```
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**RUJMAN

```
RRRRRRRR    UU      UU          JJ  MM      MM    AAAAAA    NN      NN
RRRRRRRR    UU      UU          JJ  MM      MM    AAAAAA    NN      NN
RR      RR  UU      UU          JJ  MMMM  MMMM  AA      AA  NN      NN
RR      RR  UU      UU          JJ  MMMM  MMMM  AA      AA  NN      NN
RR      RR  UU      UU          JJ  MM  MM  MM  AA      AA  NNNN    NN
RR      RR  UU      UU          JJ  MM  MM  MM  AA      AA  NNNN    NN
RRRRRRRR    UU      UU          JJ  MM      MM  AA      AA  NN  NN  NN
RRRRRRRR    UU      UU          JJ  MM      MM  AA      AA  NN  NN  NN
RR  RR      UU      UU  JJ      JJ  MM      MM  AAAAAAAAAA  NN    NNNN
RR  RR      UU      UU  JJ      JJ  MM      MM  AAAAAAAAAA  NN    NNNN
RR    RR    UU      UU  JJ      JJ  MM      MM  AA      AA  NN      NN
RR    RR    UU      UU  JJ      JJ  MM      MM  AA      AA  NN      NN      ....
RR      RR  UUUUUUUUUU    JJJJJJ    MM      MM  AA      AA  NN      NN      ....
RR      RR  UUUUUUUUUU    JJJJJJ    MM      MM  AA      AA  NN      NN      ....
                                                                            ....

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
```

RU
VO

```
0 MODULE RUJMAN (
0                 LANGUAGE (BLISS32),
0                 ADDRESSING MODE (EXTERNAL=GENERAL),
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:  MOUNT Utility Structure Level 2
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains those routine necessary to handle the creation
1 !       and manipulation of Recovery Unit Journals (RUJ) on Files-11 ODS-2
1 !       disk volumes.
1 !
1 ! ENVIRONMENT:
1 !
1 !       STARLET operating system, including privileged system services
1 !       and internal exec routines.
1 !
1 !--
1 !
1 !
1 ! AUTHOR:  Steven T. Jeffreys    CREATION DATE:  18-Jul-1983
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-008 HH0041          Hai Huang                       24-Jul-1984
1 !               Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
1 !
1 !       V03-007 HH0019          Hai Huang                       08-May-1984
```

```
!               Another round in fixing up truncation errors.
!
!       V03-006 HH0007          Hai Huang                       22-Mar-1984
!               Fix truncation error introduced by cluster-mount support.
!
!       V03-004 WMC0002         Wayne Cardoza                   16-Jan-1984
!               Immediately return from main routine to disable RU journals
!
!       V03-004 DAS0001         David Solomon                   29-Nov-1983
!               Add support for specifying maximum journal record size
!               with a new keyword, /JOURNAL=(RECORD_SIZE=n).
!
!       V03-003 WMC0001         Wayne Cardoza                   20-Sep-1983
!               DEVEXI is legal return status for CREJNL.
!
!       V03-002 CDS0001         Christian D. Saether            30-Aug-1983
!               Change name of default cjf directory from [journal] to
!               [sysjnl].
!
!       V03-001 STJ3116         Steven T. Jeffreys,             02-Aug-1983
!               Created local definition of $DISMOU macro.
!
!**


LIBRARY 'SYS$LIBRARY:LIB.L32';
REQUIRE 'SRC$:MOUDEF.B32';


FORWARD ROUTINE
        ACCESS_JOURNAL,                                 ! Look up RUJ file on volume
        ACTIVATE_JOURNAL,                               ! Main routine
        ANALYZE_DEVLIST : NOVALUE,                      ! Build volume data base from device list
        CREATE_RUJ,                                     ! Call $CREJNL to create an RUJ
        DISMOUNT_VOLSET : NOVALUE,                      ! Dismount all or part of a volume set
        GET_VOLUME_NAME,                                ! Return volume name
        INTERCEPT_SIGNAL,                               ! Condition handler
        RCP_RESTART;                                    ! Restart "frozen' RUJ
```

```
!****************************************************************************
!****************************************************************************
!**                                                                        **
!**                               N O T E                                  **
!**                                                                        **
!** The following macro is a local redefinition of the $DISMOU system service **
!** macro, and is necessary in order to make use of the disount system     **
!** within the $MOUNT system service.  Since both $DISMOU and $MOUNT are   **
!** protected shareable images, and cannot make outbound references, the   **
!** $DISMOU code must be included into the $MOUNT code at link time.  The  **
!** $MOUNT service then calls $DISMOU as if it were a normal routine, instead **
!** of going through the $DISMOU service dispatcher.                       **
!**                                                                        **
!****************************************************************************
!****************************************************************************
!
!   $DISMOU
!
!     Dismount Volume
!
!      $DISMOU devnam ,[flags]
!
!      devnam = address of device name string descriptor
!
!      flags  = 32-bit status mask  selecting options for  the  dismount
!               The  symbols  for  the flags are  defined by the $DMTDEF
!               macro.
!
!                  Flag      Meaning
!
!              DMT$M_NOUNLOAD   Do not unload the volume.
!
!              DMT$M_UNIT       Dismount the specified device, rather
!                               than the entire volume set.
!
UNDECLARE %QUOTE $DISMOU;
KEYWORDMACRO $DISMOU (DEVNAM=0,FLAGS=0) =
        (
        EXTERNAL ROUTINE SYS$DISMOU : BLISS ADDRESSING_MODE (GENERAL);
        KERNEL_CALL (SYS$DISMOU, DEVNAM, FLAGS)
        ) %;


!+
!
! Own storage for general use in this module.
!
!-

LITERAL
        ROOTVOL_NAMLEN  = 64;                           ! Maximum length of a device name

EXTERNAL LITERAL
        CJF$_DEVEXI;

OWN
        AA$_DATA_BEGIN  : VECTOR [0],                   ! Mark start of data area
```

```
;  154     0684  1        DISMOUNT_COUNT   : LONG,                                   ! Count of volumes that were dismounted
;  155     0685  1        ODS2_VOLUME      : BITVECTOR [DEVMAX],                     ! Indicates volume supports ODS-2 file struc
;  156     0686  1        ROOT_PRESENT     : BITVECTOR [DEVMAX],                     ! Indicates presence of root volume for a gi
;  157     0687  1        RUJRNL_PRESENT   : BITVECTOR [DEVMAX],                     ! Indicates presence of RUJ file on a given
;  158     0688  1        ROOTVOL_DSC      : BBLOCK [DSC$K_S_BLN],                   ! Root volume device name descriptor
;  159     0689  1        ROOTVOL_BUF      : BBLOCK [ROOTVOL_NAMLEN],                ! Buffer for root volume device name
;  160     0690  1        ROOTVOL_INDEX    : VECTOR [DEVMAX, BYTE, UNSIGNED],        ! Each entry contians index+1 of root volume
;  161     0691  1        VOLSET_RVN       : VECTOR [DEVMAX, BYTE, UNSIGNED],        ! Vector of Relative Volume Numbers
;  162     0692  1        ZZ$_DATA_END     : VECTOR [0];                             ! Mark end of data area
```

```
;  1 ROUTINE ACCESS_JOURNAL (DEVICE_NAME) =
;  1
;  1 !++
;  1 !
;  1 ! FUNCTIONAL DESCRIPTION:
;  1 !
;  1 !        This routine will attempt to access the Recovery Unit (RU) journal
;  1 !        file on a given volume.
;  1 !
;  1 ! CALLING SEQUENCE:
;  1 !        ACCESS_JOURNAL (ARG1)
;  1 !
;  1 ! INPUT PARAMETERS:
;  1 !        ARG1:    address of a device name descriptor
;  1 !
;  1 ! IMPLICIT INPUTS:
;  1 !        NONE
;  1 !
;  1 ! OUTPUT PARAMETERS:
;  1 !        NONE
;  1 !
;  1 ! IMPLICIT OUTPUTS:
;  1 !        NONE
;  1 !
;  1 ! ROUTINE VALUE:
;  1 !        Status code passed back by $OPEN
;  1 !
;  1 ! SIDE EFFECTS:
;  1 !        None.
;  1 !
;  1 !--
;  1
;  2 BEGIN
;  2
;  2 !
;  2 ! Allocate plits in the $CODE$ psect to avoid truncation error when
;  2 ! linking mountshr.
;  2 !
;  2 PSECT
;  2        PLIT = $CODE$;
;  2
;  2 BIND
;  2        RUJRNL_FILE      = DESCRIPTOR ('[SYSJNL]RUJNL.RUJ;1');
;  2
;  2 LOCAL
;  2        STATUS;
;  2
;  2 MAP
;  2        DEVICE_NAME      : REF BBLOCK,
;  2        RUJRNL_FILE      : BBLOCK;
;  2
;  2 OWN
;  2        RUJRNL_FAB       : $FAB (FOP=UFO);
;  2
;  2 !
;  2 ! Build a File Attributes Block (FAB) and determine
;  2 ! if there is a journal file on volume by attempting
```

```
;   2 ! to open it.
;   2 !
;   2
;   2 RUJRNL_FAB[FAB$B_DNS] = .DEVICE_NAME[DSC$W_LENGTH];
;   2 RUJRNL_FAB[FAB$L_DNA] = .DEVICE_NAME[DSC$A_POINTER];
;   2 RUJRNL_FAB[FAB$B_FNS] = .RUJRNL_FILE[DSC$W_LENGTH];
;   2 RUJRNL_FAB[FAB$L_FNA] = .RUJRNL_FILE[DSC$A_POINTER];
;   2
;   3 IF (STATUS = $OPEN (FAB = RUJRNL_FAB))                    ! Attempt to open the file
;   2 THEN
;   2     $DASSGN (CHAN = .RUJRNL_FAB[FAB$L_STV]);               ! Deassign channel to close file
;   2
;   2 !
;   2 ! If the status was RMS$_DNR, it indicates that the journal file
;   2 ! was present, but could not be accessed, as it or the directory
;   2 ! describing it were spread across one or more disks and at least
;   2 ! one of those disks is not mounted.  Convert the RMS$_DNR to success
;   2 ! to indicate that the journal file is present.
;   2 !
;   2 IF .STATUS EQL RMS$_DNR
;   2 THEN
;   2     STATUS = 1;
;   2
;   2 RETURN .STATUS
;   2
;   1 END;

                                                   .TITLE  RUJMAN
                                                   .IDENT  \V04-000\

                                                   .PSECT  $CODE$,NOWRT,2

52 2E 4C 4E 4A 55 52 5D 4C 4E 4A 53 59 53 5B  00000 P.AAB:   .ASCII  \[SYSJNL]RUJNL.RUJ;1\      ;
                                 31 3B 4A 55  0000F                                             ;
                                              00013          .BLKB   1
                                   00000013   00014 P.AAA:   .LONG   19                         ;
                                   00000000'  00018          .ADDRESS P.AAB                     ;

                                                   .PSECT  $OWN$,NOEXE,2

                                   00000 AA$_DATA_BEGIN:
                                                   .BLKB   0
                                   00000 DISMOUNT_COUNT:
                                                   .BLKB   4
                                   00004 ODS2_VOLUME:
                                                   .BLKB   2
                                   00006           .BLKB   2
                                   00008 ROOT_PRESENT:
                                                   .BLKB   2
                                   0000A           .BLKB   2
                                   0000C RUJRNL_PRESENT:
                                                   .BLKB   2
                                   0000E           .BLKB   2
                                   00010 ROOTVOL_DSC:
                                                   .BLKB   8
                                   00018 ROOTVOL_BUF:
```

```
                                     .BLKB    64
                        00058 ROOTVOL_INDEX:
                                     .BLKB    16
                        00068 VOLSET_RVN:
                                     .BLKB    16
                        00078 ZZ$_DATA_END:
                                     .BLKB    0
                    03  00078 RUJRNL_FAB:
                                     .BYTE    3
                    50  00079        .BYTE    80
                  0000  0007A        .WORD    0
              00020000  0007C        .LONG    131072
              00000000  00080        .LONG    0
              00000000  00084        .LONG    0
              00000000  00088        .LONG    0
                  0000  0008C        .WORD    0
                    02  0008E        .BYTE    2
                    00  0008F        .BYTE    0
              00000000  00090        .LONG    0
                    00  00094        .BYTE    0
                    00  00095        .BYTE    0
                    00  00096        .BYTE    0
                    02  00097        .BYTE    2
              00000000  00098        .LONG    0
              00000000  0009C        .LONG    0
              00000000  000A0        .LONG    0
              00000000  000A4        .LONG    0
              00000000  000A8        .LONG    0
                    00  000AC        .BYTE    0
                    00  000AD        .BYTE    0
                  0000  000AE        .WORD    0
              00000000  000B0        .LONG    0
                  0000  000B4        .WORD    0
                    00  000B6        .BYTE    0
                    00  000B7        .BYTE    0
              00000000  000B8        .LONG    0
              00000000  000BC        .LONG    0
                  0000  000C0        .WORD    0
                    00  000C2        .BYTE    0
                    00  000C3        .BYTE    0
              00000000  000C4        .LONG    0

                        RUJRNL_FILE=          P.AAA
                                     .EXTRN   CJF$_DEVEXI, SYS$OPEN
                                     .EXTRN   SYS$DASSGN

                                     .PSECT   $CODE$,NOWRT,2

                  000C 00000 ACCESS_JOURNAL:
                                     .WORD    Save R2,R3                              ; 0693
      53       0000'   CF  9E 00002  MOVAB    RUJRNL_FAB+53, R3
      50          04   AC  D0 00007  MOVL     DEVICE_NAME, R0                         ; 0753
      63               60  90 0000B  MOVB     (R0), RUJRNL_FAB+53
FB    A3          04   A0  D0 0000E  MOVL     4(R0), RUJRNL_FAB+48                    ; 0754
FF    A3          E2   AF  90 00013  MOVB     RUJRNL_FILE, RUJRNL_FAB+52              ; 0755
F7    A3          E1   AF  D0 00018  MOVL     RUJRNL_FILE+4, RUJRNL_FAB+44            ; 0756
                  CB   A3  9F 0001D  PUSHAB   RUJRNL_FAB                              ; 0758
```

```
                00000000G  00       01  FB 00020      CALLS   #1, SYS$OPEN                ;
                           52       50  D0 00027      MOVL    R0, STATUS                  ;
                           0A       52  E9 0002A      BLBC    STATUS, 1$                  ;
                              D7    A3  DD 0002D      PUSHL   RUJRNL_FAB+12               ; 0760
                00000000G  00       01  FB 00030      CALLS   #1, SYS$DASSGN              ;
                00018272   8F       52  D1 00037 1$:  CMPL    STATUS, #98930              ; 0769
                                    03  12 0003E      BNEQ    2$                          ;
                           52       01  D0 00040      MOVL    #1, STATUS                  ; 0771
                           50       52  D0 00043 2$:  MOVL    STATUS, R0                  ; 0773
                                        04 00046      RET                                 ; 0775

; Routine Size:  71 bytes,    Routine Base:  $CODE$ + 001C
```

```
1 GLOBAL ROUTINE ACTIVATE_JOURNAL =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine will do the appropriate RUJ operation for
1 !       each device listed in the PHYS_NAME descriptor vector.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       ACTIVATE_JOURNAL ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       <see the EXTERNAL declaration>
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       Status code passed back by RUJ routines.
1 !
1 ! SIDE EFFECTS:
1 !       The state of the volume may be changed depending on the action of
1 !       the journalling ACP.  A serious error returned by the journalling
1 !       service will mandate dismounting pieces (or all) of the volume sets.
1 !
1 !--
1
2 BEGIN
2
2 !EXTERNAL LITERAL
2 !     ROOTVOL_NAMLEN  = 64;                                  ! Maximum length of a device name
2
2 !EXTERNAL
2 !     AA$_DATA_BEGIN   : VECTOR [0],                         ! Mark start of data area
2 !     DISMOUNT_COUNT   : LONG,                               ! Count of volumes that were dismounted
2 !     ODS2_VOLUME      : BITVECTOR [DEVMAX],                 ! Indicates volume supports ODS-2 file struc
2 !     ROOT_PRESENT     : BITVECTOR [DEVMAX],                 ! Indicates presence of root volume for a gi
2 !     RUJRNL_PRESENT   : BITVECTOR [DEVMAX],                 ! Indicates presence of RUJ file on a given
2 !     ROOTVOL_DSC      : BBLOCK [DSC$K_S_BLN],               ! Root volume device name descriptor
2 !     ROOTVOL_BUF      : BBLOCK [ROOTVOL_NAMLEN],            ! Buffer for root volume device name
2 !     ROOTVOL_INDEX    : BBLOCK [DEVMAX, BYTE, UNSIGNED],    ! Each entry contians index+1 of root volume
2 !     VOLSET_RVN       : VECTOR [DEVMAX, BYTE, UNSIGNED],    ! Vector of Relative Volume Numbers
2 !     ZZ$_DATA_END     : VECTOR [0];                         ! Mark end of data area
2
2 !EXTERNAL ROUTINE
2 !     ANALYZE_DEVLIST : NOVALUE ADDRESSING_MODE (GENERAL),   ! Inspect the list of devices
2 !     DISMOUNT_VOLSET : NOVALUE ADDRESSING_MODE (GENERAL);   ! Dismount volumes if journalling doesn't wo
2
2 EXTERNAL
```

```
        MOUNT_OPTIONS    : BITVECTOR,                             ! Mount option flag bits
        PHYS_COUNT       : LONG,                                  ! # of descriptors in PHYS_NAME
        PHYS_NAME        : BBLOCKVECTOR [DEVMAX, DSC$K_S_BLN],    ! Descriptor of root volume name
        STORED_CONTEXT   : BITVECTOR;                             ! Determines if tape or disk mount

LITERAL
        ALL_VOLUMES      = 1,                                     ! Various and sundry mnemonics
        ONE_VOLUME       = 0,
        PERMANENT_RUJ    = 1,
        TEMPORARY_RUJ    = 0;

LOCAL
        RUJRNL_EXISTS,                                            ! Boolean
        STATUS;


!
! If the user explicitly requested that Recovery Unit Journalling (RUJ)
! not be activated, return a success status immediately.
!
!***JNL** turn off RU journals

MOUNT_OPTIONS[OPT_NOJRNL] = 1; !***JNL**
IF .MOUNT_OPTIONS[OPT_NOJRNL]
THEN
    RETURN 1;

!
! RUJ is only appropriate for disk volumes.  If we are mounting a tape
! or a foreign disk, return a success status immediately.
!
IF .STORED_CONTEXT[TAPE_MOUNT]
OR .MOUNT_OPTIONS[OPT_FOREIGN]
THEN
    RETURN 1;

!
! Inspect the list of devices that have been mounted, and record some
! useful information.  This is necessary because $MOUNT allows a great
! deal of flexibility in the amount and type of disks volumes that may
! be mounted.
!
ANALYZE_DEVLIST (PHYS_NAME[0, DSC$W_LENGTH], .PHYS_COUNT);

!+
! Now that the volume/device data base has been created, process each
! volume in the device list.  Three passes are made through the list,
! and the ordering of the processing is important.  The actions performed are:
!
!       1. For each ODS-2 volume (mounted on a device in the device list) that
!          is a member of a volume set, but not the root volume of a volume set,
!          and contains a RUJ file, create a temporary RUJ to process the
!          existing RUJ file, and delete the file when done.
!
!          This pass is meant to process new members to a volume set that
!          already contain user files, and might have an active RUJ file
!          present on the volume.  The volume must be in a consistent state
```

```
2 !             before it can be a full-fledged member of the volume set.  A
2 !             journalling failure at this point will cause the volume, but not
2 !             the volume set, to be dismounted.
2 !
2 !         2. For each ODS-2 volume (mounted on a device in the device list) that
2 !             is a root volume of a volume set, or not part of any volume set,
2 !             attempt to create a permanent RUJ on the volume if a journal file
2 !             already exists, or if the user requested one be created.
2 !
2 !             Volumes of this type are the most prevalent.  A journalling failure
2 !             at this point will cause the entire volume set to be dismounted.
2 !
2 !         3. For each ODS-2 volume (mounted on a device in the device list) that
2 !             is a member of a volume set but not a root volume, attempt to restart
2 !             a "frozen" recovery unit.
2 !
2 !             This pass handles the case of a volume set that is mounted in stages.
2 !             An active RUJ on an imcomplete volume set might become "frozen" if
2 !             data on missing volumes is needed to complete the journalling
2 !             operation.  Note that if more pieces of the volume set are still
2 !             not mounted, the RUJ may remain frozen.  A journalling failure at
2 !             this point will cause the entire volume set to be dismounted.  Note
2 !             that if the RUJ remains frozen, the volume set is not dismounted.
2 !
2 ! Note that DISMOUNT_VOLSET will update the local device list database to reflect
2 ! the removal (dismounting) of volumes.
2 !
2 ! This level of checking is necessary because $MOUNT will allow many types
2 ! of volumes to be mounted with a single invocation.  For example, more than
2 ! one volume set can be mounted; ODS1 and ODS2 disks can be mounted together
2 ! also.
2 !-
2
2 !
2 ! Handle case 1
2 !
2 INCR J FROM 0 TO .PHYS_COUNT-1 DO
2     IF  .ODS2_VOLUME[.J]
2     AND .VOLSET_RVN[.J] GTR 1
2     AND .RUJRNL_PRESENT[.J]
2     THEN
3         IF NOT (STATUS = EXEC_CALL (CREATE_RUJ, PHYS_NAME[.J, DSC$W_LENGTH], TEMPORARY_RUJ))
2         THEN
2             DISMOUNT_VOLSET (PHYS_NAME, .PHYS_COUNT, .J,  ONE_VOLUME, .STATUS);
2
2 !
2 ! Handle case 2.
2 !
2 INCR J FROM 0 TO .PHYS_COUNT-1 DO
2     IF  .ODS2_VOLUME[.J]
2     AND .VOLSET_RVN[.J] LEQ 1
3     AND (.RUJRNL_PRESENT[.J] OR .MOUNT_OPTIONS[OPT_NEWJRNL])
2     THEN
3         IF NOT (STATUS = EXEC_CALL (CREATE_RUJ, PHYS_NAME[.J, DSC$W_LENGTH], PERMANENT_RUJ))
2         THEN
2             DISMOUNT_VOLSET (PHYS_NAME, .PHYS_COUNT, .J,  ALL_VOLUMES, .STATUS);
2
```

```
2 !
2 ! Handle case 3.
2 !
2 INCR J FROM 0 TO .PHYS_COUNT-1 DO
2     IF  .ODS2_VOLUME[.J]
2     AND .VOLSET_RVN[.J] GTR 1
2     AND NOT .ROOT_PRESENT[.J]
2     THEN
3         IF NOT (STATUS = EXEC_CALL (RCP_RESTART, PHYS_NAME[.J, DSC$W_LENGTH]))
2         THEN
2             DISMOUNT_VOLSET (PHYS_NAME, .PHYS_COUNT, .J,  ALL_VOLUMES, .STATUS);
2
2 !
2 ! Compare the number of volumes dismounted against the original number
2 ! of volumes in the device list.  There are three posibilities:
2 !         1. No volumes have been dismounted - return SS$_NORMAL
2 !         2. Some but not all volumes have been dismounted - return MOUN$_DISMPART
2 !         3. All volumes have been dismounted - return MOUN$_DISMAL
2 !
2 IF .DISMOUNT_COUNT LEQ 0
2 THEN
2     RETURN SS$_NORMAL
2 ELSE
2     IF .DISMOUNT_COUNT LSS .PHYS_COUNT
2     THEN
2         RETURN MOUN$_DISMPART
2     ELSE
2         RETURN MOUN$_DISMAL
2
1 END;


                                                        .EXTRN   MOUNT_OPTIONS, PHYS_COUNT
                                                        .EXTRN   PHYS_NAME, STORED_CONTEXT
                                                        .EXTRN   SYS$CMEXEC

                                     07FC 00000         .ENTRY   ACTIVATE_JOURNAL, Save R2,R3,R4,R5,R6,R7,-  ; 0776
                                                                 R8,R9,R10                                   ;
              5A      0000V  CF  9E 00002               MOVAB    DISMOUNT_VOLSET, R10                        ;
              59 00000000G   00  9E 00007               MOVAB    MOUNT_OPTIONS+6, R9                         ;
              58 00000000G   9F  9E 0000E               MOVAB    @#SYS$CMEXEC, R8                            ;
              57      0000'  CF  9E 00015               MOVAB    ODS2_VOLUME, R7                             ;
              56 00000000G   00  9E 0001A               MOVAB    PHYS_NAME, R6                               ;
              55 00000000G   00  9E 00021               MOVAB    PHYS_COUNT, R5                              ;
              69         80  8F  88 00028               BISB2    #128, MOUNT_OPTIONS+6                       ; 0855
                             03  18 0002C               BGEQ     2$                                          ; 0856
                           00DA  31 0002E 1$:           BRW      10$                                         ;
              F6 00000000G   00  E8 00031 2$:           BLBS     STORED_CONTEXT, 1$                          ; 0864
    F1        FB         A9  03  E0 00038               BBS      #3, MOUNT_OPTIONS+1, 1$                     ; 0865
                             65  DD 0003D               PUSHL    PHYS_COUNT                                  ; 0875
                             56  DD 0003F               PUSHL    R6                                          ;
         0000V  CF           02  FB 00041               CALLS    #2, ANALYZE_DEVLIST                         ;
              53             65  D0 00046               MOVL     PHYS_COUNT, R3                              ; 0926
              52             01  CE 00049               MNEGL    #1, J                                       ;
                             33  11 0004C               BRB      4$                                          ;
    2F        67             52  E1 0004E 3$:           BBC      J, ODS2_VOLUME, 4$                          ; 0927
```

```
          01        64 A742  91 00052       CMPB    VOLSET_RVN[J], #1                    ; 0928
                         28  1B 00057       BLEQU   4$                                   ;
23        08  A7             52  E1 00059       BBC     J, RUJRNL_PRESENT, 4$                ; 0929
                         7E  D4 0005E       CLRL    -(SP)                                ; 0931
                       6642  7F 00060       PUSHAQ  PHYS_NAME[J]                         ;
                         02  DD 00063       PUSHL   #2                                   ;
                         5E  DD 00065       PUSHL   SP                                   ;
                 0000V   CF  9F 00067       PUSHAB  CREATE_RUJ                           ;
              68         05  FB 0006B       CALLS   #5, SYS$CMEXEC                       ;
              54         50  D0 0006E       MOVL    R0, STATUS                           ;
              0D         54  E8 00071       BLBS    STATUS, 4$                           ;
                         54  DD 00074       PUSHL   STATUS                               ; 0933
                         7E  D4 00076       CLRL    -(SP)                                ;
                         52  DD 00078       PUSHL   J                                    ;
                         65  DD 0007A       PUSHL   PHYS_COUNT                           ;
                         56  DD 0007C       PUSHL   R6                                   ;
              6A         05  FB 0007E       CALLS   #5, DISMOUNT_VOLSET                  ;
C9            52         53  F2 00081 4$:   AOBLSS  R3, J, 3$                            ; 0927
              53         65  D0 00085       MOVL    PHYS_COUNT, R3                       ; 0938
              52         01  CE 00088       MNEGL   #1, J                                ;
                         37  11 0008B       BRB     7$                                   ;
33            67         52  E1 0008D 5$:   BBC     J, ODS2_VOLUME, 7$                   ; 0939
              01    64 A742  91 00091       CMPB    VOLSET_RVN[J], #1                    ; 0940
                         2C  1A 00096       BGTRU   7$                                   ;
04        08  A7         52  E0 00098       BBS     J, RUJRNL_PRESENT, 6$                ; 0941
              23     01  A9  E9 0009D       BLBC    MOUNT_OPTIONS+7, 7$                  ;
                         01  DD 000A1 6$:   PUSHL   #1                                   ; 0943
                       6642  7F 000A3       PUSHAQ  PHYS_NAME[J]                         ;
                         02  DD 000A6       PUSHL   #2                                   ;
                         5E  DD 000A8       PUSHL   SP                                   ;
                 0000V   CF  9F 000AA       PUSHAB  CREATE_RUJ                           ;
              68         05  FB 000AE       CALLS   #5, SYS$CMEXEC                       ;
              54         50  D0 000B1       MOVL    R0, STATUS                           ;
              0D         54  E8 000B4       BLBS    STATUS, 7$                           ;
                         54  DD 000B7       PUSHL   STATUS                               ; 0945
                         01  DD 000B9       PUSHL   #1                                   ;
                         52  DD 000BB       PUSHL   J                                    ;
                         65  DD 000BD       PUSHL   PHYS_COUNT                           ;
                         56  DD 000BF       PUSHL   R6                                   ;
              6A         05  FB 000C1       CALLS   #5, DISMOUNT_VOLSET                  ;
C5            52         53  F2 000C4 7$:   AOBLSS  R3, J, 5$                            ; 0939
              53         65  D0 000C8       MOVL    PHYS_COUNT, R3                       ; 0950
              52         01  CE 000CB       MNEGL   #1, J                                ;
                         31  11 000CE       BRB     9$                                   ;
2D            67         52  E1 000D0 8$:   BBC     J, ODS2_VOLUME, 9$                   ; 0951
              01    64 A742  91 000D4       CMPB    VOLSET_RVN[J], #1                    ; 0952
                         26  1B 000D9       BLEQU   9$                                   ;
21        04  A7         52  E0 000DB       BBS     J, ROOT_PRESENT, 9$                  ; 0953
                       6642  7F 000E0       PUSHAQ  PHYS_NAME[J]                         ; 0955
                         01  DD 000E3       PUSHL   #1                                   ;
                         5E  DD 000E5       PUSHL   SP                                   ;
                 0000V   CF  9F 000E7       PUSHAB  RCP_RESTART                          ;
              68         04  FB 000EB       CALLS   #4, SYS$CMEXEC                       ;
              54         50  D0 000EE       MOVL    R0, STATUS                           ;
              0D         54  E8 000F1       BLBS    STATUS, 9$                           ;
                         54  DD 000F4       PUSHL   STATUS                               ; 0957
                         01  DD 000F6       PUSHL   #1                                   ;
```

```
                                  52  DD 000F8          PUSHL   J                                   ;
                                  65  DD 000FA          PUSHL   PHYS_COUNT                          ;
                                  56  DD 000FC          PUSHL   R6                                  ;
                     6A           05  FB 000FE          CALLS   #5, DISMOUNT_VOLSET                 ;
       CB            52           53  F2 00101  9$:     AOBLSS  R3, J, 8$                           ; 0951
                     50       FC  A7  D0 00105          MOVL    DISMOUNT_COUNT, R0                  ; 0966
                                  04  14 00109          BGTR    11$                                 ;
                     50           01  D0 0010B 10$:     MOVL    #1, R0                              ; 0970
                                      04 0010E          RET                                         ;
                     65           50  D1 0010F 11$:     CMPL    R0, PHYS_COUNT                      ;
                                  08  18 00112          BGEQ    12$                                 ;
                     50 00729050  8F  D0 00114          MOVL    #7508048, R0                        ; 0972
                                      04 0011B          RET                                         ;
                     50 0072822C  8F  D0 0011C 12$:     MOVL    #7504428, R0                        ; 0974
                                      04 00123          RET                                         ; 0976

; Routine Size:  292 bytes,    Routine Base:  $CODE$ + 0063
```

```
1 ROUTINE ANALYZE_DEVLIST (DEVNAM_LIST, LIST_LENGTH) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       For each device in the physical device name descriptor vector, determine
1 !               o the file structure level (ODS-1 or ODS-2)
1 !               o the name of the root volume in the volume set
1 !                 (if present in the device list, record the index)
1 !               o the Relative Volume Number (RVN)
1 !               o whether or not the root volume is in the list
1 !               o whether of not a volume has a RUJ file
1 !
1 ! CALLING SEQUENCE:
1 !       ANALYZE_DEVLIST (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1    : address of a vector of device name descriptors
1 !       ARG2    : number of device name descriptors in the list
1 !
1 ! IMPLICIT INPUTS:
1 !       <see the EXTERNAL declaration>
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2 BEGIN                                                          ! Start of ANALYZE_DEVLIST
2
2 !EXTERNAL LITERAL
2 !       ROOTVOL_NAMLEN  = 64;                                  ! Maximum length of a device name
2
2 !EXTERNAL
2 !       AA$_DATA_BEGIN  : VECTOR [0],                          ! Mark start of data area
2 !       DISMOUNT_COUNT  : LONG,                                ! Count of volumes that were dismounted
2 !       ODS2_VOLUME     : BITVECTOR [DEVMAX],                  ! Indicates volume supports ODS-2 file struc
2 !       ROOT_PRESENT    : BITVECTOR [DEVMAX],                  ! Indicates presence of root volume for a gi
2 !       RUJRNL_PRESENT  : BITVECTOR [DEVMAX],                  ! Indicates presence of RUJ file on a given
2 !       ROOTVOL_DSC     : BBLOCK [DSC$K_S_BLN],                ! Root volume device name descriptor
2 !       ROOTVOL_BUF     : BBLOCK [ROOTVOL_NAMLEN],             ! Buffer for root volume device name
2 !       ROOTVOL_INDEX   : BBLOCK [DEVMAX, BYTE, UNSIGNED],     ! Each entry contians index+1 of root volume
2 !       VOLSET_RVN      : VECTOR [DEVMAX, BYTE, UNSIGNED],     ! Vector of Relative Volume Numbers
2 !       ZZ$_DATA_END    : VECTOR [0];                          ! Mark end of data area
2
2 !EXTERNAL ROUTINE
2 !       ACCESS_JOURNAL;                                        ! Determine if RUJ file present on a volume
```

```

LOCAL
        I;                                                      ! Loop counter

MAP
        DEVNAM_LIST     : REF BBLOCKVECTOR [DEVMAX, DSC$K_S_BLN];

OWN
        ACP_TYPE        : LONG,                                 ! Volume ACP type
        ROOTNAME_BUF    : BBLOCK [ROOTVOL_NAMLEN],              ! Root volume device name buffer
        VOLUME_NUMBER   : LONG,                                 ! Relative volume number

        DVI_LIST        : BBLOCK [10*4]                         ! $GETDVI item list
                          INITIAL (
                                    WORD (64),                  !    ITEM - Root volume device name
                                    WORD (DVI$_ROOTDEVNAM),
                                    LONG (0),                   !    Filled in at run time
                                    LONG (0),                   !    Filled in at run time
                                    WORD (4),                   !    ITEM - Relative Volume Number
                                    WORD (DVI$_VOLNUMBER),
                                    LONG (VOLUME_NUMBER),
                                    LONG (0),
                                    WORD (4),                   !    ITEM - ACP type
                                    WORD (DVI$_ACPTYPE),
                                    LONG (ACP_TYPE),
                                    LONG (0),
                                    LONG (0)                    !    List terminator
                                  );

!
! Initialize the data area.
!
CH$FILL (0, (ZZ$_DATA_END-AA$_DATA_BEGIN), AA$_DATA_BEGIN);

!
! For each device listed, determine
!       o the file structure level (ODS-1 or ODS-2)
!       o its Relative Volume Number (RVN)
!       o if it has a journal file
!       o the device name the root volume is mounted on
!       o if the root volume was mounted by this invocation of $MOUNT
!
INCR J FROM 0 TO .LIST_LENGTH-1 DO
    BEGIN
    !
    ! Get the RVN and root volume device name.
    !
    ROOTVOL_DSC[DSC$A_POINTER] = ROOTVOL_BUF;
    DVI_LIST[4, 0, 32, 0] = ROOTVOL_BUF;
    DVI_LIST[8, 0, 32, 0] = ROOTVOL_DSC[DSC$W_LENGTH];
    $GETDVI (DEVNAM = DEVNAM_LIST[.J, DSC$W_LENGTH], ITMLST=DVI_LIST);
    VOLSET_RVN[.J] = .VOLUME_NUMBER;

    !
    ! If the volume supports the ODS-2 file structure, note it, and
    ! determine if the root volume for this volume set was mounted in
    ! this invocation of $MOUNT.  Note that if the RVN is LEQ 1, then
```

```
3        ! this volume is the root volume.
3        !
3        IF .ACP_TYPE EQL DVI$C_ACP_F11V2
3        THEN
4            BEGIN
4            ODS2_VOLUME[.J] = 1;                                              ! Note that this volume supports ODS-2.
4            RUJRNL_PRESENT[.J] = ACCESS_JOURNAL (DEVNAM_LIST[.J, DSC$W_LENGTH]);
4            IF .VOLSET_RVN[.J] LEQ 1
4            THEN
5                BEGIN                                                         ! This is the volume is its own root
5                ROOT_PRESENT[.J] = 1;                                         ! Note the the root volume is present in the
5                ROOTVOL_INDEX[.J] = .J + 1;                                   ! Note the index of the root volume (biased
4                END;
4            !
4            ! Check the root volume device name for this device against those
4            ! in the supplied list.  If no match, it means that this volume is
4            ! an addition to an already mounted volume set.
4            !
4            I = 0;
4            WHILE NOT .ROOT_PRESENT[.J] AND (.I LEQ .LIST_LENGTH-1) DO
4                IF CH$EQL    (.DEVNAM_LIST[.I, DSC$W_LENGTH],
4                             .DEVNAM_LIST[.I, DSC$A_POINTER],
4                             .ROOTVOL_DSC[DSC$W_LENGTH],
4                             .ROOTVOL_DSC[DSC$A_POINTER]
4                            )
4                THEN
5                    BEGIN
5                    ROOT_PRESENT[.J] = 1;                                     ! Note the the root volume is present in the
5                    ROOTVOL_INDEX[.J] = .I + 1;                               ! Note the index of the root volume (biased
5                    END
4                ELSE
4                    I = .I + 1;
3            END;
2        END;
2
1 END;                                                                         ! End of ANALYZE_DEVLIST


                                                       .PSECT  $OWN$,NOEXE,2

                                   000C8 ACP_TYPE:
                                                       .BLKB   4
                                   000CC ROOTNAME_BUF:
                                                       .BLKB   64
                                   0010C VOLUME_NUMBER:
                                                       .BLKB   4
                          0040     00110 DVI_LIST:
                                                       .WORD   64                        ;
                          0032     00112               .WORD   50                        ;
                      00000000     00114               .LONG   0                         ;
                      00000000     00118               .LONG   0                         ;
                          0004     0011C               .WORD   4                         ;
                          002E     0011E               .WORD   46                        ;
                     00000000'     00120               .ADDRESS VOLUME_NUMBER            ;
                      00000000     00124               .LONG   0                         ;
                          0004     00128               .WORD   4                         ;
```

```
                                     0042  0012A           .WORD    66                                           ;
                                00000000'  0012C           .ADDRESS ACP_TYPE                                     ;
                                 00000000  00130           .LONG    0                                            ;
                                 00000000  00134           .LONG    0                                            ;

                                                           .EXTRN   SYS$GETDVI

                                                           .PSECT   $CODE$,NOWRT,2

                                     007C  00000 ANALYZE_DEVLIST:
                                                           .WORD    Save R2,R3,R4,R5,R6                          ; 0977
                      56    0000'   CF  9E 00002           MOVAB    ROOT_PRESENT, R6                             ;
 0078   8F     00     6E            00  2C 00007           MOVC5    #0, (SP), #0, #120, AA$_DATA_BEGIN           ; 1066
                              F8    A6     0000E                                                                 ;
                      54            01  CE 00010           MNEGL    #1, J                                        ; 1084
                                  0089  31 00013           BRW      8$                                           ;
               0C     A6      10    A6  9E 00016 1$:       MOVAB    ROOTVOL_BUF, ROOTVOL_DSC+4                   ; 1081
             010C     C6      10    A6  9E 0001B           MOVAB    ROOTVOL_BUF, DVI_LIST+4                      ; 1082
             0110     C6      08    A6  9E 00021           MOVAB    ROOTVOL_DSC, DVI_LIST+8                      ; 1083
                                    7E  7C 00027           CLRQ     -(SP)                                        ; 1084
                                    7E  7C 00029           CLRQ     -(SP)                                        ;
                            0108    C6  9F 0002B           PUSHAB   DVI_LIST                                     ;
                              04  BC44  7F 0002F           PUSHAQ   @DEVNAM_LIST[J]                              ;
                                    7E  7C 00033           CLRQ     -(SP)                                        ;
         00000000G    00            08  FB 00035           CALLS    #8, SYS$GETDVI                               ;
                 60 A644    0104    C6  90 0003C           MOVB     VOLUME_NUMBER, VOLSET_RVN[J]                 ; 1085
                      02    00C0    C6  D1 00043           CMPL     ACP_TYPE, #2                                 ; 1093
                                    55  12 00048           BNEQ     8$                                           ;
               00     FC    A6      54  E2 0004A           BBSS     J, ODS2_VOLUME, 2$                           ; 1096
                              04  BC44  7F 0004F 2$:       PUSHAQ   @DEVNAM_LIST[J]                              ; 1097
                            FE3D    CF  01  FB 00053       CALLS    #1, ACCESS_JOURNAL                           ;
   04   A6     01     54            50  F0 00058           INSV     R0, J, #1, RUJRNL_PRESENT                    ;
                      01      60  A644  91 0005E           CMPB     VOLSET_RVN[J], #1                            ; 1098
                                    0A  1A 00063           BGTRU    4$                                           ;
               00     66            54  E2 00065           BBSS     J, ROOT_PRESENT, 3$                          ; 1101
          50 A644     54            01  81 00069 3$:       ADDB3    #1, J, ROOTVOL_INDEX[J]                      ; 1102
                                    55  D4 0006F 4$:       CLRL     I                                            ; 1109
               2A     66            54  E0 00071 5$:       BBS      J, ROOT_PRESENT, 8$                          ; 1110
               50     08    AC      01  C3 00075           SUBL3    #1, LIST_LENGTH, R0                          ;
                      50            55  D1 0007A           CMPL     I, R0                                        ;
                                    20  14 0007D           BGTR     8$                                           ;
                      50      04  BC45  7E 0007F           MOVAQ    @DEVNAM_LIST[I], R0                          ; 1111
   08   A6     00     04    B0      60  2D 00084           CMPC5    (R0), @4(R0), #0, ROOTVOL_DSC, -             ;
                              0C    B6     0008B                    @ROOTVOL_DSC+4                               ;
                                    0C  12 0008D           BNEQ     7$                                           ;
               00     66            54  E2 0008F           BBSS     J, ROOT_PRESENT, 6$                          ; 1118
          50 A644     55            01  81 00093 6$:       ADDB3    #1, I, ROOTVOL_INDEX[J]                      ; 1119
                                    D6  11 00099           BRB      5$                                           ; 1111
                                    55  D6 0009B 7$:       INCL     I                                            ; 1122
                                    D2  11 0009D           BRB      5$                                           ; 1111
               01     54      08    AC  F2 0009F 8$:       AOBLSS   LIST_LENGTH, J, 9$                           ; 1076
                                        04 000A4           RET                                                   ; 1126
                                  FF6E  31 000A5 9$:       BRW      1$                                           ; 1076

; Routine Size:  168 bytes,    Routine Base:  $CODE$ + 0187
```

```
1 ROUTINE CREATE_RUJ (DEVICE_NAME, PERMANENT_RUJ) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Create a JSB and call the $CREJNL system service to create a
1 !       RUJ on a specified device.  The longevity of the RUJ is also
1 !       specified.
1 !
1 ! CALLING SEQUENCE:
1 !       CREATE_RUJ (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1    : address of a device name descriptor
1 !       ARG2    : if 0 then create a temporary RUJ, otherwise create a permanent RUJ.
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       Status code passed back by RUJ routines.
1 !
1 ! SIDE EFFECTS:
1 !       The state of the volume may be changed depending on the action of
1 !       the journalling ACP.
1 !       Success messages are SIGNALED, errors are handled outside of this
1 !       routine.
1 !
1 !--
1
2 BEGIN                                                         ! Start of CREATE_JOURNAL
2
2 EXTERNAL
2       JRNL_EXTEND,                                            ! RUJ default file extension size
2       JRNL_QUOTA,                                             ! RUJ byte quota per recovery unit
2       JRNL_SIZE,                                              ! Recovery Unit Journal (RUJ)  initial size
2       JRNL_RECORD_SIZE,                                               ! RUJ maximum record size
2       MOUNT_OPTIONS   : BITVECTOR;                            ! Option flags
2
2 !EXTERNAL ROUTINE
2 !     GET_VOLUME_NAME : ADDRESSING_MODE (GENERAL),            ! Return volume name for a given device
2 !     INTERCEPT_SIGNAL: ADDRESSING_MODE (GENERAL);            ! Condition handler
2
2 LOCAL
2       JRNL_CHANNEL,                                           ! Channel to volume
2       STATUS          : BBLOCK [4],
2       VOLUME_NAME     : REF BBLOCK;                           ! Volume name descriptor
2
2 OWN
2       JSB             : BBLOCK [JSB$C_LENGTH];                ! Journal State Block
```

```
;     2
;     2
;     2 !
;     2 ! Enable a condition handler to field and print signalled messages.
;     2 !
;     2 ENABLE INTERCEPT_SIGNAL;
;     2
;     2 !
;     2 ! Build the JSB in OWN storage (stack space is at a premium).
;     2 !
;     2 JRNL_CHANNEL = 0;
;     2 CH$FILL (0, JSB$C_LENGTH, JSB);                          ! Initialize the JSB
;     2 JSB[JSB$B_JNLTYP] = DT$_RUJNL;                           ! Set the journal type
;     2 JSB[JSB$L_FILSIZ] = .JRNL_SIZE;                          ! Set RUJ initial size
;     2 JSB[JSB$W_FILEXT] = .JRNL_EXTEND;                        ! Set RUJ extend size
;     2 JSB[JSB$W_MAXSIZ] = .JRNL_RECORD_SIZE;                   ! Set RUJ max record size
;     2 JSB[JSB$L_QUOTA] =  .JRNL_QUOTA;                         ! Set RUJ process byte quota
;     2 JSB[JSB$W_BUFSIZ] = 1;                                   ! Use 1 block buffers in driver
;     2 JSB[JSB$B_COPIES] = 1;                                   ! Only one RUJ on volume
;     2 JSB[JSB$B_JNLDEV] = JSB$C_DISK;                          ! Specify device class
;     2 JSB[JSB$B_ACMODE] = PSL$C_USER;                          ! Specify journal access mode
;     2 JSB[JSB$L_PRINAMDES] = .DEVICE_NAME;                     ! devnam descriptor address
;     2 CH$FILL (-1, 8, JSB[JSB$Q_EXPDAT]);                      ! default exparation date
;     2                                                          !  Ignorant Bliss STILL does not handle quad
;     2 !
;     2 ! Set RUJ creation flags
;     2 !
;     2 IF .PERMANENT_RUJ
;     2 THEN
;     2     JSB[JSB$V_CIF] = .MOUNT_OPTIONS[OPT_NEWJRNL]
;     2 ELSE
;     3     BEGIN
;     3     JSB[JSB$V_TMPJNL] = 1;
;     3     JSB[JSB$V_TMPFIL] = 1;
;     2     END;
;     2
;     2 !
;     2 ! Activate the journal (this is a synchronous operation).
;     2 ! Parameters that are defaulted are flaged with (*).
;     2 !
;     2 ! The $CREJNL code is loadable, and not necessarily present in the system.
;     2 !
;   P 2 STATUS = $CREJNL (CHAN   = JRNL_CHANNEL,                ! Channel returned by the service
;   P 2                   JSB    = JSB,                         ! JSB address
;   P 2                   ACMODE = 0,                           ! Access mode for channel
;   P 2                   PROT   = 0,                           ! Protection for channel     (*)
;   P 2                   FACCOD = 0,                           ! Addr of facility code      (*)
;   P 2                   FLAGS  = 0,                           ! Option flags               (*)
;   P 2                   OBJUIC = 0,                           ! Object UIC                 (*)
;   P 2                   SESSID = 0,                           ! Session ID                 (*)
;   P 2                   IOSB   = 0                            ! Internal IOSB              (*)
;     2                  );
;     2
;     2 !
;     2 ! Device already exists should be changed to a success code
;     2 !
;     2 IF .STATUS EQL CJF$_DEVEXI
```

```
2 THEN
2     STATUS[STS$V_SEVERITY] = STS$K_SUCCESS;
2 !
2 ! Deassign the journal channel.  The return status is not interesting.
2 ! If the RUJ that was created was a temporary journal, deassigning the
2 ! channel will get rid of the journal and the RUJ file.
2 !
2 $DEASJNL (CHAN = .JRNL_CHANNEL);
2
2 !
2 ! If the status code returned by $CREJNL indicates an "informational" message,
2 ! attempt to inform the user of the event.  All other status values are handled
2 ! outside of this routine.  The status code is signalled, and a condition
2 ! handler prints the message.
2 !
2 IF .STATUS[STS$V_SEVERITY] EQL STS$K_INFO
2 THEN
3     BEGIN
3     VOLUME_NAME = GET_VOLUME_NAME (.DEVICE_NAME);
3     ERR_MESSAGE (MOUN$_VOLSTATUS, 1, .VOLUME_NAME, .STATUS);
2     END;
2
2 RETURN .STATUS
2
1 END;                                                           ! End of CREATE_JOURNAL


                                                    .PSECT  $OWN$,NOEXE,2

                                   00138 JSB:       .BLKB   80

                                                    .EXTRN  JRNL_EXTEND, JRNL_QUOTA
                                                    .EXTRN  JRNL_SIZE, JRNL_RECORD_SIZE
                                                    .EXTRN  CJF$CREJNL, CJF$DEASJNL

                                                    .PSECT  $CODE$,NOWRT,2

                              007C 00000 CREATE_RUJ:
                                                    .WORD   Save R2,R3,R4,R5,R6                        ; 1127
                 56      0000'  CF  9E 00002        MOVAB   JSB, R6                                    ;
                 6D      00B3   CF  DE 00007        MOVAL   5$, (FP)                                   ; 1164
                                7E  D4 0000C        CLRL    JRNL_CHANNEL                               ; 1194
  0050   8F          00  6E     00  2C 0000E        MOVC5   #0, (SP), #0, #80, JSB                     ; 1195
                                66     00015                                                           ;
            18  A6 00000000G    00  D0 00016        MOVL    JRNL_SIZE, JSB+24                          ; 1197
            1C  A6 00000000G    00  B0 0001E        MOVW    JRNL_EXTEND, JSB+28                        ; 1198
            14  A6 00000000G    00  B0 00026        MOVW    JRNL_RECORD_SIZE, JSB+20                   ; 1199
            20  A6 00000000G    00  D0 0002E        MOVL    JRNL_QUOTA, JSB+32                         ; 1200
            1E  A6              01  B0 00036        MOVW    #1, JSB+30                                 ; 1201
            3A  A6              01  90 0003A        MOVB    #1, JSB+58                                 ; 1202
            0A  A6      0101    8F  B0 0003E        MOVW    #257, JSB+10                               ; 1196
            24  A6              03  90 00044        MOVB    #3, JSB+36                                 ; 1204
            44  A6        04    AC  D0 00048        MOVL    DEVICE_NAME, JSB+68                        ; 1205
  08         FF   8F    6E      00  2C 0004D        MOVC5   #0, (SP), #-1, #8, JSB+60                  ; 1206
                          3C    A6     00053                                                           ;
                 0C       08    AC  E9 00055        BLBC    PERMANENT_RUJ, 1$                          ; 1211
```

```
     2C   A6          01          03 00000000G  00  F0 00059         INSV    MOUNT_OPTIONS+7, #3, #1, JSB+44      ; 1213
                                                04  11 00063         BRB     2$                                   ;
                              2C   A6           11  88 00065 1$:     BISB2   #17, JSB+44                          ; 1217
                                                7E  7C 00069 2$:     CLRQ    -(SP)                                ; 1235
                                                7E  7C 0006B         CLRQ    -(SP)                                ;
                                                7E  7C 0006D         CLRQ    -(SP)                                ;
                                                7E  D4 0006F         CLRL    -(SP)                                ;
                                                56  DD 00071         PUSHL   R6                                   ;
                                           20   AE  9F 00073         PUSHAB  JRNL_CHANNEL                         ;
                        00000000G  00           09  FB 00076         CALLS   #9, CJF$CREJNL                       ;
                                   52           50  D0 0007D         MOVL    R0, STATUS                           ;
                        00000000G  8F           52  D1 00080         CMPL    STATUS, #CJF$_DEVEXI                 ; 1240
                                                05  12 00087         BNEQ    3$                                   ;
     52               03           00           01  F0 00089         INSV    #1, #0, #3, STATUS                   ; 1242
                                                7E  D4 0008E 3$:     CLRL    -(SP)                                ; 1248
                                           04   AE  DD 00090         PUSHL   JRNL_CHANNEL                         ;
                        00000000G  00           02  FB 00093         CALLS   #2, CJF$DEASJNL                      ;
     03               52           03           00  ED 0009A         CMPZV   #0, #3, STATUS, #3                   ; 1256
                                                19  12 0009F         BNEQ    4$                                   ;
                                           04   AC  DD 000A1         PUSHL   DEVICE_NAME                          ; 1259
                            0000V  CF           01  FB 000A4         CALLS   #1, GET_VOLUME_NAME                  ;
                                                05  BB 000A9         PUSHR   #^M<R0,R2>                           ; 1260
                                                01  DD 000AB         PUSHL   #1                                   ;
                                     0072A07B   8F  DD 000AD         PUSHL   #7512187                             ;
                        00000000G  00           04  FB 000B3         CALLS   #4, LIB$SIGNAL                       ;
                                   50           52  D0 000BA 4$:     MOVL    STATUS, R0                           ; 1263
                                                    04 000BD         RET                                          ; 1265
                                              0000 000BE 5$:         .WORD   Save nothing                         ; 1164
                                                7E  D4 000C0         CLRL    -(SP)                                ;
                                                5E  DD 000C2         PUSHL   SP                                   ;
                                   7E      04   AC  7D 000C4         MOVQ    4(AP), -(SP)                         ;
                            0000V  CF           03  FB 000C8         CALLS   #3, INTERCEPT_SIGNAL                 ;
                                                    04 000CD         RET                                          ;

; Routine Size:  206 bytes,     Routine Base:  $CODE$ + 022F
```

```
ROUTINE DISMOUNT_VOLSET (DEVLIST, LIST_SIZE, INDEX, ENTIRE_VOLSET, ERROR_STATUS) : NOVALUE =

!++
!
! FUNCTIONAL DESCRIPTION:
!
!       Dismount all or part of a volume set, and update the local device/volume
!       data base to reflect the loss of one or more volumes.
!
! CALLING SEQUENCE:
!       DISMOUNT_VOLSET (ARG1, ARG2)
!
! INPUT PARAMETERS:
!       ARG1    : address of the list of all volumes mounted in this invocation of $MOUNT
!       ARG2    : number of volumes mounted in this invocation of $MOUNT
!       ARG3    : index into ARG1 (the current volume)
!       ARG4    : if 0 dismount a single volume, otherwise dismount the entire volume set
!       ARG5    : the journalling error status code
!
! IMPLICIT INPUTS:
!       <see the EXTERNAL declarations>
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       NONE
!
! ROUTINE VALUE:
!       NONE.
!
! SIDE EFFECTS:
!       The internal device data base is updated to reflect the dismounting of the volume(s).
!       The user is informed of the progress.
!
!--

BEGIN                                                               ! Start of CREATE_JOURNAL

!EXTERNAL LITERAL
!       ROOTVOL_NAMLEN  = 64;                                       ! Maximum length of a device name

!EXTERNAL
!       AA$_DATA_BEGIN  : VECTOR [0],                               ! Mark start of data area
!       DISMOUNT_COUNT  : BBLOCK [4],                               ! Count of volumes that were dismounted
!       ODS2_VOLUME     : BITVECTOR [DEVMAX],                       ! Indicates volume supports ODS-2 file struc
!       ROOT_PRESENT    : BITVECTOR [DEVMAX],                       ! Indicates presence of root volume for a gi
!       RUJRNL_PRESENT  : BITVECTOR [DEVMAX],                       ! Indicates presence of RUJ file on a given
!       ROOTVOL_DSC     : BBLOCK [DSC$K_S_BLN],                     ! Root volume device name descriptor
!       ROOTVOL_BUF     : BBLOCK [ROOTVOL_NAMLEN],                  ! Buffer for root volume device name
!       ROOTVOL_INDEX   : BBLOCK [DEVMAX, BYTE, UNSIGNED],          ! Each entry contians index+1 of root volume
!       VOLSET_RVN      : VECTOR [DEVMAX, BYTE, UNSIGNED],          ! Vector of Relative Volume Numbers
!       ZZ$_DATA_END    : VECTOR [0];                               ! Mark end of data area

!EXTERNAL ROUTINE
!       GET_VOLUME_NAME : ADDRESSING_MODE (GENERAL),                ! Return volume name for a given device
!       INTERCEPT_SIGNAL: ADDRESSING_MODE (GENERAL);                ! Condition handler
```

```
LOCAL
        MULTI_DISMOUNT  : LONG,                                 ! Boolean, indicates more than one volume di
        PRIMARY_STATUS  : LONG,                                 ! Primary status message code
        ROOT_INDEX      : LONG,                                 ! Index into device list of root volume for
        VOLUME_NAME     : REF BBLOCK;                           ! Volume name descriptor

MAP
        DEVLIST         : REF BBLOCKVECTOR [DEVMAX, DSC$K_S_BLN];

OWN
        DISMOUNT_FLAGS  : BBLOCK [4],                           ! Flags for $DISMOU call
        ROOTDEV1_BUF    : BBLOCK [64],                          ! Buffer for primary root device name
        ROOTDEV1_DSC    : BBLOCK [DSC$K_S_BLN]                  ! Descriptor for primary root device name
                          PRESET ([DSC$A_POINTER] = ROOTDEV1_BUF),
        ROOTDEV2_BUF    : BBLOCK [64],                          ! Buffer for secondary root device name
        ROOTDEV2_DSC    : BBLOCK [DSC$K_S_BLN]                  ! Descriptor for secondary root device name
                          PRESET ([DSC$A_POINTER] = ROOTDEV2_BUF),
        VOLNAMLIST_BUF  : BBLOCK [DEVMAX*12],                   ! Buffer for volume names
        VOLNAMLIST_DSC  : BBLOCKVECTOR [DEVMAX, DSC$K_S_BLN],   ! Vector of volume name descriptors
        VOLSET_SIZE     : LONG,                                 ! Number of volumes in the volume set
        VOLSETNAM_BUF   : BBLOCK [12],                          ! Volume set name buffer
        VOLSETNAM_DSC   : BBLOCK [DSC$K_S_BLN],                 ! Volume set name descriptor
        VOLUME_REMOVED  : BITVECTOR [DEVMAX],                   ! Mark volume as being removed

        PRIMARY_DVI     : BBLOCK [7*4]                          ! $GETDVI item list for volume set info
                          INITIAL (
                                  WORD (4),                     !    ITEM - Volume set size
                                  WORD (DVI$_VOLCOUNT),
                                  LONG (VOLSET_SIZE),
                                  LONG (0),
                                  WORD (64),                    !    ITEM - Volume set root device name
                                  WORD (DVI$_ROOTDEVNAM),
                                  LONG (ROOTDEV1_BUF),
                                  LONG (ROOTDEV1_DSC),
                                  LONG (0)                      ! List terminator
                                 ),

        SECONDARY_DVI   : BBLOCK [4*4]                          ! $GETDVI item list for volume root device n
                          INITIAL (
                                  WORD (64),                    !    ITEM - Volume set root device name
                                  WORD (DVI$_ROOTDEVNAM),
                                  LONG (ROOTDEV2_BUF),
                                  LONG (ROOTDEV2_DSC),
                                  LONG (0)                      ! List terminator
                                 );


!
! Enable a condition handler to field and print signalled messages.
!
ENABLE INTERCEPT_SIGNAL;

!
! For the specified device, fetch the root device name and the number
! of volumes in the volume set.
!
```

```
$GETDVI (DEVNAM=DEVLIST[.INDEX, DSC$W_LENGTH], ITMLST=PRIMARY_DVI);

!
! For each volume in the device list that is being dismounted,
! update the internal device data base to reflect this change of status.
! If only volume need be dismounted, this is very simple, otherwise...
!
IF NOT .ENTIRE_VOLSET
OR .VOLSET_SIZE LEQ 1
THEN
    BEGIN
    !
    ! Remove this volume from the data base, and increment the dismount count.
    !
    ODS2_VOLUME[.INDEX] = 0;
    ROOT_PRESENT[.INDEX] = 0;
    RUJRNL_PRESENT[.INDEX] = 0;
    VOLSET_RVN[.INDEX] = 0;
    ROOTVOL_INDEX[.INDEX] = 0;
    DISMOUNT_COUNT = .DISMOUNT_COUNT + 1;
    END
ELSE
    BEGIN
    !
    ! ... otherwise, if the volume being dismounted is part of a volume
    ! set, each volume in the device list that is associated with the
    ! volume set must be removed from the device data base and dismounted.
    ! The process is complicated if the root volume is not present in the
    ! device list.  A ROOTVOL_INDEX of 0 implies that the root volume is
    ! not present in the list, and a string comparison is necessary.
    !
    VOLUME_REMOVED = 0;
    ROOT_INDEX = .ROOTVOL_INDEX[.INDEX];
    INCR J FROM 0 TO .LIST_SIZE-1 DO
        BEGIN
        !
        ! See if the root volume indexes match.  If so, further checks are mandated.
        !
        IF  (.ROOT_INDEX EQL .ROOTVOL_INDEX[.J])
        AND .ODS2_VOLUME[.J]
        THEN
            BEGIN
            !
            ! The indexes match.  If the number is nonzero, then the root volume has been mounted
            ! by this invocation of $MOUNT.  Otherwise, the root device name of the current device
            ! must be fetched and compared against the root device name supplied by the caller.
            !
            IF .ROOT_INDEX NEQ 0
            OR ($GETDVI (DEVNAM=DEVLIST[.J, DSC$W_LENGTH], ITMLST=SECONDARY_DVI);
                IF CH$EQL (.ROOTDEV1_DSC[DSC$W_LENGTH],
                           .ROOTDEV1_DSC[DSC$A_POINTER],
                           .ROOTDEV2_DSC[DSC$W_LENGTH],
                           .ROOTDEV2_DSC[DSC$A_POINTER]
                          )
                THEN
                    1
                ELSE
```

```
                        0
                        )
                THEN
                    BEGIN
                    !
                    ! Remove this volume from the data base, and increment the dismount count.
                    !
                    ODS2_VOLUME[.J] = 0;
                    ROOT_PRESENT[.J] = 0;
                    RUJRNL_PRESENT[.J] = 0;
                    VOLSET_RVN[.J] = 0;
                    ROOTVOL_INDEX[.J] = 0;
                    VOLUME_REMOVED[.J] = 1;
                    DISMOUNT_COUNT = .DISMOUNT_COUNT + 1;
                    !
                    ! Get and save the volume name.  This must be done before dismounting the volume.
                    !
                    VOLUME_NAME = GET_VOLUME_NAME (DEVLIST[.J, DSC$W_LENGTH]);
                    VOLNAMLIST_DSC[.J, DSC$W_LENGTH] = .VOLUME_NAME[DSC$W_LENGTH];
                    VOLNAMLIST_DSC[.J, DSC$A_POINTER] = VOLNAMLIST_BUF + (12 * .J);
                    CH$MOVE (.VOLUME_NAME[DSC$W_LENGTH], .VOLUME_NAME[DSC$A_POINTER], .VOLNAMLIST_DSC[.J, DSC$A_
                    END;
                END;
            END;
        END;
    !
    ! The informational message will vary depending on whether or not this is
    ! a volume set.  Determine the primary message code and the volume name.
    ! The volume name must be fetched before the volume is dismounted.
    ! A local copy of the volume set name is made for future reference.
    !
    IF (.VOLSET_SIZE GTR 1) AND .ENTIRE_VOLSET
    THEN
        BEGIN
        MULTI_DISMOUNT = 1;
        VOLUME_NAME = GET_VOLUME_NAME (DEVLIST[.INDEX, DSC$W_LENGTH], 1);
        VOLSETNAM_DSC[DSC$W_LENGTH] = .VOLUME_NAME[DSC$W_LENGTH];
        VOLSETNAM_DSC[DSC$A_POINTER] = VOLSETNAM_BUF;
        CH$MOVE (.VOLUME_NAME[DSC$W_LENGTH], .VOLUME_NAME[DSC$A_POINTER], .VOLSETNAM_DSC[DSC$A_POINTER]);
        PRIMARY_STATUS = MOUN$_VOLSETSTS;
        END
    ELSE
        BEGIN
        MULTI_DISMOUNT = 0;
        VOLUME_NAME = GET_VOLUME_NAME (DEVLIST[.INDEX, DSC$W_LENGTH]);
        PRIMARY_STATUS = MOUN$_VOLSTATUS;
        END;

    !
    ! Dismount all or part of the volume set, as specified by the caller.
    !
    DISMOUNT_FLAGS = 0;
    IF NOT .ENTIRE_VOLSET
    THEN
        DISMOUNT_FLAGS[DMT$V_UNIT] = 1;
    $DISMOU (DEVNAM=DEVLIST[.INDEX, DSC$W_LENGTH], FLAGS=.DISMOUNT_FLAGS);
```

```
2 !
2 ! Inform the user that at least one volume has been dismounted.
2 ! Include the status code specified by the caller.
2 !
2 ERR_MESSAGE (.PRIMARY_STATUS, 1, .VOLUME_NAME, MOUN$_CJFERR, 0, .ERROR_STATUS);
2
2 !
2 ! If more than one volume was dismounted, inform the user of each volume
2 ! that was dismounted.
2 !
2 IF .MULTI_DISMOUNT
2 THEN
2     INCR J FROM 0 TO .LIST_SIZE-1 DO
2         IF .VOLUME_REMOVED[.J]
2         THEN
3             BEGIN
3             VOLUME_NAME = GET_VOLUME_NAME (DEVLIST[.J, DSC$W_LENGTH]);
3             ERR_MESSAGE (MOUN$_DISMOUNTD, 2, VOLNAMLIST_DSC[.J, DSC$W_LENGTH], VOLSETNAM_DSC);
2             END;
2
1 END;                                                    ! End of DISMOUNT_VOLSET


                                        .PSECT  $OWN$,NOEXE,2

                     00188 DISMOUNT_FLAGS:
                                        .BLKB   4
                     0018C ROOTDEV1_BUF:
                                        .BLKB   64
                 00# 001CC ROOTDEV1_DSC:
                                        .BYTE   0[4]
           00000000' 001D0              .ADDRESS ROOTDEV1_BUF
                     001D4 ROOTDEV2_BUF:
                                        .BLKB   64
                 00# 00214 ROOTDEV2_DSC:
                                        .BYTE   0[4]
           00000000' 00218              .ADDRESS ROOTDEV2_BUF
                     0021C VOLNAMLIST_BUF:
                                        .BLKB   192
                     002DC VOLNAMLIST_DSC:
                                        .BLKB   128
                     0035C VOLSET_SIZE:
                                        .BLKB   4
                     00360 VOLSETNAM_BUF:
                                        .BLKB   12
                     0036C VOLSETNAM_DSC:
                                        .BLKB   8
                     00374 VOLUME_REMOVED:
                                        .BLKB   2
                     00376              .BLKB   2
                0004 00378 PRIMARY_DVI:
                                        .WORD   4
                0030 0037A              .WORD   48
           00000000' 0037C              .ADDRESS VOLSET_SIZE
           00000000  00380              .LONG   0
```

```
                                     0040  00384           .WORD    64                                        ;
                                     0032  00386           .WORD    50                                        ;
                                00000000'  00388           .ADDRESS ROOTDEV1_BUF                              ;
                                00000000'  0038C           .ADDRESS ROOTDEV1_DSC                              ;
                                 00000000  00390           .LONG    0                                         ;
                                     0040  00394 SECONDARY_DVI:
                                                           .WORD    64                                        ;
                                     0032  00396           .WORD    50                                        ;
                                00000000'  00398           .ADDRESS ROOTDEV2_BUF                              ;
                                00000000'  0039C           .ADDRESS ROOTDEV2_DSC                              ;
                                 00000000  003A0           .LONG    0                                         ;

                                                           .EXTRN   SYS$DISMOU, SYS$CMKRNL

                                                           .PSECT   $CODE$,NOWRT,2

                                     OFFC  00000 DISMOUNT_VOLSET:
                                                           .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11      ; 1266
                         5B 00000000G  00  9E  00002       MOVAB    SYS$GETDVI, R11                           ;
                         5A     0000'  CF  9E  00009       MOVAB    ROOTVOL_INDEX, R10                        ;
                         6D      0196  CF  DE  0000E       MOVAL    21$, (FP)                                 ; 1340
                                       7E  7C  00013       CLRQ     -(SP)                                     ; 1380
                                       7E  7C  00015       CLRQ     -(SP)                                     ;
                                 0320  CA  9F  00017       PUSHAB   PRIMARY_DVI                               ;
                         52        0C  AC  D0  0001B       MOVL     INDEX, R2                                 ;
                         59     04 BC42    7E  0001F       MOVAQ    @DEVLIST[R2], R9                          ;
                                       59  DD  00024       PUSHL    R9                                        ;
                                       7E  7C  00026       CLRQ     -(SP)                                     ;
                         6B            08  FB  00028       CALLS    #8, SYS$GETDVI                            ;
                         07        10  AC  E9  0002B       BLBC     ENTIRE_VOLSET, 1$                         ; 1387
                         01      0304  CA  D1  0002F       CMPL     VOLSET_SIZE, #1                           ; 1388
                                       1C  14  00034       BGTR     5$                                        ;
               00        AC   AA       52  E5  00036 1$:   BBCC     R2, ODS2_VOLUME, 2$                       ; 1394
               00        B0   AA       52  E5  0003B 2$:   BBCC     R2, ROOT_PRESENT, 3$                      ; 1395
               00        B4   AA       52  E5  00040 3$:   BBCC     R2, RUJRNL_PRESENT, 4$                    ; 1396
                              10     AA42  94  00045 4$:   CLRB     VOLSET_RVN[R2]                            ; 1397
                                     6A42  94  00049       CLRB     ROOTVOL_INDEX[R2]                         ; 1398
                              A8       AA  D6  0004C       INCL     DISMOUNT_COUNT                            ; 1399
                                     0099  31  0004F       BRW      14$                                       ; 1387
                                 031C  CA  B4  00052 5$:   CLRW     VOLUME_REMOVED                            ; 1411
                         58          6A42  9A  00056       MOVZBL   ROOTVOL_INDEX[R2], ROOT_INDEX             ; 1412
                         56            01  CE  0005A       MNEGL    #1, J                                     ; 1418
                                     0081  31  0005D       BRW      12$                                       ;
       58       6A46     08            00  ED  00060 6$:   CMPZV    #0, #8, ROOTVOL_INDEX[J], ROOT_INDEX      ;
                                       79  12  00066       BNEQ     12$                                       ;
               74        AC   AA       56  E1  00068       BBC      J, ODS2_VOLUME, 12$                       ; 1419
                                       58  D5  0006D       TSTL     ROOT_INDEX                                ; 1427
                                       21  12  0006F       BNEQ     7$                                        ;
                                       7E  7C  00071       CLRQ     -(SP)                                     ; 1428
                                       7E  7C  00073       CLRQ     -(SP)                                     ;
                                 033C  CA  9F  00075       PUSHAB   SECONDARY_DVI                             ;
                                    04 BC46    7F  00079   PUSHAQ   @DEVLIST[J]                               ;
                                       7E  7C  0007D       CLRQ     -(SP)                                     ;
                         6B            08  FB  0007F       CALLS    #8, SYS$GETDVI                            ;
  01BC    CA       00   0178  DA  0174  CA  2D  00082      CMPC5    ROOTDEV1_DSC, @ROOTDEV1_DSC+4, #0, -      ; 1429
                                  01C0  DA      0008D               ROOTDEV2_DSC, @ROOTDEV2_DSC+4
                                        4F  12  00090      BNEQ     12$                                       ;
```

```
           00        AC   AA            56  E5 00092 7$:     BBCC    J, ODS2_VOLUME, 8$                       ; 1444
           00        B0   AA            56  E5 00097 8$:     BBCC    J, ROOT_PRESENT, 9$                      ; 1445
           00        B4   AA            56  E5 0009C 9$:     BBCC    J, RUJRNL_PRESENT, 10$                   ; 1446
                               10 AA46  94 000A1 10$:    CLRB    VOLSET_RVN[J]                            ; 1447
                                  6A46  94 000A5         CLRB    ROOTVOL_INDEX[J]                         ; 1448
           00      031C   CA            56  E2 000A8         BBSS    J, VOLUME_REMOVED, 11$                   ; 1449
                                  A8    AA  D6 000AE 11$:    INCL    DISMOUNT_COUNT                           ; 1450
                                  04 BC46  7F 000B1         PUSHAQ  @DEVLIST[J]                              ; 1454
                 0000V   CF             01  FB 000B5         CALLS   #1, GET_VOLUME_NAME                      ;
                         57             50  D0 000BA         MOVL    R0, VOLUME_NAME                          ;
                               0284 CA46  7F 000BD         PUSHAQ  VOLNAMLIST_DSC[J]                        ; 1455
                         9E             67  B0 000C2         MOVW    (VOLUME_NAME), @(SP)+                    ;
           50            56             0C  C5 000C5         MULL3   #12, J, R0                               ; 1456
                               0288 CA46  7F 000C9         PUSHAQ  VOLNAMLIST_DSC+4[J]                      ;
                         9E    01C4 CA40  9E 000CE         MOVAB   VOLNAMLIST_BUF[R0], @(SP)+               ;
                               0288 CA46  7F 000D4         PUSHAQ  VOLNAMLIST_DSC+4[J]                      ; 1457
                         50             9E  D0 000D9         MOVL    @(SP)+, R0                               ;
           60        04  B7             67  28 000DC         MOVC3   (VOLUME_NAME), @4(VOLUME_NAME), (R0)     ;
           02            56        08   AC  F2 000E1 12$:    AOBLSS  LIST_SIZE, J, 13$                        ; 1413
                                        03  11 000E6         BRB     14$                                      ;
                                  FF75  31 000E8 13$:    BRW     6$                                       ;
                         01      0304   CA  D1 000EB 14$:    CMPL    VOLSET_SIZE, #1                          ; 1469
                                        2F  15 000F0         BLEQ    15$                                      ;
                         2B        10   AC  E9 000F2         BLBC    ENTIRE_VOLSET, 15$                       ;
                         58             01  D0 000F6         MOVL    #1, MULTI_DISMOUNT                       ; 1472
                                        01  DD 000F9         PUSHL   #1                                       ; 1473
                                        59  DD 000FB         PUSHL   R9                                       ;
                 0000V   CF             02  FB 000FD         CALLS   #2, GET_VOLUME_NAME                      ;
                         57             50  D0 00102         MOVL    R0, VOLUME_NAME                          ;
                 0314    CA             67  B0 00105         MOVW    (VOLUME_NAME), VOLSETNAM_DSC             ; 1474
                 0318    CA      0308   CA  9E 0010A         MOVAB   VOLSETNAM_BUF, VOLSETNAM_DSC+4           ; 1475
 0318      DA        04  B7             67  28 00111         MOVC3   (VOLUME_NAME), @4(VOLUME_NAME), -        ; 1476
                                                                     @VOLSETNAM_DSC+4                         ;
                         52 0072A083    8F  D0 00118         MOVL    #7512195, PRIMARY_STATUS                 ; 1477
                                        13  11 0011F         BRB     16$                                      ; 1469
                                        58  D4 00121 15$:    CLRL    MULTI_DISMOUNT                           ; 1481
                                        59  DD 00123         PUSHL   R9                                       ; 1482
                 0000V   CF             01  FB 00125         CALLS   #1, GET_VOLUME_NAME                      ;
                         57             50  D0 0012A         MOVL    R0, VOLUME_NAME                          ;
                         52 0072A07B    8F  D0 0012D         MOVL    #7512187, PRIMARY_STATUS                 ; 1483
                                 0130   CA  D4 00134 16$:    CLRL    DISMOUNT_FLAGS                           ; 1489
                         05        10   AC  E8 00138         BLBS    ENTIRE_VOLSET, 17$                       ; 1490
                 0130    CA             02  88 0013C         BISB2   #2, DISMOUNT_FLAGS                       ; 1492
                                 0130   CA  DD 00141 17$:    PUSHL   DISMOUNT_FLAGS                           ; 1493
                                        59  DD 00145         PUSHL   R9                                       ;
                                        02  DD 00147         PUSHL   #2                                       ;
                                        5E  DD 00149         PUSHL   SP                                       ;
                            00000000G   00  9F 0014B         PUSHAB  SYS$DISMOU                               ;
            00000000G    9F             05  FB 00151         CALLS   #5, @#SYS$CMKRNL                         ;
                                   14   AC  DD 00158         PUSHL   ERROR_STATUS                             ; 1499
                                        7E  D4 0015B         CLRL    -(SP)                                    ;
                             00729058   8F  DD 0015D         PUSHL   #7508056                                 ;
                                        57  DD 00163         PUSHL   VOLUME_NAME                              ;
                                        01  DD 00165         PUSHL   #1                                       ;
                                        52  DD 00167         PUSHL   PRIMARY_STATUS                           ;
            00000000G    00             06  FB 00169         CALLS   #6, LIB$SIGNAL                           ;
                         34             58  E9 00170         BLBC    MULTI_DISMOUNT, 20$                      ; 1505
```

```
                52              01  CE 00173           MNEGL   #1, J                              ; 1507
                                2A  11 00176           BRB     19$                                ;
   24     031C  CA              52  E1 00178  18$:     BBC     J, VOLUME_REMOVED, 19$             ; 1508
                        04 BC42  7F 0017E              PUSHAQ  @DEVLIST[J]                        ; 1511
          0000V CF              01  FB 00182           CALLS   #1, GET_VOLUME_NAME                ;
                57              50  D0 00187           MOVL    R0, VOLUME_NAME                    ;
                         0314 CA  9F 0018A             PUSHAB  VOLSETNAM_DSC                      ; 1512
                         0284 CA42 7F 0018E            PUSHAQ  VOLNAMLIST_DSC[J]                  ;
                                02  DD 00193           PUSHL   #2                                 ;
                     00729060  8F  DD 00195            PUSHL   #7508064                           ;
      00000000G 00              04  FB 0019B           CALLS   #4, LIB$SIGNAL                     ;
   D1           52         08   AC  F2 001A2  19$:     AOBLSS  LIST_SIZE, J, 18$                  ; 1508
                                    04 001A7  20$:     RET                                        ; 1515
                              0000 001A8  21$:         .WORD   Save nothing                       ; 1340
                                7E  D4 001AA           CLRL    -(SP)                              ;
                                5E  DD 001AC           PUSHL   SP                                 ;
                7E         04   AC  7D 001AE           MOVQ    4(AP), -(SP)                       ;
          0000V CF              03  FB 001B2           CALLS   #3, INTERCEPT_SIGNAL               ;
                                    04 001B7           RET                                        ;

; Routine Size:  440 bytes,     Routine Base:  $CODE$ + 02FD
```

```
1 ROUTINE GET_VOLUME_NAME (DEVICE_NAME, FLAG) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Given the address of a device name descriptor, return the address of
1 !       a string descriptor for the volume mounted in that device.  The volume
1 !       name is from 1 to 12 characters.  Since the address returned points to
1 !       routine-local storage, the string must be copied by the caller to prevent
1 !       the next invocation of this routine from overwriting the result.
1 !
1 ! CALLING SEQUENCE:
1 !       GET_VOLUME_NAME (ARG1)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1    : address of device name descriptor
1 !       ARG2    : if present, indicates that the volume set name should be returned.
1 !
1 ! IMPLICIT INPUTS:
1 !       <see the EXTERNAL declarations>
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       Address of the volume name descriptor.
1 !       Return 0 if the device does not contain a volume.
1 !
1 ! SIDE EFFECTS:
1 !       NONE.
1 !
1 !--
1
2 BEGIN                                                           ! Start of GET_VOLUME_NAME
2
2
2 !
2 ! Allocate plits in the $CODE$ psect to avoid truncation error when
2 ! linking mountshr.
2 !
2 PSECT
2       PLIT = $CODE$;
2
2 BIND
2       NO_VOLSET_NAME  = DESCRIPTOR ('<name n/a>'),
2       NAME_PREFIX     = DESCRIPTOR ('DISK$');
2
2 BUILTIN
2       ACTUALCOUNT;
2
2 MAP
2       NAME_PREFIX     : BBLOCK;
2
```

```
OWN
        LOGNAM_BUF          : BBLOCK [64],                           ! Buffer for volume logical name
        LOGNAM_DSC          : BBLOCK [DSC$K_S_BLN]                   ! Descriptor for logical name string
                              PRESET ([DSC$A_POINTER] = LOGNAM_BUF),
        VOLNAM_BUF          : BBLOCK [12],                           ! Buffer for volume name string
        VOLNAM_DSC          : BBLOCK [DSC$K_S_BLN]                   ! Descriptor for volume name string
                              PRESET ([DSC$A_POINTER] = VOLNAM_BUF),
        VOLSETNAM_DSC       : BBLOCK [DSC$K_S_BLN],                  ! Decriptor for isolated volume set name

        VOLUME_DVI          : BBLOCK [7*4]                           ! $GETDVI item list for volume info
                              INITIAL (
                                        WORD (12),                   !    ITEM - Volume name
                                        WORD (DVI$_LOGVOLNAM),
                                        LONG (LOGNAM_BUF),
                                        LONG (LOGNAM_DSC),
                                        WORD (12),                   !    ITEM - Volume name
                                        WORD (DVI$_VOLNAM),
                                        LONG (VOLNAM_BUF),
                                        LONG (VOLNAM_DSC),
                                        LONG (0)                     ! List terminator
                                       );


!
! Get the requested information.
! If $GETDVI failied, return 0.
!
IF NOT $GETDVI (DEVNAM=.DEVICE_NAME, ITMLST=VOLUME_DVI)
THEN
    RETURN 0;

!
! If the volume set name was requested, strip off the system-provided
! prefix before returning it to the user.  If no prefix exists, then
! this is not a volume set, and a special message indicating that the
! volume set name is not available is returned instead.
!
IF ACTUALCOUNT() GTR 1
THEN
    IF CH$EQL  (.NAME_PREFIX[DSC$W_LENGTH],
                .NAME_PREFIX[DSC$A_POINTER],
                .NAME_PREFIX[DSC$W_LENGTH],
                .LOGNAM_DSC[DSC$A_POINTER]
               )
    THEN
        BEGIN
        VOLSETNAM_DSC[DSC$W_LENGTH] = .LOGNAM_DSC[DSC$W_LENGTH] - .NAME_PREFIX[DSC$W_LENGTH];
        VOLSETNAM_DSC[DSC$A_POINTER] = .LOGNAM_DSC[DSC$A_POINTER] + .NAME_PREFIX[DSC$W_LENGTH];
        RETURN VOLSETNAM_DSC;
        END
    ELSE
        RETURN NO_VOLSET_NAME;

RETURN VOLNAM_DSC

END;                                                                 ! End of GET_VOLUME_NAME
```

```
3E  61  2F  6E  20  65  6D  61  6E  3C  004B5 P.AAD:  .ASCII  \<name n/a>\                        ;
                                        004BF          .BLKB   1
                              0000000A  004C0 P.AAC:   .LONG   10                                  ;
                              00000000' 004C4          .ADDRESS P.AAD
                  24  4B  53  49  44    004C8 P.AAF:   .ASCII  \DISK$\                             ;
                                        004CD          .BLKB   3
                              00000005  004D0 P.AAE:   .LONG   5                                   ;
                              00000000' 004D4          .ADDRESS P.AAF

                                                       .PSECT  $OWN$,NOEXE,2

                                        003A4 LOGNAM_BUF:
                                                       .BLKB   64
                                   00#  003E4 LOGNAM_DSC:
                                                       .BYTE   0[4]
                              00000000' 003E8          .ADDRESS LOGNAM_BUF                         ;
                                        003EC VOLNAM_BUF:
                                                       .BLKB   12
                                   00#  003F8 VOLNAM_DSC:
                                                       .BYTE   0[4]
                              00000000' 003FC          .ADDRESS VOLNAM_BUF                         ;
                                        00400 VOLSETNAM_DSC:
                                                       .BLKB   8
                                  000C  00408 VOLUME_DVI:
                                                       .WORD   12                                  ;
                                  002C  0040A          .WORD   44                                  ;
                              00000000' 0040C          .ADDRESS LOGNAM_BUF                         ;
                              00000000' 00410          .ADDRESS LOGNAM_DSC                         ;
                                  000C  00414          .WORD   12                                  ;
                                  0022  00416          .WORD   34                                  ;
                              00000000' 00418          .ADDRESS VOLNAM_BUF                         ;
                              00000000' 0041C          .ADDRESS VOLNAM_DSC                         ;
                              00000000  00420          .LONG   0                                   ;

                                              NO_VOLSET_NAME=         P.AAC
                                              NAME_PREFIX=            P.AAE

                                                       .PSECT  $CODE$,NOWRT,2

                                  003C  00000 GET_VOLUME_NAME:
                                                       .WORD   Save R2,R3,R4,R5                    ; 1516
               55      0000'  CF  9E 00002             MOVAB   LOGNAM_DSC+4, R5                    ;
                              7E  7C 00007             CLRQ    -(SP)                               ; 1600
                              7E  7C 00009             CLRQ    -(SP)                               ;
                      20      A5  9F 0000B             PUSHAB  VOLUME_DVI                          ;
                      04      AC  DD 0000E             PUSHL   DEVICE_NAME                         ;
                              7E  7C 00011             CLRQ    -(SP)                               ;
         00000000G    00      08  FB 00013             CALLS   #8, SYS$GETDVI                      ;
                      2B      50  E9 0001A             BLBC    R0, 3$                              ;
                      01      6C  91 0001D             CMPB    (AP), #1                            ; 1610
                      21      1B  00020                BLEQU   2$                                  ;
               54      D3     AF  3C 00022             MOVZWL  NAME_PREFIX, R4                     ; 1612
00  B5      D2  BF            54  29 00026             CMPC3   R4, @NAME_PREFIX+4, @LOGNAM_DSC+4   ;
                              10  12 0002C             BNEQ    1$                                  ;
```

```
        18    A5      FC   A5          54  A3 0002E          SUBW3   R4, LOGNAM_DSC, VOLSETNAM_DSC              ; 1619
        1C    A5           65          54  C1 00034          ADDL3   R4, LOGNAM_DSC+4, VOLSETNAM_DSC+4          ; 1620
                           50     18   A5  9E 00039          MOVAB   VOLSETNAM_DSC, R0                          ; 1621
                                           04 0003D          RET                                                ; 1624
                           50     A7   AF  9E 0003E 1$:      MOVAB   NO_VOLSET_NAME, R0                         ;
                                           04 00042          RET                                                ;
                           50     10   A5  9E 00043 2$:      MOVAB   VOLNAM_DSC, R0                             ; 1626
                                           04 00047          RET                                                ;
                                       50  D4 00048 3$:      CLRL    R0                                         ; 1628
                                           04 0004A          RET                                                ;

; Routine Size:  75 bytes,    Routine Base:  $CODE$ + 04D8
```

```
ROUTINE INTERCEPT_SIGNAL (SIGNAL, MECHANISM) =

!++
! Functional Description:
!
!       This routine is a conditon handler whose sole
!       reason for existence is to force the primary
!       conditon code's facility-code to that of the
!       MOUNT facility.
!
! Input:
!
!       SIGNAL    = Address of the signal array
!       MECHANISM = Address of the mechanism array
!
! Output:
!
!       The condition facility code is equal to MOUN$_FACILITY
!--

BEGIN                                           ! Start of INTERCEPT_SIGNAL

MAP

        SIGNAL          : REF BBLOCK,           ! Signal array
        MECHANISM       : REF BBLOCK;           ! Mechanism array

EXTERNAL

        MOUNT_OPTIONS   : BITVECTOR VOLATILE,   ! parser option flags
        USER_STATUS     : VECTOR;               ! Status return of some routines


IF .SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND
THEN
    BEGIN
    !
    ! Make the facility code MOUN$_FCILITY.
    !
    IF .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] EQL 0
    THEN
        BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] = MOUN$_FACILITY;

    IF .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_MSG_NO] EQL 0
    THEN
        BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_MSG_NO] = .USER_STATUS [0] ^ (-$BITPOSITION (STS$V_MSG_NO));

    !
    ! If the caller requested it, print the message text associated with the message.
    !
    IF .MOUNT_OPTIONS [OPT_MESSAGE]
    THEN
        BEGIN
        SIGNAL [CHF$L_SIG_ARGS] = .SIGNAL [CHF$L_SIG_ARGS] - 2;
        $PUTMSG (MSGVEC = SIGNAL [CHF$L_SIG_ARGS], ACTRTN=0, FACNAM=0);
        SIGNAL [CHF$L_SIG_ARGS] = .SIGNAL [CHF$L_SIG_ARGS] + 2;
        BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_INHIB_MSG] = 1;
```

```
; 1163      1686 3         END;
; 1164      1687 3
; 1165      1688 3      !
; 1166      1689 3      ! If the condition severity code is SEVERE or ERROR, then unwind the
; 1167      1690 3      ! stack back to the caller of the frame that established this handler.
; 1168      1691 3      ! Return the condition code in R0.
; 1169      1692 3      !
; 1170      1693 3      IF .BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_SEVERE
; 1171      1694 3      OR .BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_ERROR
; 1172      1695 3      THEN
; 1173      1696 4         BEGIN
; 1174      1697 4         MECHANISM [CHF$L_MCH_SAVR0] = .SIGNAL [CHF$L_SIG_NAME];
; 1175      1698 4         $UNWIND ();
; 1176      1699 3         END;
; 1177      1700 2      END;
; 1178      1701 2
; 1179      1702 2   !
; 1180      1703 2   ! Attempt to continue the operation.
; 1181      1704 2   !
; 1182      1705 2   RETURN (SS$_CONTINUE);
; 1183      1706 2
; 1184      1707 1   END;                                                  ! End of INTERCEPT_SIGNAL


                                                        .EXTRN  USER_STATUS, SYS$PUTMSG
                                                        .EXTRN  SYS$UNWIND

                                    000C 00000 INTERCEPT_SIGNAL:
                                                        .WORD   Save R2,R3                          ; 1629
                          52     04   AC  D0 00002      MOVL    SIGNAL, R2                          ; 1662
                          53     04   A2  9E 00006      MOVAB   4(R2), R3                           ;
              00000920    8F          63  D1 0000A      CMPL    (R3), #2336                         ;
                                      65  13 00011      BEQL    5$                                  ;
                  0FFF    8F     02   A3  B3 00013      BITW    2(R3), #4095                        ; 1668
                                      0A  12 00019      BNEQ    1$                                  ;
  02   A3         0C      00 00000072 8F  F0 0001B      INSV    #114, #0, #12, 2(R3)                ; 1670
                  FFF8    8F          63  B3 00025 1$:  BITW    (R3), #65528                        ; 1672
                                      0E  12 0002A      BNEQ    2$                                  ;
          50 00000000G    00     FD   8F  78 0002C      ASHL    #-3, USER_STATUS, R0                ; 1674
       63         0D      03          50  F0 00035      INSV    R0, #3, #13, (R3)                   ;
          17 00000000G    00          03  E1 0003A 2$:  BBC     #3, MOUNT_OPTIONS+6, 3$             ; 1679
                          62          02  C2 00042      SUBL2   #2, (R2)                            ; 1682
                                      7E  7C 00045      CLRQ    -(SP)                               ; 1683
                                      7E  D4 00047      CLRL    -(SP)                               ;
                                      52  DD 00049      PUSHL   R2                                  ;
             00000000G    00          04  FB 0004B      CALLS   #4, SYS$PUTMSG                      ;
                          62          02  C0 00052      ADDL2   #2, (R2)                            ; 1684
                  03      A3          10  88 00055      BISB2   #16, 3(R3)                          ; 1685
       04         63      03          00  ED 00059 3$:  CMPZV   #0, #3, (R3), #4                    ; 1693
                                      07  13 0005E      BEQL    4$                                  ;
       02         63      03          00  ED 00060      CMPZV   #0, #3, (R3), #2                    ; 1694
                                      11  12 00065      BNEQ    5$                                  ;
                          50     08   AC  D0 00067 4$:  MOVL    MECHANISM, R0                       ; 1697
                  0C      A0          63  D0 0006B      MOVL    (R3), 12(R0)                        ;
                                      7E  7C 0006F      CLRQ    -(SP)                               ; 1698
             00000000G    00          02  FB 00071      CALLS   #2, SYS$UNWIND                      ;
```

```
                    50              01  D0 00078 5$:     MOVL    #1, R0                                    ; 1705
                                        04 0007B         RET                                               ; 1707

; Routine Size:  124 bytes,    Routine Base:  $CODE$ + 0523
```

```
1 ROUTINE RCP_RESTART ( DEVNAM: REF BBLOCK) =
1
1 !++
1 ! RCP_RESTART
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Issue RESTART command to the RCP for the specified volume. This
1 !       causes the RCP to restart any recovery operations which were frozen
1 !       due to inaccessibility of objects.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       STATUS  = RCP_RESTART ( DEVNAM );
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       DEVNAM  = address of descriptor of device name for which frozen
1 !                 Recovery Unit operations are to be restarted.
1 !
1 ! COMPLETION CODES:
1 !
1 !       SS$_NORMAL      = successful completion
1 !       ...
1 !--
1
2 BEGIN
2
2 EXTERNAL ROUTINE
2       CJF$RECOVERW     : ADDRESSING_MODE (GENERAL);               ! interface to RCP
2 !     GET_VOLUME_NAME  : ADDRESSING_MODE (GENERAL),               ! Return volume name for a given device
2 !     INTERCEPT_SIGNAL: ADDRESSING_MODE (GENERAL);                ! Condition handler
2
2 EXTERNAL LITERAL
2       EXE$C_SYSEFN;                                               ! system event flag
2
2 LOCAL
2       VOLUME_NAME      : REF BBLOCK;                              ! Volume name descriptor
2
2 OWN
2     !
2     ! Allocate space for RODB and RODBA structures.  OWN storage is used
2     ! since stack space is at a premium.
2     !
2       RODB             : BLOCK [RODB$K_LENGTH  + 4, BYTE],        ! RODB plus term. zero
2       RODBA            : BLOCK [RODBA$K_LENGTH + 4, BYTE],        ! RODBA + 0
2       IOSB             : VECTOR [2,LONG],                         ! I/O status block
2       STATUS           : BBLOCK [4];                              ! completion status
2
2
2 !
2 ! Enable a condition handler to field and print signalled messages.
2 !
2 ENABLE INTERCEPT_SIGNAL;
2
2 !
2 ! Fill the fixed portions of the RODB and RODBA.
```

```
2 !
2 RODB [RODB$B_TYPE] = RODB$K_RUJNL;                         ! object is RU journal
2 RODB [RODB$B_COUNT] = 1;                                   ! only one attribute
2 RODB [RODB$W_SIZE] = RODB$K_LENGTH + 4;                    ! actually, this isn't used
2 RODB [RODB$A_POINTER] = RODBA;                             ! point to attribute block
2 RODB + RODB$K_LENGTH = 0;                                  ! termination zero
2 RODBA [RODBA$B_TYPE] = RODBA$K_RUJDEVNAM;                  ! attribute is device name
2 RODBA [RODBA$W_SIZE] = .DEVNAM[DSC$W_LENGTH];              ! length of attribute
2 RODBA [RODBA$A_POINTER] = .DEVNAM[DSC$A_POINTER];          ! addr of attribute
2 RODBA + RODBA$K_LENGTH = 0;                                ! termination zero
2
2 !
2 ! Send it to the RCP.
2 !
2 STATUS = CJF$RECOVERW ( CJF$M_RESTART,                     ! function = RESTART
2                         RODB,                              ! OBJECT data block
2                         0,                                 ! filter list (none)
2                         %REF(PSL$C_EXEC),                  ! access mode
2                         EXE$C_SYSEFN,                      ! event flag
2                         IOSB,                              ! I/O status block
2                         0,                                 ! no AST address
2                         0 );                               ! no AST parameter
2 IF .STATUS THEN STATUS = .IOSB [ 0 ];
2
2 !
2 ! If the status code returned by $CREJNL indicates an "informational" message,
2 ! attempt to inform the user of the event.  All other status values are handled
2 ! outside of this routine.  The status code is signalled, and a condition
2 ! handler prints the message.
2 !
2 IF .STATUS[STS$V_SEVERITY] EQL STS$K_INFO
2 THEN
3     BEGIN
3     VOLUME_NAME = GET_VOLUME_NAME (.DEVNAM);
3     ERR_MESSAGE (MOUN$_VOLSTATUS, 1, .VOLUME_NAME, .STATUS);
2     END;
2
2 .STATUS                                                    ! Return status to caller
2
1 END;


                                        .PSECT  $OWN$,NOEXE,2

                         00424 RODB:    .BLKB   12
                         00430 RODBA:   .BLKB   12
                         0043C IOSB:    .BLKB   8
                         00444 STATUS:  .BLKB   4

                                        .EXTRN  CJF$RECOVERW, EXE$C_SYSEFN

                                        .PSECT  $CODE$,NOWRT,2

                    000C 00000 RCP_RESTART:
                                        .WORD   Save R2,R3                      ; 1708
   53     0000'  CF  9E 00002          MOVAB   STATUS, R3                       ;
```

```
                      5E          04 C2 00007          SUBL2   #4, SP                                   ;
                      6D    0078  CF DE 0000A          MOVAL   3$, (FP)                                 ; 1734
             E0       A3 000C0104 8F D0 0000F          MOVL    #786692, RODB                            ; 1766
             E4       A3      EC  A3 9E 00017          MOVAB   RODBA, RODB+4                            ; 1769
                              E8  A3 D4 0001C          CLRL    RODB+8                                   ; 1770
             EC       A3          06 90 0001F          MOVB    #6, RODBA                                ; 1771
                      52      04  AC D0 00023          MOVL    DEVNAM, R2                               ; 1772
             EE       A3          62 B0 00027          MOVW    (R2), RODBA+2                            ;
             F0       A3      04  A2 D0 0002B          MOVL    4(R2), RODBA+4                           ; 1773
                              F4  A3 D4 00030          CLRL    RODBA+8                                  ; 1774
                                  7E 7C 00033          CLRQ    -(SP)                                    ; 1779
                              F8  A3 9F 00035          PUSHAB  IOSB                                     ;
                       00000000G  8F DD 00038          PUSHL   #EXE$C_SYSEFN                            ;
             10       AE          01 D0 0003E          MOVL    #1, 16(SP)                               ; 1782
                              10  AE 9F 00042          PUSHAB  16(SP)                                   ;
                                  7E D4 00045          CLRL    -(SP)                                    ; 1779
                              E0  A3 9F 00047          PUSHAB  RODB                                     ;
                       02000000   8F DD 0004A          PUSHL   #33554432                                ;
      00000000G       00          08 FB 00050          CALLS   #8, CJF$RECOVERW                         ;
                      63          50 D0 00057          MOVL    R0, STATUS                               ;
                      04          63 E9 0005A          BLBC    STATUS, 1$                               ; 1787
                      63      F8  A3 D0 0005D          MOVL    IOSB, STATUS                             ;
  03         63       03          00 ED 00061 1$:      CMPZV   #0, #3, STATUS, #3                       ; 1795
                                  1A 12 00066          BNEQ    2$                                       ;
                                  52 DD 00068          PUSHL   R2                                       ; 1798
           FECA       CF          01 FB 0006A          CALLS   #1, GET_VOLUME_NAME                      ;
                                  63 DD 0006F          PUSHL   STATUS                                   ; 1799
                                  50 DD 00071          PUSHL   VOLUME_NAME                              ;
                                  01 DD 00073          PUSHL   #1                                       ;
                       0072A07B   8F DD 00075          PUSHL   #7512187                                 ;
      00000000G       00          04 FB 0007B          CALLS   #4, LIB$SIGNAL                           ;
                      50          63 D0 00082 2$:      MOVL    STATUS, R0                               ; 1804
                                     04 00085          RET                                              ;
                                   0000 00086 3$:      .WORD   Save nothing                             ; 1734
                                  7E D4 00088          CLRL    -(SP)                                    ;
                                  5E DD 0008A          PUSHL   SP                                       ;
                      7E      04  AC 7D 0008C          MOVQ    4(AP), -(SP)                             ;
           FEEF       CF          03 FB 00090          CALLS   #3, INTERCEPT_SIGNAL                     ;
                                     04 00095          RET                                              ;

; Routine Size:  150 bytes,    Routine Base:  $CODE$ + 059F


; 1283          1805  1
; 1284          1806  1 END
; 1285          1807  0 ELUDOM



                                                       .EXTRN  LIB$SIGNAL

;                             PSECT SUMMARY
;
;     Name                          Bytes                       Attributes
;
;  $OWN$                              1096  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;  $CODE$                    1589  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     105        0        1000         00:02.0



;                                     COMMAND QUALIFIERS

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

; Size:          1526 code + 1159 data bytes
; Run Time:         00:39.0
; Elapsed Time:     01:17.8
; Lines/CPU Min:      2782
; Lexemes/CPU-Min: 25956
; Memory Used:  212 pages
; Compilation Complete
```

0246 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RDHOME
LIS

MWTUR2
LIS

RUJMAN
LIS

MWTUR1
LIS

SRCVOL
LIS

STACP
LIS

MRDBLK
LIS

REBUILD
LIS