```
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     OOOOOOOOO     UUUUUUUUUUUUUUU  NNN         NNN        TTT
MMM         MMM     OOOOOOOOO     UUUUUUUUUUUUUUU  NNN         NNN        TTT
MMM         MMM     OOOOOOOOO     UUUUUUUUUUUUUUU  NNN         NNN        TTT
```

FILEID**MOUDK1

```
MM      MM    OOOOOO    UU      UU  DDDDDDDD    KK      KK        11
MM      MM    OOOOOO    UU      UU  DDDDDDDD    KK      KK        11
MMMM  MMMM  OO      OO  UU      UU  DD      DD  KK      KK      1111
MMMM  MMMM  OO      OO  UU      UU  DD      DD  KK      KK      1111
MM  MM  MM  OO      OO  UU      UU  DD      DD  KK    KK          11
MM  MM  MM  OO      OO  UU      UU  DD      DD  KK    KK          11
MM      MM  OO      OO  UU      UU  DD      DD  KKKKKK            11
MM      MM  OO      OO  UU      UU  DD      DD  KKKKKK            11
MM      MM  OO      OO  UU      UU  DD      DD  KK    KK          11
MM      MM  OO      OO  UU      UU  DD      DD  KK    KK          11
MM      MM  OO      OO  UU      UU  DD      DD  KK      KK        11
MM      MM  OO      OO  UU      UU  DD      DD  KK      KK        11      ....
MM      MM    OOOOOO    UUUUUUUUUU  DDDDDDDD    KK      KK      111111    ....
MM      MM    OOOOOO    UUUUUUUUUU  DDDDDDDD    KK      KK      111111    ....

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
```

```
MODULE MOUDK1 (
                LANGUAGE (BLISS32),
                IDENT = 'V04-002'
                ) =
BEGIN

!
!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************

!++
!
! FACILITY:  MOUNT Utility Structure Level 1
!
! ABSTRACT:
!
!       This routine performs all of the mechanics of mounting a disk,
!       given as input the parsed and partially validated command line.
!
! ENVIRONMENT:
!
!       STARLET operating system, including privileged system services
!       and internal exec routines.
!
!--
!
!
! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  17-Oct-1977  17:41
!
! MODIFIED BY:
!
!       V04-002 HH0057          Hai Huang               12-Sep-1984
!               Clear DEV$V_MNT bit along with UCB$L_VCB on error
!               path.
!
!       V04-001 HH0056          Hai Huang               11-Sep-1984
!               Return SS$_VOLINV status when appropriate to facilitate
```

```
1 !                     retry on volume invalid errors.
1 !
1 !     V03-013 HH0045          Hai Huang                 10-Aug-1984
1 !             Take out the volume lock for shared foreign mounts.
1 !
1 !     V03-012 HH0041          Hai Huang                 24-Jul-1984
1 !             Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
1 !
1 !     V03-011 LMP0221         L. Mark Pilant,           28-Mar-1984  9:48
1 !             Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
1 !             ORB$W_PROT.
1 !
1 !     V03-010 HH0005          Hai Huang                 29-Feb-1984
1 !             Fix truncation errors (again).
1 !
1 !     V03-009 HH0002          Hai Huang                 15-Feb-1984
1 !             Add job-wide mount support, i.e. always deallocate
1 !             mount list entries to paged-pool in condition handler.
1 !
1 !     V03-008 LY00B5          Larry Yetto               10-FEB-1984 11:25
1 !             Fix truncation errors.
1 !
1 !     V03-007 CDS0002         Christian D. Saether      26-Aug-1983
1 !             Fill in VCB$T_VOLCKNAM field.
1 !
1 !     V03-006 CDS0001         Christian D. Saether      21-Aug-1983
1 !             Add calls to check for consistent mounting on
1 !             cluster available devices.
1 !
1 !     V03-005 TCM0001         Trudy C. Matthews         21-Jun-1983
1 !             Increment refcount stored in UCB on mount.
1 !
1 !     V03-004 DMW4043         DMWalp                     7-Jun-1983
1 !             Remove (S)LOG_ENTRY
1 !
1 !     V03-003 STJ50311        Steven T. Jeffreys,       11-Feb-1983
1 !             Make all references to PHYS_NAME indexed by DEVICE_INDEX.
1 !
1 !     V03-002 STJ0300         Steven T. Jeffreys,       18-May-1982
1 !             Add support for the /NOUNLOAD qualifier.
1 !
1 !     V03-001 STJ0242         Steven T. Jeffreys,       30-Mar-1982
1 !             - Remove code that sets the device allocation access mode.
1 !               The device will be manually deallocated in VMOUNT.
1 !             - Read the first block of the storage map and write it
1 !               back to the disk to determine if the volume is hardware
1 !               write-locked.
1 !
1 !     V02-009 STJ0192         Steven T. Jeffreys,       02-Feb-1982
1 !             Use global buffers defined in MOUDK2.
1 !
1 !     V02-008 ACG0246         Andrew C. Goldstein,      4-Jan-1982  14:48
1 !             Add /OVER:LOCK, add NOCACHE bit in VCB;
1 !             remove primary exception vector logic.
1 !
1 !     V02-007 LMP0001         L. Mark Pilant            9-Nov-1981
1 !             Map the entire index file if it contains extension
```

```
1 !                 file headers.
1 !
1 !         V02-006 STJ0041         Steven T. Jeffreys,     21-May-1980
1 !                 Copy volume serial number from home block to VCB.
1 !
1 !         V02-005 ACG0169         Andrew C. Goldstein,    18-Apr-1980  13:56
1 !                 Bug check on internal errors
1 !
1 !         V02-004 ACG0167         Andrew C. Goldstein,    18-Apr-1980  13:38
1 !                 Previous revision history moved to MOUNT.REV
1 !**
1
1
1 LIBRARY 'SYS$LIBRARY:LIB.L32';
1 REQUIRE 'SRC$:MOUDEF.B32';
1
1
1 FORWARD ROUTINE
1         MOUNT_DISK1     : NOVALUE,      ! main disk mounting routine
1         MOUNT_HANDLER,                  ! condition handler for main mount code
1         MAKE_DISK_MOUNT,                ! kernel mode mount routine
1         KERNEL_HANDLER  : NOVALUE;      ! kernel mode condition handler
```

```
1 !+
1 !
1 ! Own storage for this module.
1 !
1 !-
1
1 LITERAL
1         WINDOW_SIZE     = 30*6;            ! maximum index file window size
1
1 OWN
1         PROTO_FCBE1     : BBLOCK [FCB$C_LENGTH], ! prototype index file extent 1
1         PROTO_FCBE2     : BBLOCK [FCB$C_LENGTH]; ! prototype index file extent 2
1
1 EXTERNAL
1         !
1         ! These buffers are shared with MOUDK2.
1         !
1         BUFFER          : BBLOCK,                ! buffer for disk blocks
1         PROTO_VCB       : BBLOCK,                ! prototype VCB
1         PROTO_FCB       : BBLOCK,                ! prototype index file FCB
1         PROTO_WCB       : BBLOCK,                ! prototype index file window
1         VOLUME_UIC      : LONG;                  ! owner UIC of volume
1
```

```
1 GLOBAL ROUTINE MOUNT_DISK1 : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine performs all of the mechanics of mounting a structure
1 !       level 1 disk, given as input the parsed and partially validated
1 !       command line.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       MOUNT_DISK ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       MOUNT parser data base
1 !       CHANNEL: channel number for I/O
1 !       HOME_BLOCK: buffer containing volume home block
1 !       HOMEBLOCK_LBN: LBN of home block
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       volume mounted: VCB, etc., created, ACP started
1 !
1 !--
1
2 BEGIN
2
2 BUILTIN
2       ROT,
2       FFS,
2       FFC;
2
2 LOCAL
2       PROCESS_UIC,                            ! UIC of this process
2       PRIVILEGE_MASK  : REF BBLOCK,           ! address of process privilege mask
2       P,                                      ! random pointer
2       C,                                      ! string count
2       STATUS,                                 ! utility status word
2       MAP_AREA        : REF BBLOCK,           ! pointer to file header map area
2       MAP_POINTER     : REF BBLOCK,           ! pointer to scan map pointers
2       WCB_POINTER     : REF BBLOCK,           ! pointer to scan WCB pointers
2       INDEX_LBN,                              ! LBN of current index file map pointer
2       INDEX_CNT,                              ! count for above LBN
2       EXTENT_LBN,                             ! LBN of the extension header
2       EXTENT_VBN,                             ! VBN of the extension header
```

```
        EXTENT_FID,                     ! FID of the next extent
        BIAS,                           ! offset for storage map location
        COUNT,                          ! number of blocks in storage map
        LBN,                            ! current LBN in use
        FREE,                           ! number of free blocks on volume
        X,                              ! longword of bitmap
        B1,                             ! start point of bit scan
        B2;                             ! end point of bit scan

EXTERNAL
        DEV_CTX         : BBLOCK FIELD (DC), ! device lock value block context
        VOL_CTX         : BBLOCK FIELD (VC), ! volume lock value block context
        MOUNT_OPTIONS   : BITVECTOR,    ! command option flags
        DEVICE_CHAR     : BBLOCK,       ! device characteristics
        LABEL_STRING    : VECTOR,       ! volume label string in command
        DEVICE_INDEX    : LONG,         ! index into PHYS_NAME vector
        PHYS_NAME       : VECTOR,       ! descriptor of physical device name
        DEVICE_COUNT,                   ! number of device specified
        DRIVE_COUNT     : VECTOR,       ! number of drives per device
        WINDOW,                         ! command specified window size
        ACCESSED,                       ! command specified LRU limit
        EXTENSION,                      ! command specified default file extend
        HOME_BLOCK      : BBLOCK,       ! buffer containing volume home block
        HOMEBLOCK_LBN,                  ! LBN of home block
        HEADER_LBN,                     ! LBN of current file header
        CTL$GL_PHD      : REF BBLOCK ADDRESSING_MODE (ABSOLUTE),
                                        ! vector page pointer to process header
        ACP$GB_WINDOW   : BYTE ADDRESSING_MODE (ABSOLUTE),
                                        ! default window size for /SYSTEM
        ACP$GW_SYSACC   : WORD  ADDRESSING_MODE (ABSOLUTE);
                                        ! default LRU limit for /SYSTEM

EXTERNAL ROUTINE
        CHECK_CLUSTER_SANITY : NOVALUE, ! routine to check cluster consistency
        GET_VOLUME_LOCK,                ! take out volume lock
        GET_VOLUME_LOCK_NAME,           ! generate volume lock name
        GET_UIC,                        ! get UIC of process
        CHECK_HEADER,                   ! verify file header
        WRITE_BLOCK,                    ! write a block to the disk
        READ_BLOCK,                     ! read a block from the disk
        INIT_FCB,                       ! initialize FCB
        TURN_WINDOW1;                   ! initialize window

ENABLE MOUNT_HANDLER;

! For maximum safety, we do as much setup work in user mode as possible. We
! read all of the disk blocks (index file and storage map headers and the
! storage map) in user mode so that the program is abortable in case something
! hangs. Prototype control blocks are built in local storage and are copied
! into the system pool by the kernel mode routine.
!
! Get the process UIC and the volume owner UIC. Make the privilege checks
! for overriding volume protection and options requiring operator privilege.
!

IF .DEVICE_COUNT NEQ 1 OR .DRIVE_COUNT[0] GTR 1
THEN ERR_EXIT (MOUN$_DEVICES);
```

```
2
2 PROCESS_UIC = KERNEL_CALL (GET_UIC);
2 PRIVILEGE_MASK = CTL$GL_PHD[PHD$Q_PRIVMSK];
2 VOLUME_UIC = 0;
2 IF .MOUNT_OPTIONS[OPT_IS_FILES11]
2 THEN
3     BEGIN
3     VOLUME_UIC = .(HOME_BLOCK[HM1$W_VOLOWNER])<0,8>;
3     VOLUME_UIC<16,8> = .(HOME_BLOCK[HM1$W_VOLOWNER])<8,8>;
2     END;
2
3 IF  (
3     .MOUNT_OPTIONS[OPT_OVR_PRO]
4     AND NOT (.PRIVILEGE_MASK[PRV$V_VOLPRO]
4               OR .VOLUME_UIC EQL 0
4               OR .VOLUME_UIC EQL .PROCESS_UIC)
3     )
3
3 OR  (
4     (   .MOUNT_OPTIONS[OPT_WINDOW]
4     OR .MOUNT_OPTIONS[OPT_ACCESSED]
4     OR .MOUNT_OPTIONS[OPT_UNIQUEACP]
4     OR .MOUNT_OPTIONS[OPT_SAMEACP]
4     OR .MOUNT_OPTIONS[OPT_FILEACP]
4     )
3     AND NOT .PRIVILEGE_MASK[PRV$V_OPER]
3     )
3
3 OR  (
3     .MOUNT_OPTIONS[OPT_GROUP]
3     AND NOT .PRIVILEGE_MASK [PRV$V_GRPNAM]
3     )
3
3 OR  (
3     .MOUNT_OPTIONS[OPT_SYSTEM]
3     AND NOT .PRIVILEGE_MASK [PRV$V_SYSNAM]
3     )
3
2 THEN ERR_EXIT (SS$_NOPRIV);
2
2 IF .MOUNT_OPTIONS[OPT_FOREIGN]
2 THEN VOLUME_UIC = .PROCESS_UIC;
2
2 IF .DEV_CTX [DC_NOTFIRST_MNT]
2 THEN
2     CHECK_CLUSTER_SANITY ();
2
2 ! First fill in the prototype VCB from the data in the home block.
2 !
2
2 CH$FILL (0, VCB$C_LENGTH, PROTO_VCB);    ! init to zero
2 PROTO_VCB[VCB$W_TRANS] = 1;              ! transaction count
2 PROTO_VCB[VCB$W_MCOUNT] = 1;             ! mount count
2
2 IF .MOUNT_OPTIONS[OPT_GROUP]
2 THEN PROTO_VCB[VCB$V_GROUP] = 1;
2 IF .MOUNT_OPTIONS[OPT_SYSTEM]
```

```
2 THEN PROTO_VCB[VCB$V_SYSTEM] = 1;
2
2 !
2 ! Copy volume serial number from homeblock to VCB.
2 !
2 PROTO_VCB [VCB$L_SERIALNUM] = .HOME_BLOCK [HM1$L_SERIALNUM];
2
2 IF .MOUNT_OPTIONS[OPT_IS_FILES11]
3 AND NOT (.MOUNT_OPTIONS[OPT_FOREIGN] AND .MOUNT_OPTIONS[OPT_LABEL])
2 THEN
3     BEGIN
3                                           ! volume label, blank filled
3                                           ! find trailing zero, if any
3     P = CH$FIND_CH (HM1$S_VOLNAME, HOME_BLOCK[HM1$T_VOLNAME], 0);
3     C = 12;                               ! compute string length
3     IF NOT CH$FAIL (.P)
3     THEN C = .P - HOME_BLOCK[HM1$T_VOLNAME];
3     CH$COPY (.C, HOME_BLOCK[HM1$T_VOLNAME], ' ',
3               VCB$S_VOLNAME, PROTO_VCB[VCB$T_VOLNAME]);
3     END
2 ELSE
2     CH$COPY (.LABEL_STRING[0], .LABEL_STRING[1], ' ',
2               VCB$S_VOLNAME, PROTO_VCB[VCB$T_VOLNAME]);
2
2 IF NOT .MOUNT_OPTIONS[OPT_FOREIGN]
2 THEN
3     BEGIN
3     PROTO_VCB[VCB$V_MOUNTVER] = .MOUNT_OPTIONS [OPT_MOUNTVER];
3     PROTO_VCB[VCB$L_HOMELBN] = .HOMEBLOCK_LBN; ! home block LBN
3                                           ! index file bitmap LBN
3     PROTO_VCB[VCB$L_IBMAPLBN] = ROT (.HOME_BLOCK[HM1$L_IBMAPLBN], 16);
3     PROTO_VCB[VCB$W_CLUSTER] = 1;                 ! volume cluster factor
3                                           ! default window size
3     PROTO_VCB[VCB$B_WINDOW] = .HOME_BLOCK[HM1$B_WINDOW];
3     IF .PROTO_VCB[VCB$B_WINDOW] EQL 0
3     THEN PROTO_VCB[VCB$B_WINDOW] = 7;
3     IF .MOUNT_OPTIONS[OPT_SYSTEM]
3     THEN PROTO_VCB[VCB$B_WINDOW] = .ACP$GB_WINDOW;
3     IF .MOUNT_OPTIONS[OPT_WINDOW]
3     THEN PROTO_VCB[VCB$B_WINDOW] = .WINDOW;
3                                           ! directory LRU limit
3     PROTO_VCB[VCB$B_LRU_LIM] = .HOME_BLOCK[HM1$B_LRU_LIM];
3     IF .MOUNT_OPTIONS[OPT_SYSTEM]
3     THEN PROTO_VCB[VCB$B_LRU_LIM] = .ACP$GW_SYSACC;
3     IF .MOUNT_OPTIONS[OPT_ACCESSED]
3     THEN PROTO_VCB[VCB$B_LRU_LIM] = .ACCESSED;
3     IF .MOUNT_OPTIONS[OPT_NOCACHE]
3     THEN PROTO_VCB[VCB$B_LRU_LIM] = 0;
3                                           ! default file extend
3     PROTO_VCB[VCB$W_EXTEND] = .HOME_BLOCK[HM1$B_EXTEND];
3     IF .PROTO_VCB[VCB$W_EXTEND] EQL 0
3     THEN PROTO_VCB[VCB$W_EXTEND] = 5;
3     IF .MOUNT_OPTIONS[OPT_EXTENSION]
3     THEN PROTO_VCB[VCB$W_EXTEND] = .EXTENSION;
3                                           ! index file bitmap size
3     PROTO_VCB[VCB$B_IBMAPSIZE] = .HOME_BLOCK[HM1$W_IBMAPSIZE];
3                                           ! maximum number of files
```

```
    PROTO_VCB[VCB$L_MAXFILES] = .HOME_BLOCK[HM1$W_MAXFILES];

    IF .MOUNT_OPTIONS[OPT_NOCACHE]
    THEN PROTO_VCB[VCB$V_NOCACHE] = 1;

! Now read the index file header, verify it, and initialize the prototype
! index file FCB.
!

    HEADER_LBN = .PROTO_VCB[VCB$L_IBMAPLBN] + .PROTO_VCB[VCB$B_IBMAPSIZE];
    STATUS = READ_BLOCK (.HEADER_LBN, BUFFER);
    IF NOT .STATUS THEN ERR_EXIT (.STATUS);
    IF NOT CHECK_HEADER (BUFFER, UPLIT WORD (1, 1, 0)) THEN ERR_EXIT ();

    CH$FILL (0, FCB$C_LENGTH, PROTO_FCB);

! Clear out the extension header FCB's so they are in a known state
!

    CH$FILL (0, FCB$C_LENGTH, PROTO_FCBE1);
    CH$FILL (0, FCB$C_LENGTH, PROTO_FCBE2);

    PROTO_FCB[FCB$L_STVBN] = 1;
    INIT_FCB (PROTO_FCB, BUFFER);
    PROTO_FCB[FCB$W_ACNT] = 1;

! Build the prototype index file window.
!

    CH$FILL (0, WCB$C_LENGTH, PROTO_WCB);
    PROTO_WCB[WCB$W_SIZE] = WCB$C_LENGTH + WINDOW_SIZE;
    PROTO_WCB[WCB$V_READ] = 1;
    TURN_WINDOW1 (PROTO_WCB, BUFFER, 3, 1);

! Read any extents that exist, verify them, and initialize the appropriate
! FCB for them. In addition, update the WCB to reflect the entire file.
!

MAP_AREA = BUFFER + .BUFFER[FH1$B_MPOFFSET] * 2;
IF .MAP_AREA[FM1$W_EX_FILNUM] NEQ 0 AND .MAP_AREA[FM1$W_EX_FILSEQ] NEQ 0
THEN
    BEGIN
    MAP_POINTER = .MAP_AREA + FM1$C_POINTERS;
    DECR J FROM .MAP_AREA[FM1$B_INUSE] TO 1 DO
        BEGIN
        INDEX_LBN = .MAP_POINTER[FM1$W_LOWLBN];
        INDEX_LBN<16,8> = .MAP_POINTER[FM1$B_HIGHLBN];
        INDEX_CNT = .MAP_POINTER[FM1$B_COUNT] + 1;
        IF .HEADER_LBN GEQU .INDEX_LBN
        AND .HEADER_LBN LSSU .INDEX_LBN + .INDEX_CNT THEN EXITLOOP;
        MAP_POINTER = .MAP_POINTER + 4;
        END;

! Verify that the extension file header falls within the contiguous portion
!

    EXTENT_LBN = .HEADER_LBN + .MAP_AREA[FM1$W_EX_FILNUM] - 1;
```

```
	IF .INDEX_LBN + .INDEX_CNT LSSU .EXTENT_LBN THEN ERR_EXIT (SS$_FILESTRUCT);
	EXTENT_FID = .MAP_AREA[FM1$W_EX_FILNUM];
	EXTENT_FID<16,16> = .MAP_AREA[FM1$W_EX_FILSEQ];

! Read in the extent and add to the list.
!

	STATUS = READ_BLOCK (.EXTENT_LBN, BUFFER);
	IF NOT .STATUS THEN ERR_EXIT (.STATUS);
	IF NOT CHECK_HEADER (BUFFER, EXTENT_FID) THEN ERR_EXIT ();
	PROTO_FCBE1[FCB$L_STVBN] = .PROTO_FCB[FCB$L_FILESIZE] + .PROTO_FCB[FCB$L_STVBN];
	INIT_FCB (PROTO_FCBE1, BUFFER);
	PROTO_FCB[FCB$L_FILESIZE] = .PROTO_FCB[FCB$L_FILESIZE] + .PROTO_FCBE1[FCB$L_FILESIZE];
	PROTO_FCBE1[FCB$W_ACNT] = 1;
	PROTO_FCBE1[FCB$L_HDLBN] = .EXTENT_LBN;

! Update the prototype index file window.
!

	WCB_POINTER = PROTO_WCB + WCB$C_MAP;
	EXTENT_VBN = 1;
	INCR J FROM 1 TO .PROTO_WCB[WCB$W_NMAP] DO
	    BEGIN
	    EXTENT_VBN = .EXTENT_VBN + .WCB_POINTER[WCB$W_COUNT];
	    WCB_POINTER = .WCB_POINTER + 6;
	    END;
	TURN_WINDOW1 (PROTO_WCB, BUFFER, 3, .EXTENT_VBN);

	MAP_AREA = BUFFER + .BUFFER[FH1$B_MPOFFSET] * 2;
	IF .MAP_AREA[FM1$W_EX_FILNUM] NEQ 0 AND .MAP_AREA[FM1$W_EX_FILSEQ] NEQ 0
	THEN
	    BEGIN

! Verify that the extent falls within the contiguous portion.
!

	    EXTENT_LBN = .HEADER_LBN + .MAP_AREA[FM1$W_EX_FILNUM] - 1;
	    IF .INDEX_LBN + .INDEX_CNT LSSU .EXTENT_LBN THEN ERR_EXIT (SS$_FILESTRUCT);
	    EXTENT_FID = .MAP_AREA[FM1$W_EX_FILNUM];
	    EXTENT_FID<16,16> = .MAP_AREA[FM1$W_EX_FILSEQ];

! Read in the extent and add it to the list
!

	    STATUS = READ_BLOCK (.EXTENT_LBN, BUFFER);
	    IF NOT .STATUS THEN ERR_EXIT (.STATUS);
	    IF NOT CHECK_HEADER (BUFFER, EXTENT_FID) THEN ERR_EXIT ();
	    PROTO_FCBE2[FCB$L_STVBN] = .PROTO_FCBE1[FCB$L_FILESIZE] + .PROTO_FCBE1[FCB$L_STVBN];
	    INIT_FCB (PROTO_FCBE2, BUFFER);
	    PROTO_FCB[FCB$L_FILESIZE] = .PROTO_FCB[FCB$L_FILESIZE] + .PROTO_FCBE2[FCB$L_FILESIZE];
	    PROTO_FCBE2[FCB$W_ACNT] = 1;
	    PROTO_FCBE2[FCB$L_HDLBN] = .EXTENT_LBN;

! Update the prototype index file window.
!

	    WCB_POINTER = PROTO_WCB + WCB$C_MAP;
```

```
5                  EXTENT_VBN = 1;
5                  INCR J FROM 1 TO .PROTO_WCB[WCB$W_NMAP] DO
6                      BEGIN
6                      EXTENT_VBN = .EXTENT_VBN + .WCB_POINTER[WCB$W_COUNT];
6                      WCB_POINTER = .WCB_POINTER + 6;
5                      END;
5                  TURN_WINDOW1 (PROTO_WCB, BUFFER, 3, .EXTENT_VBN);
5
4                  END;
3              END;
3
3      ! Now read the storage map file header and find the starting LBN of the
3      ! storage map. Note that we skip the "storage control block", which may or
3      ! may not be represented by a separate retrieval pointer.
3      !
3
3          STATUS = READ_BLOCK (.PROTO_VCB[VCB$L_IBMAPLBN] + .PROTO_VCB[VCB$B_IBMAPSIZE] + 1, BUFFER);
3          IF NOT .STATUS OR NOT CHECK_HEADER (BUFFER, UPLIT WORD (2, 2, 0))
3          THEN
4              BEGIN
4              IF .STATUS EQL SS$_VOLINV
4              THEN
4                  ERR_EXIT (SS$_VOLINV)
4              ELSE
4                  ERR_MESSAGE (MOUN$_MAPHDRBAD);
4              PROTO_VCB[VCB$V_NOALLOC] = 1;
4              END
4
3          ELSE
4              BEGIN
4              MAP_AREA = BUFFER + .BUFFER[FM1$B_MPOFFSET]*2;
4              MAP_POINTER = .MAP_AREA + FM1$C_POINTERS;
4
4              BIAS = 1;                               ! assume one retrieval pointer
4              IF .MAP_AREA[FM1$B_INUSE] GTR 4
4              OR .MAP_AREA[FM1$B_INUSE] LSS 2
4              THEN ERR_EXIT (SS$_FILESTRUCT); ! more than 2 or no pointers
4              IF .MAP_AREA[FM1$B_INUSE] EQL 4
4              THEN
5                  BEGIN
5                  BIAS = 0;                   ! 2 pointers - use the second
5                  MAP_POINTER = .MAP_POINTER + 4;
4                  END;
4
4              COUNT = .(.MAP_POINTER)<8,8> + 1 - .BIAS;
4              LBN = .(.MAP_POINTER)<16,16>;
4              LBN<16,8> = .(.MAP_POINTER)<0,8>;
4              LBN = .LBN + .BIAS;
4
4              PROTO_VCB[VCB$L_SBMAPLBN] = .LBN;
4              PROTO_VCB[VCB$B_SBMAPSIZE] = .COUNT;
4
4      ! Read the first block of the storage map and write it back.  If the
4      ! write fails because the device is hardware write-locked, mark the
4      ! volume software write-locked and inform the user of the situation.
4      ! For the moment, ignore read errors, as they will be handled later.
4      !
```

```
4
4          IF .MOUNT_OPTIONS [OPT_WRITE]
4          THEN
4              IF READ_BLOCK (.LBN, BUFFER)
4              THEN
5                  IF NOT (STATUS = WRITE_BLOCK (.LBN, BUFFER))
4                  THEN
5                      BEGIN
5                      IF .STATUS EQL SS$_VOLINV
5                      THEN
5                          ERR_EXIT (SS$_VOLINV);
5                      IF .STATUS EQL SS$_WRITLCK
5                      THEN ERR_MESSAGE (MOUN$_WRITELOCK)
5                      ELSE ERR_MESSAGE (MOUN$_WRITESCB, 0, .STATUS);
5                      MOUNT_OPTIONS[OPT_WRITE] = 0;
4                      END;
4
4  ! Scan the storage map to compute the number of free blocks on the volume.
4  !
4
4          FREE = 0;
4          DECR J FROM .COUNT TO 1 DO
5              BEGIN
5              MAP BUFFER : VECTOR;
5
5              STATUS = READ_BLOCK (.LBN, BUFFER);
5              IF NOT .STATUS
5              THEN
6                  BEGIN
6                  IF .STATUS EQL SS$_VOLINV
6                  THEN
6                      ERR_EXIT (SS$_VOLINV)
6                  ELSE
6                      ERR_MESSAGE (MOUN$_BITMAPERR, 0, .STATUS);
6                  PROTO_VCB[VCB$V_NOALLOC] = 1;
5                  END;
5
5              INCR I FROM 0 TO 127 DO
6                  BEGIN
6                  X = .BUFFER[.I];
6                  IF .X NEQ 0
6                  THEN
7                      BEGIN
7                      B2 = 0;
7                      WHILE 1 DO
8                          BEGIN
8                          IF FFS (B2, %REF (32-.B2), X, B1)
8                          THEN EXITLOOP;
8                          FFC (B1, %REF (32-.B1), X, B2);
8                          FREE = .FREE + .B2 - .B1;
8                          IF .B2 GEQ 32 THEN EXITLOOP;
7                          END;
6                      END;
5                  END;
5              LBN = .LBN + 1;
4              END;
4
```

```
            PROTO_VCB[VCB$L_FREE] = .FREE;
            END;

        END                                     ! end of Files-11 specific mount processing

    ELSE

    ! This is a foreign mount. If this is a shared foreign mount,
    ! take out the volume lock.
    !

        IF NOT .MOUNT_OPTIONS [OPT_NOSHARE]
        THEN
            BEGIN
            GET_VOLUME_LOCK_NAME ();
            IF NOT (STATUS = KERNEL_CALL (GET_VOLUME_LOCK))
            THEN
                ERR_EXIT (.STATUS);
            IF .DEV_CTX [DC_NOTFIRST_MNT] NEQ .VOL_CTX [VC_NOTFIRST_MNT]
            THEN
                ERR_EXIT (MOUN$_VOLALRMNT);
            END;                                ! end of foreign-specific mount processing

    ! Finally call the kernel mode routine to make it all real. Note that all the
    ! hookups, including generating the mounted volume list entry, are done
    ! within one kernel mode call so that they are uninterruptible by the user.
    !

    IF .MOUNT_OPTIONS[OPT_OVR_LOCK]
    THEN PROTO_VCB[VCB$V_NOALLOC] = 0;

    STATUS = KERNEL_CALL (MAKE_DISK_MOUNT);
    IF NOT .STATUS THEN ERR_EXIT (.STATUS);

    ! Announce that the volume is mounted.
    !

    ERR_MESSAGE (MOUN$_MOUNTED, 3, VCB$S_VOLNAME, PROTO_VCB[VCB$T_VOLNAME], PHYS_NAME[.DEVICE_INDEX*2]);

    END;                                        ! end of routine MOUNT_DISK

                                    .TITLE  MOUDK1
                                    .IDENT  \V04-002\

                                    .PSECT  $PLIT$,NOWRT,NOEXE,2

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

                                    .PSECT  $OWN$,NOEXE,2

                     00000 PROTO_FCBE1:
                                    .BLKB   180
                     000B4 PROTO_FCBE2:
                                    .BLKB   180
```

```
                                                        .EXTRN  BUFFER, PROTO_VCB
                                                        .EXTRN  PROTO_FCB, PROTO_WCB
                                                        .EXTRN  VOLUME_UIC, DEV_CTX
                                                        .EXTRN  VOL_CTX, MOUNT_OPTIONS
                                                        .EXTRN  DEVICE_CHAR, LABEL_STRING
                                                        .EXTRN  DEVICE_INDEX, PHYS_NAME
                                                        .EXTRN  DEVICE_COUNT, DRIVE_COUNT
                                                        .EXTRN  WINDOW, ACCESSED
                                                        .EXTRN  EXTENSION, HOME_BLOCK
                                                        .EXTRN  HOMEBLOCK_LBN, HEADER_LBN
                                                        .EXTRN  CTL$GL_PHD, ACP$GB_WINDOW
                                                        .EXTRN  ACP$GW_SYSACC, CHECK_CLUSTER_SANITY
                                                        .EXTRN  GET_VOLUME_LOCK
                                                        .EXTRN  GET_VOLUME_LOCK_NAME
                                                        .EXTRN  GET_UIC, CHECK_HEADER
                                                        .EXTRN  WRITE_BLOCK, READ_BLOCK
                                                        .EXTRN  INIT_FCB, TURN_WINDOW1
                                                        .EXTRN  SYS$CMKRNL

                                                        .PSECT  $CODE$,NOWRT,2

                               0FFC 00000               .ENTRY  MOUNT_DISK1, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 0691
                                                                R10,R11                                      ;
           5B 00000000G   00   9E 00002                 MOVAB   LIB$STOP, R11                                ;
           5A      0000G  CF   9E 00009                 MOVAB   PROTO_VCB+11, R10                            ;
           5E             04   C2 0000E                 SUBL2   #4, SP                                       ;
           6D       05F3  CF   DE 00011                 MOVAL   74$, (FP)                                    ; 0728
           01      0000G  CF   D1 00016                 CMPL    DEVICE_COUNT, #1                             ; 0803
                          07   12 0001B                 BNEQ    1$                                           ;
           01      0000G  CF   D1 0001D                 CMPL    DRIVE_COUNT, #1                              ;
                          09   15 00022                 BLEQ    2$                                           ;
             00728174     8F   DD 00024  1$:            PUSHL   #7504244                                     ; 0804
           6B             01   FB 0002A                 CALLS   #1, LIB$STOP                                 ;
                          7E   D4 0002D  2$:            CLRL    -(SP)                                        ; 0806
                          5E   DD 0002F                 PUSHL   SP                                           ;
                   0000G  CF   9F 00031                 PUSHAB  GET_UIC                                      ;
       00000000G   9F     03   FB 00035                 CALLS   #3, @#SYS$CMKRNL                             ;
                   52     50   D0 0003C                 MOVL    R0, PROCESS_UIC                              ;
           50 00000000G   9F   D0 0003F                 MOVL    @#CTL$GL_PHD, PRIVILEGE_MASK                 ; 0807
                   0000G  CF   D4 00046                 CLRL    VOLUME_UIC                                   ; 0808
 0E        0000G   CF     01   E1 0004A                 BBC     #1, MOUNT_OPTIONS+4, 3$                      ; 0809
           0000G   CF  0000G   CF   9A 00050            MOVZBL  HOME_BLOCK+30, VOLUME_UIC                    ; 0812
           0000G   CF  0000G   CF   90 00057            MOVB    HOME_BLOCK+31, VOLUME_UIC+2                  ; 0813
                   11  0000G   CF   E9 0005E  3$:       BLBC    MOUNT_OPTIONS+4, 4$                          ; 0817
 0D                60     15   E0 00063                 BBS     #21, (PRIVILEGE_MASK), 4$                    ; 0818
                   0000G  CF   D5 00067                 TSTL    VOLUME_UIC                                   ; 0819
                          07   13 0006B                 BEQL    4$                                           ;
                   52  0000G   CF   D1 0006D            CMPL    VOLUME_UIC, PROCESS_UIC                      ; 0820
                          34   12 00072                 BNEQ    8$                                           ;
                   18  0000G   CF   E8 00074  4$:       BLBS    MOUNT_OPTIONS+3, 5$                          ; 0824
 12        0000G   CF     01   E0 00079                 BBS     #1, MOUNT_OPTIONS+3, 5$                      ; 0825
 0C        0000G   CF     02   E0 0007F                 BBS     #2, MOUNT_OPTIONS+3, 5$                      ; 0826
 06        0000G   CF     03   E0 00085                 BBS     #3, MOUNT_OPTIONS+3, 5$                      ; 0827
 04        0000G   CF     04   E1 0008B                 BBC     #4, MOUNT_OPTIONS+3, 6$                      ; 0828
 13                60     12   E1 00091  5$:            BBC     #18, (PRIVILEGE_MASK), 8$                    ; 0830
                   0000G  CF   95 00095  6$:            TSTB    MOUNT_OPTIONS                                ; 0834
                          04   18 00099                 BGEQ    7$                                           ;
```

```
                  09            60           03  E1 0009B        BBC     #3, (PRIVILEGE_MASK), 8$                        ; 0835
                                09    0000G  CF  E9 0009F  7$:   BLBC    MOUNT_OPTIONS+T, 9$                             ; 0839
                  05            60           02  E0 000A4        BBS     #2, (PRIVILEGE_MASK), 9$                        ; 0840
                                             24  DD 000A8  8$:   PUSHL   #36                                             ; 0843
                                6B           01  FB 000AA        CALLS   #1, LIB$STOP                                    ;
                  05     0000G  CF           03  E1 000AD  9$:   BBC     #3, MOUNT_OPTIONS+1, 10$                        ; 0845
                         0000G  CF           52  D0 000B3        MOVL    PROCESS_UIC, VOLUME_UIC                         ; 0846
                                05    0000G  CF  E9 000B8  10$:  BLBC    DEV_CTX, 11$                                    ; 0848
                         0000G  CF           00  FB 000BD        CALLS   #0, CHECK_CLUSTER_SANITY                        ; 0850
   00EC   8F      00            6E           00  2C 000C2  11$:  MOVC5   #0, (SP), #0, #236, PROTO_VCB                   ; 0855
                                      F5     AA     000C9                                                                ;
                         01     AA           01  B0 000CB        MOVW    #1, PROTO_VCB+12                                ; 0856
                         41     AA           01  B0 000CF        MOVW    #1, PROTO_VCB+76                                ; 0857
                                      0000G  CF  95 000D3        TSTB    MOUNT_OPTIONS                                   ; 0859
                                             04  18 000D7        BGEQ    12$                                             ;
                                6A    40     8F  88 000D9        BISB2   #64, PROTO_VCB+11                               ; 0860
          56   0000G  CF        01           00  EF 000DD  12$:  EXTZV   #0, #1, MOUNT_OPTIONS+1, R6                     ; 0861
                                04           56  E9 000E4        BLBC    R6, 13$                                         ;
                                6A    80     8F  88 000E7        BISB2   #128, PROTO_VCB+11                              ; 0862
                         59     AA    0000G  CF  D0 000EB  13$:  MOVL    HOME_BLOCK+456, PROTO_VCB+100                   ; 0867
                  31     0000G  CF           01  E1 000F1        BBC     #1, MOUNT_OPTIONS+4, 17$                        ; 0869
                  06     0000G  CF           03  E1 000F7        BBC     #3, MOUNT_OPTIONS+1, 14$                        ; 0870
                                      0000G  CF  95 000FD        TSTB    MOUNT_OPTIONS+3                                 ;
                                             25  19 00101        BLSS    17$                                             ;
               0000G  CF        0C           00  3A 00103  14$:  LOCC    #0, #12, HOME_BLOCK+14                          ; 0875
                                             02  12 00109        BNEQ    15$                                             ;
                                             51  D4 0010B        CLRL    R1                                              ;
                                52           0C  D0 0010D  15$:  MOVL    #12, C                                          ; 0876
                                             51  D5 00110        TSTL    P                                               ; 0877
                                             09  13 00112        BEQL    16$                                             ;
                                50    0000G  CF  9E 00114        MOVAB   HOME_BLOCK+14, R0                               ; 0878
                  52            51           50  C3 00119        SUBL3   R0, P, C                                        ;
          0C      20     0000G  CF           52  2C 0011D  16$:  MOVC5   C, HOME_BLOCK+14, #32, #12, PROTO_VCB+20        ; 0880
                                      09     AA     00124                                                                ;
                                             0B  11 00126        BRB     18$                                             ; 0869
          0C      20     0000G  DF    0000G  CF  2C 00128  17$:  MOVC5   LABEL_STRING, @LABEL_STRING+4, #32, #12, -      ; 0884
                                      09     AA     00131                PROTO_VCB+20                                    ;
                  03     0000G  CF           03  E1 00133  18$:  BBC     #3, MOUNT_OPTIONS+1, 19$                        ; 0886
                                    0450     31     00139        BRW     69$                                             ;
          50   0000G  CF        01           06  EF 0013C  19$:  EXTZV   #6, #1, MOUNT_OPTIONS+6, R0                     ; 0889
      48  AA          01        02           50  F0 00143        INSV    R0, #2, #1, PROTO_VCB+83                        ;
                         19     AA    0000G  CF  D0 00149        MOVL    HOMEBLOCK_LBN, PROTO_VCB+36                     ; 0890
                 25       AA    0000G  CF    10  9C 0014F        ROTL    #16, HOME_BLOCK+2, PROTO_VCB+48                 ; 0892
                         31     AA           01  B0 00156        MOVW    #1, PROTO_VCB+60                                ; 0893
                         3D     AA    0000G  CF  90 0015A        MOVB    HOME_BLOCK+44, PROTO_VCB+72                     ; 0895
                                             04  12 00160        BNEQ    20$                                             ; 0896
                         3D     AA           07  90 00162        MOVB    #7, PROTO_VCB+72                                ; 0897
                                08           56  E9 00166  20$:  BLBC    R6, 21$                                         ; 0898
                         3D     AA 00000000G 9F  90 00169        MOVB    @#ACP$GB_WINDOW, PROTO_VCB+72                   ; 0899
                                06    0000G  CF  E9 00171  21$:  BLBC    MOUNT_OPTIONS+3, 22$                            ; 0900
                         3D     AA    0000G  CF  90 00176        MOVB    WINDOW, PROTO_VCB+72                            ; 0901
                         3E     AA    0000G  CF  90 0017C  22$:  MOVB    HOME_BLOCK+46, PROTO_VCB+73                     ; 0903
                                08           56  E9 00182        BLBC    R6, 23$                                         ; 0904
                         3E     AA 00000000G 9F  90 00185        MOVB    @#ACP$GW_SYSACC, PROTO_VCB+73                   ; 0905
                  06     0000G  CF           01  E1 0018D  23$:  BBC     #1, MOUNT_OPTIONS+3, 24$                        ; 0906
                         3E     AA    0000G  CF  90 00193        MOVB    ACCESSED, PROTO_VCB+73                          ; 0907
                  03     0000G  CF           04  E1 00199  24$:  BBC     #4, MOUNT_OPTIONS+6, 25$                        ; 0908
```

```
                              3E   AA   94 0019F        CLRB    PROTO_VCB+73                          ; 0909
              33    AA     0000G   CF   9B 001A2 25$:   MOVZBW  HOME_BLOCK+45, PROTO_VCB+62           ; 0911
                                   04   12 001A8        BNEQ    26$                                   ; 0912
              33    AA             05   B0 001AA        MOVW    #5, PROTO_VCB+62                      ; 0913
                           0000G   CF   95 001AE 26$:   TSTB    MOUNT_OPTIONS+2                       ; 0914
                                   06   18 001B2        BGEQ    27$                                   ;
              33    AA     0000G   CF   B0 001B4        MOVW    EXTENSION, PROTO_VCB+62               ; 0915
              2D    AA     0000G   CF   90 001BA 27$:   MOVB    HOME_BLOCK, PROTO_VCB+56              ; 0917
              39    AA     0000G   CF   3C 001C0        MOVZWL  HOME_BLOCK+6, PROTO_VCB+68            ; 0919
        04       0000G     CF      04   E1 001C6        BBC     #4, MOUNT_OPTIONS+6, 28$              ; 0921
                    48     AA      02   88 001CC        BISB2   #2, PROTO_VCB+83                      ; 0922
                    50        2D   AA   9A 001D0 28$:   MOVZBL  PROTO_VCB+56, R0                      ; 0928
                 0000G     CF  25 BA40  9E 001D4        MOVAB   @PROTO_VCB+48[R0], HEADER_LBN         ;
                           0000G   CF   9F 001DB        PUSHAB  BUFFER                                ; 0929
                           0000G   CF   DD 001DF        PUSHL   HEADER_LBN                            ;
                 0000G     CF      02   FB 001E3        CALLS   #2, READ_BLOCK                        ;
                           58      50   D0 001E8        MOVL    R0, STATUS                            ;
                           05      58   E8 001EB        BLBS    STATUS, 29$                           ; 0930
                                   58   DD 001EE        PUSHL   STATUS                                ;
                           6B      01   FB 001F0        CALLS   #1, LIB$STOP                          ;
                           0000'   CF   9F 001F3 29$:   PUSHAB  P.AAA                                 ; 0931
                           0000G   CF   9F 001F7        PUSHAB  BUFFER                                ;
                 0000G     CF      02   FB 001FB        CALLS   #2, CHECK_HEADER                      ;
                           05      50   E8 00200        BLBS    R0, 30$                               ;
                                   7E   D4 00203        CLRL    -(SP)                                 ;
                           6B      01   FB 00205        CALLS   #1, LIB$STOP                          ;
  00B4    8F          00   6E      00   2C 00208 30$:   MOVC5   #0, (SP), #0, #180, PROTO_FCB         ; 0933
                           0000G   CF      0020F                                                      ;
  00B4    8F          00   6E      00   2C 00212        MOVC5   #0, (SP), #0, #180, PROTO_FCBE1       ; 0938
                           0000'   CF      00219                                                      ;
  00B4    8F          00   6E      00   2C 0021C        MOVC5   #0, (SP), #0, #180, PROTO_FCBE2       ; 0939
                           0000'   CF      00223                                                      ;
                 0000G     CF      01   D0 00226        MOVL    #1, PROTO_FCB+44                      ; 0941
                           0000G   CF   9F 0022B        PUSHAB  BUFFER                                ; 0942
                           0000G   CF   9F 0022F        PUSHAB  PROTO_FCB                             ;
                 0000G     CF      02   FB 00233        CALLS   #2, INIT_FCB                          ;
                 0000G     CF      01   B0 00238        MOVW    #1, PROTO_FCB+26                      ; 0943
          30          00   6E      00   2C 0023D        MOVC5   #0, (SP), #0, #48, PROTO_WCB          ; 0948
                           0000G   CF      00242                                                      ;
                 0000G     CF  E4  8F   9B 00245        MOVZBW  #228, PROTO_WCB+8                     ; 0949
                 0000G     CF      01   88 0024B        BISB2   #1, PROTO_WCB+11                      ; 0950
                                   01   DD 00250        PUSHL   #1                                    ; 0951
                                   03   DD 00252        PUSHL   #3                                    ;
                           0000G   CF   9F 00254        PUSHAB  BUFFER                                ;
                           0000G   CF   9F 00258        PUSHAB  PROTO_WCB                             ;
                 0000G     CF      04   FB 0025C        CALLS   #4, TURN_WINDOW1                      ;
                           50  0000G   CF   9A 00261    MOVZBL  BUFFER+1, R0                          ; 0957
                           52  0000GCF40   3E 00266     MOVAW   BUFFER[R0], MAP_AREA                  ;
                           55      02   A2   3C 0026C   MOVZWL  2(MAP_AREA), R5                       ; 0958
                                   03   13 00270        BEQL    31$                                   ;
                              04   A2   B5 00272        TSTW    4(MAP_AREA)                           ;
                                   03   12 00275 31$:   BNEQ    32$                                   ;
                                 018A   31 00277        BRW     49$                                   ;
                           53      0A   A2   9E 0027A 32$: MOVAB 10(R2), MAP_POINTER                  ; 0961
                           51  0000G   CF   D0 0027E    MOVL    HEADER_LBN, R1                        ; 0967
                           56      08   A2   9A 00283   MOVZBL  8(MAP_AREA), J                        ;
                                   56   D6 00287        INCL    J                                     ;
```

```
                                    20   11 00289         BRB     35$                                            ;
                        50      02  A3   3C 0028B 33$:    MOVZWL  2(MAP_POINTER), INDEX_LBN                      ; 0964
  50          08        10          63   F0 0028F         INSV    (MAP_POINTER), #16, #8, INDEX_LBN              ; 0965
                        57      01  A3   9A 00294         MOVZBL  1(MAP_POINTER), INDEX_CNT                      ; 0966
                                    57   D6 00298         INCL    INDEX_CNT                                      ;
                        50          51   D1 0029A         CMPL    R1, INDEX_LBN                                  ; 0967
                                    09   1F 0029D         BLSSU   34$                                            ;
              54        50          57   C1 0029F         ADDL3   INDEX_CNT, INDEX_LBN, R4                       ; 0968
                        54          51   D1 002A3         CMPL    R1, R4                                         ;
                                    06   1F 002A6         BLSSU   36$                                            ;
                        53          04   C0 002A8 34$:    ADDL2   #4, MAP_POINTER                                ; 0969
                        DD          56   F5 002AB 35$:    SOBGTR  J, 33$                                         ; 0962
                        54     FF A541   9E 002AE 36$:    MOVAB   -1(R5)[R1], EXTENT_LBN                         ; 0975
              59        50          57   C1 002B3         ADDL3   INDEX_CNT, INDEX_LBN, R9                       ; 0976
                        54          59   D1 002B7         CMPL    R9, EXTENT_LBN                                 ;
                                    08   1E 002BA         BGEQU   37$                                            ;
                        7E    08C0  8F   3C 002BC         MOVZWL  #2240, -(SP)                                   ;
                        6B          01   FB 002C1         CALLS   #1, LIB$STOP                                   ;
                        6E          55   D0 002C4 37$:    MOVL    R5, EXTENT_FID                                 ; 0977
                   02   AE      04  A2   B0 002C7         MOVW    4(MAP_AREA), EXTENT_FID+2                      ; 0978
                             0000G  CF   9F 002CC         PUSHAB  BUFFER                                         ; 0983
                                    54   DD 002D0         PUSHL   EXTENT_LBN                                     ;
                0000G   CF          02   FB 002D2         CALLS   #2, READ_BLOCK                                 ;
                        58          50   D0 002D7         MOVL    R0, STATUS                                     ;
                        05          58   E8 002DA         BLBS    STATUS, 38$                                    ; 0984
                                    58   DD 002DD         PUSHL   STATUS                                         ;
                        6B          01   FB 002DF         CALLS   #1, LIB$STOP                                   ;
                                    5E   DD 002E2 38$:    PUSHL   SP                                             ; 0985
                             0000G  CF   9F 002E4         PUSHAB  BUFFER                                         ;
                0000G   CF          02   FB 002E8         CALLS   #2, CHECK_HEADER                               ;
                        05          50   E8 002ED         BLBS    R0, 39$                                        ;
                                    7E   D4 002F0         CLRL    -(SP)                                          ;
                        6B          01   FB 002F2         CALLS   #1, LIB$STOP                                   ;
        0000'  CF       0000G   CF  0000G  CF   C1 002F5 39$:    ADDL3   PROTO_FCB+44, PROTO_FCB+56, PROTO_FCBE1+44     ; 0986
                             0000G  CF   9F 002FF         PUSHAB  BUFFER                                         ; 0987
                             0000'  CF   9F 00303         PUSHAB  PROTO_FCBE1                                    ;
                0000G   CF          02   FB 00307         CALLS   #2, INIT_FCB                                   ;
                0000G   CF   0000'  CF   C0 0030C         ADDL2   PROTO_FCBE1+56, PROTO_FCB+56                   ; 0988
                0000'   CF          01   B0 00313         MOVW    #1, PROTO_FCBE1+26                             ; 0989
                0000'   CF          54   D0 00318         MOVL    EXTENT_LBN, PROTO_FCBE1+52                     ; 0990
                        57   0000G  CF   9E 0031D         MOVAB   PROTO_WCB+48, WCB_POINTER                      ; 0995
                        56          01   D0 00322         MOVL    #1, EXTENT_VBN                                 ; 0996
                        51   0000G  CF   3C 00325         MOVZWL  PROTO_WCB+22, R1                               ; 0997
                                    50   D4 0032A         CLRL    J                                              ;
                                    09   11 0032C         BRB     41$                                            ;
                        55          87   3C 0032E 40$:    MOVZWL  (WCB_POINTER)+, R5                             ; 0999
                        56          55   C0 00331         ADDL2   R5, EXTENT_VBN                                 ; 1000
                        57          04   C0 00334         ADDL2   #4, WCB_POINTER                                ; 1000
              F3        50          51   F3 00337 41$:    AOBLEQ  R1, J, 40$                                     ; 0997
                                    56   DD 0033B         PUSHL   EXTENT_VBN                                     ; 1002
                                    03   DD 0033D         PUSHL   #3                                             ;
                             0000G  CF   9F 0033F         PUSHAB  BUFFER                                         ;
                             0000G  CF   9F 00343         PUSHAB  PROTO_WCB                                      ;
                0000G   CF          04   FB 00347         CALLS   #4, TURN_WINDOW1                               ;
                        50   0000G  CF   9A 0034C         MOVZBL  BUFFER+1, R0                                   ; 1004
                        52   0000GCF40   3E 00351         MOVAW   BUFFER[R0], MAP_AREA                           ;
                        55      02  A2   3C 00357         MOVZWL  2(MAP_AREA), R5                                ; 1005
```

```
                                   03  12 0035B        BNEQ    43$
                                 00A4  31 0035D 42$:   BRW     49$
                          04       A2  B5 00360 43$:   TSTW    4(MAP_AREA)
                                   F8  13 00363        BEQL    42$
         50          55  0000G     CF  C1 00365        ADDL3   HEADER_LBN, R5, R0                    ; 1012
                     54     FF     A0  9E 0036B        MOVAB   -1(R0), EXTENT_LBN
                     54            59  D1 0036F        CMPL    R9, EXTENT_LBN                        ; 1013
                                   08  1E 00372        BGEQU   44$
                     7E   08C0     8F  3C 00374        MOVZWL  #2240, -(SP)
                     6B            01  FB 00379        CALLS   #1, LIB$STOP
                     6E            55  D0 0037C 44$:   MOVL    R5, EXTENT_FID                        ; 1014
             02      AE     04     A2  B0 0037F        MOVW    4(MAP_AREA), EXTENT_FID+2             ; 1015
                          0000G    CF  9F 00384        PUSHAB  BUFFER                                ; 1020
                                   54  DD 00388        PUSHL   EXTENT_LBN
              0000G  CF            02  FB 0038A        CALLS   #2, READ_BLOCK
                     58            50  D0 0038F        MOVL    R0, STATUS
                     05            58  E8 00392        BLBS    STATUS, 45$                           ; 1021
                                   58  DD 00395        PUSHL   STATUS
                     6B            01  FB 00397        CALLS   #1, LIB$STOP
                                   5E  DD 0039A 45$:   PUSHL   SP                                    ; 1022
                          0000G    CF  9F 0039C        PUSHAB  BUFFER
              0000G  CF            02  FB 003A0        CALLS   #2, CHECK_HEADER
                     05            50  E8 003A5        BLBS    R0, 46$
                                   7E  D4 003A8        CLRL    -(SP)
                     6B            01  FB 003AA        CALLS   #1, LIB$STOP
  0000'  CF   0000'  CF   0000'    CF  C1 003AD 46$:   ADDL3   PROTO_FCBE1+44, PROTO_FCBE1+56, -     ; 1023
                                                               PROTO_FCBE2+44
                          0000G    CF  9F 003B7        PUSHAB  BUFFER                                ; 1024
                          0000'    CF  9F 003BB        PUSHAB  PROTO_FCBE2
              0000G  CF            02  FB 003BF        CALLS   #2, INIT_FCB
              0000G  CF   0000'    CF  C0 003C4        ADDL2   PROTO_FCBE2+56, PROTO_FCB+56          ; 1025
              0000'  CF            01  B0 003CB        MOVW    #1, PROTO_FCBE2+26                    ; 1026
              0000'  CF            54  D0 003D0        MOVL    EXTENT_LBN, PROTO_FCBE2+52            ; 1027
                     57   0000G    CF  9E 003D5        MOVAB   PROTO_WCB+48, WCB_POINTER             ; 1032
                     56            01  D0 003DA        MOVL    #1, EXTENT_VBN                        ; 1033
                     51   0000G    CF  3C 003DD        MOVZWL  PROTO_WCB+22, R1                      ; 1034
                                   50  D4 003E2        CLRL    J
                                   09  11 003E4        BRB     48$
                     54            87  3C 003E6 47$:   MOVZWL  (WCB_POINTER)+, R4                    ; 1036
                     56            54  C0 003E9        ADDL2   R4, EXTENT_VBN
                     57            04  C0 003EC        ADDL2   #4, WCB_POINTER                       ; 1037
         F3          50            51  F3 003EF 48$:   AOBLEQ  R1, J, 47$                            ; 1034
                                   56  DD 003F3        PUSHL   EXTENT_VBN                            ; 1039
                                   03  DD 003F5        PUSHL   #3
                          0000G    CF  9F 003F7        PUSHAB  BUFFER
                          0000G    CF  9F 003FB        PUSHAB  PROTO_WCB
              0000G  CF            04  FB 003FF        CALLS   #4, TURN_WINDOW1
                          0000G    CF  9F 00404 49$:   PUSHAB  BUFFER                                ; 1049
                     50     2D     AA  9A 00408        MOVZBL  PROTO_VCB+56, R0
                     50     25     AA  C0 0040C        ADDL2   PROTO_VCB+48, R0
                            01     A0  9F 00410        PUSHAB  1(R0)
              0000G  CF            02  FB 00413        CALLS   #2, READ_BLOCK
                     58            50  D0 00418        MOVL    R0, STATUS
                     10            58  E9 0041B        BLBC    STATUS, 50$                           ; 1050
                          0000'    CF  9F 0041E        PUSHAB  P.AAB
                          0000G    CF  9F 00422        PUSHAB  BUFFER
              0000G  CF            02  FB 00426        CALLS   #2, CHECK_HEADER
```

```
                                26      50 E8 0042B       BLBS    R0, 53$                        ; 1053
                      00000254  8F      58 D1 0042E 50$:  CMPL    STATUS, #596
                                        0A 12 00435       BNEQ    51$
                                7E  0254 8F 3C 00437      MOVZWL  #596, -(SP)                    ; 1055
                                6B      01 FB 0043C       CALLS   #1, LIB$STOP
                                        0D 11 0043F       BRB     52$
                         00729010       8F DD 00441 51$:  PUSHL   #7507984                       ; 1057
                      00000000G 00      01 FB 00447       CALLS   #1, LIB$SIGNAL
                                6A      10 88 0044E 52$:  BISB2   #16, PROTO_VCB+11              ; 1058
                                   0171 31 00451          BRW     71$                            ; 1050
                                50  0000G CF 9A 00454 53$: MOVZBL BUFFER+1, R0                   ; 1063
                                52  0000GCF40 3E 00459    MOVAW   BUFFER[R0], MAP_AREA
                                53    0A A2 9E 0045F      MOVAB   10(R2), MAP_POINTER            ; 1064
                                54      01 D0 00463       MOVL    #1, BIAS                       ; 1066
                                04    08 A2 91 00466      CMPB    8(MAP_AREA), #4                ; 1067
                                        06 1A 0046A       BGTRU   54$
                                02    08 A2 91 0046C      CMPB    8(MAP_AREA), #2                ; 1068
                                        08 1E 00470       BGEQU   55$
                                7E  08C0 8F 3C 00472 54$: MOVZWL  #2240, -(SP)                   ; 1069
                                6B      01 FB 00477       CALLS   #1, LIB$STOP
                                04    08 A2 91 0047A 55$: CMPB    8(MAP_AREA), #4                ; 1070
                                        05 12 0047E       BNEQ    56$
                                        54 D4 00480       CLRL    BIAS                           ; 1073
                                53      04 C0 00482       ADDL2   #4, MAP_POINTER                ; 1074
                                52    01 A3 9A 00485 56$: MOVZBL  1(MAP_POINTER), R2             ; 1077
                                52      54 C2 00489       SUBL2   BIAS, R2
                                        52 D6 0048C       INCL    COUNT
                                56    02 A3 3C 0048E      MOVZWL  2(MAP_POINTER), LBN            ; 1078
      56              08        10      63 F0 00492       INSV    (MAP_POINTER), #16, #8, LBN    ; 1079
                                56      54 C0 00497       ADDL2   BIAS, LBN                      ; 1080
                           29   AA      56 D0 0049A       MOVL    LBN, PROTO_VCB+52              ; 1082
                           2E   AA      52 90 0049E       MOVB    COUNT, PROTO_VCB+57            ; 1083
                   5E  0000G    CF      01 E1 004A2       BBC     #1, MOUNT_OPTIONS+1, 60$       ; 1091
                                  0000G CF 9F 004A8       PUSHAB  BUFFER                         ; 1093
                                        56 DD 004AC       PUSHL   LBN
                          0000G CF      02 FB 004AE       CALLS   #2, READ_BLOCK
                                50      50 E9 004B3       BLBC    R0, 60$
                                  0000G CF 9F 004B6       PUSHAB  BUFFER                         ; 1095
                                        56 DD 004BA       PUSHL   LBN
                          0000G CF      02 FB 004BC       CALLS   #2, WRITE_BLOCK
                                58      50 D0 004C1       MOVL    R0, STATUS
                                3F      58 E8 004C4       BLBS    STATUS, 60$
                      00000254  8F      58 D1 004C7       CMPL    STATUS, #596                   ; 1098
                                        08 12 004CE       BNEQ    57$
                                7E  0254 8F 3C 004D0      MOVZWL  #596, -(SP)                    ; 1100
                                6B      01 FB 004D5       CALLS   #1, LIB$STOP
                      0000025C  8F      58 D1 004D8 57$:  CMPL    STATUS, #604                   ; 1101
                                        0F 12 004DF       BNEQ    58$
                         0072A013       8F DD 004E1       PUSHL   #7512083                       ; 1102
                      00000000G 00      01 FB 004E7       CALLS   #1, LIB$SIGNAL
                                        11 11 004EE       BRB     59$
                                        58 DD 004F0 58$:  PUSHL   STATUS                         ; 1103
                                        7E D4 004F2       CLRL    -(SP)
                         00729048       8F DD 004F4       PUSHL   #7508040
                      00000000G 00      03 FB 004FA       CALLS   #3, LIB$SIGNAL
                          0000G CF      02 8A 00501 59$:  BICB2   #2, MOUNT_OPTIONS+1            ; 1104
                                        54 D4 00506 60$:  CLRL    FREE                           ; 1110
```

```
                  53        01  A2  9E 00508       MOVAB   1(R2), J                          ; 1111
                                75  11 0050C       BRB     68$                               ;
                         0000G  CF  9F 0050E 61$:  PUSHAB  BUFFER                            ; 1115
                                56  DD 00512       PUSHL   LBN                               ;
                 0000G  CF      02  FB 00514       CALLS   #2, READ_BLOCK                    ;
                        58      50  D0 00519       MOVL    R0, STATUS                        ;
                        27      58  E8 0051C       BLBS    STATUS, 64$                       ; 1116
              00000254  8F      58  D1 0051F       CMPL    STATUS, #596                      ; 1119
                                0A  12 00526       BNEQ    62$                               ;
                  7E      0254  8F  3C 00528       MOVZWL  #596, -(SP)                       ; 1121
                  6B            01  FB 0052D       CALLS   #1, LIB$STOP                      ;
                                11  11 00530       BRB     63$                               ;
                                58  DD 00532 62$:  PUSHL   STATUS                            ; 1123
                                7E  D4 00534       CLRL    -(SP)                             ;
                      00729020  8F  DD 00536       PUSHL   #7508000                          ;
             00000000G  00      03  FB 0053C       CALLS   #3, LIB$SIGNAL                    ;
                        6A      10  88 00543 63$:  BISB2   #16, PROTO_VCB+11                 ; 1124
                                50  D4 00546 64$:  CLRL    I                                 ; 1127
                  57     0000GCF40  D0 00548 65$:  MOVL    BUFFER[I], X                      ; 1129
                                29  13 0054E       BEQL    67$                               ; 1130
                                52  D4 00550       CLRL    B2                                ; 1133
                  51        E0  A2  9E 00552 66$:  MOVAB   -32(B2), R1                       ; 1136
                  51            51  CE 00556       MNEGL   R1, R1                            ;
      55          57        51  52  EA 00559       FFS     B2, R1, X, B1                     ;
                                19  13 0055E       BEQL    67$                               ;
                  51        E0  A5  9E 00560       MOVAB   -32(B1), R1                       ; 1138
                  51            51  CE 00564       MNEGL   R1, R1                            ;
      52          57        51  55  EB 00567       FFC     B1, R1, X, B2                     ;
                  51        54  52  C1 0056C       ADDL3   B2, FREE, R1                      ; 1139
                  54        51  55  C3 00570       SUBL3   B1, R1, FREE                      ;
                  20            52  D1 00574       CMPL    B2, #32                           ; 1140
                                D9  19 00577       BLSS    66$                               ;
                  C7    50 0000007F 8F  F3 00579 67$:  AOBLEQ  #127, I, 65$                  ; 1127
                                56  D6 00581       INCL    LBN                               ; 1144
                        88      53  F5 00583 68$:  SOBGTR  J, 61$                            ; 1111
                    35  AA      54  D0 00586       MOVL    FREE, PROTO_VCB+64                ; 1147
                                39  11 0058A       BRB     71$                               ; 0886
                  33     0000G  CF  04  E0 0058C 69$:  BBS     #4, MOUNT_OPTIONS, 71$        ; 1158
                         0000G  CF  00  FB 00592       CALLS   #0, GET_VOLUME_LOCK_NAME      ; 1161
                                7E  D4 00597       CLRL    -(SP)                             ; 1162
                                5E  DD 00599       PUSHL   SP                                ;
                         0000G  CF  9F 0059B       PUSHAB  GET_VOLUME_LOCK                   ;
             00000000G  9F      03  FB 0059F       CALLS   #3, @#SYS$CMKRNL                  ;
                        58      50  D0 005A6       MOVL    R0, STATUS                        ;
                        05      58  E8 005A9       BLBS    STATUS, 70$                       ;
                                58  DD 005AC       PUSHL   STATUS                            ; 1164
                        6B      01  FB 005AE       CALLS   #1, LIB$STOP                      ;
      50         0000G  CF     0000G  CF  8D 005B1 70$:  XORB3   DEV_CTX, VOL_CTX, R0        ; 1165
                        09      50  E9 005B9       BLBC    R0, 71$                           ;
                      007280B4  8F  DD 005BC       PUSHL   #7504052                          ; 1167
                        6B      01  FB 005C2       CALLS   #1, LIB$STOP                      ;
      03         0000G  CF      05  E1 005C5 71$:  BBC     #5, MOUNT_OPTIONS+6, 72$          ; 1175
                        6A      10  8A 005CB       BICB2   #16, PROTO_VCB+11                 ; 1176
                                7E  D4 005CE 72$:  CLRL    -(SP)                             ; 1178
                                5E  DD 005D0       PUSHL   SP                                ;
                         0000V  CF  9F 005D2       PUSHAB  MAKE_DISK_MOUNT                   ;
             00000000G  9F      03  FB 005D6       CALLS   #3, @#SYS$CMKRNL                  ;
```

```
                  58                50 D0 005DD         MOVL     R0, STATUS                              ;
                  05                58 E8 005E0         BLBS     STATUS, 73$                             ; 1179
                                    58 DD 005E3         PUSHL    STATUS                                  ;
                  6B                01 FB 005E5         CALLS    #1, LIB$STOP                            ;
     50      0000G CF               01 78 005E8  73$:   ASHL     #1, DEVICE_INDEX, R0                    ; 1184
                          0000GCF40 DF 005EE            PUSHAL   PHYS_NAME[R0]                           ;
                             09 AA  9F 005F3            PUSHAB   PROTO_VCB+20                            ;
                                0C  DD 005F6            PUSHL    #12                                     ;
                                03  DD 005F8            PUSHL    #3                                      ;
                        0072A003 8F DD 005FA            PUSHL    #7512067                                ;
          00000000G 00          05  FB 00600            CALLS    #5, LIB$SIGNAL                          ;
                                    04 00607            RET                                              ; 1186
                                  0000 00608     74$:   .WORD    Save nothing                            ; 0728
                                7E  D4 0060A            CLRL     -(SP)                                   ;
                                5E  DD 0060C            PUSHL    SP                                      ;
                  7E         04 AC  7D 0060E            MOVQ     4(AP), -(SP)                            ;
             0000V CF           03  FB 00612            CALLS    #3, MOUNT_HANDLER                       ;
                                    04 00617            RET                                              ;

; Routine Size:  1560 bytes,    Routine Base:  $CODE$ + 0000

;  658         1187  1
```

```
1
1 ROUTINE MOUNT_HANDLER (SIGNAL, MECHANISM) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is the condition handler for the main disk mount
1 !       code. It undoes any damage done so far and returns the error
1 !       status to the user mode caller.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       MOUNT_HANDLER (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of signal vector
1 !       ARG2: address of mechanism vector
1 !
1 ! IMPLICIT INPUTS:
1 !       global pointers to blocks allocated
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       SS$_RESIGNAL
1 !
1 ! SIDE EFFECTS:
1 !       necessary cleanups done
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       SIGNAL          : REF BBLOCK,   ! signal vector
2       MECHANISM       : REF BBLOCK;   ! mechanism vector
2
2 EXTERNAL
2       MOUNT_OPTIONS   : BITVECTOR,    ! command parser options
2       CLEANUP_FLAGS   : BITVECTOR;    ! cleanup action flags
2
2 EXTERNAL ROUTINE
2       LOCK_CLEANUP    : NOVALUE;      ! cleanup dev and vol locks.
2
2
2 ! Note that cleanup is done if we are unwinding, which occurrs when
2 ! we take an error exit.
2 !
2
3 IF  (.SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND)
3 AND ((.BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_SEVERE) OR
3      (.BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_ERROR))
```

```
;  717         1245  2 THEN
;  718         1246  2     LOCK_CLEANUP ();
;  719         1247  2
;  720         1248  2 SS$_RESIGNAL
;  721         1249  1 END;                                  ! end of routine MOUNT_HANDLER


                                                            .EXTRN  CLEANUP_FLAGS, LOCK_CLEANUP

                                         0000 00000 MOUNT_HANDLER:
                                                            .WORD   Save nothing                        ; 1189
                          50       04   AC  D0 00002        MOVL    SIGNAL, R0                          ; 1242
                 00000920 8F       04   A0  D1 00006        CMPL    4(R0), #2336                        ;
                                        15  13 0000E        BEQL    2$                                  ;
       04      04   A0    03            00  ED 00010        CMPZV   #0, #3, 4(R0), #4                   ; 1243
                                        08  13 00016        BEQL    1$                                  ;
       02      04   A0    03            00  ED 00018        CMPZV   #0, #3, 4(R0), #2                   ; 1244
                                        05  12 0001E        BNEQ    2$                                  ;
                    0000G CF            00  FB 00020 1$:    CALLS   #0, LOCK_CLEANUP                    ; 1246
                          50     0918   8F  3C 00025 2$:    MOVZWL  #2328, R0                           ; 1249
                                            04 0002A        RET                                         ;

; Routine Size:  43 bytes,    Routine Base:  $CODE$ + 0618


;  722         1250  1
```

```
1 ROUTINE MAKE_DISK_MOUNT =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine does all of the data base manipulation needed to get
1 !       a volume actually mounted. It allocates the real VCB, FCB, and
1 !       window, and hooks then all together. It also starts up the ACP
1 !       gets the mounted volume list entry made.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       MAKE_DISK_MOUNT ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       MOUNT parser data base
1 !       own storage of this module
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1 if successful
1 !       status values if not
1 !
1 ! SIDE EFFECTS:
1 !       volume mounted
1 !
1 !--
1
2 BEGIN
2
2 BUILTIN
2       INSQUE;
2
2 LOCAL
2       WINDOW_SIZE,                    ! size in bytes needed for window
2       UCB             : REF BBLOCK,   ! pointer to volume UCB
2       ORB             : REF BBLOCK,   ! Pointer to device ORB
2       EXTENT1_FCB     : REF BBLOCK,   ! pointer to first extent FCB
2       EXTENT2_FCB     : REF BBLOCK;   ! pointer to second extent FCB
2
2 EXTERNAL
2       SCS$GB_NODENAME : ADDRESSING_MODE (GENERAL),
2                                       ! identify this node uniquely.
2       MOUNT_OPTIONS   : BITVECTOR,    ! command parser options
2       CLEANUP_FLAGS   : BITVECTOR,    ! cleanup action flags
2       CHANNEL,                        ! channel assigned to device
2       HOME_BLOCK      : BBLOCK,       ! buffer containing home block
2       OWNER_UIC,                      ! owner UIC from command
```

```
2           PROTECTION,                          ! volume protection from command
2           REAL_VCB        : REF BBLOCK ADDRESSING_MODE (GENERAL) ,         ! address of VCB allocated
2           REAL_FCB        : REF BBLOCK,        ! address of FCB allocated
2           REAL_WCB        : REF BBLOCK,        ! address of window allocated
2           CTL$GL_VOLUMES  : ADDRESSING_MODE (ABSOLUTE);
2                                                ! count of volumes mounted by process
2
2   EXTERNAL ROUTINE
2           STORE_CONTEXT : NOVALUE,             ! store device context
2           GET_CHANNELUCB,                      ! get UCB assigned to channel
2           ALLOCATE_MEM,                        ! allocate system dynamic memory
2           START_ACP,                           ! start and connect ACP to device
2           SET_DATACHECK,                       ! set volume data check attributes
2           LOCK_IODB       : ADDRESSING_MODE (GENERAL),
2                                                ! lock I/O database mutex
2           UNLOCK_IODB     : ADDRESSING_MODE (GENERAL),
2                                                ! unlock I/O database mutex
2           ALLOC_LOGNAME,                       ! create logical name and MTL blocks
2           ENTER_LOGNAME,                       ! enter logical name and MTL in lists
2           SEND_ERRLOG;                         ! send message to error logger
2
2
2   ! Allocate all of the required control blocks. We allocate them in
2   ! advance to avoid having to back out of some awkward situations later on.
2   ! The one exception is the AQB, which is either found or allocated by
2   ! START_ACP.
2   !
2
2   ENABLE KERNEL_HANDLER;
2
2   REAL_VCB = ALLOCATE_MEM (VCB$C_LENGTH, 0);
2   REAL_VCB[VCB$B_TYPE] = DYN$C_VCB;
2   CH$MOVE (VCB$C_LENGTH-11, PROTO_VCB+11, .REAL_VCB+11);
2
2   IF NOT .MOUNT_OPTIONS[OPT_FOREIGN]
2   THEN
3       BEGIN
3       REAL_VCB[VCB$L_FCBFL] = REAL_VCB[VCB$L_FCBFL];
3       REAL_VCB[VCB$L_FCBBL] = REAL_VCB[VCB$L_FCBFL];
3
3       REAL_FCB = ALLOCATE_MEM (FCB$C_LENGTH, 0);
3       REAL_FCB[FCB$B_TYPE] = DYN$C_FCB;
3       CH$MOVE (FCB$C_LENGTH-11, PROTO_FCB+11, .REAL_FCB+11);
3       REAL_FCB[FCB$L_WLFL] = REAL_FCB[FCB$L_WLFL];
3       REAL_FCB[FCB$L_WLBL] = REAL_FCB[FCB$L_WLFL];
3       INSQUE (.REAL_FCB, REAL_VCB[VCB$L_FCBFL]);
3
3   ! If extension headers exist, allocate room for them and link them into the list
3   !
3
3       IF .PROTO_FCBE1[FCB$L_FILESIZE] NEQ 0
3       THEN
4           BEGIN
4           EXTENT1_FCB = ALLOCATE_MEM (FCB$C_LENGTH, 0);
4           EXTENT1_FCB[FCB$B_TYPE] = DYN$C_FCB;
4           CH$MOVE (FCB$C_LENGTH-11, PROTO_FCBE1+11, .EXTENT1_FCB+11);
4           REAL_FCB[FCB$L_EXFCB] = .EXTENT1_FCB;
```

```
                EXTENT1_FCB[FCB$L_WLFL] = EXTENT1_FCB[FCB$L_WLFL];
                EXTENT1_FCB[FCB$L_WLBL] = EXTENT1_FCB[FCB$L_WLFL];
                INSQUE (.EXTENT1_FCB, REAL_FCB[FCB$L_FCBFL]);
                IF .PROTO_FCBE2[FCB$L_FILESIZE] NEQ 0
                THEN
                    BEGIN
                    EXTENT2_FCB = ALLOCATE_MEM (FCB$C_LENGTH, 0);
                    EXTENT2_FCB[FCB$B_TYPE] = DYN$C_FCB;
                    CH$MOVE (FCB$C_LENGTH-11, PROTO_FCBE2+11, .EXTENT2_FCB+11);
                    EXTENT1_FCB[FCB$L_EXFCB] = .EXTENT2_FCB;
                    EXTENT2_FCB[FCB$L_WLFL] = EXTENT2_FCB[FCB$L_WLFL];
                    EXTENT2_FCB[FCB$L_WLBL] = EXTENT2_FCB[FCB$L_WLFL];
                    INSQUE (.EXTENT2_FCB, EXTENT1_FCB[FCB$L_FCBFL]);
                    END;
                END;

            WINDOW_SIZE = WCB$C_LENGTH + MAXU (.PROTO_WCB[WCB$W_NMAP] + 2, 6) * 6;
            REAL_WCB = ALLOCATE_MEM (.WINDOW_SIZE, 0);
            REAL_WCB[WCB$B_TYPE] = DYN$C_WCB;
            CH$MOVE (.WINDOW_SIZE-11, PROTO_WCB+11, .REAL_WCB+11);
            REAL_WCB[WCB$L_FCB] = .REAL_FCB;
            INSQUE (.REAL_WCB, REAL_FCB[FCB$L_WLFL]);
            END;

        ALLOC_LOGNAME (0);

        ! All data blocks except the AQB are now allocated. First set up the
        ! volume ownership and protection in the VCB. Now hook up the blocks
        ! to the device data base and start the ACP.
        !

        UCB = GET_CHANNELUCB (.CHANNEL);
        ORB = .UCB[UCB$L_ORB];
        REAL_VCB[VCB$L_RVT] = .UCB;

        UCB[UCB$V_UNLOAD] = NOT .MOUNT_OPTIONS [OPT_NOUNLOAD];
        ORB[ORB$L_OWNER] = .VOLUME_UIC;
        IF .MOUNT_OPTIONS[OPT_OWNER_UIC]
        THEN ORB[ORB$L_OWNER] = .OWNER_UIC;

        ORB[ORB$V_PROT_16] = 1;                        ! SOGW protection word
        IF .MOUNT_OPTIONS[OPT_FOREIGN]
        THEN ORB[ORB$W_PROT] = %X'FF00'
        ELSE ORB[ORB$W_PROT] = .HOME_BLOCK[HM1$W_PROTECT];
        IF .MOUNT_OPTIONS[OPT_PROTECTION]
        THEN ORB[ORB$W_PROT] = .PROTECTION;

        IF NOT .MOUNT_OPTIONS[OPT_FOREIGN]
        THEN
            BEGIN

        ! Fill in name used to identify volume for locking purposes.
        ! This uniquely identifies this volume on this node for RMS,
        ! and eliminates device naming problems if the drive is
        ! multi-ported.  It does not, however, make any attempt to
        ! generate a useful name for cluster wide access because
        ! cluster wide access to structure level 1 volumes is not
```

```
3 ! supported other than in a one writer, multi-reader mode.
3 !
3
3       CH$MOVE (8, SCS$GB_NODENAME, REAL_VCB [VCB$T_VOLCKNAM]);
3       (REAL_VCB [VCB$T_VOLCKNAM] + 8) = .UCB;
3
3       REAL_WCB[WCB$L_ORGUCB] = .UCB;
3       START_ACP (.UCB, .REAL_VCB, AQB$K_F11V1);
3       END
2 ELSE
3       BEGIN
3       LOCK_IODB ();
3       UCB[UCB$L_VCB] = .REAL_VCB;
3       UCB[UCB$L_DEVCHAR] = .UCB[UCB$L_DEVCHAR]
3                            OR (DEV$M_MNT OR DEV$M_DIR OR DEV$M_FOR);
3       SET_DATACHECK (.UCB, 0);
3       UNLOCK_IODB ();
2       END;
2
2 IF .MOUNT_OPTIONS[OPT_NOSHARE] AND .CLEANUP_FLAGS[CLF_DEALLOCATE]
2 THEN UCB[UCB$V_DEADMO] = 1;
2
2 IF NOT .MOUNT_OPTIONS[OPT_WRITE]
2 THEN BBLOCK [UCB[UCB$L_DEVCHAR], DEV$V_SWL] = 1;
2
2 ! Enter the logical name for the volume; bump the user's volume mount count,
2 ! and make the error log entry for the mount.
2 !
2
2 ENTER_LOGNAME (.UCB, .REAL_VCB);
2 CTL$GL_VOLUMES = .CTL$GL_VOLUMES + 1;
2 SEND_ERRLOG (1, .UCB);
2
2 ! Increment the refcount, so that it never goes to zero while the device is
2 ! mounted.
2 !
2 UCB[UCB$W_REFC] = .UCB[UCB$W_REFC] + 1;
2
2 ! Store device context if cluster available.
2 !
2
2 STORE_CONTEXT ();
2
2 RETURN 1;
2
1 END;                                         ! end of routine MAKE_DISK_MOUNT


                                                  .EXTRN  SCS$GB_NODENAME
                                                  .EXTRN  CHANNEL, OWNER_UIC
                                                  .EXTRN  PROTECTION, REAL_VCB
                                                  .EXTRN  REAL_FCB, REAL_WCB
                                                  .EXTRN  CTL$GL_VOLUMES, STORE_CONTEXT
                                                  .EXTRN  GET_CHANNELUCB, ALLOCATE_MEM
                                                  .EXTRN  START_ACP, SET_DATACHECK
                                                  .EXTRN  LOCK_IODB, UNLOCK_IODB
```

```
                                                        .EXTRN  ALLOC_LOGNAME, ENTER_LOGNAME
                                                        .EXTRN  SEND_ERRLOG

                                  0FFC 00000 MAKE_DISK_MOUNT:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 1251
              5B      0000G   CF   9E 00002             MOVAB   ALLOCATE_MEM, R11                             ;
              5A      0000G   CF   9E 00007             MOVAB   REAL_FCB, R10                                 ;
              59      0000G   CF   9E 0000C             MOVAB   MOUNT_OPTIONS, R9                             ;
              58  00000000G   00   9E 00011             MOVAB   REAL_VCB, R8                                  ;
              6D       01E1   CF   DE 00018             MOVAL   13$, (FP)                                     ; 1288
                              7E   D4 0001D             CLRL    -(SP)                                         ; 1338
              7E         EC   8F   9A 0001F             MOVZBL  #236, -(SP)                                   ;
                              6B   02 FB 00023          CALLS   #2, ALLOCATE_MEM                              ;
                              68   50 D0 00026          MOVL    R0, REAL_VCB                                  ;
                              56   68 D0 00029          MOVL    REAL_VCB, R6                                  ;
                        0A    A6   11 90 0002C          MOVB    #17, 10(R6)                                   ; 1339
    0B    A6  0000G   CF    00E1   8F   28 00030        MOVC3   #225, PROTO_VCB+11, 11(R6)                    ; 1340
          03     01   A9           03   E1 00039        BBC     #3, MOUNT_OPTIONS+1, 1$                       ; 1342
                            00D7   31 0003E             BRW     4$                                            ;
                        66    56   D0 00041 1$:         MOVL    R6, (R6)                                      ; 1345
                  04    A6    56   D0 00044             MOVL    R6, 4(R6)                                     ; 1346
                              7E   D4 00048             CLRL    -(SP)                                         ; 1348
              7E         B4   8F   9A 0004A             MOVZBL  #180, -(SP)                                   ;
                              6B   02 FB 0004E          CALLS   #2, ALLOCATE_MEM                              ;
                              6A   50 D0 00051          MOVL    R0, REAL_FCB                                  ;
                              56   6A D0 00054          MOVL    REAL_FCB, R6                                  ;
                        0A    A6   07 90 00057          MOVB    #7, 10(R6)                                    ; 1349
    0B    A6  0000G   CF    00A9   8F   28 0005B        MOVC3   #169, PROTO_FCB+11, 11(R6)                    ; 1350
              10     A6      10   A6   9E 00064         MOVAB   16(R6), 16(R6)                                ; 1351
              14     A6      10   A6   9E 00069         MOVAB   16(R6), 20(R6)                                ; 1352
                        50    68   9E 0006E             MOVAB   REAL_VCB, R0                                  ; 1353
                  00    B0    66   0E 00071             INSQUE  (R6), @0(R0)                                  ;
                       0000'  CF   D5 00075             TSTL    PROTO_FCBE1+56                                ; 1358
                              5D   13 00079             BEQL    2$                                            ;
                              7E   D4 0007B             CLRL    -(SP)                                         ; 1361
              7E         B4   8F   9A 0007D             MOVZBL  #180, -(SP)                                   ;
                              6B   02 FB 00081          CALLS   #2, ALLOCATE_MEM                              ;
                              57   50 D0 00084          MOVL    R0, EXTENT1_FCB                               ;
                        0A    A7   07 90 00087          MOVB    #7, 10(EXTENT1_FCB)                           ; 1362
    0B    A7  0000'   CF    00A9   8F   28 0008B        MOVC3   #169, PROTO_FCBE1+11, 11(EXTENT1_FCB)         ; 1363
                        50    6A   D0 00094             MOVL    REAL_FCB, R0                                  ; 1364
                  0C    A0    57   D0 00097             MOVL    EXTENT1_FCB, 12(R0)                           ;
              10     A7      10   A7   9E 0009B         MOVAB   16(EXTENT1_FCB), 16(EXTENT1_FCB)              ; 1365
              14     A7      10   A7   9E 000A0         MOVAB   16(EXTENT1_FCB), 20(EXTENT1_FCB)              ; 1366
                        60    67   0E 000A5             INSQUE  (EXTENT1_FCB), (R0)                           ; 1367
                       0000'  CF   D5 000A8             TSTL    PROTO_FCBE2+56                                ; 1368
                              2A   13 000AC             BEQL    2$                                            ;
                              7E   D4 000AE             CLRL    -(SP)                                         ; 1371
              7E         B4   8F   9A 000B0             MOVZBL  #180, -(SP)                                   ;
                              6B   02 FB 000B4          CALLS   #2, ALLOCATE_MEM                              ;
                              56   50 D0 000B7          MOVL    R0, EXTENT2_FCB                               ;
                        0A    A6   07 90 000BA          MOVB    #7, 10(EXTENT2_FCB)                           ; 1372
    0B    A6  0000'   CF    00A9   8F   28 000BE        MOVC3   #169, PROTO_FCBE2+11, 11(EXTENT2_FCB)         ; 1373
                  0C    A7    56   D0 000C7             MOVL    EXTENT2_FCB, 12(EXTENT1_FCB)                  ; 1374
              10     A6      10   A6   9E 000CB         MOVAB   16(EXTENT2_FCB), 16(EXTENT2_FCB)              ; 1375
              14     A6      10   A6   9E 000D0         MOVAB   16(EXTENT2_FCB), 20(EXTENT2_FCB)              ; 1376
                        67    66   0E 000D5             INSQUE  (EXTENT2_FCB), (EXTENT1_FCB)                  ; 1377
```

```
                          52        0000G  CF   3C 000D8 2$:    MOVZWL  PROTO_WCB+22, R2                    ; 1381
                          52               02   C0 000DD        ADDL2   #2, R2                              ;
                          06               52   D1 000E0        CMPL    R2, #6                              ;
                                           03   1E 000E3        BGEQU   3$                                  ;
                          52               06   D0 000E5        MOVL    #6, R2                              ;
                          52               06   C4 000E8 3$:    MULL2   #6, R2                              ;
                          52               30   C0 000EB        ADDL2   #48, WINDOW_SIZE                    ;
                                           7E   D4 000EE        CLRL    -(SP)                               ; 1382
                                           52   DD 000F0        PUSHL   WINDOW_SIZE                         ;
                          6B               02   FB 000F2        CALLS   #2, ALLOCATE_MEM                    ;
                   0000G  CF               50   D0 000F5        MOVL    R0, REAL_WCB                        ;
                          56        0000G  CF   D0 000FA        MOVL    REAL_WCB, R6                        ; 1383
                     0A   A6               12   90 000FF        MOVB    #18, 10(R6)                         ;
                          52               0B   C2 00103        SUBL2   #11, R2                             ; 1384
       0B    A6    0000G  CF               52   28 00106        MOVC3   R2, PROTO_WCB+11, 11(R6)            ;
                     18   A6               6A   D0 0010D        MOVL    REAL_FCB, 24(R6)                    ; 1385
             50           6A               10   C1 00111        ADDL3   #16, REAL_FCB, R0                   ; 1386
                          60               66   0E 00115        INSQUE  (R6), (R0)                          ;
                                           7E   D4 00118 4$:    CLRL    -(SP)                               ; 1389
                   0000G  CF               01   FB 0011A        CALLS   #1, ALLOC_LOGNAME                   ;
                                    0000G  CF   DD 0011F        PUSHL   CHANNEL                             ; 1396
                   0000G  CF               01   FB 00123        CALLS   #1, GET_CHANNELUCB                  ;
                          56               50   D0 00128        MOVL    R0, UCB                             ;
                          50           1C  A6   D0 0012B        MOVL    28(UCB), ORB                        ; 1397
                          57               68   D0 0012F        MOVL    REAL_VCB, R7                        ; 1398
                     20   A7               56   D0 00132        MOVL    UCB, 32(R7)                         ;
      51      01     A9   01               02   EF 00136        EXTZV   #2, #1, MOUNT_OPTIONS+1, R1         ; 1400
                          51               51   D2 0013C        MCOML   R1, R1                              ;
 65   A6             01   04               51   F0 0013F        INSV    R1, #4, #1, 101(UCB)                ;
                          60        0000G  CF   D0 00145        MOVL    VOLUME_UIC, (ORB)                   ; 1401
             05      02   A9               02   E1 0014A        BBC     #2, MOUNT_OPTIONS+2, 5$             ; 1402
                          60        0000G  CF   D0 0014F        MOVL    OWNER_UIC, (ORB)                    ; 1403
                     0B   A0               01   88 00154 5$:    BISB2   #1, 11(ORB)                         ; 1405
             08      01   A9               03   E1 00158        BBC     #3, MOUNT_OPTIONS+1, 6$             ; 1406
                     18   A0         FF00  8F   B0 0015D        MOVW    #-256, 24(ORB)                      ; 1407
                                           06   11 00163        BRB     7$                                  ;
                     18   A0        0000G  CF   B0 00165 6$:    MOVW    HOME_BLOCK+32, 24(ORB)              ; 1408
             06      02   A9               01   E1 0016B 7$:    BBC     #1, MOUNT_OPTIONS+2, 8$             ; 1409
                     18   A0        0000G  CF   B0 00170        MOVW    PROTECTION, 24(ORB)                 ; 1410
                     24   01   A9          03   E0 00176 8$:    BBS     #3, MOUNT_OPTIONS+1, 9$             ; 1412
        0080   C7 00000000G  00            08   28 0017B        MOVC3   #8, SCS$GB_NODENAME, 128(R7)        ; 1425
                   0088   C7               56   D0 00185        MOVL    UCB, 136(R7)                        ; 1426
                          50        0000G  CF   D0 0018A        MOVL    REAL_WCB, R0                        ; 1428
             10           A0               56   D0 0018F        MOVL    UCB, 16(R0)                         ;
                                           01   DD 00193        PUSHL   #1                                  ; 1429
                          7E               56   7D 00195        MOVQ    UCB, -(SP)                          ;
                   0000G  CF               03   FB 00198        CALLS   #3, START_ACP                       ;
                                           23   11 0019D        BRB     10$                                 ; 1412
               00000000G  00               00   FB 0019F 9$:    CALLS   #0, LOCK_IODB                       ; 1433
                     34   A6               68   D0 001A6        MOVL    REAL_VCB, 52(UCB)                   ; 1434
                     38   A6   01080008    8F   C8 001AA        BISL2   #17301512, 56(UCB)                  ; 1436
                                           7E   D4 001B2        CLRL    -(SP)                               ; 1437
                                           56   DD 001B4        PUSHL   UCB                                 ;
                   0000G  CF               02   FB 001B6        CALLS   #2, SET_DATACHECK                   ;
               00000000G  00               00   FB 001BB        CALLS   #0, UNLOCK_IODB                     ; 1438
             0A           69               04   E1 001C2 10$:   BBC     #4, MOUNT_OPTIONS, 11$              ; 1441
             04    0000G  CF               01   E1 001C6        BBC     #1, CLEANUP_FLAGS, 11$              ;
```

```
                    65   A6                  04  88 001CC          BISB2   #4, 101(UCB)                                ; 1442
        04          01   A9                  01  E0 001D0 11$:     BBS     #1, MOUNT_OPTIONS+1, 12$                    ; 1444
                    3B   A6                  02  88 001D5          BISB2   #2, 59(UCB)                                 ; 1445
                                             68  DD 001D9 12$:     PUSHL   REAL_VCB                                    ; 1451
                                             56  DD 001DB          PUSHL   UCB                                         ;
                 0000G   CF                  02  FB 001DD          CALLS   #2, ENTER_LOGNAME                           ;
                              00000000G      9F  D6 001E2          INCL    @#CTL$GL_VOLUMES                            ; 1452
                                             56  DD 001E8          PUSHL   UCB                                         ; 1453
                                             01  DD 001EA          PUSHL   #1                                          ;
                 0000G   CF                  02  FB 001EC          CALLS   #2, SEND_ERRLOG                             ;
                                      5C     A6  B6 001F1          INCW    92(UCB)                                     ; 1458
                 0000G   CF                  00  FB 001F4          CALLS   #0, STORE_CONTEXT                           ; 1463
                         50                  01  D0 001F9          MOVL    #1, R0                                      ; 1465
                                                 04 001FC          RET                                                 ; 1467
                                               0000 001FD 13$:     .WORD   Save nothing                                ; 1288
                                             7E  D4 001FF          CLRL    -(SP)                                       ;
                                             5E  DD 00201          PUSHL   SP                                          ;
                         7E           04     AC  7D 00203          MOVQ    4(AP), -(SP)                                ;
                 0000V   CF                  03  FB 00207          CALLS   #3, KERNEL_HANDLER                          ;
                                                 04 0020C          RET                                                 ;

; Routine Size:  525 bytes,    Routine Base:  $CODE$ + 0643
```

```
1 ROUTINE KERNEL_HANDLER (SIGNAL, MECHANISM) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is the condition handler for all of the kernel mode
1 !       code. It undoes any damage done so far and returns the error
1 !       status to the user mode caller.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       KERNEL_HANDLER (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of signal vector
1 !       ARG2: address of mechanism vector
1 !
1 ! IMPLICIT INPUTS:
1 !       global pointers to blocks allocated
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, allocations undone
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       SIGNAL          : REF BBLOCK,   ! signal vector
2       MECHANISM       : REF BBLOCK;   ! mechanism vector
2
2 LOCAL
2       P               : REF BBLOCK,   ! pointer to scan system lists
2       UCB             : REF BBLOCK;   ! UCB being mounted
2
2 EXTERNAL
2       MOUNT_OPTIONS   : BITVECTOR,    ! command parser options
2       CLEANUP_FLAGS   : BITVECTOR,    ! cleanup action flags
2       CHANNEL,                        ! channel assigned to device
2       MAILBOX_CHANNEL,                ! channel number of ACP mailbox
2       REAL_VCB        : REF BBLOCK,   ! address of VCB allocated
2       REAL_FCB        : REF BBLOCK,   ! address of FCB allocated
2       REAL_WCB        : REF BBLOCK,   ! address of window allocated
2       REAL_AQB        : REF BBLOCK,   ! address of AQB allocated
2       MTL_ENTRY       : REF BBLOCK,   ! address of mounted volume list entry
2       IOC$GL_AQBLIST  : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
2                                       ! system AQB list
```

```
2
2 EXTERNAL ROUTINE
2         LOCK_CLEANUP     : NOVALUE,          ! cleanup device lock on errors.
2         GET_CHANNELUCB,                      ! get UCB address of channel
2         LOCK_IODB        : ADDRESSING_MODE (GENERAL),
2                                              ! interlock system I/O database
2         UNLOCK_IODB      : ADDRESSING_MODE (GENERAL),
2                                              ! unlock system I/O database
2         DEALLOCATE_MEM;                      ! deallocate system dynamic memory
2
2
2 ! Deallocate whatever control blocks exist to wherever they came from.
2 !
2
2 IF .SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND
2 THEN
3     BEGIN
3
3     IF .SIGNAL[CHF$L_SIG_ARGS] NEQ 3
3     THEN BUG_CHECK (UNXSIGNAL, FATAL, 'Unexpected signal in MOUNT');
3
3     KERNEL_CALL (LOCK_CLEANUP);
3
3 ! If there is a mailbox in existence, deassign its channel, thereby
3 ! deleting the mailbox.
3 !
3
3     IF .CLEANUP_FLAGS[CLF_DEASSMBX]
3     THEN
3         $DASSGN (CHAN = .MAILBOX_CHANNEL);
3
3 ! Clean up the UCB.
3 !
3
3     UCB = GET_CHANNELUCB (.CHANNEL);
3     LOCK_IODB ();
3     BBLOCK [UCB [UCB$L_DEVCHAR], DEV$V_MNT] = 0;
3     UCB[UCB$L_VCB] = 0;
3     UNLOCK_IODB ();
3
3
3 ! If we have created an AQB but no ACP, we must remove the AQB from the
3 ! system list.
3 !
3
3     IF .CLEANUP_FLAGS[CLF_DELAQB]
3     THEN
4         BEGIN
4         LOCK_IODB ();
4         P = .IOC$GL_AQBLIST;
4         IF .P EQL .REAL_AQB
4         THEN
4             IOC$GL_AQBLIST = .REAL_AQB[AQB$L_LINK]
4         ELSE
5             BEGIN
5             UNTIL .P[AQB$L_LINK] EQL .REAL_AQB
5             DO P = .P[AQB$L_LINK];
```

```
; 1056      1582 5              P[AQB$L_LINK] = .REAL_AQB[AQB$L_LINK];
; 1057      1583 4              END;
; 1058      1584 4          DEALLOCATE_MEM (.REAL_AQB, 0);
; 1059      1585 4          UNLOCK_IODB ();
; 1060      1586 3          END;
; 1061      1587 3
; 1062      1588 3
; 1063      1589 3      IF .REAL_VCB NEQ 0
; 1064      1590 3      THEN DEALLOCATE_MEM (.REAL_VCB, 0);
; 1065      1591 3
; 1066      1592 3      IF .REAL_FCB NEQ 0
; 1067      1593 3      THEN DEALLOCATE_MEM (.REAL_FCB, 0);
; 1068      1594 3
; 1069      1595 3      IF .REAL_WCB NEQ 0
; 1070      1596 3      THEN DEALLOCATE_MEM (.REAL_WCB, 0);
; 1071      1597 3
; 1072      1598 3      IF .MTL_ENTRY NEQ 0
; 1073      1599 3      THEN DEALLOCATE_MEM (.MTL_ENTRY, 1);
; 1074      1600 3
; 1075      1601 3  ! Return the condition code in R0.
; 1076      1602 3  !
; 1077      1603 3
; 1078      1604 3      MECHANISM[CHF$L_MCH_SAVR0] = .SIGNAL[CHF$L_SIG_NAME];
; 1079      1605 3      $UNWIND ();
; 1080      1606 3
; 1081      1607 2      END;
; 1082      1608 1  END;                                ! end of routine KERNEL_HANDLER


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

                              007C 00000 KERNEL_HANDLER:
                                                  .WORD   Save R2,R3,R4,R5,R6            ; 1468
          56 00000000G  9F   9E 00002             MOVAB   @#IOC$GL_AQBLIST, R6           ;
          55 00000000G  00   9E 00009             MOVAB   UNLOCK_IODB, R5                ;
          54 00000000G  00   9E 00010             MOVAB   LOCK_IODB, R4                  ;
          53     0000G  CF   9E 00017             MOVAB   DEALLOCATE_MEM, R3             ;
          50        04  AC   D0 0001C             MOVL    SIGNAL, R0                     ; 1539
 00000920 8F        04  A0   D1 00020             CMPL    4(R0), #2336                   ;
                        01   12 00028             BNEQ    1$                             ;
                             04 0002A             RET                                    ;
          03            60   D1 0002B 1$:         CMPL    (R0), #3                       ; 1543
                        04   13 0002E             BEQL    2$                             ;
                           FEFF 00030             BUGW                                   ; 1544
                          0000* 00032             .WORD   <BUG$_UNXSIGNAL!4>             ;
                        7E   D4 00034 2$:         CLRL    -(SP)                          ; 1546
                        5E   DD 00036             PUSHL   SP                             ;
                 0000G  CF   9F 00038             PUSHAB  LOCK_CLEANUP                   ;
 00000000G 9F           03   FB 0003C             CALLS   #3, @#SYS$CMKRNL               ;
0B   0000G CF           03   E1 00043             BBC     #3, CLEANUP_FLAGS, 3$          ; 1552
                 0000G  CF   DD 00049             PUSHL   MAILBOX_CHANNEL                ; 1554
 00000000G 00           01   FB 0004D             CALLS   #1, SYS$DASSGN                 ;
```

```
                          0000G  CF  DD 00054 3$:    PUSHL   CHANNEL                                  ; 1559
          0000G  CF              01  FB 00058        CALLS   #1, GET_CHANNELUCB
                 52              50  D0 0005D        MOVL    R0, UCB
                 64              00  FB 00060        CALLS   #0, LOCK_IODB                            ; 1560
          3A     A2              08  8A 00063        BICB2   #8, 58(UCB)                              ; 1561
                           34    A2  D4 00067        CLRL    52(UCB)                                  ; 1562
                 65              00  FB 0006A        CALLS   #0, UNLOCK_IODB                          ; 1563
   31     0000G  CF              02  E1 0006D        BBC     #2, CLEANUP_FLAGS, 7$                    ; 1570
                 64              00  FB 00073        CALLS   #0, LOCK_IODB                            ; 1573
                 50              66  D0 00076        MOVL    IOC$GL_AQBLIST, P                        ; 1574
                 51       0000G  CF  D0 00079        MOVL    REAL_AQB, R1                             ; 1575
                 51              50  D1 0007E        CMPL    P, R1
                                 06  12 00081        BNEQ    4$
                 66        10    A1  D0 00083        MOVL    16(R1), IOC$GL_AQBLIST                   ; 1577
                                 11  11 00087        BRB     6$
                 51        10    A0  D1 00089 4$:    CMPL    16(P), R1                                ; 1580
                                 06  13 0008D        BEQL    5$
                 50        10    A0  D0 0008F        MOVL    16(P), P                                 ; 1581
                                 F4  11 00093        BRB     4$
          10     A0        10    A1  D0 00095 5$:    MOVL    16(R1), 16(P)                            ; 1582
                                 7E  D4 0009A 6$:    CLRL    -(SP)                                    ; 1584
                                 51  DD 0009C        PUSHL   R1
                 63              02  FB 0009E        CALLS   #2, DEALLOCATE_MEM
                 65              00  FB 000A1        CALLS   #0, UNLOCK_IODB                          ; 1585
                 50       0000G  CF  D0 000A4 7$:    MOVL    REAL_VCB, R0                             ; 1589
                                 07  13 000A9        BEQL    8$
                                 7E  D4 000AB        CLRL    -(SP)                                    ; 1590
                                 50  DD 000AD        PUSHL   R0
                 63              02  FB 000AF        CALLS   #2, DEALLOCATE_MEM
                 50       0000G  CF  D0 000B2 8$:    MOVL    REAL_FCB, R0                             ; 1592
                                 07  13 000B7        BEQL    9$
                                 7E  D4 000B9        CLRL    -(SP)                                    ; 1593
                                 50  DD 000BB        PUSHL   R0
                 63              02  FB 000BD        CALLS   #2, DEALLOCATE_MEM
                 50       0000G  CF  D0 000C0 9$:    MOVL    REAL_WCB, R0                             ; 1595
                                 07  13 000C5        BEQL    10$
                                 7E  D4 000C7        CLRL    -(SP)                                    ; 1596
                                 50  DD 000C9        PUSHL   R0
                 63              02  FB 000CB        CALLS   #2, DEALLOCATE_MEM
                 50       0000G  CF  D0 000CE 10$:   MOVL    MTL_ENTRY, R0                            ; 1598
                                 07  13 000D3        BEQL    11$
                                 01  DD 000D5        PUSHL   #1                                       ; 1599
                                 50  DD 000D7        PUSHL   R0
                 63              02  FB 000D9        CALLS   #2, DEALLOCATE_MEM
                 50        04    AC  7D 000DC 11$:   MOVQ    SIGNAL, R0                               ; 1604
          0C     A1        04    A0  D0 000E0        MOVL    4(R0), 12(R1)
                                 7E  7C 000E5        CLRQ    -(SP)                                    ; 1605
      00000000G  00              02  FB 000E7        CALLS   #2, SYS$UNWIND
                                     04 000EE        RET                                              ; 1608

; Routine Size:  239 bytes,    Routine Base:  $CODE$ + 0850

; 1083        1609  1
; 1084        1610  1 END
; 1085        1611  0 ELUDOM
```

```
                                                  .EXTRN  LIB$SIGNAL, LIB$STOP

;                         PSECT SUMMARY
;
;     Name                      Bytes                  Attributes
;
;   $OWN$                          360  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $PLIT$                          12  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                        2367  NOVEC,NOWRT,  RD , EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                       Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;     File                               Total    Loaded   Percent      Mapped      Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1        18619       120         0      1000         00:02.0


;                         COMMAND QUALIFIERS

;     BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:MOUDK1/OBJ=OBJ$:MOUDK1 MSRC$:MOUDK1/UPDATE=(ENH$:MOUDK1)

; Size:           2367 code + 372 data bytes
; Run Time:          00:47.1
; Elapsed Time:      01:29.3
; Lines/CPU Min:      2050
; Lexemes/CPU-Min: 20816
; Memory Used:  424 pages
; Compilation Complete
```

0244 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CHKHM2
LIS

CHNUCB
LIS

GETUIC
LIS

ERASE
LIS

LEFTONE
LIS

MOUDK2
LIS

CHKHM1
LIS

CHKSM2
LIS

CLUSTRMNT
LIS

INIFC2
LIS

MOUDK1
LIS

MAKRVT
LIS

MAKLOG
LIS