           ! translate logical name
        READ_VOLLABEL,                          ! read magtape volume header label
        READ_HOMEBLOCK,                         ! read disk home block
        MOUNT_TAPE,                             ! mount magtape
        MOUNT_DISK1,                            ! mount level 1 disk
        MOUNT_DISK2,                            ! mount level 1 disk
        GET_DEVICE_CONTEXT;                     ! get device lock value block context

BIND
        OPTIONS         = MOUNT_OPTIONS : VECTOR VOLATILE;


ENABLE MAIN_HANDLER;                            ! Enable the MOUNT condition handler

!
! Reset the mount options bit mask.
!

OPTIONS[0] = .OPTIONS[0] AND NOT RESET_OPTIONS1;
OPTIONS[1] = .OPTIONS[1] AND NOT RESET_OPTIONS2;
MOUNT_FAILED  = 1;

BEGIN
!
! rebind things to make life easier ( so we see them as their
! real logical units)
!
MAP
    DEVICE_STRING   : BBLOCKVECTOR [ DEVMAX, 8 ],
    NAME_BUFFER     : BBLOCKVECTOR [ DEVMAX, NAMEBUF_LEN ],
    PHYS_NAME       : BBLOCKVECTOR [ DEVMAX, 8 ];

!
! Start of buffer
!
MACRO   STADR   = 0,0,0,0%;

!
! Define descriptor vector displacements
!
MACRO   LEN     = 0,0,32,0%;
MACRO   ADDR    = 4,0,32,0%;
MACRO   ILEN    = 8,0,32,0%;     ! Item list returned length position.


!
! If the device is being mounted /SHARE, /GROUP, or /SYSTEM, search the
```

```
  3 ! device database for a  matching volume label.  To properly  serialize
  3 ! simultaneous shared mounts, take out the label lock in EX mode.  This
  3 ! label lock will be released in routine  SYS$VMOUNT when everything is
  3 ! done.
  3 !
  3 STATUS = 0;
  3 IF NOT .MOUNT_OPTIONS [OPT_NOSHARE]
  3 THEN
  4     BEGIN
  4
  4     !
  4     ! The label lock has the form  MOU$-csid-vollabel. The csid part makes
  4     ! the label lock node-specific, which is necessary to avoid potential
  4     ! deadlocks in a cluster.  If the node is not in a cluster, the csid
  4     ! field is set to zero.
  4     !
  4     LOCAL
  4         CSID            : LONG INITIAL (0),       ! Initialize to zero
  4         SYI_ITMLST      : BLOCK  [(1*12)+4, BYTE] INITIAL
  4                         ( WORD (4),               ! Return buffer length
  4                           WORD (SYI$_NODE_CSID),  ! CSID item code
  4                           LONG (CSID),            ! Reuturn buffer address
  4                           LONG (0),
  4                           LONG (0));
  4
P 4     $GETSYIW ( EFN = MOUNT_EFN,                   ! Get CSID of the local node
  4                ITMLST = SYI_ITMLST );
  4
  4     !
  4     ! Set up the label lock resource name and descriptor
  4     !
  4     LABLCKNAM_DESC [0] = .LABEL_STRING [.J*2] + 8; ! 'MOU$' prefix +  CSID
  4     LABLCKNAM_BUF + 4  = .CSID;                    ! Merge in CSID
  4     CH$COPY  (  .LABEL_STRING [.J*2],              ! Length of input string
  4                 .LABEL_STRING [.J*2+1],            ! Address of label string buffer
  4                 0,                                 !
  4                 .LABEL_STRING [.J*2],              ! Length of output string
  4                 LABLCKNAM_BUF + 8  );              ! Address of output buffer
  4
P 4     $ENQW  (  LKMODE = LCK$K_EXMODE,               ! Take out the label lock
P 4                 LKSB = LABLCK_STATUS,
P 4                FLAGS = LCK$M_SYSTEM,
P 4               RESNAM = LABLCKNAM_DESC,
P 4                  EFN = MOUNT_EFN,
  4               ACMODE = PSL$C_EXEC  );
  4
  4     STATUS = KERNEL_CALL (SEARCH_VOL, LABEL_STRING[.J*2]);
  3     END;
  3
  3 !
  3 ! The SEARCH_VOL routine will only return success if this is a /SHARE
  3 ! mount and a matching volume label is found.  It will signal an error
  3 ! if this is a /SYSTEM or /GROUP mount and a duplicate volume label is
  3 ! already in use.
  3 !
  3 IF .STATUS
  3     !
```

```
        ! A successful /SHARE mount.  Just print the message here; we rejoin
        ! the "volume not found" path much later in the routine.
        !
        THEN ERR_MESSAGE (MOUN$_MOUNTED, 3, .LABEL_STRING[.J*2],
                          .LABEL_STRING[.J*2+1], PHYS_NAME[.J, LEN])

        ELSE
            IF .STATUS GTRU 7
                THEN ERR_EXIT (.STATUS)
                ELSE

    !
    ! Volume not found: either not there or this is a /NOSHARE mount.
    ! We must go through the mechanics of mounting the device.
    !
    BEGIN

    !
    ! The following block of code should not be re-executed if this routine
    ! is called a second time by operator-assisted mount code.
    !
    IF NOT .DEV_ACQUIRED[.J]
    THEN
        BEGIN

        LOCAL
            STSBLK  : VECTOR [2];

        !
        ! Call the SEARCH_DEVICE routine to search for a mountable device,
        ! allocate it, and set up the physical device name and descriptor
        ! in mount database.  Note that if the device is available cluster-
        ! wide, SEARCH_DEVICE will take out an EX mode lock for a private
        ! mount, or a PW mode lock for a shared mount.
        !
        IF NOT .DEV_ALLOCATED [.J]
        THEN
            BEGIN
            STATUS = KERNEL_CALL (SEARCH_DEVICE, .J);

            !
            ! If the device does not exists, disable operator assist before
            ! exiting with the error status.
            !
            ! Otherwise, indicate that this device has been allocated.
            ! If the device was not previously allocated, indicate such.
            ! If the mount fails, these devices must be deallocated.
            !
            IF NOT .STATUS
            THEN
                BEGIN
                IF ((.STATUS AND STS$M_MSG_NO) EQL (SS$_NOSUCHDEV AND STS$M_MSG_NO))
                OR ((.STATUS AND STS$M_MSG_NO) EQL (SS$_IVDEVNAM  AND STS$M_MSG_NO))
                THEN
                    MOUNT_OPTIONS [OPT_ASSIST] = 0;
                ERR_EXIT (.STATUS);
                END;
    6             IF .STATUS NEQ SS$_DEVALRALLOC
    6             THEN
    7                 BEGIN
    7                 CLEANUP_FLAGS [CLF_DEALLOCATE] = 1;
    7                 CLEANUP_ALLOC [.J] = 1;
    6                 END;
    6             DEV_ALLOCATED [.J] = 1;
    5             END;                                        ! End device search/allocation block
    5
    5         !
    5         ! Set the PHYS_NAME high-water mark.
    5         !
    5         PHYS_COUNT = .J + 1;
    5
    5         END           ! End of code that shouldn't be executed more than once
    5                       ! per device.
    5
    4 ELSE
    4
    5         BEGIN
    5
    5         ! Take out a lock on the allocation class device name.  This will
    5         ! interlock all mounts of this device.
    5         !
P   5         STATUS = $ENQW (LKMODE = LCK$K_EXMODE,
P   5                         LKSB = LOCK_STATUS,
P   5                         FLAGS = LCK$M_SYSTEM,
P   5                         RESNAM = ALLDEVNAM_DESC,
P   5                         EFN = MOUNT_EFN,
    5                         ACMODE = PSL$C_EXEC);
    5         IF NOT .STATUS THEN ERR_EXIT (.STATUS);
    5
    4         END;
    4
    4 !
    4 ! The remainder of the code is executed each time this routine is called by
    4 ! ASSIST if an operator-assisted mount is required.
    4 !
    4 DEV_ACQUIRED[.J] = 1;
    4
    4 !
    4 ! Get a channel to it.  If this is a cluster accessible device,
    4 ! a device lock will be taken out by this node on the device.
    4 !
    4
P   4 STATUS = $ASSIGN (DEVNAM = PHYS_NAME[.J,LEN],
    4                   CHAN = CHANNEL);
    4 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
    4
    4 !
    4 ! Get the device characteristics and do device type validation: Make sure
    4 ! the device is mountable at all, and check that the mount qualifiers are
    4 ! consistent with the device type. A mismatch between primary and secondary
    4 ! device characteristics indicates a spooled device or something else strange.
    4 ! Reject such.
    4 !
    4
```

```
4 $GETCHN (CHAN = .CHANNEL, PRIBUF = DEVCHAR_DESC, SCDBUF = DEVCHAR_DESC2);
4
4 IF CH$NEQ (DIB$K_LENGTH, DEVICE_CHAR, DIB$K_LENGTH, DEVICE_CHAR2, 0)
4     OR NOT .DEVICE_CHAR[DEV$V_FOD]
4     THEN ERR_EXIT (SS$_NOTFILEDEV);
4
4 IF NOT .DEVICE_CHAR[DEV$V_AVL] THEN ERR_EXIT (SS$_DEVOFFLINE);
4
4 IF .DEVICE_CHAR[DEV$V_MNT] THEN ERR_EXIT (SS$_DEVMOUNT);
4
4 CLEANUP_FLAGS[CLF_CLEARVALID] = 1; ! device is now known not mounted
4
4 !
4 ! Some things to be tested on the 1st only and then stored anyway
4 !
4 IF .J EQL 0
4     THEN
5         BEGIN
5         !
5         ! is it a tape or disk mount
5         !
5         STORED_CONTEXT [TAPE_MOUNT] = .DEVICE_CHAR [DEV$V_SQD];
5
5         ! we need only to test if we are going to overide something
5         ! once ( and then just save it )
5         !
6         IF (   .MOUNT_OPTIONS[OPT_FOREIGN] OR .MOUNT_OPTIONS[OPT_NOLABEL]
6             OR .MOUNT_OPTIONS[OPT_OVR_ACC] OR .MOUNT_OPTIONS[OPT_PROTECTION]
6             OR .MOUNT_OPTIONS[OPT_OVR_EXP] OR .MOUNT_OPTIONS[OPT_USER_UIC]
6             OR .MOUNT_OPTIONS[OPT_NOQUOTA] OR .MOUNT_OPTIONS[OPT_OWNER_UIC]
6             OR .MOUNT_OPTIONS[OPT_OVR_LOCK] OR .MOUNT_OPTIONS[OPT_OVR_VOLO])
5                 THEN STORED_CONTEXT [OVERIDE_SOMETHING] = 1
5                 ELSE STORED_CONTEXT [OVERIDE_SOMETHING] = 0;
5
5         ! device number must match label number for disk
5         !
5         IF (NOT .STORED_CONTEXT [TAPE_MOUNT]) AND
6            (.DEVICE_COUNT NEQ .LABEL_COUNT) AND (.LABEL_COUNT NEQ 0)
5                 THEN ERR_EXIT (MOUN$_DEVCOUNT);
5
4         END;     ! End of block to be executed for first device only.
4
4 !
4 ! test legal options for device type
4 !
4 IF
5     BEGIN
5     IF .DEVICE_CHAR[DEV$V_SQD]
5         THEN
6             ((.OPTIONS[0] AND NOT TAPE_OPTIONS1) NEQ 0
6             OR (.OPTIONS[1] AND NOT TAPE_OPTIONS2) NEQ 0)
5         ELSE
6             ((.OPTIONS[0] AND NOT DISK_OPTIONS1) NEQ 0
6             OR (.OPTIONS[1] AND NOT DISK_OPTIONS2) NEQ 0)
5     END
4         THEN ERR_EXIT (MOUN$_ILLOPT);
4
```

```
  4 ! device types must be consistent
  4 !    tapes with tapes or disks with disks
  4 !
  5 IF (NOT .DEVICE_CHAR [DEV$V_SQD] AND .STORED_CONTEXT [TAPE_MOUNT])
  4     OR
  5    (.DEVICE_CHAR [DEV$V_SQD] AND NOT .STORED_CONTEXT [TAPE_MOUNT])
  4         THEN ERR_EXIT (MOUN$_INCONSDEV);
  4
  4 !
  4 ! Now attempt to read the home block or volume header label, as appropriate
  4 ! for the device type.
  4 !
  4
  4 IF .DEVICE_CHAR[DEV$V_SQD]
  4     THEN
  4         STATUS = READ_VOLLABEL (LABEL_STRING[.J*2])
  4     ELSE
  4         STATUS = READ_HOMEBLOCK (LABEL_STRING[.J*2], NOT .MOUNT_OPTIONS[OPT_FOREIGN]);
  4
  4 !
  4 ! Now check the status of the volume against the various mount options. Note,
  4 ! in particular, whether the user is attempting to override volume protection.
  4 !
  4
  4 MOUNT_OPTIONS[OPT_IS_FILES11] = 1;        ! assume volume is Files-11
  4 IF NOT .STATUS
  5     THEN BEGIN
  5     IF .STATUS EQL SS$_NOHOMEBLK OR .STATUS EQL SS$_NOTLABELMT
  5                                     ! if home block is not found
  6         THEN BEGIN
  6         MOUNT_OPTIONS[OPT_IS_FILES11] = 0;
  7         IF NOT  (  .MOUNT_OPTIONS[OPT_FOREIGN]
  7                 OR .MOUNT_OPTIONS[OPT_NOLABEL])
  6         THEN
  6             IF .DEVICE_CHAR[DEV$V_SQD]
  6             THEN ERR_EXIT (.STATUS)
P 6             ELSE ERR_EXIT (.STATUS, 0, MOUN$_VOLIDENT, 6,
P 6                             HM2$S_VOLNAME, HOME_BLOCK[HM2$T_VOLNAME],
P 6                             HM2$S_OWNERNAME, HOME_BLOCK[HM2$T_OWNERNAME],
  6                             HM2$S_FORMAT, HOME_BLOCK[HM2$T_FORMAT]);
  6         END
  6
  5         ELSE IF .STATUS EQL SS$_INCVOLLABEL ! if volume label mismatch
  5             THEN
  6                 BEGIN
  6                 IF .MOUNT_OPTIONS[OPT_LABEL]
  6                 AND NOT .MOUNT_OPTIONS[OPT_FOREIGN]
  6                 AND NOT .MOUNT_OPTIONS[OPT_OVR_ID]
  6                 THEN
  6                     IF .DEVICE_CHAR[DEV$V_SQD]
  6                     THEN ERR_EXIT (.STATUS)
P 6                     ELSE ERR_EXIT (.STATUS, 0, MOUN$_VOLIDENT, 6,
P 6                                     HM2$S_VOLNAME, HOME_BLOCK[HM2$T_VOLNAME],
P 6                                     HM2$S_OWNERNAME, HOME_BLOCK[HM2$T_OWNERNAME],
  6                                     HM2$S_FORMAT, HOME_BLOCK[HM2$T_FORMAT]);
  6                 END
  6
```

```
; 1179        1835  5              ELSE
; 1180        1836  6                   BEGIN
; 1181        1837  6                   MOUNT_OPTIONS[OPT_IS_FILES11] = 0; ! Clean up option flag.
; 1182        1838  6                   ERR_EXIT (.STATUS);
; 1183        1839  5                   END;
; 1184        1840  4      END;
; 1185        1841  4
; 1186        1842  4  !
; 1187        1843  4  ! are overiding something with a files-11 mount
; 1188        1844  4  !
; 1189        1845  4 IF .MOUNT_OPTIONS[OPT_IS_FILES11] AND .STORED_CONTEXT [OVERIDE_SOMETHING]
; 1190        1846  4         THEN MOUNT_OPTIONS[OPT_OVR_PRO] = 1;
; 1191        1847  4
; 1192        1848  4  !
; 1193        1849  4  ! Call the device specific routine that actually does the mount.
; 1194        1850  4  !
; 1195        1851  4
; 1196        1852  4 IF .DEVICE_CHAR[DEV$V_SQD]
; 1197        1853  4         THEN
; 1198        1854  5              BEGIN
; 1199        1855  5              MOUNT_TAPE ();
; 1200        1856  5              KERNEL_CALL (XFER_DEV_OWNER, .CHANNEL);
; 1201        1857  5              END
; 1202        1858  4         ELSE
; 1203        1859  5              BEGIN
; 1204        1860  5
; 1205        1861  5  ! Get the device context, if it exists.  This is necessary to
; 1206        1862  5  ! make sure that mounts of the same device from different nodes
; 1207        1863  5  ! are consistent.
; 1208        1864  5  !
; 1209        1865  5
; 1210        1866  6              IF NOT (STATUS = KERNEL_CALL (GET_DEVICE_CONTEXT))
; 1211        1867  5              THEN
; 1212        1868  5                   ERR_EXIT (.STATUS);
; 1213        1869  5
; 1214        1870  5              IF .MOUNT_OPTIONS[OPT_IS_FILES11B]
; 1215        1871  5              THEN
; 1216        1872  5                   MOUNT_DISK2 ()
; 1217        1873  5              ELSE
; 1218        1874  5                   MOUNT_DISK1 ();
; 1219        1875  5  !
; 1220        1876  5  ! If we are mounting a shared volume on an allocated device, deallocate the
; 1221        1877  5  ! device now.  We delayed the deallocation until now so that if the mount
; 1222        1878  5  ! failed, the device remained allocated.
; 1223        1879  5  !
; 1224        1880  5              IF NOT .MOUNT_OPTIONS [OPT_NOSHARE]
; 1225        1881  6                   THEN KERNEL_CALL (DALLOC_SHR_DEV, .CHANNEL)
; 1226        1882  5                   ELSE KERNEL_CALL (XFER_DEV_OWNER, .CHANNEL);
; 1227        1883  5
; 1228        1884  4              END;
; 1229        1885  4
; 1230        1886  4  !
; 1231        1887  4  ! Deassign the channel.
; 1232        1888  4  !
; 1233        1889  4 $DASSGN (CHAN = .CHANNEL);
; 1234        1890  4
; 1235        1891  3 END;                                  ! shared mount path rejoins us here
; 1236      1892  2 END;                                  ! end of rebind block
; 1237      1893  2
; 1238      1894  2 !
; 1239      1895  2 ! Clean out status values for the next time around the loop.
; 1240      1896  2 !
; 1241      1897  2
; 1242      1898  2 CLEANUP_FLAGS = .CLEANUP_FLAGS AND (1^CLF_REBUILD OR 1^CLF_REBUILDQUO);
; 1243      1899  2 CHANNEL = 0;
; 1244      1900  2 REAL_MVL = 0;
; 1245      1901  2 REAL_RVT = 0;
; 1246      1902  2 REAL_VCB = 0;
; 1247      1903  2 REAL_FCB = 0;
; 1248      1904  2 REAL_WCB = 0;
; 1249      1905  2 REAL_AQB = 0;
; 1250      1906  2 MTL_ENTRY = 0;
; 1251      1907  2 SMTC_ENTRY = 0;
; 1252      1908  2 OPTIONS[0] = .OPTIONS[0] AND NOT RESET_OPTIONS1;
; 1253      1909  2 OPTIONS[1] = .OPTIONS[1] AND NOT RESET_OPTIONS2;
; 1254      1910  2 MOUNT_FAILED = 0;                     ! Indicate that the mount worked.
; 1255      1911  3 RETURN (SS$_NORMAL)                   ! Return success status
; 1256      1912  1 END;                                  ! end of MOUNT_VOLUME


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                           001C  00000 P.AAA:           .WORD   28                                              ;
                           00EC  00002                  .WORD   236                                             ;
                      00000000'  00004                  .ADDRESS ALLDEVNAM_BUF+4                                ;
                      00000000'  00008                  .ADDRESS ALLDEVNAM_DESC                                 ;
                       00000000  0000C                  .LONG   0                                               ;
                           0004  00010 P.AAB:           .WORD   4                                               ;
                           10D0  00012                  .WORD   4304                                            ;
                       00000000  00014                  .LONG   0                                               ;
                       00000000  00018                  .LONG   0                                               ;
                       00000000  0001C                  .LONG   0                                               ;

                                                        .EXTRN  DEV_CTX, MOUNT_FAILED
                                                        .EXTRN  LABEL_COUNT, DEVICE_STRING
                                                        .EXTRN  LABEL_STRING, SEARCH_VOL
                                                        .EXTRN  TRAN_LOGNAME, READ_VOLLABEL
                                                        .EXTRN  READ_HOMEBLOCK, MOUNT_TAPE
                                                        .EXTRN  MOUNT_DISK1, MOUNT_DISK2
                                                        .EXTRN  GET_DEVICE_CONTEXT
                                                        .EXTRN  SYS$GETSYIQ, SYS$ENQW
                                                        .EXTRN  SYS$GETCHN

                                                        .PSECT  $CODE$,NOWRT,2

                              0FFC 00000 MOUNT_VOLUME:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11            ; 1436
            5B 00000000G    00    9E 00002              MOVAB   LABEL_STRING, R11                               ;
            5A 00000000G    00    9E 00009              MOVAB   LIB$STOP, R10                                   ;
            59 00000000G    00    9E 00010              MOVAB   MOUNT_OPTIONS, R9                               ;
            58 00000000'    EF    9E 00017              MOVAB   DEVICE_CHAR, R8                                 ;
            5E              24    C2 0001E              SUBL2   #36, SP                                         ;
   14   AE 00000000'  EF    10    28 00021              MOVC3   #16, P.AAA, DEVICE_ITMLST1                      ; 1487
```

```
                     6D      03BE   CF  DE 0002A       MOVAL   45$, (FP)                               ;
                 04  A9      0207   8F  AA 0002F       BICW2   #519, OPTIONS+4                         ; 1522
          00000000G  00             01  D0 00035       MOVL    #1, MOUNT_FAILED                        ; 1523
                                    57  D4 0003C       CLRL    STATUS                                  ; 1555
      70             69             04  E0 0003E       BBS     #4, MOUNT_OPTIONS, 1$                   ; 1556
                                    6E  D4 00042       CLRL    CSID                                    ; 1558
  04  AE  00000000'  EF             10  28 00044       MOVC3   #16, P.AAB, SYI_ITMLST                  ; 1573
                 08  AE             6E  9E 0004D       MOVAB   CSID, SYI_ITMLST+4                      ; 1558
                                    7E  7C 00051       CLRQ    -(SP)                                   ; 1576
                                    7E  D4 00053       CLRL    -(SP)                                   ;
                               10   AE  9F 00055       PUSHAB  SYI_ITMLST                              ;
                                    7E  7C 00058       CLRQ    -(SP)                                   ;
                                    1A  DD 0005A       PUSHL   #26                                     ;
          00000000G  00             07  FB 0005C       CALLS   #7, SYS$GETSYIW                         ;
      50         04  AC             01  78 00063       ASHL    #1, J, R0                               ; 1581
                     56           6B40  DE 00068       MOVAL   LABEL_STRING[R0], R6                    ;
018C  C8             66             08  C1 0006C       ADDL3   #8, (R6), LABLCKNAM_DESC                ;
               016C  C8             6E  D0 00072       MOVL    CSID, LABLCKNAM_BUF+4                   ; 1582
                     50        04 AB40  D0 00077       MOVL    LABEL_STRING+4[R0], R0                  ; 1584
0170  C8             60             66  28 0007C       MOVC3   (R6), (R0), LABLCKNAM_BUF+8             ; 1587
                     7E             01  7D 00082       MOVQ    #1, -(SP)                               ; 1594
                                    7E  7C 00085       CLRQ    -(SP)                                   ;
                                    7E  7C 00087       CLRQ    -(SP)                                   ;
                             018C   C8  9F 00089       PUSHAB  LABLCKNAM_DESC                          ;
                                    10  DD 0008D       PUSHL   #16                                     ;
                             0128   C8  9F 0008F       PUSHAB  LABLCK_STATUS                           ;
                                    05  DD 00093       PUSHL   #5                                      ;
                                    1A  DD 00095       PUSHL   #26                                     ;
          00000000G  00             0B  FB 00097       CALLS   #11, SYS$ENQW                           ;
                                    56  DD 0009E       PUSHL   R6                                      ; 1596
                                    01  DD 000A0       PUSHL   #1                                      ;
                                    5E  DD 000A2       PUSHL   SP                                      ;
                            0000G   CF  9F 000A4       PUSHAB  SEARCH_VOL                              ;
          00000000G  9F             04  FB 000A8       CALLS   #4, @#SYS$CMKRNL                        ;
                     57             50  D0 000AF       MOVL    R0, STATUS                              ;
                     28             57  E9 000B2 1$:   BLBC    STATUS, 2$                              ; 1605
                     50        04   AC  D0 000B5       MOVL    J, R0                                   ; 1611
                             FB40 C840  7F 000B9       PUSHAQ  PHYS_NAME[R0]                           ;
      51             50             01  78 000BE       ASHL    #1, R0, R1                              ;
                               04 AB41  DD 000C2       PUSHL   LABEL_STRING+4[R1]                      ;
                     50             02  C4 000C6       MULL2   #2, R0                                  ;
                                  6B40  DD 000C9       PUSHL   LABEL_STRING[R0]                        ;
                                    03  DD 000CC       PUSHL   #3                                      ;
                         0072A003   8F  DD 000CE       PUSHL   #7512067                                ;
          00000000G  00             05  FB 000D4       CALLS   #5, LIB$SIGNAL                          ;
                                    0A  11 000DB       BRB     3$                                      ;
                     07             57  D1 000DD 2$:   CMPL    STATUS, #7                              ; 1614
                                    08  1B 000E0       BLEQU   4$                                      ;
                                    57  DD 000E2       PUSHL   STATUS                                  ; 1615
                     6A             01  FB 000E4       CALLS   #1, LIB$STOP                            ;
                                  02D1  31 000E7 3$:   BRW     44$                                     ;
                     54        04   AC  D0 000EA 4$:   MOVL    J, R4                                   ; 1628
      64       FB18  C8             54  E0 000EE       BBS     R4, DEV_ACQUIRED, 10$                   ;
      56       FB14  C8             54  E0 000F4       BBS     R4, DEV_ALLOCATED, 9$                   ; 1642
                                    54  DD 000FA       PUSHL   R4                                      ; 1645
                                    01  DD 000FC       PUSHL   #1                                      ;
                                    5E  DD 000FE       PUSHL   SP                                      ;
```

```
VMOUNT                                           L 6
V04-002                             16-Sep-1984 01:00:56    VAX-11 Bliss-32 V4.0-742                   Page  35
                                    12-Sep-1984 11:14:53    DISK$VMSMASTER:[MOUNT.SRC]VMOUNT.B32;3      (7)

                              00000000V   EF 9F 00100       PUSHAB  SEARCH_DEVICE                       ;
          00000000G  9F                   04 FB 00106       CALLS   #4, @#SYS$CMKRNL                    ;
                     57                   50 D0 0010D       MOVL    R0, STATUS                          ;
                     23                   57 E8 00110       BLBS    STATUS, 7$                          ; 1655
     50              57    FFFF0007       8F CB 00113       BICL3   #-65529, STATUS, R0                 ; 1658
           00000908  8F                   50 D1 0011B       CMPL    R0, #2312                           ;
                                          09 13 00122       BEQL    5$                                  ;
           00000140  8F                   50 D1 00124       CMPL    R0, #320                            ; 1659
                                          04 12 0012B       BNEQ    6$                                  ;
                 06  A9                   04 8A 0012D 5$:   BICB2   #4, MOUNT_OPTIONS+6                 ; 1661
                                          57 DD 00131 6$:   PUSHL   STATUS                              ; 1662
                     6A                   01 FB 00133       CALLS   #1, LIB$STOP                        ;
           00000641  8F                   57 D1 00136 7$:   CMPL    STATUS, #1601                       ; 1664
                                          0B 13 0013D       BEQL    8$                                  ;
               FB28  C8                   02 88 0013F       BISB2   #2, CLEANUP_FLAGS                   ; 1667
     00        FB24  C8                   54 E2 00144       BBSS    R4, CLEANUP_ALLOC, 8$               ; 1668
     00        FB14  C8                   54 E2 0014A 8$:   BBSS    R4, DEV_ALLOCATED, 9$               ; 1670
               FB3C  C8          01       A4 9E 00150 9$:   MOVAB   1(R4), PHYS_COUNT                   ; 1676
                                          27 11 00156       BRB     11$                                 ; 1628
                     7E                   01 7D 00158 10$:  MOVQ    #1, -(SP)                           ; 1693
                                          7E 7C 0015B       CLRQ    -(SP)                               ;
                                          7E 7C 0015D       CLRQ    -(SP)                               ;
                               0150       C8 9F 0015F       PUSHAB  ALLDEVNAM_DESC                      ;
                                          10 DD 00163       PUSHL   #16                                 ;
                               FB1C       C8 9F 00165       PUSHAB  LOCK_STATUS                         ;
                                          05 DD 00169       PUSHL   #5                                  ;
                                          1A DD 0016B       PUSHL   #26                                 ;
          00000000G  00                   0B FB 0016D       CALLS   #11, SYS$ENQW                       ;
                     57                   50 D0 00174       MOVL    R0, STATUS                          ;
                     05                   57 E8 00177       BLBS    STATUS, 11$                         ; 1694
                                          57 DD 0017A       PUSHL   STATUS                              ;
                     6A                   01 FB 0017C       CALLS   #1, LIB$STOP                        ;
     00        FB18  C8                   54 E2 0017F 11$:  BBSS    R4, DEV_ACQUIRED, 12$               ; 1702
                                          7E 7C 00185 12$:  CLRQ    -(SP)                               ; 1710
                               FB2C       C8 9F 00187       PUSHAB  CHANNEL                             ;
                               FB40     C844 7F 0018B       PUSHAQ  PHYS_NAME[R4]                       ;
          00000000G  00                   04 FB 00190       CALLS   #4, SYS$ASSIGN                      ;
                     57                   50 D0 00197       MOVL    R0, STATUS                          ;
                     05                   57 E8 0019A       BLBS    STATUS, 13$                         ; 1711
                                          57 DD 0019D       PUSHL   STATUS                              ;
                     6A                   01 FB 0019F       CALLS   #1, LIB$STOP                        ;
                               0160       C8 9F 001A2 13$:  PUSHAB  DEVCHAR_DESC2                       ; 1721
                                          7E D4 001A6       CLRL    -(SP)                               ;
                               0158       C8 9F 001A8       PUSHAB  DEVCHAR_DESC                        ;
                                          7E D4 001AC       CLRL    -(SP)                               ;
                               FB2C       C8 DD 001AE       PUSHL   CHANNEL                             ;
          00000000G  00                   05 FB 001B2       CALLS   #5, SYS$GETCHN                      ;
74   A8              68        0074       8F 29 001B9       CMPC3   #116, DEVICE_CHAR, DEVICE_CHAR2     ; 1723
                                          05 12 001C0       BNEQ    14$                                 ;
     08          01  A8                   06 E0 001C2       BBS     #6, DEVICE_CHAR+1, 15$              ; 1724
                     7E        01CC       8F 3C 001C7 14$:  MOVZWL  #460, -(SP)                         ; 1725
                     6A                   01 FB 001CC       CALLS   #1, LIB$STOP                        ;
     07          02  A8                   02 E0 001CF 15$:  BBS     #2, DEVICE_CHAR+2, 16$              ; 1727
                     7E          84       8F 9A 001D4       MOVZBL  #132, -(SP)                         ;
                     6A                   01 FB 001D8       CALLS   #1, LIB$STOP                        ;
     07          02  A8                   03 E1 001DB 16$:  BBC     #3, DEVICE_CHAR+2, 17$              ; 1729
                     7E          6C       8F 9A 001E0       MOVZBL  #108, -(SP)                         ;
```

```
                              6A               01 FB 001E4        CALLS   #1, LIB$STOP                           ;
                 FB28         C8         80    8F 88 001E7 17$:   BISB2   #128, CLEANUP_FLAGS                    ; 1731
                                               54 D5 001ED        TSTL    R4                                     ; 1736
                                               6D 12 001EF        BNEQ    21$                                    ;
        50             68                      01 05 EF 001F1     EXTZV   #5, #1, DEVICE_CHAR, R0                ; 1742
  FB0C  C8             01                      00 50 F0 001F6     INSV    R0, #0, #1, STORED_CONTEXT             ;
                       2D        01     A9     03 E0 001FD        BBS     #3, MOUNT_OPTIONS+1, 18$               ; 1747
                       28        01     A9     04 E0 00202        BBS     #4, MOUNT_OPTIONS+1, 18$               ;
                       23        04     A9     06 E0 00207        BBS     #6, MOUNT_OPTIONS+4, 18$               ; 1748
                       1E        02     A9     01 E0 0020C        BBS     #1, MOUNT_OPTIONS+2, 18$               ;
                       19        02     A9     04 E0 00211        BBS     #4, MOUNT_OPTIONS+2, 18$               ; 1749
                       14        02     A9     03 E0 00216        BBS     #3, MOUNT_OPTIONS+2, 18$               ;
                       0F        05     A9     02 E0 0021B        BBS     #2, MOUNT_OPTIONS+5, 18$               ; 1750
                       0A        02     A9     02 E0 00220        BBS     #2, MOUNT_OPTIONS+2, 18$               ;
                       05        06     A9     05 E0 00225        BBS     #5, MOUNT_OPTIONS+6, 18$               ; 1751
                       07        07     A9     04 E1 0022A        BBC     #4, MOUNT_OPTIONS+7, 19$               ;
                 FB0C         C8               02 88 0022F 18$:   BISB2   #2, STORED_CONTEXT                     ; 1752
                                               05 11 00234        BRB     20$                                    ;
                 FB0C         C8               02 8A 00236 19$:   BICB2   #2, STORED_CONTEXT                     ; 1753
                              1E       FB0C    C8 E8 0023B 20$:   BLBS    STORED_CONTEXT, 21$                    ; 1757
             00000000G        00  00000000G    00 D1 00240        CMPL    DEVICE_COUNT, LABEL_COUNT              ; 1758
                                               11 13 0024B        BEQL    21$                                    ;
                                  00000000G    00 D5 0024D        TSTL    LABEL_COUNT                            ;
                                               09 13 00253        BEQL    21$                                    ;
                                  0072818C     8F DD 00255        PUSHL   #7504268                               ; 1759
                              6A               01 FB 0025B        CALLS   #1, LIB$STOP                           ;
                       13     68               05 E1 0025E 21$:   BBC     #5, DEVICE_CHAR, 22$                   ; 1768
             038861C0         8F               69 D3 00262        BITL    OPTIONS, #59269568                     ; 1770
                                               1D 12 00269        BNEQ    24$                                    ;
             C1B3E787         8F         04    A9 D3 0026B        BITL    OPTIONS+4, #-1045174393                ; 1771
                                               11 11 00273        BRB     23$                                    ;
             00317003         8F               69 D3 00275 22$:   BITL    OPTIONS, #3239939                      ; 1773
                                               0A 12 0027C        BNEQ    24$                                    ;
             3C0018E7         8F         04    A9 D3 0027E        BITL    OPTIONS+4, #1006639335                 ; 1774
                                               09 13 00286 23$:   BEQL    25$                                    ;
                                  00728034     8F DD 00288 24$:   PUSHL   #7503924                               ; 1776
                              6A               01 FB 0028E        CALLS   #1, LIB$STOP                           ;
                       09     68               05 E0 00291 25$:   BBS     #5, DEVICE_CHAR, 26$                   ; 1781
                              09       FB0C    C8 E8 00295        BLBS    STORED_CONTEXT, 27$                    ;
                       0E     68               05 E1 0029A        BBC     #5, DEVICE_CHAR, 28$                   ; 1783
                              09       FB0C    C8 E8 0029E 26$:   BLBS    STORED_CONTEXT, 28$                    ;
                                  00728184     8F DD 002A3 27$:   PUSHL   #7504260                               ; 1784
                              6A               01 FB 002A9        CALLS   #1, LIB$STOP                           ;
                       50     54               01 78 002AC 28$:   ASHL    #1, R4, R0                             ; 1793
                              52             6B40 DE 002B0        MOVAL   LABEL_STRING[R0], R2                   ;
                       09     68               05 E1 002B4        BBC     #5, DEVICE_CHAR, 29$                   ; 1791
                                               52 DD 002B8        PUSHL   R2                                     ; 1793
                 0000G        CF               01 FB 002BA        CALLS   #1, READ_VOLLABEL                      ;
                                               10 11 002BF        BRB     30$                                    ;
        7E        01   A9                      01 03 EF 002C1 29$: EXTZV  #3, #1, MOUNT_OPTIONS+1, -(SP)          ; 1795
                                               6E 6E D2 002C7     MCOML   (SP), (SP)                             ;
                                               52 DD 002CA        PUSHL   R2                                     ;
                 0000G        CF               02 FB 002CC        CALLS   #2, READ_HOMEBLOCK                     ;
                              57               50 D0 002D1 30$:   MOVL    R0, STATUS                             ;
                       04     A9               02 88 002D4        BISB2   #2, MOUNT_OPTIONS+4                    ; 1802
                              67               57 E8 002D8        BLBS    STATUS, 36$                            ; 1803
             000008E0         8F               57 D1 002DB        CMPL    STATUS, #2272                          ; 1805
```

```
                                 09  13 002E2          BEQL    31$                                 ;
    000001DC  8F                 57  D1 002E4          CMPL    STATUS, #476                        ;
                                 10  12 002EB          BNEQ    32$                                 ;
          04  A9                 02  8A 002ED  31$:    BICB2   #2, MOUNT_OPTIONS+4                 ;  1808
    4C    01  A9                 03  E0 002F1          BBS     #3, MOUNT_OPTIONS+1, 36$            ;  1809
    47    01  A9                 04  E0 002F6          BBS     #4, MOUNT_OPTIONS+1, 36$            ;  1810
                                 18  11 002FB          BRB     33$                                 ;  1812
    0000010C  8F                 57  D1 002FD  32$:    CMPL    STATUS, #268                        ;  1820
                                 33  12 00304          BNEQ    34$                                 ;
                         03      A9  95 00306          TSTB    MOUNT_OPTIONS+3                     ;  1823
                                 37  18 00309          BGEQ    36$                                 ;
    32    01  A9                 03  E0 0030B          BBS     #3, MOUNT_OPTIONS+1, 36$            ;  1824
    2D    02  A9                 06  E0 00310          BBS     #6, MOUNT_OPTIONS+2, 36$            ;  1825
    24        68                 05  E0 00315  33$:    BBS     #5, DEVICE_CHAR, 35$                ;  1827
                         F0      A8  9F 00319          PUSHAB  HOME_BLOCK+496                      ;  1832
                                 0C  DD 0031C          PUSHL   #12                                 ;
                         E4      A8  9F 0031E          PUSHAB  HOME_BLOCK+484                      ;
                                 0C  DD 00321          PUSHL   #12                                 ;
                         D8      A8  9F 00323          PUSHAB  HOME_BLOCK+472                      ;
                                 0C  DD 00326          PUSHL   #12                                 ;
                                 06  DD 00328          PUSHL   #6                                  ;
                  0072A00B       8F  DD 0032A          PUSHL   #7512075                            ;
                                 7E  D4 00330          CLRL    -(SP)                               ;
                                 57  DD 00332          PUSHL   STATUS                              ;
              6A                 0A  FB 00334          CALLS   #10, LIB$STOP                       ;
                                 09  11 00337          BRB     36$                                 ;  1827
          04  A9                 02  8A 00339  34$:    BICB2   #2, MOUNT_OPTIONS+4                 ;  1837
                                 57  DD 0033D  35$:    PUSHL   STATUS                              ;  1838
              6A                 01  FB 0033F          CALLS   #1, LIB$STOP                        ;
    0A    04  A9                 01  E1 00342  36$:    BBC     #1, MOUNT_OPTIONS+4, 37$            ;  1845
    04  FB0C  C8                 01  E1 00347          BBC     #1, STORED_CONTEXT, 37$             ;
          04  A9                 01  88 0034D          BISB2   #1, MOUNT_OPTIONS+4                 ;  1846
    07        68                 05  E1 00351  37$:    BBC     #5, DEVICE_CHAR, 38$                ;  1852
        0000G  CF                00  FB 00355          CALLS   #0, MOUNT_TAPE                      ;  1855
                                 3F  11 0035A          BRB     42$                                 ;  1856
                                 7E  D4 0035C  38$:    CLRL    -(SP)                               ;  1866
                                 5E  DD 0035E          PUSHL   SP                                  ;
                      0000G      CF  9F 00360          PUSHAB  GET_DEVICE_CONTEXT                  ;
     00000000G  9F               03  FB 00364          CALLS   #3, @#SYS$CMKRNL                    ;
              57                 50  D0 0036B          MOVL    R0, STATUS                          ;
              05                 57  E8 0036E          BLBS    STATUS, 39$                         ;
                                 57  DD 00371          PUSHL   STATUS                              ;  1868
              6A                 01  FB 00373          CALLS   #1, LIB$STOP                        ;
    07    04  A9                 02  E1 00376  39$:    BBC     #2, MOUNT_OPTIONS+4, 40$            ;  1870
        0000G  CF                00  FB 0037B          CALLS   #0, MOUNT_DISK2                     ;  1872
                                 05  11 00380          BRB     41$                                 ;
        0000G  CF                00  FB 00382  40$:    CALLS   #0, MOUNT_DISK1                     ;  1874
    10        69                 04  E0 00387  41$:    BBS     #4, MOUNT_OPTIONS, 42$              ;  1880
                       FB2C      C8  DD 0038B          PUSHL   CHANNEL                             ;  1881
                                 01  DD 0038F          PUSHL   #1                                  ;
                                 5E  DD 00391          PUSHL   SP                                  ;
                 00000000V       EF  9F 00393          PUSHAB  DALLOC_SHR_DEV                      ;
                                 0E  11 00399          BRB     43$                                 ;
                       FB2C      C8  DD 0039B  42$:    PUSHL   CHANNEL                             ;  1882
                                 01  DD 0039F          PUSHL   #1                                  ;
                                 5E  DD 003A1          PUSHL   SP                                  ;
                 00000000V       EF  9F 003A3          PUSHAB  XFER_DEV_OWNER                      ;
     00000000G  9F              04  FB 003A9 43$:    CALLS   #4, @#SYS$CMKRNL                   ;
                      FB2C      C8  DD 003B0         PUSHL   CHANNEL                            ; 1889
     00000000G  00              01  FB 003B4         CALLS   #1, SYS$DASSGN                     ;
        FB28    C8  FFFFF9FF    8F  CA 003BB 44$:    BICL2   #-1537, CLEANUP_FLAGS              ; 1898
                      FB2C      C8  D4 003C4         CLRL    CHANNEL                            ; 1899
                      0100      C8  7C 003C8         CLRQ    REAL_MVL                           ; 1900
                      0108      C8  D4 003CC         CLRL    REAL_VCB                           ; 1902
                      0110      C8  7C 003D0         CLRQ    REAL_FCB                           ; 1903
                      0118      C8  7C 003D4         CLRQ    REAL_AQB                           ; 1905
                      0120      C8  D4 003D8         CLRL    SMTL_ENTRY                         ; 1907
          04    A9    0207      8F  AA 003DC         BICW2   #519, OPTIONS+4                    ; 1909
                 00000000G      00  D4 003E2         CLRL    MOUNT_FAILED                       ; 1910
                50              01  D0 003E8         MOVL    #1, R0                             ; 1911
                                    04 003EB         RET                                        ; 1912
                                  0000 003EC 45$:    .WORD   Save nothing                       ; 1487
                                7E  D4 003EE         CLRL    -(SP)                              ;
                                5E  DD 003F0         PUSHL   SP                                 ;
                7E        04    AC  7D 003F2         MOVQ    4(AP), -(SP)                       ;
     00000000V  EF              03  FB 003F6         CALLS   #3, MAIN_HANDLER                   ;
                                    04 003FD         RET                                        ;

; Routine Size:  1022 bytes,    Routine Base:  $CODE$ + 0282
```

```
1 ROUTINE MAIN_HANDLER (SIGNAL, MECHANISM) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is the main level condition handler for the MOUNT
1 !       utility. It undoes anything that MOUNT has done so far and returns
1 !       the condition code as status to MOUNT's caller (i.e., the CLI).
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       MAIN_HANDLER (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of signal array
1 !       ARG2: address of mechanism array
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       stack unwound, control passed to CLI
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       SIGNAL          : REF BBLOCK,   ! signal array
2       MECHANISM       : REF BBLOCK;   ! mechanism array
2
2 EXTERNAL
2       USER_STATUS     : VECTOR;       ! status return of some routines
2
2
2 IF .SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND
2 THEN
3     BEGIN
3     !
3     ! Do cleanup as indicated by the status flags.
3     !
3     IF .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_SEVERE
3     THEN
4         BEGIN
4         IF .CLEANUP_FLAGS[CLF_DISMOUNT]
4         THEN
4             KERNEL_CALL (FORCE_DISMOUNT);
```

```
; 1315          1970  4
; 1316          1971  4               IF .CHANNEL NEQ 0
; 1317          1972  4               THEN
; 1318          1973  5                   BEGIN
; 1319          1974  5                   IF NOT .CLEANUP_FLAGS[CLF_DISMOUNT]
; 1320          1975  5                       AND .CLEANUP_FLAGS[CLF_CLEARVALID]
; 1321          1976  5                   THEN
; 1322          1977  6                       BEGIN
; 1323        P 1978  6                       DO_IO (CHAN = .CHANNEL,
; 1324        P 1979  6                              FUNC = (IO$_AVAILABLE OR IO$M_INHERLOG),
; 1325        P 1980  6                              EFN  = MOUNT_EFN
; 1326          1981  6                             );
; 1327          1982  6                       IF .STORED_CONTEXT [TAPE_MOUNT]
; 1328          1983  6                       THEN KERNEL_CALL (CLEAR_VALID);
; 1329          1984  5                       END;
; 1330          1985  5                   $DASSGN (CHAN = .CHANNEL);
; 1331          1986  5                   CHANNEL = 0;
; 1332          1987  4                   END;
; 1333          1988  4
; 1334          1989  4               !
; 1335          1990  4               ! Zero the various cleanup flags.
; 1336          1991  4               !
; 1337          1992  4               CLEANUP_FLAGS[CLF_DISMOUNT] = 0;
; 1338          1993  4               CLEANUP_FLAGS[CLF_CLEARVALID] = 0;
; 1339          1994  4               CLEANUP_FLAGS[CLF_DEASSTEMP] = 0;
; 1340          1995  4
; 1341          1996  3               END;
; 1342          1997  2       END;
; 1343          1998  2 !
; 1344          1999  2 ! Resignal the condtion.  Does not affect UNWIND.
; 1345          2000  2 !
; 1346          2001  2 RETURN SS$_RESIGNAL;
; 1347          2002  2
; 1348          2003  1 END;                                    ! end of routine MAIN_HANDLER


                                                                .EXTRN  COMMON_IO

                                          000C 00000 MAIN_HANDLER:
                                                                .WORD   Save R2,R3                              ; 1913
                        53 00000000G  9F  9E 00002              MOVAB   @#SYS$CMKRNL, R3                        ;
                        52 00000000'  EF  9E 00009              MOVAB   CLEANUP_FLAGS, R2                       ;
                        50         04  AC  D0 00010              MOVL    SIGNAL, R0                              ; 1958
               00000920  8F         04  A0  D1 00014              CMPL    4(R0), #2336                            ;
                                        6A  13 0001C              BEQL    4$                                      ;
     04        04  A0         03        00  ED 0001E              CMPZV   #0, #3, 4(R0), #4                       ; 1964
                                        62  12 00024              BNEQ    4$                                      ;
               0D             62        06  E1 00026              BBC     #6, CLEANUP_FLAGS, 1$                   ; 1967
                                        7E  D4 0002A              CLRL    -(SP)                                   ; 1969
                                        5E  DD 0002C              PUSHL   SP                                      ;
                        00000000V  EF  9F 0002E              PUSHAB  FORCE_DISMOUNT                          ;
                              63        03  FB 00034              CALLS   #3, SYS$CMKRNL                          ;
                                   04  A2  D5 00037 1$:          TSTL    CHANNEL                                 ; 1971
                                        41  13 0003A              BEQL    3$                                      ;
               30             62        06  E0 0003C              BBS     #6, CLEANUP_FLAGS, 2$                   ; 1974
                                        62  95 00040              TSTB    CLEANUP_FLAGS                           ; 1975
```

```
                                       2C  18 00042         BGEQ    2$                                  ;
                                       7E  7C 00044         CLRQ    -(SP)                               ; 1981
                                       7E  7C 00046         CLRQ    -(SP)                               ;
                                       7E  7C 00048         CLRQ    -(SP)                               ;
                                       7E  7C 0004A         CLRQ    -(SP)                               ;
                                       7E  D4 0004C         CLRL    -(SP)                               ;
                        7E      0811   8F  3C 0004E         MOVZWL  #2065, -(SP)                        ;
                                  04   A2  DD 00053         PUSHL   CHANNEL                             ;
                                       1A  DD 00056         PUSHL   #26                                 ;
              00000000G 00             0C  FB 00058         CALLS   #12, COMMON_IO                      ;
                        0D        E4   A2  E9 0005F         BLBC    STORED_CONTEXT, 2$                  ; 1982
                                       7E  D4 00063         CLRL    -(SP)                               ; 1983
                                       5E  DD 00065         PUSHL   SP                                  ;
                           00000000V   EF  9F 00067         PUSHAB  CLEAR_VALID                         ;
                        63             03  FB 0006D         CALLS   #3, SYS$CMKRNL                      ;
                                  04   A2  DD 00070 2$:     PUSHL   CHANNEL                             ; 1985
              00000000G 00             01  FB 00073         CALLS   #1, SYS$DASSGN                      ;
                                  04   A2  D4 0007A         CLRL    CHANNEL                             ; 1986
                        62        40   8F  8A 0007D 3$:     BICB2   #64, CLEANUP_FLAGS                  ; 1992
                        62        80   8F  8A 00081         BICB2   #128, CLEANUP_FLAGS                 ; 1993
                        62             10  8A 00085         BICB2   #16, CLEANUP_FLAGS                  ; 1994
                        50      0918   8F  3C 00088 4$:     MOVZWL  #2328, R0                           ; 2001
                                           04 0008D         RET                                         ; 2003

; Routine Size:  142 bytes,    Routine Base:  $CODE$ + 0680
```

```
1 ROUTINE FORCE_DISMOUNT =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine initiates a dismount on the volume just mounted
1 !       (usually because an error occurred during the /BIND processing).
1 !       This routine must be called in kernel mode.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       FORCE_DISMOUNT ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       MTL_ENTRY: address of mounted volume list entry just created
1 !       SMTL_ENTRY: as above, for volume set if non-zero
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !       volume dismounted
1 !
1 !--
1
2 BEGIN
2
2 BUILTIN
2       REMQUE;
2
2 LINKAGE
2       IOC_DISMOUNT    = JSB (REGISTER = 6, REGISTER = 3, REGISTER = 4) :
2                         NOPRESERVE (2);
2
2 EXTERNAL
2       SCH$GL_CURPCB   : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
2                                       ! address of process PCB
2
2 EXTERNAL ROUTINE
2       IOC$DISMOUNT    : IOC_DISMOUNT ADDRESSING_MODE (GENERAL);
2                                       ! system dismount routine
2
2 LOCAL
2       MTL             : REF BBLOCK;   ! address of mount list entry
2
2
2 REMQUE (.MTL_ENTRY, MTL);
```

```
; 1407          2061  2 IOC$DISMOUNT (.MTL, 1, .SCH$GL_CURPCB);
; 1408          2062  2
; 1409          2063  2 IF .SMTL_ENTRY NEQ 0
; 1410          2064  2 THEN
; 1411          2065  3     BEGIN
; 1412          2066  3     REMQUE (.SMTL_ENTRY, MTL);
; 1413          2067  3     IOC$DISMOUNT (.MTL, 1, .SCH$GL_CURPCB);
; 1414          2068  2     END;
; 1415          2069  2
; 1416          2070  2 RETURN 1;
; 1417          2071  2
; 1418          2072  1 END;                                              ! end of routine FORCE_DISMOUNT


                                                                   .EXTRN  SCH$GL_CURPCB, IOC$DISMOUNT

                                                   OFFC 00000 FORCE_DISMOUNT:
                                                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 2004
                               57 0000000G  00  9E 00002           MOVAB   IOC$DISMOUNT, R7                            ;
                               55 0000000G  9F  9E 00009           MOVAB   @#SCH$GL_CURPCB, R5                         ;
                               56 00000000'  FF  0F 00010           REMQUE  @MTL_ENTRY, MTL                             ; 2060
                               54            65  D0 00017           MOVL    SCH$GL_CURPCB, R4                           ; 2061
                               53            01  D0 0001A           MOVL    #1, R3                                      ;
                                             67  16 0001D           JSB     IOC$DISMOUNT                                ;
                               50 00000000'  EF  D0 0001F           MOVL    SMTL_ENTRY, R0                              ; 2063
                                             0B  13 00026           BEQL    1$                                          ;
                               56            60  0F 00028           REMQUE  (R0), MTL                                   ; 2066
                               54            65  D0 0002B           MOVL    SCH$GL_CURPCB, R4                           ; 2067
                               53            01  D0 0002E           MOVL    #1, R3                                      ;
                                             67  16 00031           JSB     IOC$DISMOUNT                                ;
                               50            01  D0 00033 1$:       MOVL    #1, R0                                      ; 2070
                                             04 00036              RET                                                 ; 2072

; Routine Size:  55 bytes,    Routine Base:  $CODE$ + 070E


; 1419          2073  1
```

```
1 ROUTINE CLEAR_VALID =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine clears the volume valid bit in the UCB.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       CLEAR_VALID ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       CHANNEL: channel number assigned to device
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !       valid bit clear in UCB
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       UCB             : REF BBLOCK;   ! pointer to UCB
2
2 EXTERNAL
2       CHANNEL;                        ! channel assigned to device
2
2 EXTERNAL ROUTINE
2       GET_CHANNELUCB;                 ! get UCB of channel
2
2
2 ! Get the UCB address from the channel and clear the bit.
2 !
2
2 UCB = GET_CHANNELUCB (.CHANNEL);
2 UCB[UCB$V_VALID] = 0;
2
2 RETURN 1;
2
1 END;                                  ! end of routine CLEAR_VALID


                                                        .EXTRN  GET_CHANNELUCB
```

```
                                   0000 00000 CLEAR_VALID:
                                                      .WORD    Save nothing                                    ; 2074
                         0000G  CF    DD 00002        PUSHL    CHANNEL                                         ; 2121
           0000G  CF            01    FB 00006        CALLS    #1, GET_CHANNELUCB                              ;
              65  A0            08    8A 0000B        BICB2    #8, 101(UCB)                                    ; 2122
                  50            01    D0 0000F        MOVL     #1, R0                                          ; 2124
                                      04 00012        RET                                                      ; 2126

; Routine Size:  19 bytes,    Routine Base:  $CODE$ + 0745
```

```
1 GLOBAL ROUTINE DALLOC_SHR_DEV (CHANNEL) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine locates the UCB associated with the channel passed to
1 !       it as an input argument.  It then deallocates the device (i.e. marks
1 !       the UCB as unallocated) on the local system.  If an
1 !       exclusive cluster-wide lock exists for this device, it will also
1 !       convert it into a CR mode lock.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       kernel_call (dalloc_shr_dev, .channel)
1 !
1 !       This routine must be called in kernel mode.
1 !
1 ! INPUT:
1 !
1 !       CHANNEL = channel to the device which is being mounted
1 !
1 ! OUTPUT:
1 !
1 !       None.
1 !
1 ! IMPLICIT INPUT:
1 !
1 !       Mount data base.
1 !       Device is being mounted /SHARE, /GROUP, or /SYSTEM.
1 !
1 ! IMPLICIT OUTPUT:
1 !
1 !       None.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       1 if control is returned to the caller.  Otherwise, the procedure
1 !       signals an error.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Device is deallocated.  Device lock is converted to CR mode.
1 !--
1
2 BEGIN
2
2 EXTERNAL ROUTINE
2       GET_CHANNELUCB: ADDRESSING_MODE (GENERAL);
2
2 LOCAL
2       STATUS,                         ! Status of $ENQ call.
2       LOCK_STATUS: VECTOR [2],        ! Lock status block.
2       UCB: REF BBLOCK;                ! UCB of device.
2
2 UCB = GET_CHANNELUCB (.CHANNEL);      ! Get the UCB address.
2 !
```

```
; 1532          2184   2 ! We already know that this is a shared mount; check to see if the device
; 1533          2185   2 ! was previously allocated.
; 1534          2186   2 !
; 1535          2187   2 IF .UCB [UCB$L_PID] NEQ 0
; 1536          2188   3    THEN BEGIN
; 1537          2189   3         !
; 1538          2190   3         ! Deallocate the local UCB.
; 1539          2191   3         !
; 1540          2192   3         UCB [UCB$L_PID] = 0;
; 1541          2193   3         BBLOCK [UCB [UCB$L_DEVCHAR], DEV$V_ALL] = 0;
; 1542          2194   3         UCB [UCB$W_REFC] = .UCB [UCB$W_REFC] - 1;
; 1543          2195   3         !
; 1544          2196   3         ! If an exclusive lock exists, convert it to CR mode.
; 1545          2197   3         !
; 1546          2198   3         IF .UCB [UCB$L_LOCKID] NEQ 0
; 1547          2199   4             THEN BEGIN
; 1548          2200   4                  LOCK_STATUS [1] = .UCB [UCB$L_LOCKID];
; 1549        P 2201   4                  STATUS = $ENQW (ACMODE = PSL$C_KERNEL,
; 1550        P 2202   4                                  EFN    = MOUNT_EFN,
; 1551        P 2203   4                                  LKSB   = LOCK_STATUS,
; 1552        P 2204   4                                  FLAGS  = (LCK$M_CONVERT OR LCK$M_CVTSYS),
; 1553          2205   4                                  LKMODE = LCK$K_CRMODE);
; 1554          2206   4                  IF NOT .STATUS THEN ERR_EXIT (.STATUS);
; 1555          2207   3                  END;
; 1556          2208   2         END;
; 1557          2209   2
; 1558          2210   2 RETURN (1);
; 1559          2211   2
; 1560          2212   1 END;                                  ! End of routine DALLOC_SHR_DEV.



                                           0000 00000         .ENTRY   DALLOC_SHR_DEV, Save nothing                    ; 2127
                    5E                   08  C2 00002         SUBL2    #8, SP                                          ;
                                  04     AC  DD 00005         PUSHL    CHANNEL                                         ; 2182
          00000000G  00                  01  FB 00008         CALLS    #1, GET_CHANNELUCB                              ;
                                  2C     A0  D5 0000F         TSTL     44(UCB)                                         ; 2187
                                         3B  13 00012         BEQL     1$                                              ;
                                  2C     A0  D4 00014         CLRL     44(UCB)                                         ; 2192
                 3A  A0           80     8F  8A 00017         BICB2    #128, 58(UCB)                                   ; 2193
                                  5C     A0  B7 0001C         DECW     92(UCB)                                         ; 2194
                                  20     A0  D5 0001F         TSTL     32(UCB)                                         ; 2198
                                         2B  13 00022         BEQL     1$                                              ;
                 04  AE           20     A0  D0 00024         MOVL     32(UCB), LOCK_STATUS+4                          ; 2200
                                         7E  7C 00029         CLRQ     -(SP)                                           ; 2205
                                         7E  7C 0002B         CLRQ     -(SP)                                           ;
                                         7E  7C 0002D         CLRQ     -(SP)                                           ;
                                         7E  D4 0002F         CLRL     -(SP)                                           ;
                     7E           42     8F  9A 00031         MOVZBL   #66, -(SP)                                      ;
                                  20     AE  9F 00035         PUSHAB   LOCK_STATUS                                     ;
                                         01  DD 00038         PUSHL    #1                                              ;
                                         1A  DD 0003A         PUSHL    #26                                             ;
          00000000G  00                  0B  FB 0003C         CALLS    #11, SYS$ENQW                                   ;
                     09                  50  E8 00043         BLBS     STATUS, 1$                                      ; 2206
                                         50  DD 00046         PUSHL    STATUS                                          ;
                     00000000G  00            01  FB 00048        CALLS   #1, LIB$STOP                               ;
                                50            01  D0 0004F 1$:    MOVL    #1, R0                                     ; 2210
                                                  04 00052        RET                                                ; 2212

; Routine Size:  83 bytes,    Routine Base:  $CODE$ + 0758
```

```
1 GLOBAL ROUTINE XFER_DEV_OWNER (CHANNEL) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine locates the UCB associated with the channel passed to
1 !       it as an input argument.  If current process is a subprocess, then
1 !       the device ownership is transfered to the top level process in the
1 !       process tree. This is necessary to support job-wide mount.
1 !
1 !       Note: we perform the transfer of ownership simply by setting the
1 !       master's PID into the UCB. This is sufficient because the lock on
1 !       this device is not tied to this process, i.e. it is a system-owned
1 !       lock.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       KERNEL_CALL (XFER_DEV_OWNER, .CHANNEL)
1 !
1 !       This routine must be called in kernel mode.
1 !
1 ! INPUT:
1 !
1 !       CHANNEL = channel to the device which is being mounted
1 !
1 ! OUTPUT:
1 !
1 !       None.
1 !
1 ! IMPLICIT INPUT:
1 !
1 !       Mount data base.
1 !       Device is being mounted /NOSHARE.
1 !
1 ! IMPLICIT OUTPUT:
1 !
1 !       If the current process is a subprocess, then the device is
1 !               allocated to its master,
1 !       else
1 !               none.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       1.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None.
1 !
1 !--
1
2 BEGIN
2
2 EXTERNAL
2       SCH$GL_CURPCB   : REF BBLOCK ADDRESSING_MODE (GENERAL);
2                                               ! address of our PCB
```

```
; 1619          2270  2
; 1620          2271  2 EXTERNAL ROUTINE
; 1621          2272  2         GET_CHANNELUCB  : ADDRESSING_MODE (GENERAL);
; 1622          2273  2
; 1623          2274  2 LOCAL
; 1624          2275  2         JIB     : REF BBLOCK,           ! JIB of current process
; 1625          2276  2         UCB     : REF BBLOCK;           ! UCB of device.
; 1626          2277  2
; 1627          2278  2 UCB = GET_CHANNELUCB (.CHANNEL);         ! Get the UCB address.
; 1628          2279  2 !
; 1629          2280  2 ! We already know that this is a private mount; check to see if the device
; 1630          2281  2 ! was previously allocated for sanity's sake.
; 1631          2282  2 !
; 1632          2283  2 IF .UCB [UCB$L_PID] NEQ 0
; 1633          2284  2 THEN
; 1634          2285  3     BEGIN
; 1635          2286  3     !
; 1636          2287  3     ! Check if the current process is a subprocess. If so, set the PID
; 1637          2288  3     ! of the top level process in the process tree in the UCB.
; 1638          2289  3     !
; 1639          2290  3     IF .SCH$GL_CURPCB [PCB$L_OWNER] NEQ 0
; 1640          2291  3     THEN
; 1641          2292  4         BEGIN
; 1642          2293  4         JIB = .SCH$GL_CURPCB [PCB$L_JIB];
; 1643          2294  4         UCB [UCB$L_PID] = .JIB [JIB$L_MPID];
; 1644          2295  3         END;
; 1645          2296  2     END;
; 1646          2297  2
; 1647          2298  2 RETURN 1;
; 1648          2299  2
; 1649          2300  1 END;                                  ! End of routine XFER_DEV_OWNER.



                                        0000 00000          .ENTRY  XFER_DEV_OWNER, Save nothing                            ; 2213
                           04   AC   DD 00002          PUSHL   CHANNEL                                                 ; 2278
        00000000G  00              01   FB 00005          CALLS   #1, GET_CHANNELUCB
                           2C   A0   D5 0000C          TSTL    44(UCB)                                                 ; 2283
                           16   13 0000F          BEQL    1$
             51 00000000G  00   D0 00011          MOVL    SCH$GL_CURPCB, R1                                       ; 2290
                           1C   A1   D5 00018          TSTL    28(R1)
                                0A   13 0001B          BEQL    1$
             51          0080   C1   D0 0001D          MOVL    128(R1), JIB                                            ; 2293
        2C   A0            54   A1   D0 00022          MOVL    84(JIB), 44(UCB)                                        ; 2294
             50                 01   D0 00027 1$:      MOVL    #1, R0                                                  ; 2298
                                     04 0002A          RET                                                             ; 2300

; Routine Size:  43 bytes,    Routine Base:  $CODE$ + 07AB


; 1650          2301  1
```

```
1
1 GLOBAL ROUTINE MOUNT_CLUSTER (ITEM_LIST) =
1
1 !+
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine performs the cluster-wide mount operation.
1 !       It calls another routine to create a cluster-mount packet
1 !       and then sends this mount request to other nodes in the
1 !       cluster.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       MOUNT_CLUSTER (ARG1)
1 !
1 ! INPUTS:
1 !
1 !       ARG1     : Address of the mount item list
1 !
1 ! OUTPUTS:
1 !
1 !       None.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None.
1 !
1 ! OUTPUT PARAMETERS:
1 !
1 !       None.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       1          : If success
1 !       Otherwise : Status from comm primitive.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The mount request is sent to other nodes in the cluster.
1 !
1 !-
1
1
2 BEGIN                                             ! Start of MOUNT_CLUSTER
2
2 MAP
2       ITEM_LIST         : REF BBLOCK;
2
2 EXTERNAL ROUTINE
2       IN_CLUSTER        : ADDRESSING_MODE (GENERAL),
2       SEND_CLUSTER      : ADDRESSING_MODE (GENERAL),
2       GET_DIC           : ADDRESSING_MODE (GENERAL);
```

```
2
2  EXTERNAL
2          MOUNT_OPTIONS    : BITVECTOR VOLATILE; ! Parser option flags
2
2  !
2  ! Define constants to calculate the size of the cluster-mount buffer
2  !
2  LITERAL
2          ITEM_SIZE        = 12,
2          NO_OF_ITEMS      = 18,
2          BUFFER_SIZE      = 63,
2          COMMENT_SIZE     = 80,
2          ITEM_LIST_SIZE   = ( (ITEM_SIZE*DEVMAX)*2 + (NO_OF_ITEMS*ITEM_SIZE) +4 );
2
2  LOCAL
2          STATUS,
2          LENGTH,
2          UIC;
2
2  OWN
2          BUFFER           : VECTOR [0],                           ! Buffer area for
2                                                                   ! cluster-mount packet
2          ITEM_BUF         : BBLOCK [ITEM_LIST_SIZE],              ! Item descriptors
2
2          LABEL_BUF        : BBLOCK [BUFFER_SIZE * DEVMAX],        ! Volume labels
2          LOGNAM_BUF       : BBLOCK [BUFFER_SIZE * DEVMAX],        ! Logical names
2          ACPNAM_BUF       : BBLOCK [BUFFER_SIZE],                 ! ACP name
2          VOLSET_BUF       : BBLOCK [BUFFER_SIZE],                 ! Volume set name
2          COMMENT_BUF      : BBLOCK [COMMENT_SIZE],                ! Comments
2
2          NAME_BUF         : VECTOR [NAMEBUF_LEN * DEVMAX, BYTE],  ! Device names
2          BUFFER_END       : VECTOR [0];
2
2  LITERAL
2          BUFFER_LEN       = BUFFER_END - BUFFER;
2
2
3  IF ( NOT .MOUNT_OPTIONS [OPT_CLUSTER] )            ! If not /cluster or not in a
3  OR NOT ( STATUS = IN_CLUSTER() )                   ! cluster environment, return
2  THEN                                               ! immediately
2      RETURN 1;
2
2  CH$FILL (0, BUFFER_LEN, BUFFER);                   ! Zero buffer area
2  STATUS = MOUNT_ENCIPHER (.ITEM_LIST, BUFFER_LEN, BUFFER, LENGTH);
2                                                     ! Encipher the mount request
2  IF NOT .STATUS                                     ! If error, return
2  THEN
2      RETURN .STATUS;
2
2  UIC = KERNEL_CALL (GET_UIC);                       ! Get our UIC
2  STATUS = KERNEL_CALL (SEND_CLUSTER, BUFFER, .LENGTH, .UIC); ! Broadcast the request
2                                                     ! Arg3 (UIC) non-zero means a cluster-mount
2  RETURN .STATUS;
2
1  END;                                               ! End of MOUNT_CLUSTER
```

```
                                                          .PSECT  $OWN$,NOEXE,2

                                   00000 BUFFER:  .BLKB   0
                                   00000 ITEM_BUF:
                                                  .BLKB   604
                                   0025C LABEL_BUF:
                                                  .BLKB   1008
                                   0064C LOGNAM_BUF:
                                                  .BLKB   1008
                                   00A3C ACPNAM_BUF:
                                                  .BLKB   63
                                   00A7B          .BLKB   1
                                   00A7C VOLSET_BUF:
                                                  .BLKB   63
                                   00ABB          .BLKB   1
                                   00ABC COMMENT_BUF:
                                                  .BLKB   80
                                   00B0C NAME_BUF:
                                                  .BLKB   512
                                   00D0C BUFFER_END:
                                                  .BLKB   0

                                                  .EXTRN  IN_CLUSTER, SEND_CLUSTER
                                                  .EXTRN  GET_UIC

                                                  .PSECT  $CODE$,NOWRT,2

                                01FC 00000        .ENTRY  MOUNT_CLUSTER, Save R2,R3,R4,R5,R6,R7,R8        ; 2303
                 58 00000000G   9F   9E 00002     MOVAB   @#SYS$CMKRNL, R8                                :
                 57 00000000'   EF   9E 00009     MOVAB   BUFFER, R7                                      :
                 5E              04   C2 00010     SUBL2   #4, SP                                          :
       0D     0000G  CF          06   E1 00013     BBC     #6, MOUNT_OPTIONS+7, 1$                         ; 2396
           00000000G  00          00   FB 00019     CALLS   #0, IN_CLUSTER                                  ; 2397
                 56              50   D0 00020     MOVL    R0, STATUS                                      :
                 04              56   E8 00023     BLBS    STATUS, 2$                                      :
                 50              01   D0 00026 1$: MOVL    #1, R0                                          ; 2399
                                 04      00029     RET                                                     :
 0D0C    8F     00       6E      00   2C 0002A 2$: MOVC5   #0, (SP), #0, #3340, BUFFER                     ; 2401
                                 67      00031                                                             :
                          4080  8F   BB 00032     PUSHR   #^M<R7,SP>                                      ; 2402
                 7E       0D0C  8F   3C 00036     MOVZWL  #3340, -(SP)                                    :
                            04  AC   DD 0003B     PUSHL   ITEM_LIST                                       :
           00000000V  EF          04   FB 0003E     CALLS   #4, MOUNT_ENCIPHER                              :
                 56              50   D0 00045     MOVL    R0, STATUS                                      :
                 24              56   E9 00048     BLBC    STATUS, 3$                                      ; 2404
                                 7E   D4 0004B     CLRL    -(SP)                                           ; 2408
                                 5E   DD 0004D     PUSHL   SP                                              :
                      00000000G  00   9F 0004F     PUSHAB  GET_UIC                                         :
                 68              03   FB 00055     CALLS   #3, SYS$CMKRNL                                  :
                                 50   DD 00058     PUSHL   UIC                                             ; 2409
                            04   AE   DD 0005A     PUSHL   LENGTH                                          :
                                 57   DD 0005D     PUSHL   R7                                              :
                                 03   DD 0005F     PUSHL   #3                                              :
                                 5E   DD 00061     PUSHL   SP                                              :
                      00000000G  00   9F 00063     PUSHAB  SEND_CLUSTER                                    :
                 68              06   FB 00069     CALLS   #6, SYS$CMKRNL                                  :
                        56          50  D0 0006C         MOVL    R0, STATUS                              ;
                        50          56  D0 0006F 3$:     MOVL    STATUS, R0                              ; 2411
                                        04 00072         RET                                             ; 2413

; Routine Size:  115 bytes,    Routine Base:  $CODE$ + 07D6


; 1764          2414  1
; 1765          2415  1
```

```
ROUTINE MOUNT_ENCIPHER (ITEM_LIST, LIMIT, BUFFER, LENGTH) =

!+
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine takes the parameters of the mount request
!       and enciphers the parameters into a cluster-mount packet.
!
! CALLING SEQUENCE:
!
!       MOUNT_ENCIPHER (ARG1,ARG2,ARG3,ARG4)
!
! INPUTS:
!
!       ARG1     : Address of the item list
!       ARG2     : Output buffer limit
!
! OUTPUTS:
!
!       None.
!
! IMPLICIT INPUTS:
!
!       None.
!
! OUTPUT PARAMETERS:
!
!       ARG3     : Address of the output buffer to receive the
!                       cluster-mount packet
!       ARG4     : Address of a longword to receive the length of
!                       the output buffer
!
! IMPLICIT OUTPUTS:
!
!       None.
!
! ROUTINE VALUES:
!
!       1               : If successful
!       SS$_BUFFEROVF   : Insufficient internal buffer space,
!                         i.e. length exceeds limit
!
! SIDE EFFECTS:
!
!       None.
!
! NOTES:
!
! This encipher routine takes the given mount item list and turns it
! into a cluster-mount packet of the form:
!
!                                       Offset
!       +---------------+
!       | code1 | len1  |  0  ITEM_LENG item_desc_1
```

```
1 !             +-----------------+
1 !             |offset to str_1|  8  ITEM_ADDR
1 !             +-----------------+
1 !             |     unused      | 12  ITEM_NULL
1 !             +-----------------+
1 !             | code2 | len2    |  0  ITEM_LENG item_desc_2
1 !             +-----------------+
1 !             |offset to str_2|  8  ITEM_ADDR
1 !             +-----------------+
1 !             |     unused      | 12  ITEM_NULL
1 !             +-----------------+
1 !             .                 .
1 !             .                 .
1 !             .                 .
1 !             +-----------------+
1 !             |        0        |     End of item decsiptors
1 !             +-----------------+
1 !             |      str_1      |
1 !             +-----------------+
1 !             |      .....      |
1 !             +-----------------+
1 !             |      str_2      |
1 !             +-----------------+
1 !             |      .....      |
1 !             +-----------------+
1 !
1 !             1. This cluster-mount packet is to be sent to other nodes in
1 !                the cluster and processed by CSP (the Cluster Server Process).
1 !
1 !             2. The address in the item descriptor is "relocated" to be the
1 !                offset from the beginning of the packet (i.e. self-relative).
1 !                This is done so that when CSP gets the cluster-mount packet,
1 !                it can simply add the packet address to the offset in the
1 !                item desciptor to obtain the address of the string.
1 !
1 !-
1
2 BEGIN                                                    ! Start of MOUNT_ENCIPHER
2
2 MAP
2           ITEM_LIST       : REF BBLOCK,
2           BUFFER          : REF BBLOCK;
2
2 LOCAL
2           ITEM            : REF BBLOCK,                  ! Pointer to item descriptor
2           PTR             : REF BBLOCK,                  ! Pointer to output item desc
2           STR_PTR         : REF BBLOCK,                  ! Pointer to item string
2           ITEM_COUNT,                                    ! Number of items in item list
2           DEVICE_COUNT,                                  ! Device number index
2           J;
2
2 EXTERNAL
2           MOUNT_OPTIONS   : BITVECTOR VOLATILE; ! Parser option flags
2
2 MACRO ITEM_LENG  =  0,0,16,0%;                          ! Define buffer offsets
2 MACRO ITEM_CODE  =  2,0,16,0%;
```

```
2 MACRO ITEM_ADDR  =  4,0,32,0%;
2 MACRO ITEM_NULL  =  8,0,32,0%;
2 LITERAL ITEM_SIZE = 12;
2
2 !
2 ! Count of number of items in the item list
2 !
2
2 ITEM = .ITEM_LIST;                                   ! Point to the beginning of list
2 ITEM_COUNT = 0;                                      ! Initialize counter
2 WHILE ( .ITEM [ITEM_CODE] NEQ 0 ) DO
3 BEGIN
3     ITEM_COUNT = .ITEM_COUNT + 1;                    ! Increment number of items
3     ITEM = .ITEM + ITEM_SIZE;                        ! Bump item desciptor pointer
2 END;
2
2 !
2 ! Calculate space needed for the item descriptors
2 !
2 STR_PTR = .BUFFER + (.ITEM_COUNT * ITEM_SIZE);       ! Space needed for descriptors
2 STR_PTR [ITEM_CODE] = STR_PTR [ITEM_LENG] = 0;       ! Mark end of descriptor area
2 STR_PTR = .STR_PTR + 4;                              ! Mark beginning of string area
2 PTR = .BUFFER;                                       ! Mark beginning of desciptor area
2 ITEM = .ITEM_LIST;                                   ! Point to the beginning of item list
2 DEVICE_COUNT = 0;                                    ! Initialize device index
2 .LENGTH = 4;                                         ! Initialize length (itmlst stopper)
2
2 !
2 ! For each item in the item list, copy the item descriptor and the
2 ! item string to the output buffer
2 !
2 DECR J FROM .ITEM_COUNT TO 1 DO
3 BEGIN
3     SELECT .ITEM [ITEM_CODE] OF
3         SET
3
3         [MNT$_DEVNAM]   :
4                               BEGIN
4                               !
4                               ! For DEVNAM:
4                               !
4                               !    a. Create item descriptor, relocate address
4                               !    b. Compute length, return SS$_BUFFEROVF if appropriate
4                               !    c. Copy device string from physical device descriptor
4                               !
4                               BIND
4                                   DEV_DSC = PHYS_NAME [.DEVICE_COUNT * 2] : $BBLOCK;
4
4                               PTR [ITEM_LENG] = .DEV_DSC [DSC$W_LENGTH];
4                               PTR [ITEM_CODE] = MNT$_DEVNAM;
4                               PTR [ITEM_ADDR] = .STR_PTR - .BUFFER;
4                               .LENGTH = ..LENGTH + ITEM_SIZE + .PTR [ITEM_LENG];
4                               IF ..LENGTH GTRU .LIMIT
4                               THEN
4                                   RETURN SS$_BUFFEROVF;
4                               CH$COPY (.PTR [ITEM_LENG],
4                                        .DEV_DSC [DSC$A_POINTER],
4                                     0,
4                                     .PTR [ITEM_LENG],
4                                     .STR_PTR);
4                            DEVICE_COUNT = .DEVICE_COUNT + 1;
3                            END;
3
3       [MNT$_FLAGS]    :
4                            BEGIN
4                            !
4                            ! For FLAGS:
4                            !
4                            !    a. Create item descriptor, relocate address
4                            !    b. Compute length, return SS$_BUFFEROVF if appropriate
4                            !    c. Copy flags, clear NMT$V_CLUSTER bit, and
4                            !       set MNT$V_NOASSIST (disables operator assist)
4                            !
4                            PTR [ITEM_LENG] = .ITEM [ITEM_LENG];
4                            PTR [ITEM_CODE] = MNT$_FLAGS;
4                            PTR [ITEM_ADDR] = .STR_PTR - .BUFFER;
4                            .LENGTH = ..LENGTH + ITEM_SIZE + .PTR [ITEM_LENG];
4                            IF ..LENGTH GTRU .LIMIT
4                            THEN
4                                RETURN SS$_BUFFEROVF;
5                            BEGIN
5                                BIND
5                                    TEMP_PTR = .STR_PTR : BBLOCK;
5                                TEMP_PTR = ..ITEM [ITEM_ADDR];
5                                TEMP_PTR [MNT$V_CLUSTER] = 0;
5                                TEMP_PTR [MNT$V_NOASSIST] = 1;
5                                IF  NOT .MOUNT_OPTIONS [OPT_GROUP]
5                                THEN
5                                    TEMP_PTR [MNT$V_SYSTEM] = 1;
4                            END;
4
3                            END;
3
3       [OTHERWISE]     :
4                            BEGIN
4                            !
4                            ! All others:
4                            !
4                            !    a. Create item descriptor, relocate address
4                            !    b. Compute length, return SS$_BUFFEROVF if appropriate
4                            !    c. Copy item to output buffer
4                            !
4                            PTR [ITEM_LENG] = .ITEM [ITEM_LENG];
4                            PTR [ITEM_CODE] = .ITEM [ITEM_CODE];
4                            PTR [ITEM_ADDR] = .STR_PTR - .BUFFER;
4                            .LENGTH = ..LENGTH + ITEM_SIZE + .PTR [ITEM_LENG];
4                            IF ..LENGTH GTRU .LIMIT
4                            THEN
4                                RETURN SS$_BUFFEROVF;
4                            CH$COPY (.ITEM [ITEM_LENG],
4                                     .ITEM [ITEM_ADDR],
4                                     0,
4                                     .ITEM [ITEM_LENG],
4                                     .STR_PTR);
```

```
; 1995        2644  3                       END,
; 1996        2645  3
; 1997        2646  3           TES;
; 1998        2647  3
; 1999        2648  3       STR_PTR = .STR_PTR + .PTR [ITEM_LENG];          ! Bump string buffer pointer
; 2000        2649  3       ITEM = .ITEM + ITEM_SIZE;                       ! Bump item descriptor pointer
; 2001        2650  3       PTR = .PTR + ITEM_SIZE;                         ! Bump output descriptor pointer
; 2002        2651  3
; 2003        2652  2   END;                                                ! End of item list loop
; 2004        2653  2
; 2005        2654  2
; 2006        2655  2   RETURN 1;
; 2007        2656  1   END;                                                ! End of MOUNT_ENCIPHER





                                          0FFC 00000 MOUNT_ENCIPHER:
                                                           .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 2417
                        5E            08  C2 00002         SUBL2    #8, SP                                        ;
                        59       04   AC  D0 00005         MOVL     ITEM_LIST, ITEM                               ; 2538
                                      50  D4 00009         CLRL     ITEM_COUNT                                    ; 2539
                                 02   A9  B5 0000B 1$:     TSTW     2(ITEM)                                       ; 2540
                                      07  13 0000E         BEQL     2$                                            ;
                                      50  D6 00010         INCL     ITEM_COUNT                                    ; 2542
                        59            0C  C0 00012         ADDL2    #12, ITEM                                     ; 2543
                                      F4  11 00015         BRB      1$                                            ; 2540
              56        50            0C  C5 00017 2$:     MULL3    #12, ITEM_COUNT, R6                           ; 2549
              5B        56       0C   AC  C1 0001B         ADDL3    BUFFER, R6, STR_PTR                           ;
                                      8B  D4 00020         CLRL     (STR_PTR)+                                    ; 2550
                        58       0C   AC  D0 00022         MOVL     BUFFER, PTR                                   ; 2552
                        59       04   AC  D0 00026         MOVL     ITEM_LIST, ITEM                               ; 2553
                                      57  D4 0002A         CLRL     DEVICE_COUNT                                  ; 2554
                        5A       10   AC  D0 0002C         MOVL     LENGTH, R10                                   ; 2555
                        6A            04  D0 00030         MOVL     #4, (R10)                                     ;
                        56       01   A0  9E 00033         MOVAB    1(R0), J                                      ; 2561
                                    00B3  31 00037         BRW      9$                                            ;
                        6E       02   A9  3C 0003A 3$:     MOVZWL   2(ITEM), (SP)                                 ; 2563
                   04   AE            01  D0 0003E         MOVL     #1, 4(SP)                                     ;
                        01            6E  B1 00042         CMPW     (SP), #1                                      ; 2566
                                      33  12 00045         BNEQ     4$                                            ;
                                 04   AE  D4 00047         CLRL     4(SP)                                         ;
              50        57            01  78 0004A         ASHL     #1, DEVICE_COUNT, R0                          ; 2576
                        50 00000000'EF40  DE 0004E         MOVAL    PHYS_NAME[R0], R0                             ;
                        68            60  B0 00056         MOVW     (R0), (PTR)                                   ; 2578
                   02   A8            01  B0 00059         MOVW     #1, 2(PTR)                                    ; 2579
         04   A8        5B       0C   AC  C3 0005D         SUBL3    BUFFER, STR_PTR, 4(PTR)                       ; 2580
                        51            68  3C 00063         MOVZWL   (PTR), R1                                     ; 2581
                        51            6A  C0 00066         ADDL2    (R10), R1                                     ;
                        6A       0C   A1  9E 00069         MOVAB    12(R1), (R10)                                 ;
                   08   AC            6A  D1 0006D         CMPL     (R10), LIMIT                                  ; 2582
                                      63  1A 00071         BGTRU    6$                                            ;
              6B   04   B0            68  28 00073         MOVC3    (PTR), @4(R0), (STR_PTR)                      ; 2589
                                      57  D6 00078         INCL     DEVICE_COUNT                                  ; 2590
                   04   6E            B1  0007A 4$:        CMPW     (SP), #4                                      ; 2593
                                      36  12 0007D         BNEQ     5$                                            ;
                                    04    AE  D4 0007F          CLRL     4(SP)                                  ;
                          68              69  B0 00082          MOVW     (ITEM), (PTR)                          ; 2603
                   02     A8              04  B0 00085          MOVW     #4, 2(PTR)                             ; 2604
       04    A8           5B      0C      AC  C3 00089          SUBL3    BUFFER, STR_PTR, 4(PTR)                ; 2605
                          50              68  3C 0008F          MOVZWL   (PTR), R0                              ; 2606
                          50              6A  C0 00092          ADDL2    (R10), R0                              ;
                          6A      0C      A0  9E 00095          MOVAB    12(R0), (R10)                          ;
                   08     AC              6A  D1 00099          CMPL     (R10), LIMIT                           ; 2607
                                          37  1A 0009D          BGTRU    6$                                     ;
                          6B      04      B9  D0 0009F          MOVL     @4(ITEM), (STR_PTR)                    ; 2613
                   03     AB              10  8A 000A3          BICB2    #16, 3(STR_PTR)                        ; 2614
                          6B              04  88 000A7          BISB2    #4, (STR_PTR)                          ; 2615
                               0000G      CF  95 000AA          TSTB     MOUNT_OPTIONS                          ; 2616
                                          05  19 000AE          BLSS     5$                                     ;
                   01     AB      40      8F  88 000B0          BISB2    #64, 1(STR_PTR)                        ; 2618
                          28      04      AE  E9 000B5  5$:     BLBC     4(SP), 8$                              ; 2623
                          68              69  B0 000B9          MOVW     (ITEM), (PTR)                          ; 2632
                   02     A8              6E  B0 000BC          MOVW     (SP), 2(PTR)                           ; 2633
       04    A8           5B      0C      AC  C3 000C0          SUBL3    BUFFER, STR_PTR, 4(PTR)                ; 2634
                          50              68  3C 000C6          MOVZWL   (PTR), R0                              ; 2635
                          50              6A  C0 000C9          ADDL2    (R10), R0                              ;
                          6A      0C      A0  9E 000CC          MOVAB    12(R0), (R10)                          ;
                   08     AC              6A  D1 000D0          CMPL     (R10), LIMIT                           ; 2636
                                          06  1B 000D4          BLEQU    7$                                     ;
                          50    0601      8F  3C 000D6  6$:     MOVZWL   #1537, R0                              ; 2638
                                              04 000DB          RET                                             ;
             6B    04     B9              69  28 000DC  7$:     MOVC3    (ITEM), @4(ITEM), (STR_PTR)            ; 2643
                          50              88  3C 000E1  8$:     MOVZWL   (PTR)+, R0                             ; 2648
                          5B              50  C0 000E4          ADDL2    R0, STR_PTR                            ;
                          59              0C  C0 000E7          ADDL2    #12, ITEM                              ; 2649
                          58              0A  C0 000EA          ADDL2    #10, PTR                               ; 2650
                          02              56  F5 000ED  9$:     SOBGTR   J, 10$                                 ; 2561
                                          03  11 000F0          BRB      11$                                    ;
                                        FF45  31 000F2  10$:    BRW      3$                                     ;
                          50              01  D0 000F5  11$:    MOVL     #1, R0                                 ; 2655
                                              04 000F8          RET                                             ; 2656

; Routine Size:  249 bytes,    Routine Base:  $CODE$ + 0849


; 2008          2657  1
```

```

ROUTINE SEARCH_DEVICE (J) =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine searches the I/O database for a mountable device
!       and allocates the device, if it is not already allocated. If
!       the mount operation is a private mount, an EX mode lock will
!       be taken out. If the mount operation is a shared mount, a PW
!       mode lock will be taken out.
!
!       Note: this routine must be called in kernel mode.
!
! CALLING SEQUENCE:
!
!       SEARCH_DEVICE (ARG1)
!
! INPUT:
!
!       ARG1     : Index into device list.
!
! OUTPUT:
!
!       None.
!
! IMPLICIT INPUT:
!
!       Mount database.
!
! IMPLICIT OUTPUT:
!
!       The physical device name of the device will be set up in
!       the mount data base, with the appropriate device descriptor
!       set up in PHYS_NAME.
!
! ROUTINE VALUE:
!
!       Assorted status codes.
!
! SIDE EFFECTS:
!
!       None.
!
! NOTES:
!
!       To properly interlock the device in a cluster environment,
!       we must carefully take out the MOU$ interlock and the device
!       lock to serialize the mounts in the cluster without deadlocks.
!       Following is the algorithm used:
!
!       Step 0: Lock I/O database;
!               IOC$SEARCH (...);
!               If success
!                   then S0
!                   else F0
```

```
!
!       S0:     If device allocated
!               then
!                   set SS$_DEVALRALLOC
!               else
!                   mark UCB as allocated;
!
!       S1:     IOC$CVT_DEVNAM (...) to convert device name;
!
!       S2:     Unlock I/O database;
!
!       S3:     $GETDVIW to obtain allocation class name;
!
!       S4:     $ENQW MOU$ lock with LCK$M_NOQUEUE
!
!       S5:     If success
!                   then
!                       Exit loop;
!
!       S6:     If SS$_DEVALRALLOC
!                   then
!                       IOC$UNLOCK_DEV to dequeue device lock
!                   else
!                       IOC$DALLOC_DEV to deallocate and release device lock;
!
!       S7:     Wait delta time;
!
!       S8:     $ENQW MOU$ lock;
!
!       S9:     $DEQ MOU$ lock;
!
!       S10:    Goto step 0;
!
!
!       F0:     If ( not SS$_DEVALLOC )
!               or ( private mount )
!               or ( device_allocated )
!                   then
!                       Unlock I/O database;
!                       Exit loop;
!
!       F1:     IOC$CVT_DEVNAM (...) to convert alloc class device name;
!
!       F2:     Unlock I/O database;
!
!       F3:     Wait delta time;
!
!       F4:     $ENQW MOU$ lock;
!
!       F5:     Construct device lock;
!               $ENQW device in CR mode with NOQUEUE;
!
!       F6:     If failed
!                   then
!                       Exit loop;
!
!       F7:     $DEQ device lock;
```

```
1 !
1 !          F8:     $DEQ MOU$ lock;
1 !
1 !          F9:     Goto step 0;
1 !
1 !
1 !          Note:   This algorithm does not handle simultaneous /shared mounts
1 !                  from the same system. In this case, the first mounter will
1 !                  mount the device, and the second mounter will fail with an
1 !                  SS$_DEVALLOC error.  To  properly  serialize  simultaneous
1 !                  shared mounts,  another level of lock (the label lock) had
1 !                  to be added.
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2          SEARCH_FLAGS     : BBLOCK [4],              ! IOC$SEARCH routine flags
2          UCB              : REF BBLOCK,              ! Address of the UCB
2          STATUS,                                     ! Routine status
2          SEARCH_STATUS,                              ! Final success status
2          COUNTER,                                    ! Count number of iterations
2          DEVICE_ITMLST1   : BBLOCK [(1 * 12) + 4] INITIAL
2                             !
2                             ! item: allocation class plus device name
2                             !
2                            (WORD (NAMEBUF_LEN-4),
2                             WORD (DVI$_ALLDEVNAM),
2                             LONG (ALLDEVNAM_BUF+4),
2                             LONG (ALLDEVNAM_DESC),
2                             !
2                             ! end of list
2                             !
2                             LONG (0));
2
2 EXTERNAL
2          DEV_CTX          : BBLOCK FIELD (DC),      ! device value block context fields
2          MOUNT_OPTIONS    : BITVECTOR VOLATILE,     ! Parser option flags
2          DEVICE_STRING    : VECTOR VOLATILE,        ! Device name string descriptor
2          SCH$GL_CURPCB    : REF BBLOCK ADDRESSING_MODE (GENERAL); ! PCB address of current process
2
2 LINKAGE
2          IOC_SEARCH       = JSB (REGISTER = 1, REGISTER = 2, REGISTER = 3,
2                               REGISTER = 4; REGISTER = 1) :
2                               NOPRESERVE (2, 3, 5),
2
2          IOC_CVT_DEVNAM   = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 4,
2                               REGISTER = 5; REGISTER = 1) :
2                               PRESERVE (2,3,4,5,6,7),
2
2          IOC_LOCK_DEV     = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 4,
2                               REGISTER = 5) : NOPRESERVE (4, 5),
2
2          IOC_UNLOCK_DEV   = JSB (REGISTER = 5),
2
2          IOC_DALLOC_DEV   = JSB (REGISTER = 4, REGISTER = 5) :
```

```
                                NOPRESERVE (1,2,3,8),

        EXE_MAXACMODE       = JSB (REGISTER = 0) :
                                NOTUSED (2,3,4,5,6,7,8,9,10,11);

EXTERNAL ROUTINE
        LOCK_IODB           : ADDRESSING_MODE (GENERAL),
                                                ! Lock I/O database mutex
        UNLOCK_IODB         : ADDRESSING_MODE (GENERAL),
                                                ! Unlock the above
        IOC$SEARCH          : IOC_SEARCH ADDRESSING_MODE (GENERAL),
                                                ! Search I/O database for device
        IOC$CVT_DEVNAM      : IOC_CVT_DEVNAM ADDRESSING_MODE (GENERAL),
                                                ! Get fully expanded device name
        IOC$LOCK_DEV        : IOC_LOCK_DEV ADDRESSING_MODE (GENERAL),
                                                ! Take out the device lock
        IOC$UNLOCK_DEV      : IOC_UNLOCK_DEV ADDRESSING_MODE (GENERAL),
                                                ! Release the device lock
        IOC$DALLOC_DEV      : IOC_DALLOC_DEV ADDRESSING_MODE (GENERAL),
                                                ! Deallocate device and device lock
        EXE$MAXACMODE       : EXE_MAXACMODE ADDRESSING_MODE (GENERAL);
                                                ! Maximize access mode

!
! Rebind things to make life easier ( so we see them as their
! real logical units).
!
MAP
        DEVICE_STRING       : BBLOCKVECTOR [ DEVMAX, 8 ],
        NAME_BUFFER         : BBLOCKVECTOR [ DEVMAX, NAMEBUF_LEN ],
        PHYS_NAME           : BBLOCKVECTOR [ DEVMAX, 8 ];

!
! Start of buffer
!
MACRO   STADR   = 0,0,0,0%;

!
! Define descriptor vector displacements
!
MACRO   LEN     = 0,0,32,0%;
MACRO   ADDR    = 4,0,32,0%;
MACRO   ILEN    = 8,0,32,0%;                    ! Item list returned length position.

LITERAL
        RETRY_LIMIT = 1000;                     ! Define retry limit

SEARCH_FLAGS [0,0,32,0] = 0;                    ! Initialize search flags
SEARCH_FLAGS [IOC$V_MOUNT] = 1;                 ! Set flag to indicate searching for a mountable device

!
! If this is a private mount, set flag to take out an exclusive lock on
! the device.
!
IF .MOUNT_OPTIONS [OPT_NOSHARE]
THEN
    SEARCH_FLAGS [IOC$V_ALLOC] = 1;
```

```
  2
  2 COUNTER = 0;                                          ! Initialize counter
  2
  2 WHILE  (1) DO                                         ! Forever do block
  2
  3 BEGIN
  3
  3 LOCK_IODB ();                                         ! Lock I/O database
  3 STATUS = IOC$SEARCH ( DEVICE_STRING [.J, LEN],        ! Search device with proper flags
  3                       .SEARCH_FLAGS,
  3                       DEV_CTX,                        ! Return lock value block
  3                       .SCH$GL_CURPCB;
  3                       UCB );                          ! Target device UCB
  3
  3 IF .STATUS
  3 THEN
  3
  4     BEGIN                                             ! IOC$SEARCH succeeded
  4     !
  4     ! If the device is not already allocated, allocate the device by
  4     ! setting up the proper status in the I/O database.
  4     !
  4     IF  NOT .BBLOCK [UCB [UCB$L_DEVCHAR], DEV$V_ALL]
  4     THEN
  5         BEGIN
  5         UCB [UCB$B_AMOD] = EXE$MAXACMODE (.CALLERS_ACMOD); ! Set access mode
  5         BBLOCK [UCB [UCB$L_DEVCHAR], DEV$V_ALL] = 1;   ! Set the device as allocated
  5         UCB [UCB$W_REFC] = .UCB [UCB$W_REFC] + 1;      ! Bump reference count
  5         UCB [UCB$L_PID] = .SCH$GL_CURPCB [PCB$L_PID]; ! Set device owner
  5         SEARCH_STATUS = SS$_NORMAL;                   ! Set normal return status
  5         END
  4     ELSE
  4         SEARCH_STATUS = SS$_DEVALRALLOC;              ! Set proper return status
  4
  4     !
  4     ! Set up physical device name in mount database (also set up the device
  4     ! descriptor).
  4     !
  4     IOC$CVT_DEVNAM ( NAMEBUF_LEN,                     ! Output buffer length
  4                      NAME_BUFFER [.J, STADR],         ! Output buffer address
  4                      -1,                              ! Format device name
  4                      .UCB;                            ! Address of UCB
  4                      PHYS_NAME [.J, LEN] );           ! Returned length of device name
  4
  4     PHYS_NAME [.J, ADDR] = NAME_BUFFER [.J, STADR]; ! Set up device descriptor
  4
  4     UNLOCK_IODB ();                                   ! Unlock I/O database
  4
P 4     $GETDVIW ( DEVNAM = PHYS_NAME [.J, LEN],          ! Target device descriptor
P 4                ITMLST = DEVICE_ITMLST1,               ! Item list
  4                EFN    = MOUNT_EFN );                  !
  4     ALLDEVNAM_DESC [0] = .ALLDEVNAM_DESC [0] + 4; ! Fix up length to include MOU$
  4
  4     !
  4     ! Take out a lock on the allocation class device name.  This will
  4     ! interlock all mounts of this device.
  4     !
```

```
        STATUS = $ENQW (LKMODE = LCK$K_EXMODE,
                        LKSB = LOCK_STATUS,
                        FLAGS = (LCK$M_SYSTEM OR LCK$M_NOQUEUE),
                        RESNAM = ALLDEVNAM_DESC,
                        EFN = MOUNT_EFN,
                        ACMODE = PSL$C_EXEC);
        IF .STATUS
        THEN
            BEGIN                                       ! MOU$ interlock granted
            STATUS = .SEARCH_STATUS;                    ! Return proper status code
            EXITLOOP;                                   ! Get out of the loop
            END                                         ! End of MOU$ success story
        ELSE
            BEGIN                                       ! MOU$ interlock failed
            LOCK_IODB ();                               ! Lock I/O database
            IF .SEARCH_STATUS EQL SS$_DEVALRALLOC
            THEN
                IOC$UNLOCK_DEV (.UCB)                   ! Release device lock
            ELSE
                IOC$DALLOC_DEV (.SCH$GL_CURPCB, .UCB); ! Deallocate device and
                                                        ! release device lock
            UNLOCK_IODB ();                             ! Unlock I/O database

            WAIT_DELTA (.COUNTER);                      ! Wait a while

            $ENQW (LKMODE = LCK$K_EXMODE,               ! Enqueue MOU$ lock again
                   LKSB = LOCK_STATUS,
                   FLAGS = LCK$M_SYSTEM,
                   RESNAM = ALLDEVNAM_DESC,
                   EFN = MOUNT_EFN,
                   ACMODE = PSL$C_EXEC);
            $DEQ ( LKID = .LOCK_STATUS [1] );           ! Dequeue MOU$ lock

            END                                         ! End of MOU$ failure block

        END                                             ! End of IOC$SEACH success block

    ELSE

        BEGIN                                           ! IOC$SEARCH failure block

        IF ( .STATUS NEQ SS$_DEVALLOC )                 ! If not SS$_DEVALLOC
        OR ( .MOUNT_OPTIONS [OPT_NOSHARE] )             ! or this is a private mount
        OR ( .BBLOCK [ UCB [UCB$L_DEVCHAR], DEV$V_ALL ] ) ! or an allocated device
        THEN                                            ! Get out
            BEGIN
            UNLOCK_IODB ();
            EXITLOOP;
            END;

        !
        ! We have a valid UCB address, get the allocation device name to
        ! derive the MOU$ interlock.
        !
        IOC$CVT_DEVNAM ( NAMEBUF_LEN-4,                 ! Output buffer length
                         ALLDEVNAM_BUF+4,               ! Output buffer address
                         1,                             ! Format allocation class device name
```

```
   4                        .UCB;                         ! Address of UCB
   4                        ALLDEVNAM_DESC [0]);          ! Returned length of device name
   4      ALLDEVNAM_DESC [0] = .ALLDEVNAM_DESC [0] + 4;   ! Fix up length to include MOU$
   4
   4      UNLOCK_IODB ();                                 ! Unlock I/O database
   4
   4      WAIT_DELTA (.COUNTER);                          ! Wait a while
   4
   4      !
   4      ! Take out a lock on the allocation class device name.
   4      !
P  4      $ENQW (LKMODE = LCK$K_EXMODE,
P  4                LKSB = LOCK_STATUS,
P  4               FLAGS = LCK$M_SYSTEM,
P  4              RESNAM = ALLDEVNAM_DESC,
P  4                 EFN = MOUNT_EFN,
   4              ACMODE = PSL$C_EXEC);
   4
   4      !
   4      ! Construct the device lock name and take out the device lock
   4      ! in CR mode with NOQUEUE.
   4      !
   5      BEGIN
   5
   5      LOCAL
   5          DEVLCKNAM_BUF    : VECTOR [NAMEBUF_LEN, BYTE]
   5                             INITIAL (BYTE('SYS$', REP NAMEBUF_LEN-4 OF (' '))),
   5          DEVLCKNAM_DESC   : VECTOR [2, LONG]
   5                             INITIAL (0, DEVLCKNAM_BUF),
   5          DEVLCK_STS       : VECTOR [2, LONG];
   5
   5      DEVLCKNAM_DESC [0] = .ALLDEVNAM_DESC [0];   ! Set up device lock descriptor
   5
   5      CH$COPY (   .ALLDEVNAM_DESC [0] - 4,          ! Length of input string
   5                  .ALLDEVNAM_DESC [1] + 4,          ! Start of alloc name string
   5                  0,                                !
   5                  .ALLDEVNAM_DESC [0] - 4,          ! Length of output string
   5                  .DEVLCKNAM_DESC [1] + 4 );        ! Start of target string
   5
P  5      STATUS = $ENQW  (LKMODE = LCK$K_CRMODE,      ! Enqueue device lock in CR mode
P  5                         LKSB = DEVLCK_STS,         ! Lock status block
P  5                        FLAGS = (LCK$M_SYSTEM OR LCK$M_NOQUEUE), ! Return if not available
P  5                       RESNAM = DEVLCKNAM_DESC,     ! Device lock
   5                          EFN = MOUNT_EFN);
   5
   5      IF .STATUS
   5      THEN
   5          !
   5          ! Device lock in CR mode granted. This implies that the device
   5          ! is not allocated. Release both locks and try again.
   5          !
   6          BEGIN
   6          $DEQ ( LKID = .DEVLCK_STS [1] );          ! Release device lock
   6          $DEQ ( LKID = .LOCK_STATUS [1] );         ! Release MOU$ lock
   6          END
   5      ELSE
   5          !
```

```
5            ! Device lock in CR mode is not granted. This will happen if the
5            ! lock is already taken out in EX mode, i.e. the device is allocated.
5            ! Get out with an SS$_DEVALLOC status.
5            !
6           BEGIN
6           STATUS = SS$_DEVALLOC;                             ! Set return code
6           EXITLOOP;                                          ! Get out
5           END;
5
4       END;                                                   ! End of block defining DEVLCK
4
3       END;                                                   ! End of IOC$SEARCH failure block
3
3        !
3        ! Do a sanity check on how many times we have gone thru this loop. If
3        ! too many times, give up with an error.
3        !
3       COUNTER = .COUNTER + 1;                                ! Update counter
3       IF .COUNTER GEQ RETRY_LIMIT                            ! If loop thru too many times
3       THEN                                                   ! give up with an error
4           BEGIN
4           STATUS = SS$_DEVNOTMOUNT;
4           EXITLOOP;
3           END;
3
2   END;                                                       ! End of forever block
2
2
2   IF NOT .STATUS
2   THEN                                                       ! If SEARCH_DEVICE failed
3       BEGIN
3
3       LOCAL
3           ITMLST2             : BBLOCK [(1 * 12) + 4] INITIAL
3                                 !
3                                 ! item: device name
3                                 !
3                                (WORD (NAMEBUF_LEN),     ! Device name buffer length
3                                 WORD (DVI$_DEVNAM),     ! Device name item code
3                                 LONG (0),               ! Device name buffer address
3                                 LONG (0),               ! Returned device name length
3                                 !
3                                 ! end of list
3                                 !
3                                 LONG (0)),
3           LOC_STATUS;                                        ! Local status work
3
3        !
3        ! The IOC$SEARCH routine failed, use input device string to get the
3        ! device name. Also set up the device descriptor. This is necessary
3        ! so Operator Assist can output the message with a device name. If
3        ! the $GETDVI failed, we've got some real problems, return the status
3        ! as the status of routine SEARCH_DEVICE.
3        !
3       ITMLST2 [ADDR] = NAME_BUFFER [.J, STADR];    ! Set up device buffer address
3       ITMLST2 [ILEN] = PHYS_NAME [.J, LEN];        ! Set returned length
3       PHYS_NAME [.J, ADDR] = NAME_BUFFER [.J, STADR]; ! Set up descriptor
```

```
; 2466          3114 3
; 2467        P 3115 3      LOC_STATUS = $GETDVIW (DEVNAM = DEVICE_STRING [.J, LEN], ! Target device descriptor
; 2468        P 3116 3                             ITMLST = ITMLST2,               ! Item list
; 2469          3117 3                             EFN    = MOUNT_EFN );           !
; 2470          3118 3
; 2471          3119 3      IF NOT .LOC_STATUS                                     ! If we can't even get the device name
; 2472          3120 3      THEN                                                   ! Return the status from $GETDVI
; 2473          3121 3          STATUS = .LOC_STATUS
; 2474          3122 3
; 2475          3123 2      END;                                                   ! End of SEARCH_DEVICE failure block
; 2476          3124 2
; 2477          3125 2  RETURN .STATUS;                                            ! Return status
; 2478          3126 1  END;                                                       ! End of routine SEARCH_DEVICE


                                                                   .PSECT  $PLIT$,NOWRT,NOEXE,2

                                        001C  00020 P.AAC:   .WORD    28
                                        00EC  00022          .WORD    236
                                   00000000'  00024          .ADDRESS ALLDEVNAM_BUF+4
                                   00000000'  00028          .ADDRESS ALLDEVNAM_DESC
                                    00000000  0002C          .LONG    0
                             24  53  59  53  00030 P.AAD:   .ASCII   \SYS$\
                                          20  00034          .ASCII   \ \
                                          20  00035          .ASCII   \ \
                                          20  00036          .ASCII   \ \
                                          20  00037          .ASCII   \ \
                                          20  00038          .ASCII   \ \
                                          20  00039          .ASCII   \ \
                                          20  0003A          .ASCII   \ \
                                          20  0003B          .ASCII   \ \
                                          20  0003C          .ASCII   \ \
                                          20  0003D          .ASCII   \ \
                                          20  0003E          .ASCII   \ \
                                          20  0003F          .ASCII   \ \
                                          20  00040          .ASCII   \ \
                                          20  00041          .ASCII   \ \
                                          20  00042          .ASCII   \ \
                                          20  00043          .ASCII   \ \
                                          20  00044          .ASCII   \ \
                                          20  00045          .ASCII   \ \
                                          20  00046          .ASCII   \ \
                                          20  00047          .ASCII   \ \
                                          20  00048          .ASCII   \ \
                                          20  00049          .ASCII   \ \
                                          20  0004A          .ASCII   \ \
                                          20  0004B          .ASCII   \ \
                                          20  0004C          .ASCII   \ \
                                          20  0004D          .ASCII   \ \
                                          20  0004E          .ASCII   \ \
                                          20  0004F          .ASCII   \ \
                                        0020  00050 P.AAE:   .WORD    32
                                        0020  00052          .WORD    32
                                    00000000  00054          .LONG    0
                                    00000000  00058          .LONG    0
                                    00000000  0005C          .LONG    0
```

```
                                                    .EXTRN  LOCK_IODB, UNLOCK_IODB
                                                    .EXTRN  IOC$SEARCH, IOC$CVT_DEVNAM
                                                    .EXTRN  IOC$LOCK_DEV, IOC$UNLOCK_DEV
                                                    .EXTRN  IOC$DALLOC_DEV, EXE$MAXACMODE
                                                    .EXTRN  SYS$GETDVIW

                                                    .PSECT  $CODE$,NOWRT,2

                                  0FFC 00000 SEARCH_DEVICE:
                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11              ; 2659
                  5E        B8    AE  9E 00002      MOVAB   -72(SP), SP                                       ;
38  AE 00000000'  EF              10  28 00006      MOVC3   #16, P.AAC, DEVICE_ITMLST1                        ; 2806
                                  6E  D4 0000F      CLRL    SEARCH_FLAGS                                      ; 2876
                  6E        80    8F  88 00011      BISB2   #128, SEARCH_FLAGS                                ; 2877
04         0000G  CF              04  E1 00015      BBC     #4, MOUNT_OPTIONS, 1$                             ; 2883
              01  AE              04  88 0001B      BISB2   #4, SEARCH_FLAGS+1                                ; 2885
                                  59  D4 0001F 1$:  CLRL    COUNTER                                           ; 2887
                  57        04    AC  D0 00021      MOVL    J, R7                                             ; 2894
              04  AE      0000GCF47   7E 00025      MOVAQ   DEVICE_STRING[R7], 4(SP)                          ;
       00000000G  00              00  FB 0002C 2$:  CALLS   #0, LOCK_IODB                                     ; 2893
                  53     0000G    CF  9E 00033      MOVAB   DEV_CTX, R3                                       ; 2894
                  54 00000000G    00  D0 00038      MOVL    SCH$GL_CURPCB, R4                                 ;
                  52              6E  D0 0003F      MOVL    SEARCH_FLAGS, R2                                  ;
                  51        04    AE  D0 00042      MOVL    4(SP), R1                                         ;
                     00000000G    00  16 00046      JSB     IOC$SEARCH                                        ;
                  5A              50  D0 0004C      MOVL    R0, STATUS                                        ;
                  56              51  D0 0004F      MOVL    R1, R6                                            ;
                  03              5A  E8 00052      BLBS    STATUS, 3$                                        ; 2900
                                0117  31 00055      BRW     9$                                                ;
                            3A    A6  95 00058 3$:  TSTB    58(UCB)                                           ; 2908
                                  2A  19 0005B      BLSS    4$                                                ;
                  50 00000000'    EF  D0 0005D      MOVL    CALLERS_ACMOD, R0                                 ; 2911
                     00000000G    00  16 00064      JSB     EXE$MAXACMODE                                     ;
              5F  A6              50  90 0006A      MOVB    R0, 95(UCB)                                       ;
              3A  A6        80    8F  88 0006E      BISB2   #128, 58(UCB)                                     ; 2912
                            5C    A6  B6 00073      INCW    92(UCB)                                           ; 2913
                  50 00000000G    00  D0 00076      MOVL    SCH$GL_CURPCB, R0                                 ; 2914
              2C  A6        60    A0  D0 0007D      MOVL    96(R0), 44(UCB)                                   ;
                  5B              01  D0 00082      MOVL    #1, SEARCH_STATUS                                 ; 2915
                                  05  11 00085      BRB     5$                                                ; 2908
                  5B      0641    8F  3C 00087 4$:  MOVZWL  #1601, SEARCH_STATUS                              ; 2918
52                57              05  78 0008C 5$:  ASHL    #5, R7, R2                                        ; 2925
                  51 00000000'EF42    9E 00090      MOVAB   NAME_BUFFER[R2], R1                               ;
                  55              56  D0 00098      MOVL    UCB, R5                                           ; 2928
                  54              01  CE 0009B      MNEGL   #1, R4                                            ;
                  50              20  D0 0009E      MOVL    #32, R0                                           ;
                     00000000G    00  16 000A1      JSB     IOC$CVT_DEVNAM                                    ;
                     00000000'EF47    7F 000A7      PUSHAQ  PHYS_NAME[R7]                                     ;
                  9E              51  D0 000AE      MOVL    R1, @(SP)+                                        ;
                     00000000'EF47    7F 000B1      PUSHAQ  PHYS_NAME+4[R7]                                   ; 2930
                  9E 00000000'EF42    9E 000B8      MOVAB   NAME_BUFFER[R2], @(SP)+                           ;
       00000000G  00              00  FB 000C0      CALLS   #0, UNLOCK_IODB                                   ; 2932
                                  7E  7C 000C7      CLRQ    -(SP)                                             ; 2936
                                  7E  7C 000C9      CLRQ    -(SP)                                             ;
                            48    AE  9F 000CB      PUSHAB  DEVICE_ITMLST1                                    ;
                     00000000'EF47    7F 000CE      PUSHAQ  PHYS_NAME[R7]                                     ;
```

```
                          7E               1A  7D 000D5            MOVQ    #26, -(SP)                                   ;
              00000000G   00               08  FB 000D8            CALLS   #8, SYS$GETDVIW                              ;
              00000000'   EF               04  C0 000DF            ADDL2   #4, ALLDEVNAM_DESC                           ; 2937
                          7E               01  7D 000E6            MOVQ    #1, -(SP)                                    ; 2948
                                           7E  7C 000E9            CLRQ    -(SP)                                        ;
                                           7E  7C 000EB            CLRQ    -(SP)                                        ;
                              00000000'    EF  9F 000ED            PUSHAB  ALLDEVNAM_DESC                               ;
                                           14  DD 000F3            PUSHL   #20                                          ;
                              00000000'    EF  9F 000F5            PUSHAB  LOCK_STATUS                                  ;
                                           05  DD 000FB            PUSHL   #5                                           ;
                                           1A  DD 000FD            PUSHL   #26                                          ;
              00000000G   00               0B  FB 000FF            CALLS   #11, SYS$ENQW                                ;
                          5A               50  D0 00106            MOVL    R0, STATUS                                   ;
                          05               5A  E9 00109            BLBC    STATUS, 6$                                   ; 2949
                          5A               5B  D0 0010C            MOVL    SEARCH_STATUS, STATUS                        ; 2952
                                           79  11 0010F            BRB     11$                                          ; 2951
              00000000G   00               00  FB 00111  6$:       CALLS   #0, LOCK_IODB                                ; 2957
              00000641    8F               5B  D1 00118            CMPL    SEARCH_STATUS, #1601                         ; 2958
                                           0B  12 0011F            BNEQ    7$                                           ;
                          55               56  D0 00121            MOVL    UCB, R5                                      ; 2960
                              00000000G    00  16 00124            JSB     IOC$UNLOCK_DEV                               ;
                                           10  11 0012A            BRB     8$                                           ;
                          55               56  D0 0012C  7$:       MOVL    UCB, R5                                      ; 2962
                          54  00000000G    00  D0 0012F            MOVL    SCH$GL_CURPCB, R4                            ;
                              00000000G    00  16 00136            JSB     IOC$DALLOC_DEV                               ;
              00000000G   00               00  FB 0013C  8$:       CALLS   #0, UNLOCK_IODB                              ; 2964
                                           59  DD 00143            PUSHL   COUNTER                                      ; 2966
              00000000V   EF               01  FB 00145            CALLS   #1, WAIT_DELTA                               ;
                          7E               01  7D 0014C            MOVQ    #1, -(SP)                                    ; 2973
                                           7E  7C 0014F            CLRQ    -(SP)                                        ;
                                           7E  7C 00151            CLRQ    -(SP)                                        ;
                              00000000'    EF  9F 00153            PUSHAB  ALLDEVNAM_DESC                               ;
                                           10  DD 00159            PUSHL   #16                                          ;
                              00000000'    EF  9F 0015B            PUSHAB  LOCK_STATUS                                  ;
                                           05  DD 00161            PUSHL   #5                                           ;
                                           1A  DD 00163            PUSHL   #26                                          ;
              00000000G   00               0B  FB 00165            CALLS   #11, SYS$ENQW                                ;
                                         00D1  31 0016C            BRW     13$                                          ; 2974
              00000840    8F               5A  D1 0016F  9$:       CMPL    STATUS, #2112                                ; 2984
                                           0B  12 00176            BNEQ    10$                                          ;
        05        0000G   CF               04  E0 00178            BBS     #4, MOUNT_OPTIONS, 10$                       ; 2985
                                      3A   A6  95 0017E            TSTB    58(UCB)                                      ; 2986
                                           0A  18 00181            BGEQ    12$                                          ;
              00000000G   00               00  FB 00183  10$:      CALLS   #0, UNLOCK_IODB                              ; 2989
                                         00DF  31 0018A  11$:      BRW     17$                                          ; 2988
                          51  00000000'    EF  9E 0018D  12$:      MOVAB   ALLDEVNAM_BUF+4, R1                          ; 2998
                          55               56  D0 00194            MOVL    UCB, R5                                      ; 2997
                          54               01  D0 00197            MOVL    #1, R4                                       ;
                          50               1C  D0 0019A            MOVL    #28, R0                                      ;
                              00000000G    00  16 0019D            JSB     IOC$CVT_DEVNAM                               ;
              00000000'   EF               51  D0 001A3            MOVL    R1, ALLDEVNAM_DESC                           ; 3001
              00000000'   EF               04  C0 001AA            ADDL2   #4, ALLDEVNAM_DESC                           ; 3002
              00000000G   00               00  FB 001B1            CALLS   #0, UNLOCK_IODB                              ; 3004
                                           59  DD 001B8            PUSHL   COUNTER                                      ; 3006
              00000000V   EF               01  FB 001BA            CALLS   #1, WAIT_DELTA                               ;
                          7E               01  7D 001C1            MOVQ    #1, -(SP)                                    ; 3016
                                           7E  7C 001C4            CLRQ    -(SP)                                        ;
```

```
                                                          J 9
VMOUNT                                             16-Sep-1984 01:00:56    VAX-11 Bliss-32 V4.0-742                    Page  72
V04-002                                            12-Sep-1984 11:14:53    DISK$VMSMASTER:[MOUNT.SRC]VMOUNT.B32;3       (15)

                                        7E   7C 001C6           CLRQ    -(SP)                                         ;
                          00000000'     EF   9F 001C8           PUSHAB  ALLDEVNAM_DESC                                ;
                                        10   DD 001CE           PUSHL   #16                                           ;
                          00000000'     EF   9F 001D0           PUSHAB  LOCK_STATUS                                   ;
                                        05   DD 001D6           PUSHL   #5                                            ;
                                        1A   DD 001D8           PUSHL   #26                                           ;
               00000000G  00            0B   FB 001DA           CALLS   #11, SYS$ENQW                                 ;
      18   AE  00000000'  EF            20   28 001E1           MOVC3   #32, P.AAD, DEVLCKNAM_BUF                     ; 3026
                                   10   AE   D4 001EA           CLRL    DEVLCKNAM_DESC                                ;
                   14     AE       18   AE   9E 001ED           MOVAB   DEVLCKNAM_BUF, DEVLCKNAM_DESC+4               ;
                   10     AE  00000000'  EF   D0 001F2           MOVL    ALLDEVNAM_DESC, DEVLCKNAM_DESC                ; 3031
           52  00000000'  EF            04   C3 001FA           SUBL3   #4, ALLDEVNAM_DESC, R2                        ; 3033
                   51  00000000'     EF   D0 00202           MOVL    ALLDEVNAM_DESC+4, R1                          ; 3034
                   50          14    AE   D0 00209           MOVL    DEVLCKNAM_DESC+4, R0                          ; 3037
      04   A0           04    A1        52   28 0020D           MOVC3   R2, 4(R1), 4(R0)                              ;
                                        7E   7C 00213           CLRQ    -(SP)                                         ; 3043
                                        7E   7C 00215           CLRQ    -(SP)                                         ;
                                        7E   7C 00217           CLRQ    -(SP)                                         ;
                                   28   AE   9F 00219           PUSHAB  DEVLCKNAM_DESC                                ;
                                        14   DD 0021C           PUSHL   #20                                           ;
                                   28   AE   9F 0021E           PUSHAB  DEVLCK_STS                                    ;
                                        01   DD 00221           PUSHL   #1                                            ;
                                        1A   DD 00223           PUSHL   #26                                           ;
               00000000G  00            0B   FB 00225           CALLS   #11, SYS$ENQW                                 ;
                          5A            50   D0 0022C           MOVL    R0, STATUS                                    ;
                          21            5A   E9 0022F           BLBC    STATUS, 14$                                   ; 3045
                                        7E   7C 00232           CLRQ    -(SP)                                         ; 3052
                                        7E   D4 00234           CLRL    -(SP)                                         ;
                                   18   AE   DD 00236           PUSHL   DEVLCK_STS+4                                  ;
               00000000G  00            04   FB 00239           CALLS   #4, SYS$DEQ                                   ;
                                        7E   7C 00240  13$:     CLRQ    -(SP)                                         ; 3053
                                        7E   D4 00242           CLRL    -(SP)                                         ;
                          00000000'     EF   DD 00244           PUSHL   LOCK_STATUS+4                                 ;
               00000000G  00            04   FB 0024A           CALLS   #4, SYS$DEQ                                   ;
                                        07   11 00251           BRB     15$                                           ; 3045
                          5A     0840   8F   3C 00253  14$:     MOVZWL  #2112, STATUS                                 ; 3062
                                        12   11 00258           BRB     17$                                           ; 3061
                                        59   D6 0025A  15$:     INCL    COUNTER                                       ; 3074
               000003E8   8F            59   D1 0025C           CMPL    COUNTER, #1000                                ; 3075
                                        03   18 00263           BGEQ    16$                                           ;
                                      FDC4   31 00265           BRW     2$                                            ;
                          5A       7C   8F   9A 00268  16$:     MOVZBL  #124, STATUS                                  ; 3078
                          46            5A   E8 0026C  17$:     BLBS    STATUS, 18$                                   ; 3085
      28   AE  00000000'  EF            10   28 0026F           MOVC3   #16, P.AAE, ITMLST2                           ; 3101
           52             57            05   78 00278           ASHL    #5, R7, R2                                    ; 3111
                          50  00000000'EF42   9E 0027C           MOVAB   NAME_BUFFER[R2], R0                           ;
                   2C     AE            50   D0 00284           MOVL    R0, ITMLST2+4                                 ;
                   30     AE  00000000'EF47   7E 00288           MOVAQ   PHYS_NAME[R7], ITMLST2+8                      ; 3112
                              00000000'EF47   7F 00291           PUSHAQ  PHYS_NAME+4[R7]                               ; 3113
                          9E            50   D0 00298           MOVL    R0, @(SP)+                                    ;
                                        7E   7C 0029B           CLRQ    -(SP)                                         ; 3117
                                        7E   7C 0029D           CLRQ    -(SP)                                         ;
                                   38   AE   9F 0029F           PUSHAB  ITMLST2                                       ;
                                   18   AE   DD 002A2           PUSHL   24(SP)                                        ;
                          7E            1A   7D 002A5           MOVQ    #26, -(SP)                                    ;
               00000000G  00            08   FB 002A8           CALLS   #8, SYS$GETDVIW                               ;
                          03            50   E8 002AF           BLBS    LOC_STATUS, 18$                               ; 3119
                          5A           50  D0 002B2          MOVL    LOC_STATUS, STATUS                          ; 3121
                          50           5A  D0 002B5 18$:     MOVL    STATUS, R0                                  ; 3125
                                           04 002B8          RET                                                 ; 3126

; Routine Size:  697 bytes,    Routine Base:  $CODE$ + 0942


; 2479          3127  1
; 2480          3128  1
; 2482          3129  1 GLOBAL ROUTINE DEQ_MOUNT_LOCK : NOVALUE =
; 2483          3130  1
; 2484          3131  1 !++
; 2485          3132  1 !
; 2486          3133  1 ! FUNCTIONAL DESCRIPTION:
; 2487          3134  1 !
; 2488          3135  1 !       This routine dequeus the mount interlock, if it exists.
; 2489          3136  1 !
; 2490          3137  1 ! CALLING SEQUENCE:
; 2491          3138  1 !
; 2492          3139  1 !       KERNEL_CALL ( DEQ_MOUNT_LOCK );
; 2493          3140  1 !
; 2494          3141  1 !       This routine is called in kernel mode because the mount interlock
; 2495          3142  1 !       is taken out in kernel mode.
; 2496          3143  1 !
; 2497          3144  1 ! INPUT:
; 2498          3145  1 !
; 2499          3146  1 !       None.
; 2500          3147  1 !
; 2501          3148  1 ! OUTPUT:
; 2502          3149  1 !
; 2503          3150  1 !       None.
; 2504          3151  1 !
; 2505          3152  1 ! IMPLICIT INPUT:
; 2506          3153  1 !
; 2507          3154  1 !       LOCK_STATUS     : Lock status block of the mount interlock
; 2508          3155  1 !
; 2509          3156  1 ! IMPLICIT OUTPUT:
; 2510          3157  1 !
; 2511          3158  1 !       None.
; 2512          3159  1 !
; 2513          3160  1 ! ROUTINE VALUE:
; 2514          3161  1 !
; 2515          3162  1 !       None.
; 2516          3163  1 !
; 2517          3164  1 ! SIDE EFFECTS:
; 2518          3165  1 !
; 2519          3166  1 !       Mount interlock released.
; 2520          3167  1 !
; 2521          3168  1 !--
; 2522          3169  1
; 2523          3170  2 BEGIN
; 2524          3171  2
; 2525          3172  2 IF .LOCK_STATUS[1] NEQ 0                      ! If mount lock exists,
; 2526          3173  2     THEN $DEQ (LKID = .LOCK_STATUS[1]);        ! Release it
; 2527          3174  2
; 2528          3175  2
; 2529          3176  2 RETURN;                                        ! Back to caller
; 2530          3177  1 END;                                           ! End of routine DEQ_MOUNT_LOCK



                                    0000 00000          .ENTRY  DEQ_MOUNT_LOCK, Save nothing                    ; 3129
                    50 00000000'  EF  D0 00002          MOVL    LOCK_STATUS+4, R0                               ; 3172
                                  0D  13 00009          BEQL    1$                                              ;
```

```
                              7E  7C 0000B           CLRQ     -(SP)                                      ; 3173
                              7E  D4 0000D           CLRL     -(SP)                                      ;
                              50  DD 0000F           PUSHL    R0                                         ;
          00000000G  00       04  FB 00011           CALLS    #4, SYS$DEQ                                ;
                                  04 00018 1$:       RET                                                 ; 3177

; Routine Size:  25 bytes,    Routine Base:  $CODE$ + 0BFB


; 2531          3178  1
```

```
1
1 ROUTINE WAIT_DELTA (N) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine goes into the waitfor state for a small period of
1 !       time. This wait period is introduced to give simultaneous mounts
1 !       a chance to get both the device lock and the mount interlock.
1 !       The amount of time spent in waitfor state is node-dependent (e.g.
1 !       based on the scssystemid). The wait time also varies from one
1 !       call to the next (with a small positive or negative bias).
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       WAIT_DELTA (ARG1)
1 !
1 ! INPUT:
1 !
1 !       ARG1 :  Number of times this routine has been called.
1 !
1 ! OUTPUT:
1 !
1 !       None.
1 !
1 ! IMPLICIT INPUT:
1 !
1 !       None.
1 !
1 ! IMPLICIT OUTPUT:
1 !
1 !       None.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       None.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None.
1 !
1 !--
1
2 BEGIN
2
2 OWN
2       SCSSYSID,
2       XDELTA,
2       BIAS,
2       GETS_ITMLST     : BLOCK  [(1*12)+4, BYTE] INITIAL
2                       ( WORD (4),
2                         WORD (SYI$_SCSSYSTEMID),
2                         LONG (SCSSYSID),
2                         LONG (0),
2                         LONG (0));
2 LOCAL
```

```
  2         STATUS,
  2         DELTA             : VECTOR [2,LONG] INITIAL (-1,-1);
  2
  2 !
  2 ! Set up some initial values for the first call to this routine.
  2 !
  2 IF .N EQL 0
  2 THEN
  3     BEGIN
  3
  3     SCSSYSID = 0;
  3
P 3     STATUS = $GETSYIW ( EFN    = MOUNT_EFN,
  3                         ITMLST = GETS_ITMLST );
  3
  3     !
  3     ! If the $GETSYI failed or scssystemid is zero, use a default value.
  3     !
  3
  3     IF NOT .STATUS
  3     OR .SCSSYSID EQL 0
  3     THEN
  3         SCSSYSID = 64;
  3
  3     !
  3     ! Comput the initial delta time.
  3     !
  3     XDELTA = .(SCSSYSID)<0,7>;
  3
  3     !
  3     ! Set up the bias. We set up a positive bias if the initial value
  3     ! is "sufficiently" small. Otherwise, we set up a positive bias.
  3     !
  3     IF .XDELTA GEQ 64
  3     THEN
  3         BIAS = -1
  3     ELSE
  3         BIAS = +1;
  3
  3     END;
  2
  2 !
  2 ! The actual delta is the previous delta plus the bias, i.e.
  2 !
  2 ! (previous_delta+bias) * 1 million * 100 nanosecond
  2 !
  2 ! This gives the range of
  2 !
  2 !       scssystemid<0,7> = 1+bias           .1 second + bias
  2 !
  2 !       scssystemid<0,7> = 128+bias       12.8 seconds + bias
  2 !
  2 ! The bias is + or - .1 second, depending on the previous delta time.
  2 ! If delta is large, we set up a negative bias for the next iteration.
  2 ! If delta is small, we set up a positive bias for the next iteration.
  2 !
  2
```

```
; 2647        3293  2 IF .XDELTA GEQ 128                                  ! Large xdelta, set negative bias
; 2648        3294  2 THEN
; 2649        3295  2     BIAS = - 1;
; 2650        3296  2
; 2651        3297  2 IF .XDELTA LEQ 10                                   ! Small xdelta, set positive bias
; 2652        3298  2 THEN
; 2653        3299  2     BIAS = + 1;
; 2654        3300  2
; 2655        3301  2 XDELTA = .XDELTA + .BIAS;                           ! Compute new xdelta
; 2656        3302  2
; 2657        3303  2 DELTA [0] = .XDELTA * (-1 * 1000 * 1000);           ! Compute delta in 100 nanoseconds
; 2658        3304  2
; 2659      P 3305  2 STATUS = $SETIMR ( EFN    = MOUNT_EFN,                ! Set timer
; 2660        3306  2                    DAYTIM = DELTA    );
; 2661        3307  2
; 2662        3308  2 IF .STATUS
; 2663        3309  2 THEN
; 2664        3310  2     $WAITFR ( EFN = MOUNT_EFN );                    ! Wait
; 2665        3311  2
; 2666        3312  2 RETURN;                                             ! Back to caller
; 2667        3313  1 END;                                                ! End of routine WAIT_DELTA


                                                          .PSECT  $OWN$,NOEXE,2

                                         00D0C SCSSYSID:
                                                          .BLKB    4
                                         00D10 XDELTA:    .BLKB    4
                                         00D14 BIAS:      .BLKB    4
                                    0004 00D18 GETS_ITMLST:
                                                          .WORD    4                                       ;
                                    1065 00D1A            .WORD    4197                                    ;
                               00000000' 00D1C            .ADDRESS SCSSYSID                                ;
                                00000000 00D20            .LONG    0                                       ;
                                00000000 00D24            .LONG    0                                       ;

                                                          .EXTRN  SYS$SETIMR, SYS$WAITFR

                                                          .PSECT  $CODE$,NOWRT,2

                                    0004 00000 WAIT_DELTA:
                                                          .WORD    Save R2                                 ; 3180
                 52 00000000'    EF   9E 00002            MOVAB    XDELTA, R2                              ;
                 5E              04   C2 00009            SUBL2    #4, SP                                  ;
                 7E              01   CE 0000C            MNEGL    #1, DELTA                               ; 3223
          04     AE              01   CE 0000F            MNEGL    #1, DELTA+4                             ;
                           04    AC   D5 00013            TSTL     N                                       ; 3242
                                 37   12 00016            BNEQ     4$                                      ;
                           FC    A2   D4 00018            CLRL     SCSSYSID                                ; 3246
                                 7E   7C 0001B            CLRQ     -(SP)                                   ; 3249
                                 7E   D4 0001D            CLRL     -(SP)                                   ;
                           08    A2   9F 0001F            PUSHAB   GETS_ITMLST                             ;
                                 7E   7C 00022            CLRQ     -(SP)                                   ;
                                 1A   DD 00024            PUSHL    #26                                     ;
    00000000G    00              07   FB 00026            CALLS    #7, SYS$GETSYIW                         ;
                 05              50   E9 0002D            BLBC     STATUS, 1$                              ; 3255
```

```
                                                              D 10
VMOUNT                                                   16-Sep-1984 01:00:56    VAX-11 Bliss-32 V4.0-742                    Page  79
V04-002                                                  12-Sep-1984 11:14:53    DISK$VMSMASTER:[MOUNT.SRC]VMOUNT.B32;3     (17)

                                      FC    A2  D5 00030           TSTL    SCSSYSID                                          ; 3256
                                            05  12 00033           BNEQ    2$                                                ;
                           FC    A2    40   8F  9A 00035 1$:       MOVZBL  #64, SCSSYSID                                     ; 3258
         62        FC    A2          07     00  EF 0003A 2$:       EXTZV   #0, #7, SCSSYSID, XDELTA                          ; 3263
                                     3F     62  D1 00040           CMPL    XDELTA, #63                                       ; 3269
                                            06  15 00043           BLEQ    3$                                                ;
                           04        A2     01  CE 00045           MNEGL   #1, BIAS                                          ; 3271
                                            04  11 00049           BRB     4$                                                ;
                           04        A2     01  D0 0004B 3$:       MOVL    #1, BIAS                                          ; 3273
                   00000080          8F     62  D1 0004F 4$:       CMPL    XDELTA, #128                                      ; 3293
                                            04  19 00056           BLSS    5$                                                ;
                           04        A2     01  CE 00058           MNEGL   #1, BIAS                                          ; 3295
                                     0A     62  D1 0005C 5$:       CMPL    XDELTA, #10                                       ; 3297
                                            04  14 0005F           BGTR    6$                                                ;
                           04        A2     01  D0 00061           MOVL    #1, BIAS                                          ; 3299
                                     62    04   A2  C0 00065 6$:   ADDL2   BIAS, XDELTA                                      ; 3301
                   6E                62 FFF0BDC0 8F  C5 00069      MULL3   #-1000000, XDELTA, DELTA                          ; 3303
                                            7E  7C 00071           CLRQ    -(SP)                                             ; 3306
                                       08   AE  9F 00073           PUSHAB  DELTA                                             ;
                                            1A  DD 00076           PUSHL   #26                                               ;
                   00000000G         00     04  FB 00078           CALLS   #4, SYS$SETIMR                                    ;
                                     09     50  E9 0007F           BLBC    STATUS, 7$                                        ; 3308
                                            1A  DD 00082           PUSHL   #26                                               ; 3310
                   00000000G         00     01  FB 00084           CALLS   #1, SYS$WAITFR                                    ;
                                                04 0008B 7$:       RET                                                       ; 3313

; Routine Size:  140 bytes,    Routine Base:  $CODE$ + 0C14


; 2668          3314  1
; 2669          3315  1 END
; 2670          3316  0 ELUDOM




                                                                   .EXTRN  LIB$SIGNAL, LIB$STOP

;                                PSECT SUMMARY
;
;       Name                               Bytes                        Attributes
;
;  $GLOBAL$                                1672  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                                  3232  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                                    96  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                                   3368  NOVEC,  WRT,  RD ,NOEXE,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      122         0          1000           00:01.8
;                                  COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:VMOUNT/OBJ=OBJ$:VMOUNT MSRC$:VMOUNT/UPDATE=(ENH$:VMOUNT)

; Size:          3232 code + 5136 data bytes
; Run Time:         01:10.3
; Elapsed Time:     02:27.4
; Lines/CPU Min:     2832
; Lexemes/CPU-Min: 25190
; Memory Used:  347 pages
; Compilation Complete
```

0247 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

VMOUNT
LIS

MPCLRPFM
LIS

MPAST
LIS

MP

MP
MAP

MP
MDL

TRNLOG
LIS

MPCMOD
LIS

MPMACROS
MAR