L 5

```
   SSSSSSSSSSSS   YYY         YYY      SSSSSSSSSSSS   LLL                  OOOOOOOOO         AAAAAAAAA
   SSSSSSSSSSSS   YYY         YYY      SSSSSSSSSSSS   LLL                  OOOOOOOOO         AAAAAAAAA
   SSSSSSSSSSSS   YYY         YYY      SSSSSSSSSSSS   LLL                  OOOOOOOOO         AAAAAAAAA
SSS               YYY         YYY   SSS               LLL               OOO         OOO   AAA         AAA
SSS               YYY         YYY   SSS               LLL               OOO         OOO   AAA         AAA
SSS               YYY         YYY   SSS               LLL               OOO         OOO   AAA         AAA
SSS                  YYY   YYY      SSS               LLL               OOO         OOO   AAA         AAA
SSS                  YYY   YYY      SSS               LLL               OOO         OOO   AAA         AAA
SSS                  YYY   YYY      SSS               LLL               OOO         OOO   AAA         AAA
   SSSSSSSSS            YYY            SSSSSSSSS      LLL               OOO         OOO   AAA         AAA
   SSSSSSSSS            YYY            SSSSSSSSS      LLL               OOO         OOO   AAA         AAA
   SSSSSSSSS            YYY            SSSSSSSSS      LLL               OOO         OOO   AAA         AAA
            SSS         YYY                     SSS   LLL               OOO         OOO   AAAAAAAAAAAAAAA
            SSS         YYY                     SSS   LLL               OOO         OOO   AAAAAAAAAAAAAAA
            SSS         YYY                     SSS   LLL               OOO         OOO   AAAAAAAAAAAAAAA
            SSS         YYY                     SSS   LLL               OOO         OOO   AAA         AAA
            SSS         YYY                     SSS   LLL               OOO         OOO   AAA         AAA
            SSS         YYY                     SSS   LLL               OOO         OOO   AAA         AAA
SSSSSSSSSSSS            YYY         SSSSSSSSSSSS      LLLLLLLLLLLLLLL      OOOOOOOOO      AAA         AAA
SSSSSSSSSSSS            YYY         SSSSSSSSSSSS      LLLLLLLLLLLLLLL      OOOOOOOOO      AAA         AAA
SSSSSSSSSSSS            YYY         SSSSSSSSSSSS      LLLLLLLLLLLLLLL      OOOOOOOOO      AAA         AAA
```

FILEID**MOUNTVER E 2

```
MM      MM    OOOOOO    UU      UU  NN      NN  TTTTTTTTTT  VV      VV  EEEEEEEEEE  RRRRRRRR
MM      MM    OOOOOO    UU      UU  NN      NN  TTTTTTTTTT  VV      VV  EEEEEEEEEE  RRRRRRRR
MMMM  MMMM  OO      OO  UU      UU  NN      NN      TT      VV      VV  EE          RR      RR
MMMM  MMMM  OO      OO  UU      UU  NN      NN      TT      VV      VV  EE          RR      RR
MM  MM  MM  OO      OO  UU      UU  NNNN    NN      TT      VV      VV  EE          RR      RR
MM  MM  MM  OO      OO  UU      UU  NNNN    NN      TT      VV      VV  EE          RR      RR
MM      MM  OO      OO  UU      UU  NN  NN  NN      TT      VV      VV  EEEEEEEE    RRRRRRRR
MM      MM  OO      OO  UU      UU  NN  NN  NN      TT      VV      VV  EEEEEEEE    RRRRRRRR
MM      MM  OO      OO  UU      UU  NN    NNNN      TT      VV      VV  EE          RR  RR
MM      MM  OO      OO  UU      UU  NN    NNNN      TT      VV      VV  EE          RR  RR
MM      MM  OO      OO  UU      UU  NN      NN      TT        VV  VV    EE          RR    RR      ....
MM      MM  OO      OO  UU      UU  NN      NN      TT        VV  VV    EE          RR    RR      ....
MM      MM    OOOOOO    UUUUUUUUUU  NN      NN      TT          VV      EEEEEEEEEE  RR      RR    ....
MM      MM    OOOOOO    UUUUUUUUUU  NN      NN      TT          VV      EEEEEEEEEE  RR      RR    ....

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

Table of contents

```
0000        .TITLE  MOUNTVER - Mount Verification routines
0000        .IDENT  'V04-002'
```



```
0000
0000  ;++
0000  ; Facility:
0000  ;
0000  ;       Executive I/O system.
0000  ;
0000  ; Abstract:
0000  ;
0000  ;       This module contains most of the code necessary to implement
0000  ;       mount verification.  Support routines not in this module are
0000  ;       contained in other modules in the EXEC, notably IOSUBNPAG.
0000  ;
0000  ; Environment:
0000  ;
0000  ;       This code executes in KERNEL mode, at device IPL
0000  ;       or higher, and the code must therefore be resident.
0000  ;
0000  ; Author:
0000  ;
0000  ;       Steven T. Jeffreys
0000  ;
0000  ; Creation date:
0000  ;
0000  ;       June 10, 1981
0000  ;
0000  ; Update history:
0000  ;
0000  ;       V04-002 ROW0415         Ralph O. Weber          10-SEP-1984
0000  ;               Fix no-quorum branch destination after PAUSE to go to the
0000  ;               TIME_DELAY subroutine call.
0000  ;
0000  ;       V04-001 ROW0414         Ralph O. Weber          6-SEP-1984
0000  ;               Rework handling of MVTIMEOUT checking after PAUSE so that
```

```
0000  ;             /FOREIGN volumes -- which get into mount verification during
0000  ;             cluster state transitions -- will time out mount verification.
0000  ;
0000  ;     V03-022 ROW0410         Ralph O. Weber                  6-AUG-1984
0000  ;             Setup use of UCB$V_CLUTRAN to guarantee that mount
0000  ;             verification always runs for all disks after a VAXcluster
0000  ;             state transition.
0000  ;
0000  ;     V03-021 ROW0407         Ralph O. Weber                  25-JUL-1984
0000  ;             Add $GETDVI escape vector place holder, EXE$MNTVER_DVI_ASSIST.
0000  ;             Also move EXE$MNTVERSP2 back to RSB; EXE$CLUTRANIO is
0000  ;             correctly used by the connection manager now.
0000  ;
0000  ;     V03-020 ROW0403         Ralph O. Weber                  22-JUL-1984
0000  ;             Change volume name checking so that if either the volume label
0000  ;             or the volume lock name is correct the volume is considered to
0000  ;             be correct.  This volume name checking technique is designed
0000  ;             to handle the cases where the volume is write locked or
0000  ;             becomes write locked such that the volume lock name cannot be
0000  ;             written into the SCB.
0000  ;
0000  ;     V03-019 ROW0385         Ralph O. Weber                  7-JUL-1984
0000  ;             - Add GET_VCB, a generalized routine which gets the VCB
0000  ;               address when possible and signals its legal absence.
0000  ;             - Add EXE$CLUTRANIO symbol which will eventually replace
0000  ;               EXE$MNTVERSP2.
0000  ;             - Add symbol for EXE$UPDGNERNUM, the update a shadow set
0000  ;               generation number routine.
0000  ;             - Change UCB$V_SUPMVMSG processing to clear that bit if a
0000  ;               message which cannot be suppressed is ever broadcast.
0000  ;             - Change GET_BUFFER and FREE_BUFFER to save the address of the
0000  ;               IRP currently using the mount verification work buffer.
0000  ;             - Change volume validation algorithm to check volume lock name
0000  ;               stored in the SCB instead of the volume name stored in the
0000  ;               home block.  This prevents incorrect failures in a
0000  ;               VAXcluster where SET VOLUME /LABEL= has be used.
0000  ;
0000  ;     V03-018 ROW0372         Ralph O. Weber                  30-MAY-1984
0000  ;             Correct setting of UCB$V_MNTVERIP and UCB$V_MNTVERPND bits in
0000  ;             EXE$MNTVERSP2 to have them set in UCB$L_STS not UCB$L_DEVCHAR2.
0000  ;
0000  ;     V03-017 ROW0366         Ralph O. Weber                  18-MAY-1984
0000  ;             Cause QUORUM_LOOP to REMQUE the quorum disk IRP before passing
0000  ;             it off to the driver for processing.  This is useful because
0000  ;             the driver will eventually insert the IRP on some queue and
0000  ;             failing to remove it first causes queue tangles.
0000  ;
0000  ;     V03-016 WMC0004         Wayne Cardoza                   15-May-1984
0000  ;             Class driver wants QUORLOST bit off after call.
0000  ;
0000  ;     V03-015 WMC0003         Wayne Cardoza                   11-May-1984
0000  ;             More VCB fixes.
0000  ;
0000  ;     V03-014 WMC0002         Wayne Cardoza                   10-May-1984
0000  ;             Fix problem for mount verification of disk without VCB.
0000  ;
0000  ;     V03-013 WMC0001         Wayne Cardoza                   02-May-1984
```

```
0000 ;                  Add support for loss of quorum stalling of I/O.
0000 ;
0000 ;          V03-012 ROW0343         Ralph O. Weber          10-APR-1984
0000 ;                  Setup usage of UCB$V_SUPMVMSG to actually supress "normal"
0000 ;                  mount verification messages.
0000 ;
0000 ;          V03-011 ROW0330         Ralph O. Weber          24-MAR-1984
0000 ;                  Add EXE$MNTVERSP1 and EXE$MNTVERSP2, two spare mount
0000 ;                  verification entry points.  Also move EXE$MNTVERSHDOL to a
0000 ;                  place where its future implementation is less likely to break
0000 ;                  branch displacements.
0000 ;
0000 ;          V03-010 ROW0326         Ralph O. Weber          20-MAR-1984
0000 ;                  Add testing of SCB$Q_MOUNTTIME against VCB$Q_MOUNTTIME to
0000 ;                  enhance wrong volume detection.  Eliminate mount verification
0000 ;                  timeout for system disk.
0000 ;
0000 ;          V03-009 ROW0321         Ralph O. Weber          4-MAR-1984
0000 ;                  Fix incorrect usage of self-relative queues while waiting for
0000 ;                  mount verification work page.  Change to using absolute queues.
0000 ;
0000 ;          V03-008 ROW0314         Ralph O. Weber          28-FEB-1984
0000 ;                  Setup definition and use of MVMSL structure for the list of
0000 ;                  messages and other information relivant for using the
0000 ;                  SEND_MESSAGE routine.
0000 ;
0000 ;          V03-007 ROW0311         Ralph O. Weber          26-FEB-1984
0000 ;                  > Change handling of work page based upon use of new mount
0000 ;                    verification private work page instead of blackhole page.
0000 ;                  > Setup no mount verification processing of any internal IRP.
0000 ;                  > Fix bug in failure to allocate a mount verification IRP
0000 ;                    error path.
0000 ;                  > Eliminate unneeded code in INIT_IRP and SEND_MESSAGE.
0000 ;
0000 ;          V03-006 ROW0308         Ralph O. Weber          21-FEB-1984
0000 ;                  Fix message handling to be position independent.
0000 ;
0000 ;          V03-005 ROW0292         Ralph O. Weber          4-FEB-1984
0000 ;                  o Make changes necessary to move this module to SYSLOAxxx.
0000 ;                    Make all address references position independent.
0000 ;                  o Restore TIME_DELAY routine, but have it FORK_WAIT using the
0000 ;                    CDRP portion of the internal IRP.  Fix GET_BUFFER and
0000 ;                    FREE_BUFFER to fork on the CDRP as well.  This makes mount
0000 ;                    verfication compatible with the class drivers which fork on
0000 ;                    the UCB at random times.
0000 ;                  o Add replace START_IO label with EXE$MNTVERSIO thus making
0000 ;                    that routine accessible to in-driver mount verification
0000 ;                    routines.
0000 ;                  o Define EXE$MNTVERSHDOL as a NOP routine, for now.
0000 ;
0000 ;          V03-004 TCM0001         Trudy C. Matthews       09-Jun-1983
0000 ;                  Set up register R4 before calling IOC$CVT_DEVNAM to get
0000 ;                  node$ddcu name if a remote device and ddcu if a local device.
0000 ;
0000 ;          V03-003 ROW0177         Ralph O. Weber          5-APR-1983
0000 ;                  Remove routine TIME_DELAY and convert all its callers to use
0000 ;                  the fork-and-wait executive service, FORK_WAIT.
```

```
0000  172 ;
0000  173 ;      V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000  174 ;              Added $PRDEF.
0000  175 ;
0000  176 ;--
0000  177
```

```
0000          .SBTTL  Declarations
0000    ;
0000    ; Macro definitions
0000    ;
0000          $CADEF                                    ; Define conditional assembly symbols
0000          $CDDBDEF                                  ; Class driver data block
0000          $CDRPDEF                                  ; Define Class Driver Req. Pkt. fields
0000          $CLUBDEF                                  ; Cluster block
0000          $CLUDCBDEF                                ; Quorum disk control block
0000          $DCDEF                                    ; Device classes
0000          $DDBDEF                                   ; Define Device Data Block fields
0000          $DDTDEF                                   ; Define Driver Dispatch Table fields
0000          $DEVDEF                                   ; Define device characteristics bits
0000          $DYNDEF                                   ; Define block type symbols
0000          $FKBDEF                                   ; Define Fork Block fields
0000          $HM1DEF                                   ; Define ODS-1 home block fields
0000          $HM2DEF                                   ; Define ODS-2 home block fields
0000          $IODEF                                    ; Define I/O function codes
0000          $IRPDEF                                   ; Define I/O Request Packet fields
0000          $MSGDEF                                   ; Define system message type codes
0000          $MVMSLDEF                                 ; Define MV message list symbols
0000          $PRDEF                                    ; Define processor registers
0000          $PTEDEF                                   ; Define Page Table Entry fields
0000          $SCBDEF                                   ; Define Storage Control Block offsets
0000          $SSDEF                                    ; Define system status codes
0000          $UCBDEF                                   ; Define Unit Control Block fields
0000          $VCBDEF                                   ; Define Volume Control Block fields
0000
0000    ;+
0000    ; BUILD_TABLE
0000    ;
0000    ;       This macro builds entries in a table of mount verification messages.
0000    ;       The table is used to coordinate messages broadcast to OPA0 with
0000    ;       messages sent to OPCOM.  It existance allows both kinds of messages to
0000    ;       be referenced by a single value, the table index value.
0000    ;
0000    ;       This table is also used by mount verification routines in various
0000    ;       drivers.  The driver-specific mount verification routines are passed
0000    ;       the base address of the table.  From this they can determine the
0000    ;       address of the SEND_MESSAGE routine, and by scanning the table, they
0000    ;       can determine the table index to supply to SEND_MESSAGE.
0000    ;
0000    ; Parameters:
0000    ;
0000    ;       index   symbol to be assigned the index value for a given message
0000    ;       opcom   OPCOM message number for a given message
0000    ;       text    OPA0 broadcast text for a given message
0000    ;       flags   bits giving special properties of a given message
0000    ;               the allowable flags are:
0000    ;                  MVMSL$M_NOSUFFIX       OPA0 broadcast should not include
0000    ;                                           "Mount verification in progress."
0000    ;                  MVMSL$M_SUPRESS        message sending can be supressed by
0000    ;                                            UCB$V_SUPMVMSG
0000    ;
0000    ;--
0000
0000          .MACRO  BUILD_TABLE index, opcom, text, flags=0, ?l1
0000          .IIF    NDF TABLE_INDEX, TABLE_INDEX=0
```

```
                                    0000          .ENABLE LSB
                                    0000          .SAVE
                                    0000          .PSECT  WMOUNTVERMSG
                                    0000  l1:     .ASCIC  text
                                    0000          .RESTORE
                                    0000          .WORD   opcom
                                    0000          .WORD   flags
                                    0000          .ADDRESS l1-EXE$AB_MVMSLBAS
                                    0000          .DISABLE LSB
                                    0000          index = TABLE_INDEX
                                    0000          TABLE_INDEX = TABLE_INDEX + 1
                                    0000          .ENDM   BUILD_TABLE
                                    0000  ;
                                    0000  ; Own storage (read only)
                                    0000  ;
                                    0000
                                00000000          .PSECT  WIONONPAGED LONG
                                    0000
                                    0000  ; The following is the MVMSL passed to driver specific mount verification
                                    0000  ; support routines.
                                    0000
                                    0000          ASSUME  MVMSL$L_SNDMSGOFF EQ -4
                                    0000          ASSUME  MVMSL$W_MSG_CODE  EQ  0
                                    0000          ASSUME  MVMSL$W_FLAGS     EQ  2
                                    0000          ASSUME  MVMSL$L_TEXTOFF   EQ  4
                                    0000          ASSUME  MVMSL$K_LENGTH    EQ  8
                                    0000
                       00000402'    0000          .ADDRESS SEND_MESSAGE - EXE$AB_MVMSLBAS ; Offset to SEND_MESSAGE
                                    0004  EXE$AB_MVMSLBAS::
                                    0004          BUILD_TABLE     index=OFFLINE, opcom=MSG$_DEVOFFLINX, -
                                    0004                          flags=MVMSL$M_SUPRESS, -
                                    0004                          text=<" is offline.">
                                    000C
                                    000C          BUILD_TABLE     index=WRONGVOL, opcom=MSG$_WRONGVOL, -
                                    000C                          text=<" contains the wrong volume.">
                                    0014
                                    0014          BUILD_TABLE     index=WRITELOCK, opcom=MSG$_DEVWRTLCK, -
                                    0014                          text=<" has been write-locked.">
                                    001C
                                    001C          BUILD_TABLE     index=MVCOMPLETE, opcom=MSG$_MVCOMPLETE, -
                                    001C                          flags=<MVMSL$M_NOSUFFIX ! MVMSL$M_SUPRESS>, -
                                    001C                          text=<" has completed mount verification.">
                                    0024
                                    0024          BUILD_TABLE     index=MVABORTED, opcom=MSG$_MVABORTED, -
                                    0024                          flags=MVMSL$M_NOSUFFIX, -
                                    0024                          text=<" has aborted mount verification.">
                                    002C  ;
                                    002C  ; The following message text is part of every message, and does not
                                    002C  ; require an entry in the message table.
                                    002C  ;
                                    002C
                                00000085          .PSECT  WMOUNTVERMSG
49 2D 4D 45 54 53 59 53 25 07 07 00'  0085  PREFIX: .ASCIC  "%SYSTEM-I-MOUNTVER, "
   20 2C 52 45 56 54 4E 55 4F 4D 2D   0091
                                16    0085
72 65 76 20 74 6E 75 6F 4D 20 20 00'  009C  SUFFIX: .ASCIC  "  Mount verification in progress."
6E 69 20 6E 6F 69 74 61 63 69 66 69   00A8
```

```
   2E 73 73 65 72 67 6F 72 70 20  00B4
                              21  009C
                                  00BE
                                  00BE      ;
                                  00BE      ; The following literal is used as carriage control
                                  00BE      ; in the message sent to the operator's console.
                                  00BE      ;
                        00000A0D  00BE      CAR_CTRL        = ^X0A0D                ; <cr><lf> characters
                        00000002  00BE      CAR_CTRL_SIZE   = 2                     ; Size in bytes
```

```
00BE          .SBTTL  EXE$MOUNTVER - initial entry point
00BE    ;++
00BE    ; EXE$MOUNTVER
00BE    ;
00BE    ; Functional description:
00BE    ;
00BE    ;       Mount verification is the mechanism whereby Files-11 volumes are
00BE    ;       are brought back online after a catastrophic (but hopefully transient)
00BE    ;       hardware problem has rendered the volume unusable.
00BE    ;
00BE    ;       This is the main routine, and initial entry point of the
00BE    ;       mount verification code.  For a detailed discussion of
00BE    ;       the implementation, please see the section entitled "Design Notes".
00BE    ;
00BE    ; Input:
00BE    ;
00BE    ;       R0,R1     = I/O status
00BE    ;       R2        = scratch
00BE    ;       R3        = IRP address
00BE    ;       R4        = scratch
00BE    ;       R5        = UCB address
00BE    ;       0(SP)     = return address of caller
00BE    ;       4(SP)     = return address of caller's caller
00BE    ;
00BE    ; Output:
00BE    ;
00BE    ;       None.
00BE    ;
00BE    ; Side Effects:
00BE    ;
00BE    ;       Providing that whatever external event has caused a given
00BE    ;       volume to undergo mount verification has been corrected,
00BE    ;       normal I/O activity will resume on the device.  Otherwise,
00BE    ;       the volume will do no useful work until mount verification
00BE    ;       is complete, and will appear to be hung.
00BE    ;
00BE    ; Design Notes:
00BE    ;
00BE    ;       Mount verification (MV for short) is only done for FILES-11
00BE    ;       disk volumes.  Tapes are already covered by the magnetic
00BE    ;       tape ACP, and handling foreign disk volumes would open a
00BE    ;       security hole that would be difficult to close.
00BE    ;
00BE    ;       There are two error conditions that can cause a volume to
00BE    ;       undergo mount verification.  The first arises when a volume
00BE    ;       is somehow hardware writelocked sometime after it was mount
00BE    ;       write-enabled.  (The writelock recovery mechanism is covered
00BE    ;       in detail in the funtional description of the WRITLCK_HNDLR
00BE    ;       routine, and will not be covered here.)
00BE    ;
00BE    ;       The second, and more serious, situation is the result of the
00BE    ;       volume being declared software invalid.  The volume-valid
00BE    ;       bit in the volume's UCB is cleared by the driver when it
00BE    ;       detects a situation that warrants such action, such as the
00BE    ;       device issuing an ONLINE interrupt.  This implies that the
00BE    ;       volume has been spun down, and then spun back up, so the
00BE    ;       contents of the drive may not be the same.  It is MV's task
```

```
00BE    ;       to notify the operator of the situation, and to verify that
00BE    ;       the volume (if any) now in the drive is the same one that
00BE    ;       was there originally.
00BE    ;
00BE    ;       What follows is a series of one-liners and short paragraphs
00BE    ;       that are meant to explain various design decisions, and
00BE    ;       to explain obscure parts of the code.  There is no particular
00BE    ;       order to the notes.
00BE    ;
00BE    ;       The cell EXE$GL_SVAPTE maps the MV work page into S0 space,
00BE    ;       and is set up by INIT at system boot time.  MV used to use
00BE    ;       the blackhole page for its I/O operations.  However, the
00BE    ;       read-modify-write operations performed by shadow set state
00BE    ;       change processing prohibit this.  Permanent provision of a
00BE    ;       512 byte buffer for MV is preferable to dynamic allocation
00BE    ;       of the space because a pileup of MV requests might make
00BE    ;       allocation of the needed space impossible.
00BE    ;
00BE    ;       Broadcasting a message directly to the operator's console
00BE    ;       is done to ensure that someone is notified of MV being in
00BE    ;       progess.  OPCOM cannot be relied on since it may not be
00BE    ;       present, or the very nature of the problem may prevent it
00BE    ;       from operating (eg.  OPCOM is swapped out, and the system
00BE    ;       disk is undergoing MV).
00BE    ;
00BE    ;       The code is optimized for size over speed, since the code
00BE    ;       is infrequently executed, but must be resident.
00BE    ;
00BE    ;       Once MV starts, normal I/O activity on the device ceases until
00BE    ;       MV completes or the volume is dismounted.  Unfortunately, there
00BE    ;       is a deadlock problem with the ACP, because to dismount a volume
00BE    ;       an ACP I/O request must be done to the volume, and that can't
00BE    ;       happen since we're in MV, so the ACP hangs waiting for the I/O
00BE    ;       to complete.  Worse, requests will pile up at the ACP, and
00BE    ;       other processes will be hung by the hung ACP.  The situation
00BE    ;       clears itself up nicely when MV completes.
00BE    ;
00BE    ;       MV consists of the code in this module, support routines in
00BE    ;       other modules, and a driver-dependent piece of code that
00BE    ;       is pointed to by the driver's DDT.  The covention is that
00BE    ;       if R3 is nonzero, the routine should assume that R3 points
00BE    ;       to an IRP and requeue the IRP.  If R3 is zero, dequeue the
00BE    ;       first IRP and resume normal I/O activity.  The cell in the
00BE    ;       DDT is DDT$L_MNTVER, and it defaults to IOC$MNTVER.
00BE    ;
00BE    ;       MV is entered by the common I/O completion routine, IOC$REQCOM.
00BE    ;       This may be different for DISK_CLASS drivers.
00BE    ;--
```

```
                  0000002C        .PSECT  WIONONPAGED
                      002C        .ENABL LSB
                      002C
                      002C EXE$MOUNTVER::                        ; Initial entry point
                      002C ;
                      002C ; See if an IRP was supplied.  If not, skip all the checks and I/O cleanup.
                      002C ;
            53    D5  002C        TSTL    R3
            58    13  002E        BEQL    35$
                      0030 ;
                      0030 ; Determine if mount verification is possible and necessary.
                      0030 ;
                      0030        ;
                      0030        ; Ignore all internal IRPs (including MV IRPs).
                      0030        ;
         0C A3    D5  0030        TSTL    IRP$L_PID(R3)          ; Is this an internal IRP?
            47    19  0033        BLSS    10$                    ; Branch to exit if internal IRP.
                      0035        ;
                      0035        ; The device must be file oriented, and contain a mounted
                      0035        ; volume that is not mounted /FOREIGN.  In addition, the volume
                      0035        ; must have a VCB with the
                      0035        ; The one time this is not true is if this is a cluster that is out
                      0035        ; out of quorum.  In this case, we must go into mount verification in
                      0035        ; order to stall all I/O
                      0035        ;
            0E    E1  0035        BBC     #DEV$V_FOD,-           ; Branch if device not file oriented
      42 38 A5        0037                UCB$L_DEVCHAR(R5),10$  ;
            05    E0  003A        BBS     #DEV$V_SQD,-           ; Is this a sequential device?
      3D 38 A5        003C                UCB$L_DEVCHAR(R5),10$  ;
          0527    30  003F        BSBW    QUORUM                 ; Is it out-of-quorum cluster
         38 54    E8  0042        BLBS    R4,20$                 ; Yes
33 64 A5     15   E4  0045        BBSC    #UCB$V_CLUTRAN, -      ; Branch if here due to a VAXcluster
                      004A                UCB$L_STS(R5), 20$     ; state transition (and clear flag).
            13    E1  004A        BBC     #DEV$V_MNT,-           ; Branch if device not mounted
      2D 38 A5        004C                UCB$L_DEVCHAR(R5),10$  ;
            18    E0  004F        BBS     #DEV$V_FOR,-           ; Is this a foreign device?
      28 38 A5        0051                UCB$L_DEVCHAR(R5),10$  ;
                      0054        ;
                      0054        ; Check the VCB$V_MOUNTVER bit to ensure the volume
                      0054        ; is a candidate for mount verification.
                      0054        ;
   54     34 A5   D0  0054        MOVL    UCB$L_VCB(R5),R4       ; Get the VCB address
            22    13  0058        BEQL    10$                    ; Exit if none
            02    E1  005A        BBC     #VCB$V_MOUNTVER,-      ; Exit if bit not set
      1D 53 A4        005C                VCB$B_STATUS2(R4),10$  ;
                      005F        ;
                      005F        ; If the medium is offline, or the volume is
                      005F        ; invalid, the error can be recovered from.
                      005F        ;
   50   01A4 8F   B1  005F        CMPW    #SS$_MEDOFL,R0         ; Is the media (disk volume) offline?
            17    13  0064        BEQL    20$                    ; Branch if true
   50   0254 8F   B1  0066        CMPW    #SS$_VOLINV,R0         ; Is the volume invalid?
            10    13  006B        BEQL    20$                    ; Branch if true
                      006D        ;
                      006D        ; If the volume has been writelocked, make sure that it was
                      006D        ; an accidental writelock.  If the software writelock bit is
                      006D        ; on, then the volume was mounted with the volume write protected.
```

```
                              006D       ; If the bit is not set, then the volume has been mounted for
                              006D       ; read/write access, and has since been (accidentally) write protected.
                              006D       ;
          50   025C 8F    B1  006D       CMPW    #SS$_WRITLCK,R0         ; Is the device writelocked?
                    08    12  0072       BNEQ    10$                     ; Branch if not
                    19    E0  0074       BBS     #DEV$V_SWL,-            ; Branch if software writelocked
               03 38 A5       0076               UCB$L_DEVCHAR(R5),10$   ;
                  0366    31  0079       BRW     WRITLCK_HNDLR           ; Recover from accidental writelock
                          05  007C 10$:  RSB                             ; Mount verifcation is not called for - exit
                              007D ;
                              007D ; The error can be recovered from.
                              007D ;
                    0D    E5  007D 20$:  BBCC    #IRP$V_MVIRP,-          ; Clear the MV bit in the IRP.
               00 2A A3       007F               IRP$W_STS(R3),30$       ;
                  0342    30  0082 30$:  BSBW    CLEANUP_IO              ; Clean up the I/O operation
                              0085       ;
                              0085       ; Perform any driver-specific initialization for mount verification.
                              0085       ; R5 points to the device UCB, and R3 points to the IRP.
                              0085       ;
                  0352    30  0085       BSBW    DRIVER_CODE
                    8E    D5  0088 35$:  TSTL    (SP)+                   ; Discard the return address
                              008A ;
                              008A ; This entry point is used to start mount verification without having an error
                              008A ; IRP
                              008A ;
                              008A MNTVER_NOIRP:
                    0E    E2  008A       BBSS    #UCB$V_MNTVERIP,-       ; Set mount verification in progress
               ED 64 A5       008C               UCB$W_STS(R5),10$       ; (return to caller's caller if already set)
                              008F ;
                              008F ; Inform interested parties that the device needs attention.
                              008F ;
              54    00    D0  008F       MOVL    #OFFLINE,R4             ; Set message code
                  0371    30  0092       BSBW    SEND_MESSAGE            ; Send message
                              0095 ;
                              0095 ; Allocate an IRP.  If non exists, exit.
                              0095 ;
          51   00C4 8F    3C  0095       MOVZWL  #IRP$K_LENGTH,R1        ; Set IRP size
           00000000'GF    16  009A       JSB     G^EXE$ALONONPAGED       ; Allocate an IRP
                 53 52    D0  00A0       MOVL    R2,R3                   ; Copy IRP address
                 05 50    E8  00A3       BLBS    R0,40$                  ; Branch if success
                    53    D4  00A6       CLRL    R3                      ; Signal the no IRP was allocated.
                  0193    31  00A8       BRW     ERROR_EXIT              ; Exit if no IRP available
                              00AB ;
                              00AB ; Calculate the maximum time for which we will
                              00AB ; continue to attempt mount verification.  Store
                              00AB ; the value in the IRP.
                              00AB ;
          50   00000000'GF    3C  00AB 40$:  MOVZWL  G^IOC$GW_MVTIMEOUT,R0   ; Get delta time
  14 A3   50   00000000'GF    C1  00B2       ADDL3   G^EXE$GL_ABSTIM,R0,-    ; Add site-specific delta time to current ti
                              00BB               IRP$L_ASTPRM(R3)        ; Fall through...
                              00BB
                              00BB ;
                              00BB ; Decide whether or not this is a cluster that is out of quorum.
                              00BB ; From this point on the meaning of R4 is:
                              00BB ;       0 -> normal mount verification
                              00BB ;       1 -> out of quorum
                              00BB ;       -1-> out of quorum and pack-ack completed
```

```
                          00BB      ;
          04AB    30      00BB                 BSBW    QUORUM                  ; Get indicator in R4
                          00BE
                          00BE                 .DSABL  LSB
                          00BE
                          00BE      BUILD_PACKACK_IRP:
          0202    30      00BE                 BSBW    INIT_IRP                ; Set request-independent fields in the IRP
                          00C1                 ;
                          00C1                 ; Determine if the device supports a PACKACK function.
                          00C1                 ; If it does, then issue a PACKACK request, else attempt
                          00C1                 ; to read the volume's home block.
                          00C1                 ;
            0B    E2      00C1                 BBSS    #UCB$V_VALID,-          ; Set volume valid
      00 64 A5            00C3                         UCB$W_STS(R5),10$       ;
  51    0088 C5   D0      00C6      10$:       MOVL    UCB$L_DDT(R5),R1        ; Get address of DDT
   51     08 A1   D0      00CB                 MOVL    DDT$L_FDT(R1),R1        ; Get address of FDT masks
    61    08      E1      00CF                 BBC     #IO$_PACKACK,(R1),-     ; Branch if PACKACK not supported
          16              00D2                         30$                     ;
                          00D3                 ;
                          00D3                 ; Set the request dependent fields in the IRP and start the I/O.
                          00D3                 ;
                          00D3      20$:       ASSUME  IO$_PHYSICAL GE IO$_PACKACK
                  B0      00D3                 MOVW    #<IRP$M_PHYSIO  ! -     ; Set physical I/O function
                          00D4                         IRP$M_MVIRP>,-          ; Mark this a mount verification IRP
  2A A3    2100 8F        00D4                         IRP$W_STS(R3)
          0808 8F B0      00D9                 MOVW    #<IO$_PACKACK ! IO$M_INHERLOG>,-
            20 A3         00DD                         IRP$W_FUNC(R3)          ; Set function code, inhibit error logging
          02B5    30      00DF                 BSBW    EXE$MNTVERSIO           ; Start I/O request
                          00E2      ;
                          00E2      ; When the PACKACK I/O is done, control returns here.
                          00E2      ;
          0484    30      00E2                 BSBW    QUORUM                  ; Is it out-of-quorum cluster
          38 A3   E9      00E5                 BLBC    IRP$L_MEDIA(R3),-       ; If failure, try again
          10              00E8                         PAUSE                   ;
    54    54      CE      00E9      30$:       MNEGL   R4,R4                   ; Indicate pack-ack succeeded
          39      12      00EC                 BNEQ    QUORUM_LOOP             ; No quorum
          0491    30      00EE                 BSBW    GET_VCB                 ; Check for legally absent VCB.
          03      13      00F1                 BEQL    39$                     ; Branch if VCB legally absent.
          00AC    31      00F3                 BRW     BUILD_RDHOME_IRP        ; Otherwise, go check the volume.
          0107    31      00F6      39$:       BRW     NORMAL_EXIT             ; Branch assist.
                          00F9
                          00F9      PAUSE:     ;
                          00F9                 ; If out-of-quorum, skip all checks and just wait a while.  Else,
                          00F9                 ; check for MVTIMEOUT expired on this request or VCB$V_MOUNTVER
                          00F9                 ; clear (indicate abort mount verification).  If mount verification
                          00F9                 ; still ok, wait.  Else, abort.
                          00F9                 ;
         20 54    E8      00F9                 BLBS    R4, 15$                 ; If no quorum, skip checks.
          0483    30      00FC                 BSBW    GET_VCB                 ; Get VCB address.
14 A3  00000000'GF D1     00FF                 CMPL    G^EXE$GL_ABSTIM, -      ; Have we run out of time?
                          0107                         IRP$L_ASTPRM(R3)
            0A    1F      0107                 BLSSU   10$                     ; Branch if we still have time.
            50    D5      0109                 TSTL    R0                      ; Out of time.  Is there a VCB?
            17    13      010B                 BEQL    20$                     ; Branch if no VCB to update.
  53 A0     04    8A      010D                 BICB    #<1@VCB$V_MOUNTVER>, -  ; Else, disable mount verification.
                          0111                         VCB$B_STATUS2(R0)
            11    11      0111                 BRB     20$                     ; Then abort.
```

```
                    50  D5  0113  10$:    TSTL    R0                              ; Still have time.  Is there a VCB?
                    05  13  0115          BEQL    15$                             ; Branch if no VCB to check.
           08 53 A0 02  E1  0117          BBC     #VCB$V_MOUNTVER, -              ; Abort MV if MV is now disabled.
                            011C                  VCB$B_STATUS2(R0), 20$
                  018A  30  011C  15$:    BSBW    TIME_DELAY                      ; Pause for a bit
                 05 54  E8  011F          BLBS    R4,QUORUM_LOOP                  ; We are waiting for quorum
                    9A  11  0122          BRB     BUILD_PACKACK_IRP               ; Retry the packack
                  0117  31  0124  20$:    BRW     ERROR_EXIT                      ; Exit
                            0127  ;
                            0127  ; Check for quorum IRP and issue it if found.
                            0127  ; Wait a while longer if still no quorum
                            0127  ;
                            0127  QUORUM_LOOP:
      51   00000000'GF  D0  0127          MOVL    G^CLU$GL_CLUB,R1
         51   00B4 C1   D0  012E          MOVL    CLUB$L_CLUDCB(R1),R1            ; Get quorum disk control block
                    50  13  0133          BEQL    20$                             ; None
           51    10 A1  D0  0135          MOVL    CLUDCB$L_IRP(R1),R1             ; Quorum IRP
                    4A  13  0139          BEQL    20$                             ; None
           50    4C A5  9E  013B          MOVAB   UCB$L_IOQFL(R5),R0              ; IO queue
                 52 50  D0  013F          MOVL    R0,R2
                 52 62  D0  0142  10$:    MOVL    (R2),R2                         ; Next IRP
                 52 50  D1  0145          CMPL    R0,R2
                    3B  13  0148          BEQL    20$                             ; End of queue
                 51 52  D1  014A          CMPL    R2,R1                           ; Is this the quorum IRP
                    F3  12  014D          BNEQ    10$                             ; Not an interesting IRP
                 52 62  0F  014F          REMQUE  (R2),R2                         ; Dequeue the quorum IRP
           14 3C A5 05  E1  0152          BBC     #DEV$V_MSCP,UCB$L_DEVCHAR2(R5),15$ ; Standard disk
                    0D  E2  0157          BBSS    #IRP$V_MVIRP,-                  ; Let the I/O go through
              00 2A A2      0159                  IRP$W_STS(R2),12$               ; pretend it is MV IRP
                    38  BB  015C  12$:    PUSHR   #^M<R3,R4,R5>
                 53 52  D0  015E          MOVL    R2,R3                           ; Quorum IRP
           00000000'GF  16  0161          JSB     G^IOC$INITIATE                  ; Go start the I/O
                    38  BA  0167          POPR    #^M<R3,R4,R5>
                    1A  11  0169          BRB     20$                             ; Continue
                            016B  ;
 64 A5     00080000 8F  C8  016B  15$:    BISL    #UCB$M_MNTVERPND,UCB$L_STS(R5) ; Get control back after I/O
                 50 53  D0  0173          MOVL    R3,R0                           ; Mount verification IRP
           00000000'GF  16  0176          JSB     G^COM$DRVDEALMEM                ; Free it
                 53 52  D0  017C          MOVL    R2,R3                           ; Quorum IRP
           00000000'GF  17  017F          JMP     G^IOC$INITIATE                  ; Go do I/O
                            0185  ;
      52   00000000'GF  D0  0185  20$:    MOVL    G^CLU$GL_CLUB,R2
        0A 1C A2    1C  E0  018C          BBS     #CLUB$V_QUORUM,CLUB$L_FLAGS(R2),30$ ; At last we have quorum
                    54  D5  0191          TSTL    R4
                    03  19  0193          BLSS    25$                             ; Go wait some more
                  FF26  31  0195          BRW     BUILD_PACKACK_IRP               ; Pack-ack not done yet
                  FF5E  31  0198  25$:    BRW     PAUSE
                    54  D4  019B  30$:    CLRL    R4                              ; Indicate we now have quorum
                  03E2  30  019D          BSBW    GET_VCB                         ; Check for legally absent VCB.
                    5E  13  01A0          BEQL    NORMAL_EXIT                     ; Branch if VCB is legally absent.
                            01A2                                                  ; Else, test volume validity.
                            01A2
                            01A2  ;
                            01A2  ; Validate that the correct volume is still in the drive.
                            01A2  ;
                            01A2  ;       The validation is performed in two steps.  First the home block is
                            01A2  ;       read and checked against fields stored in the VCB.  If that test
```

```
                            01A2  ;         succeeds, the storage control block is read and checked against fields
                            01A2  ;         stored in the VCB.  If either test fails, wrong volume mount
                            01A2  ;         verification is entered.
                            01A2  ;
                            01A2  BUILD_RDHOME_IRP:
                            01A2
            00B4      30    01A2          BSBW    GET_BUFFER                ; Obtain ownership of MV work buffer.
                            01A5
                            01A5          ; Read and validate the home block.
                            01A5
            011B      30    01A5          BSBW    INIT_IRP                  ; Initialize IRP.
       50   34 A5     D0    01A8          MOVL    UCB$L_VCB(R5), R0         ; Get VCB address.
              03      12    01AC          BNEQ    10$
            008B      31    01AE          BRW     FREE_BUF_ERRXIT           ; Branch if no VCB address.
       50   24 A0     D0    01B1  10$:    MOVL    VCB$L_HOMELBN(R0), R0     ; Get home block LBN.
            012B      30    01B5          BSBW    INIT_IRP_READ             ; Setup IRP for home block read.
            01DC      30    01B8          BSBW    EXE$MNTVERSIO             ; Do the read.
            03AB      30    01BB          BSBW    QUORUM                    ; Is it out-of-quorum cluster
           04 54      E9    01BE          BLBC    R4,20$                    ; Yes
              50      D4    01C1          CLRL    R0                        ; Error status
              54      11    01C3          BRB     VALIDATE_FAILED           ; WE don't want to succeed without quorum
                            01C5  20$:
        50 38 A3      E9    01C5          BLBC    IRP$L_MEDIA(R3), -        ; Branch if error occured on the read.
                            01C9                  VALIDATE_FAILED
            0137      30    01C9          BSBW    VALIDATE_HOME             ; Validate the home block.
           4A 50      E9    01CC          BLBC    R0, VALIDATE_FAILED       ; Branch if home blk. validation failed.
                            01CF
                            01CF          ; Read and validate the storage control block.
                            01CF
            00F1      30    01CF          BSBW    INIT_IRP                  ; Initialize IRP.
       50   34 A5     D0    01D2          MOVL    UCB$L_VCB(R5), R0         ; Get VCB address.
              64      13    01D6          BEQL    FREE_BUF_ERRXIT           ; Branch if no VCB address.
  50   34 A0   01     C3    01D8          SUBL3   #1, -                     ; Get storage control block LBN.
                            01DD                  VCB$L_SBMAPLBN(R0), R0
            0103      30    01DD          BSBW    INIT_IRP_READ             ; Setup IRP for home block read.
            01B4      30    01E0          BSBW    EXE$MNTVERSIO             ; Do the read.
            0383      30    01E3          BSBW    QUORUM                    ; Is it out-of-quorum cluster
           04 54      E9    01E6          BLBC    R4,30$                    ; Yes
              50      D4    01E9          CLRL    R0                        ; Error status
              2C      11    01EB          BRB     VALIDATE_FAILED           ; WE don't want to succeed without quorum
                            01ED  30$:
        28 38 A3      E9    01ED          BLBC    IRP$L_MEDIA(R3), -        ; Branch if error occured on the read.
                            01F1                  VALIDATE_FAILED
            0148      30    01F1          BSBW    VALIDATE_SCB              ; Validate the storage block.
           22 50      E9    01F4          BLBC    R0, VALIDATE_FAILED       ; Branch if SCB validation failed.
                            01F7
                            01F7          ; Mount verification has succeeded.
                            01F7          ; Release the MV work buffer, inform the world that this volume has
                            01F7          ; returned to the living, and drop through to a normal exit.
                            01F7
            0089      30    01F7          BSBW    FREE_BUFFER               ; Release MV work buffer.
        54    03      D0    01FA          MOVL    #MVCOMPLETE, R4           ; Set success message code.
            0206      30    01FD          BSBW    SEND_MESSAGE              ; Signal the world.
                            0200
                            0200  ;
                            0200  ; This is the common exit path for mount verification.
                            0200  ; Return all resources, clear MNTVERIP, and resume normal
```

```
                            0200  ; I/O activity for the device.
                            0200  ;
                            0200  NORMAL_EXIT:
          50    53    D0    0200          MOVL    R3,R0                     ; Copy IRP address
                06    13    0203          BEQL    10$                       ; Branch if no IRP was allocated.
       00000000'GF    16    0205          JSB     G^COM$DRVDEALMEM          ; Deallocate the IRP
 64 A5  0004C000 8F   CA    020B  10$:    BICL    #<UCB$M_MNTVERIP ! -      ; Clear MNTVERIP and WRONGVOL
                            0213                     UCB$M_WRONGVOL ! -     ; and SUPMVMSG
                            0213                     UCB$M_SUPMVMSG>, -     ;
                            0213                    UCB$L_STS(R5)           ;
                53    D4    0213          CLRL    R3                        ; Clear pointer to IRP
              01C2    30    0215          BSBW    DRIVER_CODE               ; Do driver-specific clean-up
                      05    0218          RSB                               ; Exit
                            0219  ;
                            0219  ; Attempt to recover from an error encountered while reading
                            0219  ; the home block or while validating the volume.
                            0219  ;
                            0219  VALIDATE_FAILED:
                68    10    0219          BSBB    FREE_BUFFER               ; Release MV work buffer.
       50   010C 8F   B1    021B          CMPW    #SS$_INCVOLLABEL,R0       ; Is this the correct volume?
                08    13    0220          BEQL    20$                       ; Branch if not
                0F    E4    0222          BBSC    #UCB$V_WRONGVOL,-         ; Clear WRONGVOL bit
             00 64 A5       0224                  UCB$W_STS(R5),10$         ;
              FECF    31    0227  10$:    BRW     PAUSE                     ; Retry mount verification
                            022A          ;
                            022A          ; The wrong volume is in the drive.  If this is the first time
                            022A          ; for this volume, inform all interested parties of the event.
                            022A          ;
                0F    E2    022A  20$:    BBSS    #UCB$V_WRONGVOL,-         ; Branch if not the first time
             F8 64 A5       022C                  UCB$W_STS(R5),10$         ;
                54    DD    022F          PUSHL   R4                        ; Save quorum indicator
          54    01    D0    0231          MOVL    #WRONGVOL,R4              ; Set message code
              01CF    30    0234          BSBW    SEND_MESSAGE              ; Send message to console
                54  8ED0    0237          POPL    R4
                EB    11    023A          BRB     10$                       ; Try again
                            023C  ;
                            023C  ; This is the error exit path for mount verification.
                            023C  ; Since the operation never completed, clear volume valid
                            023C  ; to prevent the volume from being used.
                            023C  ;
                            023C  FREE_BUF_ERRXIT:
                45    10    023C          BSBB    FREE_BUFFER               ; Free MV work buffer, then error exit.
                            023E  ERROR_EXIT:
          54    04    D0    023E          MOVL    #MVABORTED,R4             ; Set message code
              01C2    30    0241          BSBW    SEND_MESSAGE              ; Send message to console
                0B    E5    0244          BBCC    #UCB$V_VALID,-            ; Clear volume valid and join common code
             64 A5          0246                  UCB$W_STS(R5),-           ;
                B7          0248                  NORMAL_EXIT               ;
                B5    11    0249          BRB     NORMAL_EXIT               ; Branch to common exit code
                            024B
```

```
                          024B          .SBTTL  GET_BUFFER - allocate an I/O buffer
                          024B    ;++
                          024B    ; GET_BUFFER
                          024B    ;
                          024B    ; Functional description:
                          024B    ;
                          024B    ;       This routine will attempt to allocate the MV work page
                          024B    ;       for to the caller.  If the page is not busy, the page is
                          024B    ;       marked busy and control returns to the caller.  If the page
                          024B    ;       is busy, the caller's context is folded up into the CDRP
                          024B    ;       attached to the input internal IRP and the CDRP is put on
                          024B    ;       the blakhole page wait queue.  When the wait queue entry is
                          024B    ;       processed, input context is restored and control is return
                          024B    ;       to the caller, who now owns the MV work page.
                          024B    ;
                          024B    ; Input:
                          024B    ;
                          024B    ;       R3      = Mount verification IRP address (CDRP used as fork block)
                          024B    ;       R5      = Device UCB address
                          024B    ;       (SP)    = Return address for caller
                          024B    ;      4(SP)    = Return address for caller's caller
                          024B    ;
                          024B    ; Output:
                          024B    ;
                          024B    ;       R3..R5  are preserved.
                          024B    ;++
                          024B
                          024B    ; Define cells used to control access to MV work page.
                          024B
                          024B          .ALIGN  LONG
             00000000     024C    EXE$AL_WRKWQFL: .LONG 0          ; Work page wait queue.
             00000000     0250    EXE$AL_WRKWQBL: .LONG 0
             00000000     0254    EXE$AL_BSYIRP: .LONG 0           ; Current buffer owner IRP.
                   00     0258    EXE$AB_MVWORK: .BYTE 0           ; Byte of storage belonging to the
                          0259                                     ; MV work buffer owner.
                          0259
                          0259    GET_BUFFER:
                          0259
        F8 AF      D5     0259          TSTL    B^EXE$AL_BSYIRP           ; Is the work buffer currently owned?
           05      12     025C          BNEQ    10$                       ; Branch if work buffer is owned.
     F2 AF 53      D0     025E          MOVL    R3, B^EXE$AL_BSYIRP       ; Else, setup address of current owner.
                   05     0262          RSB                               ; Return to caller
                          0263          ;
                          0263          ; The page is busy. Put the caller in the wait queue.
                          0263          ;
     55 60 A3      9E     0263    10$:  MOVAB   IRP$L_FQFL(R3), R5        ; Get CDRP for use as fork block.
     10 A5 53      7D     0267          MOVQ    R3, CDRP$L_FR3(R5)        ; Save R3 and R4.
        0C A5    8ED0     026B          POPL    CDRP$L_FPC(R5)            ; Save return address.
        DA AF      D5     026F          TSTL    B^EXE$AL_WRKWQFL          ; Is work page wait queue initialized?
           0A      12     0272          BNEQ    20$                       ; Branch if queue initialized.
  D3 AF   D5 AF    DE     0274          MOVAL   B^EXE$AL_WRKWQFL, -       ; Else, initalize queue header.
                          0279                  B^EXE$AL_WRKWQFL
  D2 AF   D0 AF    DE     0279          MOVAL   B^EXE$AL_WRKWQFL, -
                          027E                  B^EXE$AL_WRKWQBL
     CE BF 65      0E     027E    20$:  INSQUE  (R5), @B^EXE$AL_WRKWQBL   ; Put CDRP on the work page wait queue.
                   05     0282          RSB                               ; Return to caller's caller.
```

```
                      0283          .SBTTL  FREE_BUFFER - release an I/O buffer
                      0283   ;++
                      0283   ; FREE_BUFFER
                      0283   ;
                      0283   ; Functional description:
                      0283   ;
                      0283   ;       This routine is called by the current owner of the MV work page
                      0283   ;       to make the page available.  If the MV work page wait queue is
                      0283   ;       not empty, a fork process will be created for the waiting device,
                      0283   ;       and the MV work page will be given to that process.
                      0283   ;
                      0283   ; Input:
                      0283   ;
                      0283   ;       None.
                      0283   ;
                      0283   ; Output:
                      0283   ;
                      0283   ;       R0, R3...R5 are preserved.
                      0283   ;--
                      0283
                      0283   FREE_BUFFER:                                    ; Release an I/O buffer
                      0283           ;
                      0283           ; Remove first entry from wait queue.  If empty, then return.
                      0283           ;
       C6 AF    D5    0283           TSTL    B^EXE$AL_WRKWQFL                ; Is work page wait queue initialized?
          06    13    0286           BEQL    10$                             ; Branch if Q not init'ed (ie. unused).
51     C1 BF    0F    0288           REMQUE  @B^EXE$AL_WRKWQFL, R1           ; Remove first entry from wait queue.
          04    1C    028C           BVC     20$                             ; Branch if somebody was waiting.
       C3 AF    D4    028E   10$:    CLRL    B^EXE$AL_BSYIRP                 ; Mark the MV work page as unowned.
                05    0291           RSB                                     ; Return to caller.
                      0292
                      0292           ;
                      0292           ; Restore the waiting fork process's context
                      0292           ; and call it at its return address.
                      0292           ;
          39    BB    0292   20$:    PUSHR   #^M<R0,R3,R4,R5>                ; Save current fork context and R0.
      55  51    D0    0294           MOVL    R1, R5                          ; Copy fork block address.
53     10 A5    7D    0297           MOVQ    CDRP$L_FR3(R5), R3              ; Restore saved R3 and R4.
55     1C A3    D0    029B           MOVL    IRP$L_UCB(R3), R5               ; Restore UCB address.
B1 AF     53    D0    029F           MOVL    R3, B^EXE$AL_BSYIRP             ; Save address of work page owner.
                      02A3           ASSUME  IRP$L_FPC EQ <IRP$L_FQFL + CDRP$L_FPC>
       6C B3    16    02A3           JSB     @IRP$L_FPC(R3)                  ; Call waiting fork process.
          39    BA    02A6           POPR    #^M<R0,R3,R4,R5>                ; Restore input context.
                05    02A8           RSB                                     ; Return.
```

```
                            02A9          .SBTTL  TIME_DELAY - Put mount verification into a wait state
                            02A9  ;+
                            02A9  ; TIME_DELAY
                            02A9  ;
                            02A9  ; Functional description:
                            02A9  ;
                            02A9  ;       This routine will put the specified mount verification
                            02A9  ;       request into a wait state for a short while.  This is
                            02A9  ;       accomplished by performing a FORK_WAIT using the CDRP
                            02A9  ;       in the IRP/CDRP pair of the input internal IRP.
                            02A9  ;
                            02A9  ; Input:
                            02A9  ;
                            02A9  ;       R3 = internal irp address
                            02A9  ;       R5 = device ucb address
                            02A9  ;
                            02A9  ; Output:
                            02A9  ;
                            02A9  ;       R3..R5  are preserved
                            02A9  ;-
                            02A9
                            02A9  TIME_DELAY:                               ; Pause for a moment
           78 A3 8ED0       02A9          POPL    IRP$L_SAVD_RTN(R3)        ; Save caller's return address.
     6B A3    0B A5   90    02AD          MOVB    UCB$B_FIPL(R5), -         ; Copy fork IPL to CDRP at tail of
                            02B2                  IRP$B_FIPL(R3)            ; IRP.
        55    60 A3   9E    02B2          MOVAB   IRP$L_FQFL(R3), R5        ; Get CDRP address.
                            02B6          FORK_WAIT                         ; Wait with CDRP as fork block.
        55    1C A3   D0    02BC          MOVL    IRP$L_UCB(R3), R5         ; Restore UCB address.
              78 B3   17    02C0          JMP     @IRP$L_SAVD_RTN(R3)       ; Return to original caller.
```

```
                                02C3          .SBTTL  INIT_IRP - set request independent fields of the IRP
                                02C3    ;++
                                02C3    ; INIT_IRP
                                02C3    ;
                                02C3    ; Functional description:
                                02C3    ;
                                02C3    ;       Zero an IRP and then set certain request-independent
                                02C3    ;       fields to their initial values.
                                02C3    ;
                                02C3    ; Input:
                                02C3    ;
                                02C3    ;       R3      = IRP address
                                02C3    ;       R5      = UCB address
                                02C3    ;
                                02C3    ; Output:
                                02C3    ;
                                02C3    ;       R0,R1,R2  are overwritten
                                02C3    ;       R4      preserved
                                02C3    ;       R3      = IRP address
                                02C3    ;       R5      = UCB address
                                02C3    ;--
                                02C3
                                02C3    INIT_IRP:                               ; Set request-independent fields of IRP
                                02C3    ;
                                02C3    ; Zero the IRP.
                                02C3    ;
                  14 A3    DD   02C3            PUSHL   IRP$L_ASTPRM(R3)        ; Save the MV timeout time
                     38    BB   02C6            PUSHR   #^M<R3,R4,R5>           ; Save IRP and UCB address from MOVC5
 63   00C4 8F   00   60 00    2C   02C8            MOVC5   #0,(R0),#0,#IRP$K_LENGTH,(R3)   ; Zero the IRP
                     38    BA   02D0            POPR    #^M<R3,R4,R5>           ; Restore IRP and UCB address
                  14 A3  8ED0   02D2            POPL    IRP$L_ASTPRM(R3)        ; Restore the MV timeout time
                                02D6    ;
                                02D6    ; Set the request-independent fields of the IRP.
                                02D6    ;
                                02D6            ;
                                02D6            ; Set the size, type and access mode fields.
                                02D6            ;
                                02D6            ASSUME  IRP$W_SIZE  EQ 8
                                02D6            ASSUME  IRP$B_TYPE  EQ 10
                                02D6            ASSUME  IRP$B_RMOD  EQ 11
     08 A3    000A00C4 8F    D0   02D6            MOVL    #<<DYN$C_IRP@16>!IRP$K_LENGTH>, IRP$W_SIZE(R3)
                                02DE            ;
                                02DE            ; Set the UCB address.  All other fields remain zero for now.
                                02DE            ;
          1C A3    55    D0   02DE            MOVL    R5,IRP$L_UCB(R3)        ; Set UCB address
                         05   02E2            RSB
```

```
                                  02E3          .SBTTL  INIT_IRP_READ - setup IRP for a one block read
                                  02E3     ;++
                                  02E3     ; INIT_IRP_READ
                                  02E3     ;
                                  02E3     ; Functional description:
                                  02E3     ;
                                  02E3     ;        Setup IRP fields required for a read operation.  Assume a one
                                  02E3     ;        block read of the LBN specified in R0.
                                  02E3     ;
                                  02E3     ; Input:
                                  02E3     ;
                                  02E3     ;        R0      LBN to read
                                  02E3     ;        R3      IRP address
                                  02E3     ;        R5      UCB address
                                  02E3     ;
                                  02E3     ; Output:
                                  02E3     ;
                                  02E3     ;        R0 ... R2 overwritten
                                  02E3     ;        All other registers preserved.
                                  02E3     ;--
                                  02E3
                                  02E3     INIT_IRP_READ:
                                  02E3
       20 A3    080C 8F      B0   02E3          MOVW    #<IO$_READPBLK -          ; Set function: read block
                                  02E9                   !IO$M_INHERLOG>, -       ;               w/o error logging
                                  02E9                  IRP$W_FUNC(R3)
                                  02E9          ASSUME  IO$_READPBLK LE IO$_PHYSICAL
       2A A3    2002 8F      A8   02E9          BISW    #<IRP$M_FUNC -            ; Set req. status:      read function
                                  02EF                   !IRP$M_MVIRP>, -         ;                       MV request
                                  02EF                  IRP$W_STS(R3)
     2C A3    00000000'GF    D0   02EF          MOVL    G^EXE$GL_SVAPTE, -        ; Set transfer SVAPTE for MV work
                                  02F7                  IRP$L_SVAPTE(R3)          ; page.
       32 A3    0200 8F      3C   02F7          MOVZWL  #512, IRP$L_BCNT(R3)      ; Set transfer byte count of 1 block.
          00000000'GF        17   02FD          JMP     G^IOC$CVTLOGPHY           ; Convert LBN (in R0) to PBN, and
                                  0303                                            ; return to caller.
```

```
                           0303          .SBTTL  VALIDATE_HOME
                           0303  ;++
                           0303  ; VALIDATE_HOME
                           0303  ;
                           0303  ;       This routine will check to see if the home block that was just
                           0303  ;       read contains valid data, and if so it will check to see if the
                           0303  ;       home block belongs to same volume as the one described in the
                           0303  ;       VCB for this device.  If both checks succeed, return success.
                           0303  ;       If either check fails, return failure.
                           0303  ;
                           0303  ; Input:
                           0303  ;
                           0303  ;       R3      = IRP address
                           0303  ;       R5      = UCB address
                           0303  ;
                           0303  ; Output:
                           0303  ;
                           0303  ;       R0              LBS ==> volume is correct
                           0303  ;                       LBC ==> some type of error
                           0303  ;       EXE$AB_MVWORK   is zero if the volume label comparison failed
                           0303  ;                       and one if it succeeded
                           0303  ;
                           0303  ;       R3, R4 and R5 are preserved.
                           0303  ;
                           0303  ; Side effects:
                           0303  ;
                           0303  ;       If the volume now physicaly mounted in the device is not
                           0303  ;       the same volume as the one described in the VCB, then the
                           0303  ;       interested parties will be informed, and the mount verification
                           0303  ;       will be restarted.
                           0303  ;--
                           0303
                           0303  VALIDATE_HOME:
                           0303
                  54  DD   0303          PUSHL   R4                      ; Save quorum indicator
                  7B  10   0305          BSBB    GET_BUF_ADDR            ; Get virtual address of MV buffer.
                           0307
                           0307  ;
                           0307  ; Compare the volume serial number and volume name in the VCB
                           0307  ; with those stored on the stack.  If they are identical, then
                           0307  ; this is the correct volume.  The following assumptions must
                           0307  ; be true if we are to avoid special-casing ODS-1 and ODS-2 volumes.
                           0307  ;
                           0307          ASSUME  HM2$L_SERIALNUM EQ HM1$L_SERIALNUM
                           0307          ASSUME  HM2$T_VOLNAME   EQ HM1$T_VOLNAME2
                           0307  ;
      50      34 A5   D0   0307          MOVL    UCB$L_VCB(R5), R0       ; Get VCB address.
                  2B  13   030B          BEQL    VALIDATE_EXIT           ; Branch if no VCB address.
   64 A0    01C8 C4   D1   030D          CMPL    HM2$L_SERIALNUM(R4), -  ; Check the volume serial number.
                           0313                  VCB$L_SERIALNUM(R0)
                  19  12   0313          BNEQ    VALIDATE_ERROR          ; Branch if serial nos. not equal.
             FF3F CF  94   0315          CLRB    W^EXE$AB_MVWORK         ; Assume label comparison will fail.
                  38  BB   0319          PUSHR   #^M<R3,R4,R5>           ; Save registers.
 14 A0    01D8 C4   0C  29 031B          CMPC3   #VCB$S_VOLNAME, -       ; Check the volume name.
                           0322                  HM2$T_VOLNAME(R4), -
                           0322                  VCB$T_VOLNAME(R0)
                  38  BA   0322          POPR    #^M<R3,R4,R5>           ; Restore registers.
```

```
                 04   12  0324         BNEQ    10$                     ; Branch if not the same volume.
            FF2E CF   96  0326         INCB    W^EXE$AB_MVWORK         ; Else, set same volume flag.
                 42   10  032A  10$:   BSBB    CHECKSUM                ; Test home block checksum, and
                 0A   11  032C         BRB     VALIDATE_EXIT
                          032E
                          032E  VALIDATE_ERROR:
                          032E
                          032E         ; A difference has been detected between the block in the MV work
                          032E         ; buffer (either the home block or the storage control block) and the
                          032E         ; VCB.  If the block checksum agrees with the caculated checksum, then
                          032E         ; the volume is incorrect.  If the checksums do not agree, then the
                          032E         ; preceived difference may be due to a corrupted disk block.  In such
                          032E         ; cases the read must be retried.
                          032E
                 3E   10  032E         BSBB    CHECKSUM                ; Checksum the MV work buffer.
              05 50   E9  0330         BLBC    R0, VALIDATE_EXIT       ; Branch if checksums differ.
       50   010C 8F   3C  0333         MOVZWL  #SS$_INCVOLLABEL, R0    ; Else, set incorrect volume status.
                          0338
                          0338  VALIDATE_EXIT:
               54 8ED0    0338         POPL    R4
                     05   033B         RSB                             ; Return.
```

```
                             033C          .SBTTL  VALIDATE_SCB
                             033C   ;++
                             033C   ; VALIDATE_SCB
                             033C   ;
                             033C   ;       This routine verifies the checksum of the storage control block in
                             033C   ;       the MV work buffer.  If the checksum is ok, the mount time in the SCB
                             033C   ;       is compared to the equivalent value in the VCB.  If the volume label
                             033C   ;       tested in VALIDATE_HOME did not match, the volume-lock-names in the
                             033C   ;       VCB and SCB are also compared.  If all tested values are equal,
                             033C   ;       success is returned.  Otherwise, SS$_INCVOLLABEL is returned.
                             033C   ;
                             033C   ; Input:
                             033C   ;
                             033C   ;       R5              UCB address
                             033C   ;       EXE$AB_MVWORK   is zero if the volume label comparison in
                             033C   ;                       VALIDATE_VOLUME failed and one if it succeeded
                             033C   ;
                             033C   ; Output:
                             033C   ;
                             033C   ;       R0      LBS ==> volume is correct
                             033C   ;               LBC ==> some type of error
                             033C   ;
                             033C   ;       R1 and R2 are overwritten.
                             033C   ;       All other registers are preserved.
                             033C   ;--
                             033C
                             033C   VALIDATE_SCB:
                  54     DD  033C          PUSHL   R4                      ; Save quorum indicator
                  42     10  033E          BSBB    GET_BUF_ADDR            ; Get the MV work buffer VA.
         50    34 A5     D0  0340          MOVL    UCB$L_VCB(R5), R0       ; Get VCB address.
                  F2     13  0344          BEQL    VALIDATE_EXIT           ; Branch if no VCB.
    0090 C0    2E A4     D1  0346          CMPL    SCB$Q_MOUNTTIME(R4), -  ; Compare mount times in the SCB
                             034C                  VCB$Q_MOUNTTIME(R0)     ; and in the VCB.
                  E0     12  034C          BNEQ    VALIDATE_ERROR          ; Take error exit if times don't
    0094 C0    32 A4     D1  034E          CMPL    SCB$Q_MOUNTTIME+4(R4), -; match.
                             0354                  VCB$Q_MOUNTTIME+4(R0)
                  D8     12  0354          BNEQ    VALIDATE_ERROR
         0D FEFE CF      E8  0356          BLBS    W^EXE$AB_MVWORK, 50$    ; Branch if VALIDATE_VOLUME succeeded.
                  38     BB  035B          PUSHR   #^M<R3,R4,R5>           ; Save registers.
0080 C0    22 A4  0C     29  035D          CMPC3   #VCB$S_VOLCKNAM, -      ; Check the volume-lock-name.
                             0364                  SCB$T_VOLOCKNAME(R4), -
                             0364                  VCB$T_VOLCKNAM(R0)
                  38     BA  0364          POPR    #^M<R3,R4,R5>           ; Restore registers.
                  C6     12  0366          BNEQ    VALIDATE_ERROR          ; Branch if not the same volume.
                  04     10  0368   50$:   BSBB    CHECKSUM                ; Test SCB checksum, and
              54 8ED0        036A          POPL    R4
                  05         036D          RSB
```

```
                        036E        .SBTTL  CHECKSUM - compute FILES-11 structure block checksum
                        036E  ;++
                        036E  ; CHECKSUM
                        036E  ;
                        036E  ; Functional description:
                        036E  ;
                        036E  ;       This routine computes the FILES-11 structure block checksum and
                        036E  ;       compares it to the checksum stored in the FILES-11 structure block.
                        036E  ;       If the checksums match, a success status is returned.
                        036E  ;
                        036E  ; Input:
                        036E  ;
                        036E  ;       R4      = FILES-11 structure block buffer address
                        036E  ;
                        036E  ; Ouput:
                        036E  ;
                        036E  ;       R0      = status value
                        036E  ;       R3 and R5 are preserved
                        036E  ;--
                        036E
                        036E  CHECKSUM:                                       ; Compute FILES-11 block checksum
                        036E
                        036E        ; Note: the FILES-11 structure block checksum is stored in the last
                        036E        ; word of the structure.  After the checksum loop completes, R4 points
                        036E        ; to the stored checksum and R1 contains the computed checksum.  The
                        036E        ; following assumptions must be true for this technique to work.
                        036E
                        036E        ASSUME  HM2$W_CHECKSUM2 EQ HM1$W_CHECKSUM2
                        036E        ASSUME  HM2$W_CHECKSUM2/2 EQ 255
                        036E
            50     7C   036E        CLRQ    R0                              ; Assume failure; clear total
  52    FF  8F     9A   0370        MOVZBL  #255,R2                         ; Set loop counter
     51     84     A0   0374  10$:  ADDW2   (R4)+,R1                        ; Sum adjacent words
        FA  52     F5   0377        SOBGTR  R2,10$                          ; Branch if more to go
    64      51     B1   037A        CMPW    R1,(R4)                         ; Compare checksums
            02     12   037D        BNEQ    13$                             ; Branch if not equal
            50     D6   037F        INCL    R0                              ; Set success status
                   05   0381  13$:  RSB                                     ; Return
```

```
                                    0382          .SBTTL  GET_BUF_ADDR
                                    0382  ;++
                                    0382  ; GET_BUF_ADDR
                                    0382  ;
                                    0382  ;       This routine returns the virtual address of the MV work buffer in R4.
                                    0382  ;
                                    0382  ; Input:
                                    0382  ;
                                    0382  ;       None.
                                    0382  ;
                                    0382  ;
                                    0382  ; Output:
                                    0382  ;
                                    0382  ;       R4      system virtual address of the MV work buffer
                                    0382  ;
                                    0382  ;       All other registers are preserved.
                                    0382  ;--
                                    0382
                                    0382  GET_BUF_ADDR:
                                    0382
00000000'GF    00000000'GF     C3   0382          SUBL3   G^MMG$GL_SPTBASE, -        ; Calculate byte offset to MV work
                        54          038D                  G^EXE$GL_SVAPTE, R4        ; SPTE.
          54     54     07     78   038E          ASHL    #9-2, R4, R4               ; Shift offset making VA minus 80000000.
             00  54     1F     E2   0392          BBSS    #PTE$V_VALID, R4, 10$      ; Fix the 80000000 part.
                               05   0396  10$:    RSB                                ; Return.
```

```
                        0397          .SBTTL  EXE$MNTVERSIO
                        0397  ;++
                        0397  ; EXE$MNTVERSIO
                        0397  ;
                        0397  ; Functional description:
                        0397  ;
                        0397  ;       This routine is called to initiate an I/O to a device.  The
                        0397  ;       request is packed in an 'internal' IRP.  This means that the
                        0397  ;       PID field of the IRP contains the address of a routine to be
                        0397  ;       called by IOPOST to finish processing the I/O request.  Note
                        0397  ;       that the PID field will be set to the address of a generic
                        0397  ;       post-processor, which will in turn branch to the actual post
                        0397  ;       processing routine.
                        0397  ;
                        0397  ; Input:
                        0397  ;
                        0397  ;       R0,R1,R2,R4 are scratch registers
                        0397  ;       R3      = IRP address
                        0397  ;       R5      = UCB address
                        0397  ;        (SP)   = address of action routine to post-process the I/O
                        0397  ;
                        0397  ; Output:
                        0397  ;
                        0397  ;       None.
                        0397  ;--
                        0397
                        0397  EXE$MNTVERSIO::                           ; Start an internal I/O request
                        0397
0C A3    AC'AF     9E   0397          MOVAB   B^END_IO,IRP$L_PID(R3)    ; Set I/O post address
         10 A3   8ED0   039C          POPL    IRP$L_AST(R3)             ; Set action routine address
                        03A0                                            ; Note that the stack is now clean.
                        03A0          ;
                        03A0          ; Begin optional I/O performance measurement.
                        03A0          ;
                        03A0          .IF     DF      CA$_MEASURE_IOT
  00000000'GF      16   03A0          JSB     G^PMS$START_RQ            ; Start I/O request measurement
                        03A6          .ENDC
                        03A6
  00000000'GF      17   03A6          JMP     G^IOC$INITIATE            ; Start the I/O immediately
```

```
                          03AC          .SBTTL  END_IO
                          03AC   ;++
                          03AC   ; END_IO
                          03AC   ;
                          03AC   ; Functional description:
                          03AC   ;
                          03AC   ;       This routine performs some common I/O post-processing
                          03AC   ;       before dispatching to the specific post-processing routine,
                          03AC   ;       whose address is stored in IRP$L_AST(R3).  This routine is
                          03AC   ;       called by IOPOST, at IPL 4, so it is necessary to get into
                          03AC   ;       the proper driver fork context before continuing.
                          03AC   ;
                          03AC   ; Input:
                          03AC   ;
                          03AC   ;       R5      = IRP address
                          03AC   ;       IPL     = IPL$_POST
                          03AC   ;
                          03AC   ; Output:
                          03AC   ;
                          03AC   ;       R5      = UCB address
                          03AC   ;       R3      = IRP address
                          03AC   ;
                          03AC   ;--
                          03AC
                          03AC   END_IO:                                ; End of I/O request
                          03AC          ;
                          03AC          ; Complete I/O performance measurement.
                          03AC          ;
                          03AC          .IF     DF      CA$_MEASURE_IOT
    00000000'GF     16    03AC          JSB     G^PMS$END_RQ          ; Gather performance data
                          03B2          .ENDC
                          03B2          ;
        53   55     D0    03B2          MOVL    R5,R3                 ; Copy IRP addres
    55   1C  A3     D0    03B5          MOVL    IRP$L_UCB(R3),R5      ; Get UCB address
                          03B9          DSBINT  UCB$B_FIPL(R5)        ; Raise IPL to driver FORK level
        10   B3     16    03C0          JSB     @IRP$L_AST(R3)        ; Dispatch to post processing routine
                          03C3          ENBINT                        ; Restore IPL
                    05    03C6          RSB                           ; Return
```

```
                        03C7          .SBTTL  CLEANUP_IO
                        03C7  ;++
                        03C7  ; CLEANUP_IO
                        03C7  ;
                        03C7  ; Functional description:
                        03C7  ;
                        03C7  ;       This routine is called to perform any necessary cleanup on the
                        03C7  ;       I/O request that has come into the main entry point of mount
                        03C7  ;       verification.
                        03C7  ;
                        03C7  ; Input:
                        03C7  ;
                        03C7  ;       R0,R1   = Status of the I/O operation
                        03C7  ;       R3      = IRP address
                        03C7  ;       R5      = UCB address
                        03C7  ;
                        03C7  ; Output:
                        03C7  ;
                        03C7  ;       All registers are preserved.
                        03C7  ;--
                        03C7
                        03C7  CLEANUP_IO:                             ; Finish processing an I/O packet
                        03C7          ;
                        03C7          ; If I/O performance measurement is enabled, call the proper routine
                        03C7          ; to gather the performance data.  Note that the performance routine
                        03C7          ; assumes that IRP$L_MEDIA and IRP$L_MEDIA+4 contain the I/O status.
                        03C7          ; However, these longwords currently contain some information that will
                        03C7          ; be needed to later restart the request.  Therefore, the contents of
                        03C7          ; IRP$L_MEDIA and IRP$L_MEDIA+4 must be saved and later restored.
                        03C7          ;
                        03C7          .IF     DF      CA$_MEASURE_IOT
    7E   38 A3      7D  03C7          MOVQ    IRP$L_MEDIA(R3),-(SP)   ; Save info from IRP
    38 A3   50      7D  03CB          MOVQ    R0,IRP$L_MEDIA(R3)      ; Set I/O status
00000000'GF         16  03CF          JSB     G^PMS$END_IO            ; Take performance measurement
    38 A3   8E      7D  03D5          MOVQ    (SP)+,IRP$L_MEDIA(R3)   ; Restore info to IRP
                        03D9          .ENDC
                    05  03D9          RSB                             ; Return
```

```
                        03DA          .SBTTL  DRIVER_CODE - Driver specific code
                        03DA     ;++
                        03DA     ; DRIVER_CODE
                        03DA     ;
                        03DA     ; Functional description:
                        03DA     ;
                        03DA     ;       This routine will call the driver specific routine that is necessary
                        03DA     ;       to begin and end mount verification.  The address of the routine is
                        03DA     ;       stored in the cell DDT$L_MNTVER.  If a driver does not specify the
                        03DA     ;       contents of DDT$L_MNTVER, it defaults to the address of IOC$MNTVER.
                        03DA     ;
                        03DA     ; Input:
                        03DA     ;
                        03DA     ;       R3      = IRP address or 0
                        03DA     ;       R5      = UCB address
                        03DA     ;
                        03DA     ; Output:
                        03DA     ;
                        03DA     ;       None.
                        03DA     ;
                        03DA     ; Side effects:
                        03DA     ;
                        03DA     ;       Driver specific, but the contents of R0,R1,R2 and R4 may be overwritten.
                        03DA     ;--
                        03DA
                        03DA     DRIVER_CODE:                               ; Call driver specific code
 50    0088 C5    D0    03DA          MOVL    UCB$L_DDT(R5),R0           ; Get DDT address
          20 B0   17    03DF          JMP     @DDT$L_MNTVER(R0)          ; Branch to driver specified routine
```

MOU
Sym

MSG
MVA
MVC
MVM
MVM
MVM
MVM
MVM
MVM
MVM
MVM
MVM
NOR
OFF
OPA
PAU
PMS
PMS
PMS
POO
PR$
PRE
PTE
QUO
QUO
SAV
SAV
SAV
SAV
SAV
SCB
SCB
SEN
SS$
SS$
SS$
SS$
SS$
SUF
SYS
SYS
TAB
TIM
UCB
UCB
UCB
UCB
UCB
UCB
UCB
UCB
UCB
UCB
UCB
UCB
UCB
UCB

```
                  03E2         .SBTTL  WRITLCK_HNDLR
                  03E2  ;++
                  03E2  ; WRITLCK_HNDLR
                  03E2  ;
                  03E2  ; Functional description:
                  03E2  ;
                  03E2  ;       This routine will allow the file system to recover from the
                  03E2  ;       accidental hardware writelocking of a FILES-11 volume.  The
                  03E2  ;       method used is to mark the current IRP as a mount verification
                  03E2  ;       IRP, and try the request over again.  If the request then
                  03E2  ;       succeeds, nothing more need be done.  If the request fails again,
                  03E2  ;       keep trying until it succeeds.  If this is the first time this
                  03E2  ;       request has failed (the MVIRP bit will be clear), then inform
                  03E2  ;       all interested parties that the device has been writelocked.
                  03E2  ;
                  03E2  ;       Note that writelock recovery is not interlocked by the MVNVERIP
                  03E2  ;       bit.  This will allow recovery of a device offline error to supersede
                  03E2  ;       writelock recovery.  If this were not done, it would be possible to
                  03E2  ;       become deadlocked by an offline error occurring after a writelock
                  03E2  ;       error.
                  03E2  ;
                  03E2  ; Input:
                  03E2  ;
                  03E2  ;       R0,R1   = I/O status
                  03E2  ;       R3      = IRP address
                  03E2  ;       R5      = UCB address
                  03E2  ;
                  03E2  ; Output:
                  03E2  ;
                  03E2  ;       None.
                  03E2  ;--
                  03E2
                  03E2  WRITLCK_HNDLR:                          ; Recover from accidental writelock
       8E    D5   03E2         TSTL    (SP)+                   ; Remove return address from stack
       E1    10   03E4         BSBB    CLEANUP_IO              ; Clean up current I/O operation
                  03E6  ;
                  03E6  ; To prevent the error log to become saturated with entries
                  03E6  ; due the repeated failure of this request, inhibit error logging
                  03E6  ; for this operation.  Note that if error logging had previously
                  03E6  ; been enabled, then the error log will already contain an entry
                  03E6  ; for this request.
                  03E6  ;
       0B    E2   03E6         BBSS    #IO$V_INHERLOG,-        ; Inhibit error logging
 00 20 A3         03E8                 IRP$W_FUNC(R3),10$      ;
                  03EB  ;
                  03EB  ; If mount verification is already in progress, then
                  03EB  ; requeue this request to the device and try again later.
                  03EB  ;
       0E    E1   03EB  10$:   BBC     #UCB$V_MNTVERIP,-       ; Branch if mount verification not in progre
 03 64 A5         03ED                 UCB$W_STS(R5),20$       ;
    E8       10   03F0         BSBB    DRIVER_CODE             ; Requeue the IRP to the driver
             05   03F2         RSB                             ; Return to caller's caller
                  03F3  ;
                  03F3  ; Mark this IRP as a mount verification IRP.  If it already is,
                  03F3  ; then retry the I/O immediately.
                  03F3  ;
       0D    E2   03F3  20$:   BBSS    #IRP$V_MVIRP,-          ; Branch if already set and set the bit
```

```
        05 2A A3         03F5  1349              IRP$W_STS(R3),30$        ;
                         03F8  1350 ;
                         03F8  1351 ; This is the first time this IRP has come through.  Inform all
                         03F8  1352 ; interested parties that the device has been writelocked.
                         03F8  1353 ;
       54     02    9A   03F8  1354         MOVZBL  #WRITELOCK,R4   ; Set message type code
              09    10   03FB  1355         BSBB    SEND_MESSAGE            ; Send message to interested parties
                         03FD  1356 ;
                         03FD  1357 ; Try the I/O over again.
                         03FD  1358 ;
           FEA9     30   03FD  1359 30$:    BSBW    TIME_DELAY              ; Pause for a bit
      00000000'GF   17   0400  1360         JMP     G^IOC$INITIATE          ; Retry the I/O
```

**F

```
                                 0406          .SBTTL  SEND_MESSAGE
                                 0406  ;++
                                 0406  ; SEND_MESSAGE
                                 0406  ;
                                 0406  ; Functional description:
                                 0406  ;
                                 0406  ;       This routine is used to inform all interested parties of a mount
                                 0406  ;       verification related event.  A message is sent to OPCOM, which will
                                 0406  ;       in turn send the message to all operators enabled to receive DEVICE
                                 0406  ;       or DISK messages.
                                 0406  ;
                                 0406  ;       Since there is a possibility that OPCOM will not be able to relay
                                 0406  ;       the message to the operator, also send a message that is targeted
                                 0406  ;       explicitly at the operator's console.
                                 0406  ;
                                 0406  ; Input:
                                 0406  ;
                                 0406  ;       R4      = MV message list index for message
                                 0406  ;       R5      = UCB address
                                 0406  ;
                                 0406  ; Output:
                                 0406  ;
                                 0406  ;       None.  Contents of R0 .. R2 are destroyed
                                 0406  ;
                                 0406  ;--
                                 0406
              00000000           0406  SAVED_R1 =      0                       ;.
              00000004           0406  SAVED_R2 =      4                       ; .
              00000008           0406  SAVED_R3 =      8                       ;  . Offsets to saved registers
              0000000C           0406  SAVED_R4 =      12                      ; .
              00000010           0406  SAVED_R5 =      16                      ;.
              00000014           0406  DEVNAM_SIZE     = 20                    ; Allow 20 character device names
              0000000A           0406  POOL_OVERHEAD   = 2+IRP$W_SIZE          ; Allow for listhead and size fields
              00000022           0406  MSG_OVERHEAD    = POOL_OVERHEAD + DEVNAM_SIZE + <2 * CAR_CTRL_SIZE>
              0000000A           0406  MSG_START       = POOL_OVERHEAD         ; First usuable byte in pool
                                 0406
                                 0406  SEND_MESSAGE:                           ; Send message to interested parties
                                 0406  ;
                                 0406  ; Send message to OPCOM.
                                 0406  ;
                    3E     BB    0406          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save registers
                  00AF     30    0408          BSBW    GET_MSG_ID              ; Put address of message Id in R4
        08 02 A4    01     E1    040B          BBC     #MVMSL$V_SUPRESS, -     ; Branch if MVMSL does not indicate
                                 0410                  MVMSL$W_FLAGS(R4), 15$  ; that this message is supressable.
        07 64 A5    12     E1    0410          BBC     #UCB$V_SUPMVMSG, -      ; Branch if UCB does not indicate that
                                 0415                  UCB$L_STS(R5), 17$      ; normal MV messages are supressed.
                  009F     31    0415          BRW     90$                     ; Else, supress this message.
                                 0418  15$:    ASSUME  UCB$V_SUPMVMSG GE 16
           66 A5    04     8A    0418          BICB    #<UCB$M_SUPMVMSG@-16>, -; For must print messages, clear the
                                 041C                  UCB$L_STS+2(R5)         ; supress mount verification msgs. bit.
              54    64     3C    041C  17$:    MOVZWL  MVMSL$W_MSG_CODE(R4),R4 ; Get the OPCOM message number.
    53   00000000'GF       9E    041F          MOVAB   G^SYS$GL_OPRMBX,R3      ; Get operator mailbox UCB address
         00000000'GF       16    0426          JSB     G^EXE$SNDEVMSG          ; Mail message to OPCOM; ignore failure
                                 042C  ;
                                 042C  ; Send message to _OPA0:.
                                 042C  ;
                                 042C          ;
```

```
                              042C       ; Calculate the total message size and allocate a block
                              042C       ; of nonpaged pool in which to build the message.
                              042C       ; SIZE = prefix_size + message_size + suffix_size + pool_header_size + devic
                              042C       ;
            008B    30        042C            BSBW    GET_MSG_ID                ; Get message description and id.
         51   61    9A        042F            MOVZBL  (R1), R1                  ; Setup allocation size accumulator.
      51   0085'CF  80        0432            ADDB    W^PREFIX, R1              ; Add prefix size to message size.
      51   009C'CF  80        0437            ADDB    W^SUFFIX, R1              ; Add suffix size to accumulated size.
           51   22  80        043C            ADDB    #MSG_OVERHEAD, R1         ; Add message overhead to acc. size.
       00000000'GF  16        043F            JSB     G^EXE$ALONONPAGED         ; Allocate the necessary pool
            6F  50  E9        0445            BLBC    R0, 90$                   ; If no pool available, give up.
                              0448       ;
                              0448       ; Build the message in the pool just allocated.
                              0448       ; The entire message consists of the prefix, followed by
                              0448       ; the formatted device name, followed by the message text,
                              0448       ; followed by the sufix.
                              0448       ;
                              0448       ; R1 = actual pool size
                              0448       ; R2 = address of pool
                              0448       ;
      08 A2   51    B0        0448            MOVW    R1,IRP$W_SIZE(R2)         ; Save size of pool in the block itself
      04 AE   52    D0        044C            MOVL    R2,SAVED_R2(SP)           ; Save address of pool
      53   0A A2    9E        0450            MOVAB   MSG_START(R2),R3          ; Get address of message buffer.
   83   0A0D 8F     B0        0454            MOVW    #CAR_CTRL,(R3)+           ; Insert leading <cr><lf>
   51   0085'CF     9E        0459            MOVAB   W^PREFIX,R1               ; Get address of prefix ASCIC string
              6C    10        045E            BSBB    COPY_ASCIC                ; Copy string to buffer (R5 destroyed)
   55   10 AE       D0        0460            MOVL    SAVED_R5(SP),R5           ; Restore R5
      50   14       D0        0464            MOVL    #DEVNAM_SIZE,R0           ; Set device name size
      51   53       D0        0467            MOVL    R3,R1                     ; Set buffer address
           54       DD        046A            PUSHL   R4                        ; Save R4
      54   01       CE        046C            MNEGL   #1,R4                     ; Signal get node+devnam for remote devs
       00000000'GF  16        046F            JSB     G^IOC$CVT_DEVNAM          ; Format the device name (R1 = result length
           54 8ED0            0475            POPL    R4                        ; Restore R4
      53   51       C0        0478            ADDL    R1,R3                     ; Point to next byte in msg buffer
           3D       10        047B            BSBB    GET_MSG_ID                ; R4 = address of message identifier
           4D       10        047D            BSBB    COPY_ASCIC                ; Copy message text to buffer.
           39       10        047F            BSBB    GET_MSG_ID                ; Get message text descriptor
 07 02 A4  00       E0        0481            BBS     #MVMSL$V_NOSUFFIX, -      ; Branch if suffix not requested.
                              0486                    MVMSL$W_FLAGS(R4), 30$
   51   009C'CF     9E        0486            MOVAB   W^SUFFIX,R1               ; Get address of suffix ASCIC string
           3F       10        048B            BSBB    COPY_ASCIC                ; Copy suffix to the buffer
   83   0A0D 8F     B0        048D   30$:     MOVW    #CAR_CTRL, (R3)+          ; Insert trailing <CR><LF>.
                              0492       ;
                              0492       ; Send the message to _OPA0:
                              0492       ;
   54   04 AE       D0        0492            MOVL    SAVED_R2(SP),R4           ; Restore pool address
   52   0A A4       9E        0496            MOVAB   MSG_START(R4),R2          ; Get start of message address
 51   53   52       C3        049A            SUBL3   R2,R3,R1                  ; Calculate the length of the message
55  00000000'GF    9E        049E            MOVAB   G^OPA$UCB0,R5             ; Get console terminal UCB address
    00000000'GF     16        04A5            JSB     G^IOC$BROADCAST           ; Send the message
      50   54       D0        04AB            MOVL    R4,R0                     ; Copy pool address
    00000000'GF     16        04AE            JSB     G^COM$DRVDEALMEM          ; Deallocate the pool
      50   01       3C        04B4            MOVZWL  #SS$_NORMAL,R0            ; Set success status
           3E       BA        04B7   90$:     POPR    #^M<R1,R2,R3,R4,R5>       ; Restore R1 ... R5.
           05                 04B9            RSB                               ; Return
                              04BA
                              04BA       ;+
```

```
                     04BA  ; This is a local subroutine to index into the message
                     04BA  ; identifier table and return the address of the correct
                     04BA  ; entry in R4.  Note that the stack depth has changed due
                     04BA  ; to the return address on the stack.
                     04BA  ;-
                     04BA  GET_MSG_ID:                                  ; Return address of message identifier
      54    10 AE   D0 04BA          MOVL    4+SAVED_R4(SP),R4         ; Get the table index number
  51    FB42 CF     9E 04BE          MOVAB   W^EXE$AB_MVMSLBAS, R1     ; Get base of message info table.
                     04C3          ASSUME  MVMSL$K_LENGTH EQ 8
      54    6144    7E 04C3          MOVAQ   (R1)[R4], R4              ; Get address of entry for message id.
      51    04 A4   C0 04C7          ADDL    MVMSL$L_TEXTOFF(R4), R1   ; Get base of counted string address.
                    05 04CB          RSB                               ; Return
                     04CC
                     04CC
                     04CC  ;+
                     04CC  ; This routine is a spcial case of COPY_STRING.  On input, R1 is
                     04CC  ; assumed to contain the address of an ASCIC string.  Execution
                     04CC  ; drops through to COPY_STRING, and will return from there to the caller.
                     04CC  ;-
                     04CC  COPY_ASCIC:                                  ; Special case of COPY_STRING
        50    81    9A 04CC          MOVZBL  (R1)+,R0                  ; Get length of string and advance the point
                     04CF
                     04CF  ;+
                     04CF  ; This routine is used to copy a string to a buffer.
                     04CF  ;
                     04CF  ; Input:
                     04CF  ;
                     04CF  ;         R0 = string length
                     04CF  ;         R1 = string address
                     04CF  ;         R3 = buffer address
                     04CF  ;
                     04CF  ; Output:
                     04CF  ;
                     04CF  ;         R3 = address of next byte in the buffer.
                     04CF  ;              (This is a fortuitous side-effect of the MOVC3 instruction.)
                     04CF  ;-
                     04CF  COPY_STRING:                                 ; Copy a string to a buffer
   63    61    50   28 04CF          MOVC3   R0,(R1),(R3)              ; Go to it!
                    05 04D3          RSB                               ; Return
```

```
             04D4          .SBTTL  EXE$UPDGNERNUM - update shadow set generation number
             04D4          .SBTTL  EXE$MNTVERSHDOL - bring a shadow set member online
             04D4          .SBTTL  EXE$MNTVERSP1 - spare mount verification entry point
             04D4          .SBTTL  EXE$MNTVERSP2 - spare mount verification entry point
             04D4          .SBTTL  EXE$MNTVER_DVI_ASSIST - $GETDVI escape transfer vector
             04D4  ;++
             04D4  ; EXE$UPDGNERNUM - update shadow set generation number
             04D4  ; EXE$MNTVERSHDOL - bring a shadow set member online
             04D4  ; EXE$MNTVERSP1 - spare mount verification entry point
             04D4  ; EXE$MNTVERSP2 - spare mount verification entry point
             04D4  ; EXE$MNTVER_DVI_ASSIST - $GETDVI escape transfer vector
             04D4  ;
             04D4  ; Functional description:
             04D4  ;
             04D4  ;       These mount verification entry points do not yet have any code written
             04D4  ;       for them.  There are defined here to hold their places for possibly /
             04D4  ;       probable implementation during the life-time of Version 4.
             04D4  ;--
             04D4
             04D4  EXE$UPDGNERNUM::
             04D4  EXE$MNTVERSHDOL::
             04D4  EXE$MNTVERSP1::
             04D4  EXE$MNTVERSP2::
             04D4
          05 04D4          RSB
             04D5
             04D5  EXE$MNTVER_DVI_ASSIST::
             04D5
       60 17 04D5          JMP     (R0)
```

```
                                 04D7        .SBTTL  EXE$CLUTRANIO - VAXcluster State Change I/O Blocking
                                 04D7  ;++
                                 04D7  ; EXE$CLUTRANIO - VAXcluster State Change I/O Blocking
                                 04D7  ;
                                 04D7  ; Functional description:
                                 04D7  ;
                                 04D7  ;       This routine is used to place all MSCP or cluster accessible disks into
                                 04D7  ;       mount verification.  The intended use is to ensure that no I/O is
                                 04D7  ;       issued when a cluster may be out of quorum.
                                 04D7  ;
                                 04D7  ; Input:
                                 04D7  ;
                                 04D7  ;       None
                                 04D7  ;
                                 04D7  ; Output:
                                 04D7  ;
                                 04D7  ;       None.  Contents of all registers preserved.
                                 04D7  ;
                                 04D7  ;--
                                 04D7
                                 04D7  EXE$CLUTRANIO::
               007F 8F     BB    04D7          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6>
                                 04DB  ;
                                 04DB  ; First look at all local devices.
                                 04DB  ; If the UCB is not busy, it is marked busy and mount verifiction is started.
                                 04DB  ; If busy, volume valid is cleared and the mount verification pending bit is
                                 04DB  ; set.
                                 04DB  ;
     56    00000000'GF     D0    04DB          MOVL    G^IOC$GL_DEVLIST,R6        ; Start of DDB list
          55    04 A6      D0    04E2  10$:    MOVL    DDB$L_UCB(R6),R5           ; Get a UCB
                   49      13    04E6          BEQL    50$                        ; None
          01    40 A5      91    04E8          CMPB    UCB$B_DEVCLASS(R5),#DC$_DISK
                   43      12    04EC          BNEQ    50$                        ; Not a disk
       3E 38 A5    0E      E1    04EE          BBC     #DEV$V_FOD,UCB$L_DEVCHAR(R5),50$ ; Not file oriented
       39 3C A5    05      E0    04F3          BBS     #DEV$V_MSCP,UCB$L_DEVCHAR2(R5),50$ ; Not interested in MSCP now
                34 A5      D5    04F8  20$:    TSTL    UCB$L_VCB(R5)              ; Check the VCB - consistency check only
                   2E      13    04FB          BEQL    40$                        ; Next UCB if none
       29 64 A5    0E      E0    04FD          BBS     #UCB$V_MNTVERIP,UCB$W_STS(R5),40$ ; Already have it
       24 3C A5    00      E1    0502          BBC     #DEV$V_CLU,UCB$L_DEVCHAR2(R5),40$ ; Not cluster accessible
       00 64 A5    12      E2    0507          BBSS    #UCB$V_SUPMVMSG,UCB$L_STS(R5),25$ ; No messages
       10 64 A5    08      E1    050C  25$:    BBC     #UCB$V_BSY,UCB$W_STS(R5),30$ ; UCB not busy
     64 A5    0800 8F      AA    0511          BICW    #UCB$M_VALID,UCB$W_STS(R5) ; Clear vol-valid to stop I/Os
 64 A5    00284000 8F      C8    0517          BISL    #<UCB$M_MNTVERIP -         ; Pending when I/O completes
                                 051F                   !UCB$M_MNTVERPND -        ; initiate mount verification for
                                 051F                   !UCB$M_CLUTRAN>, -        ; for a VAXcluster state transition.
                                 051F                  UCB$L_STS(R5)
                   0A      11    051F          BRB     40$
     64 A5    0100 8F      A8    0521  30$:    BISW    #UCB$M_BSY,UCB$W_STS(R5); Make UCB busy to stall I/O
               FB5F CF     16    0527          JSB     MNTVER_NOIRP               ; Put it in mount verification
          55    30 A5      D0    052B  40$:    MOVL    UCB$L_LINK(R5),R5          ; Next UCB
                   C7      12    052F          BNEQ    20$                        ; Go look at it
          56    66         D0    0531  50$:    MOVL    DDB$L_LINK(R6),R6          ; next DDB
                   AC      12    0534          BNEQ    10$                        ; Go look at it
                                 0536  ;
                                 0536  ; Now handle the MSCP disks.  It is only necessary to set the CDDB bit to
                                 0536  ; indicate out-of-quorum and then call the revalidate routine.
                                 0536  ;
```

```
53   00000000'GF  D0  0536  1603          MOVL   G^IOC$GL_DU_CDDB,R3        ; Get start of CDDB list
            25    13  053D  1604          BEQL   90$                        ; None
     0200 8F      A8  053F  1605  70$:    BISW   #CDDB$M_QUORLOST,-         ; Indicate a quorum lost request
       12 A3          0543  1606                 CDDB$W_STATUS(R3)
     0F88 8F      BB  0545  1607          PUSHR  #^M<R3,R7,R8,R9,R10,R11>
50     1C A3      D0  0549  1608          MOVL   CDDB$L_DDB(R3),R0          ; Get DDB
50     0C A0      D0  054D  1609          MOVL   DDB$L_DDT(R0),R0           ; Get DDT
       04 B0      16  0551  1610          JSB    @DDT$L_UNSOLINT(R0)        ; Call the revalidate routine
     0F88 8F      BA  0554  1611          POPR   #^M<R3,R7,R8,R9,R10,R11>
     0200 8F      AA  0558  1612          BICW   #CDDB$M_QUORLOST,-         ; Back to usual state
       12 A3          055C  1613                 CDDB$W_STATUS(R3)
53     58 A3      D0  055E  1614          MOVL   CDDB$L_CDDBLINK(R3),R3     ; Get next CDDB
          DB      12  0562  1615          BNEQ   70$                        ; Go process it
     007F 8F      BA  0564  1616  90$:    POPR   #^M<R0,R1,R2,R3,R4,R5,R6>
                  05  0568  1617          RSB
                      0569  1618
```

```
                                 0569        .SBTTL  QUORUM
                                 0569  ;++
                                 0569  ; QUORUM
                                 0569  ;
                                 0569  ; Functional description:
                                 0569  ;
                                 0569  ;       This routine is used to determine if we are in a cluster that is
                                 0569  ;       out of quorum.
                                 0569  ;
                                 0569  ; Input:
                                 0569  ;
                                 0569  ;       None
                                 0569  ;
                                 0569  ; Output:
                                 0569  ;
                                 0569  ;       R4   0 -> not an out of quorum cluster
                                 0569  ;            1 -> out of quorum cluster
                                 0569  ;
                                 0569  ;--
                                 0569
                      54    D4   0569  QUORUM: CLRL   R4                              ; Assume no cluster
   52    00000000'GF    D0   056B          MOVL   G^CLU$GL_CLUB,R2
                      0D    13   0572          BEQL   50$                             ; No CLUB, so no cluster
         08 1C A2     00    E1   0574          BBC    #CLUB$V_CLUSTER,CLUB$L_FLAGS(R2),50$ ; Not a cluster member
         03 1C A2     1C    E0   0579          BBS    #CLUB$V_QUORUM,CLUB$L_FLAGS(R2),50$ ; Not out of quorum
                 54   01    D0   057E          MOVL   #1,R4                           ; Indicate no quorum
                            05   0581  50$:    RSB
                                 0582
```

```
                          0582          .SBTTL  GET_VCB - Obtain VCB address
                          0582  ;++
                          0582  ;
                          0582  ; GET_VCB - Obtain VCB address
                          0582  ;
                          0582  ; Functional Description:
                          0582  ;
                          0582  ;       Because mount verification is used for more than just fixing mistaken
                          0582  ;       hardware changes, there are several conditions in which mount
                          0582  ;       verification can be entered legally without there being a  FILES-11
                          0582  ;       VCB.  Such cases are:
                          0582  ;
                          0582  ;         1. The volume being processed is mounted /FOREIGN.
                          0582  ;
                          0582  ;         2. The volume being processed is the system disk and it has not yet
                          0582  ;            been properly mounted.
                          0582  ;
                          0582  ;         3. The volume being processed is the quorum disk and it has not yet
                          0582  ;            been properly mounted.
                          0582  ;
                          0582  ;       This routine returns control if a VCB is present or legally absent.
                          0582  ;       The condition code is NEQ if the VCB is present or EQL if it is
                          0582  ;       legally absent.  Otherwise, mount verification is aborted.
                          0582  ;
                          0582  ; Inputs:
                          0582  ;
                          0582  ;       R3      MVIRP address
                          0582  ;       R5      UCB address
                          0582  ;
                          0582  ; Outputs:
                          0582  ;
                          0582  ;       Condition codes:
                          0582  ;
                          0582  ;       NEQ     VCB is present
                          0582  ;       EQL     VCB is legally absent
                          0582  ;
                          0582  ;       R0      VCB address
                          0582  ;
                          0582  ;       All other registers are preserved.
                          0582  ;--
                          0582
                          0582  GET_VCB:
                          0582
  25 38 A5    18    E0    0582          BBS     #DEV$V_FOR, -               ; Branch if /FOREIGN disk.
                          0587                  UCB$L_DEVCHAR(R5), 60$
                          0587
     50    34 A5    D0    0587          MOVL    UCB$L_VCB(R5), R0           ; Get VCB address.
              21    12    058B          BNEQ    70$                         ; Branch if VCB is present.
                          058D
55   00000000'8F    D1    058D          CMPL    #SYS$GL_BOOTUCB, R5         ; Is this the system disk?
              16    13    0594          BEQL    60$                         ; Branch if system disk.
                          0596
50   00000000'GF    D0    0596          MOVL    G^CLU$GL_CLUB, R0           ; Get CLUB address.
              10    13    059D          BEQL    80$                         ; Branch if no CLUB.
   50    00B4 C0    D0    059F          MOVL    CLUB$L_CLUDCB(R0), R0       ; Get quorum disk control block addr.
              09    13    05A4          BEQL    80$                         ; Branch if no CLUDCB.
     55    0C A0    D1    05A6          CMPL    CLUDCB$L_UCB(R0), R5        ; Is this the quorum disk?
```

```
      03   12  05AA  1706         BNEQ    80$              ; Branch if not the quorum disk.
               05AC  1707
      50   D4  05AC  1708 60$:    CLRL    R0               ; Indicate VCB legally absent.
           05  05AE  1709 70$:    RSB                      ; Return to caller.
               05AF  1710
               05AF  1711                                  ; VCB not present when it should be.
      8E   D5  05AF  1712 80$:    TSTL    (SP)+            ; Pop return address from stack.
    FC8A   31  05B1  1713         BRW     ERROR_EXIT       ; Abort mount verification.
               05B4  1714
               05B4  1715
               05B4  1716         .END
```

```
BUILD_PACKACK_IRP                000000BE R      02      EXE$SNDEVMSG                   ********   X   02
BUILD_RDHOME_IRP                 000001A2 R      02      EXE$UPDGNERNUM                 000004D4 RG    02
CAR_CTRL                       = 00000A0D                FREE_BUFFER                    00000283 R     02
CAR_CTRL_SIZE                  = 00000002                FREE_BUF_ERRXIT                0000023C R     02
CDDB$L_CDDBLINK                = 00000058                GET_BUFFER                     00000259 R     02
CDDB$L_DDB                     = 0000001C                GET_BUF_ADDR                   00000382 R     02
CDDB$M_QUORLOST                = 00000200                GET_MSG_ID                     000004BA R     02
CDDB$W_STATUS                  = 00000012                GET_VCB                        00000582 R     02
CDRP$L_FPC                     = 0000000C                HM1$L_SERIALNUM              = 000001C8
CDRP$L_FR3                     = 00000010                HM1$T_VOLNAME2               = 000001D8
CHECKSUM                         0000036E R      02      HM1$W_CHECKSUM2              = 000001FE
CLEANUP_IO                       000003C7 R      02      HM2$L_SERIALNUM              = 000001C8
CLU$GL_CLUB                      ********   X    02      HM2$T_VOLNAME                = 000001D8
CLUB$L_CLUDCB                  = 000000B4                HM2$W_CHECKSUM2              = 000001FE
CLUB$L_FLAGS                   = 0000001C                INIT_IRP                       000002C3 R     02
CLUB$V_CLUSTER                 = 00000000                INIT_IRP_READ                  000002E3 R     02
CLUB$V_QUORUM                  = 0000001C                IO$M_INHERLOG                = 00000800
CLUDCB$L_IRP                   = 00000010                IO$V_INHERLOG                = 0000000B
CLUDCB$L_UCB                   = 0000000C                IO$_PACKACK                  = 00000008
COM$DRVDEALMEM                   ********   X    02      IO$_PHYSICAL                 = 0000001F
COPY_ASCIC                       000004CC R      02      IO$_READPBLK                 = 0000000C
COPY_STRING                      000004CF R      02      IOC$BROADCAST                  ********   X   02
DC$_DISK                       = 00000001                IOC$CVTLOGPHY                  ********   X   02
DDB$L_DDT                      = 0000000C                IOC$CVT_DEVNAM                 ********   X   02
DDB$L_LINK                     = 00000000                IOC$GL_DEVLIST                 ********   X   02
DDB$L_UCB                      = 00000004                IOC$GL_DU_CDDB                 ********   X   02
DDT$L_FDT                      = 00000008                IOC$GW_MVTIMEOUT               ********   X   02
DDT$L_MNTVER                   = 00000020                IOC$INITIATE                   ********   X   02
DDT$L_UNSOLINT                 = 00000004                IRP$B_FIPL                   = 0000006B
DEV$V_CLU                      = 00000000                IRP$B_RMOD                   = 0000000B
DEV$V_FOD                      = 0000000E                IRP$B_TYPE                   = 0000000A
DEV$V_FOR                      = 00000018                IRP$K_LENGTH                 = 000000C4
DEV$V_MNT                      = 00000013                IRP$L_AST                    = 00000010
DEV$V_MSCP                     = 00000005                IRP$L_ASTPRM                 = 00000014
DEV$V_SQD                      = 00000005                IRP$L_BCNT                   = 00000032
DEV$V_SWL                      = 00000019                IRP$L_FPC                    = 0000006C
DEVNAM_SIZE                    = 00000014                IRP$L_FQFL                   = 00000060
DRIVER_CODE                      000003DA R      02      IRP$L_MEDIA                  = 00000038
DYN$C_IRP                      = 0000000A                IRP$L_PID                    = 0000000C
END_IO                           000003AC R      02      IRP$L_SAVD_RTN               = 00000078
ERROR_EXIT                       0000023E R      02      IRP$L_SVAPTE                 = 0000002C
EXE$AB_MVMSLBAS                  00000004 RG     02      IRP$L_UCB                    = 0000001C
EXE$AB_MVWORK                    00000258 R      02      IRP$M_FUNC                   = 00000002
EXE$ALONONPAGED                  ********   X    02      IRP$M_MVIRP                  = 00002000
EXE$AL_BSYIRP                    00000254 R      02      IRP$M_PHYSIO                 = 00000100
EXE$AL_WRKWQBL                   00000250 R      02      IRP$V_MVIRP                  = 0000000D
EXE$AL_WRKWQFL                   0000024C R      02      IRP$W_FUNC                   = 00000020
EXE$CLUTRANIO                    000004D7 RG     02      IRP$W_SIZE                   = 00000008
EXE$FORK_WAIT                    ********   X    02      IRP$W_STS                    = 0000002A
EXE$GL_ABSTIM                    ********   X    02      MMG$GL_SPTBASE                 ********   X   02
EXE$GL_SVAPTE                    ********   X    02      MNTVER_NOIRP                   0000008A R     02
EXE$MNTVERSHDOL                  000004D4 RG     02      MSG$_DEVOFFLINX              = 00000050
EXE$MNTVERSIO                    00000397 RG     02      MSG$_DEVWRTLCK               = 00000052
EXE$MNTVERSP1                    000004D4 RG     02      MSG$_MVABORTED               = 00000055
EXE$MNTVERSP2                    000004D4 RG     02      MSG$_MVCOMPLETE              = 00000054
EXE$MNTVER_DVI_ASSIST            000004D5 RG     02      MSG$_WRONGVOL                = 00000051
EXE$MOUNTVER                     0000002C RG     02      MSG_OVERHEAD                 = 00000022
```

```
MSG_START                        = 0000000A                     UCB$M_VALID                      = 00000800
MVABORTED                        = 00000004                     UCB$M_WRONGVOL                   = 00008000
MVCOMPLETE                       = 00000003                     UCB$V_BSY                        = 00000008
MVMSL$K_LENGTH                   = 00000008                     UCB$V_CLUTRAN                    = 00000015
MVMSL$L_SNDMSGOFF                = FFFFFFFC                     UCB$V_MNTVERIP                   = 0000000E
MVMSL$L_TEXTOFF                  = 00000004                     UCB$V_SUPMVMSG                   = 00000012
MVMSL$M_NOSUFFIX                 = 00000001                     UCB$V_VALID                      = 0000000B
MVMSL$M_SUPRESS                  = 00000002                     UCB$V_WRONGVOL                   = 0000000F
MVMSL$V_NOSUFFIX                 = 00000000                     UCB$W_STS                        = 00000064
MVMSL$V_SUPRESS                  = 00000001                     VALIDATE_ERROR                     0000032E R      02
MVMSL$W_FLAGS                    = 00000002                     VALIDATE_EXIT                      00000338 R      02
MVMSL$W_MSG_CODE                 = 00000000                     VALIDATE_FAILED                    00000219 R      02
NORMAL_EXIT                        00000200 R      02           VALIDATE_HOME                      00000303 R      02
OFFLINE                          = 00000000                     VALIDATE_SCB                       0000033C R      02
OPA$UCB0                           ********   X    02           VCB$B_STATUS2                    = 00000053
PAUSE                              000000F9 R      02           VCB$L_HOMELBN                    = 00000024
PMS$END_IO                         ********   X    02           VCB$L_SBMAPLBN                   = 00000034
PMS$END_RQ                         ********   X    02           VCB$L_SERIALNUM                  = 00000064
PMS$START_RQ                       ********   X    02           VCB$Q_MOUNTTIME                  = 00000090
POOL_OVERHEAD                    = 0000000A                     VCB$S_VOLCKNAM                   = 0000000C
PR$_IPL                          = 00000012                     VCB$S_VOLNAME                    = 0000000C
PREFIX                             00000085 R      03           VCB$T_VOLCKNAM                   = 00000080
PTE$V_VALID                      = 0000001F                     VCB$T_VOLNAME                    = 00000014
QUORUM                             00000569 R      02           VCB$V_MOUNTVER                   = 00000002
QUORUM_LOOP                        00000127 R      02           WRITELOCK                        = 00000002
SAVED_R1                         = 00000000                     WRITLCK_HNDLR                      000003E2 R      02
SAVED_R2                         = 00000004                     WRONGVOL                         = 00000001
SAVED_R3                         = 00000008
SAVED_R4                         = 0000000C
SAVED_R5                         = 00000010
SCB$Q_MOUNTTIME                  = 0000002E
SCB$T_VOLOCKNAME                 = 00000022
SEND_MESSAGE                       00000406 R      02
SS$_INCVOLLABEL                  = 0000010C
SS$_MEDOFL                       = 000001A4
SS$_NORMAL                       = 00000001
SS$_VOLINV                       = 00000254
SS$_WRITLCK                      = 0000025C
SUFFIX                             0000009C R      03
SYS$GL_BOOTUCB                     ********   X    02
SYS$GL_OPRMBX                      ********   X    02
TABLE_INDEX                      = 00000005
TIME_DELAY                         000002A9 R      02
UCB$B_DEVCLASS                   = 00000040
UCB$B_FIPL                       = 0000000B
UCB$L_DDT                        = 00000088
UCB$L_DEVCHAR                    = 00000038
UCB$L_DEVCHAR2                   = 0000003C
UCB$L_IOQFL                      = 0000004C
UCB$L_LINK                       = 00000030
UCB$L_STS                        = 00000064
UCB$L_VCB                        = 00000034
UCB$M_BSY                        = 00000100
UCB$M_CLUTRAN                    = 00200000
UCB$M_MNTVERIP                   = 00004000
UCB$M_MNTVERPND                  = 00080000
UCB$M_SUPMVMSG                   = 00040000
```

```
                                     +----------------+
                                     ! Psect synopsis !
                                     +----------------+

PSECT name                 Allocation            PSECT No.  Attributes
----------                 ----------            ---------  ----------
.  ABS  .                  00000000  (     0.)   00 (   0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                      00000000  (     0.)   01 (   1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
WIONONPAGED                000005B4  (  1460.)   02 (   2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
WMOUNTVERMSG               000000BE  (   190.)   03 (   3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE

                                +------------------------+
                                ! Performance indicators !
                                +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     35    00:00:00.05    00:00:00.73
Command processing                117    00:00:00.37    00:00:01.83
Pass 1                            615    00:00:17.38    00:01:02.24
Symbol table sort                   0    00:00:02.87    00:00:10.35
Pass 2                            311    00:00:04.16    00:00:17.92
Symbol table output                29    00:00:00.14    00:00:00.56
Psect synopsis output               4    00:00:00.02    00:00:00.02
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             1113    00:00:24.99    00:01:34.29

The working set limit was 1800 pages.
151990 bytes (297 pages) of virtual memory were used to buffer the intermediate code.
There were 150 pages of symbol table space allocated to hold 2738 non-local and 57 local symbols.
1716 source lines were read in Pass 1, producing 20 object records in Pass 2.
35 pages of virtual memory were used to define 34 macros.

                               +--------------------------+
                               ! Macro library statistics !
                               +--------------------------+

Macro library name                         Macros defined
------------------                         --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                   20
_$255$DUA28:[SYSLIB]STARLET.MLB;2                10
TOTALS (all libraries)                           30

2833 GETS were required to define 30 macros.

There were no errors, warnings or information messages.

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

0398 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MOUNTVER
LIS

OPDRVWS1
LIS

QUORUM
LIS

OPDRV790
LIS

OPDRIVER
LIS

REBLDLOCK
LIS