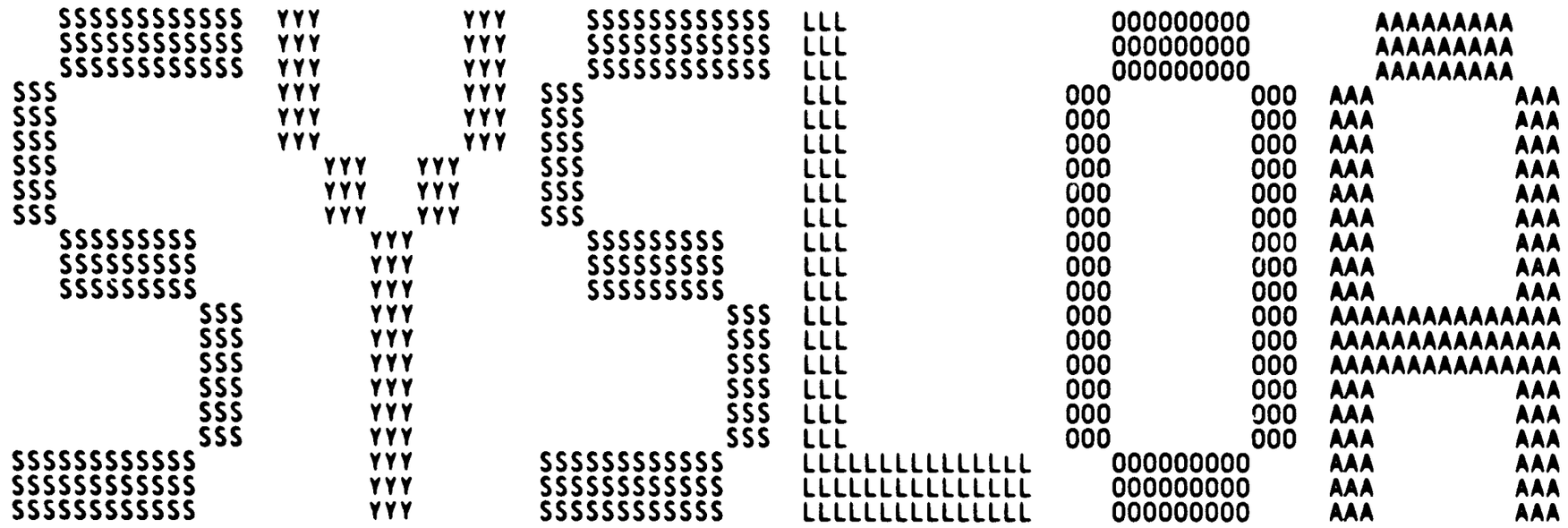
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA

```
MM      MM    000000    UU      UU  NN      NN  TTTTTTTTTT  VV      VV  EEEEEEEEEE  RRRRRRRR
MM      MM    000000    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    000000    UUUUUUUUUU  NN      NN      TT          VV      EEEEEEEEEE  RR      RR      ....
MM      MM    000000    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     1          .TITLE  MOUNTVER - Mount Verification routines
0000     2          .IDENT  'V04-002'
0000     3  ;*****************************************************************************
0000     4  ;*                                                                           *
0000     5  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     6  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     7  ;*  ALL RIGHTS RESERVED.                                                     *
0000     8  ;*                                                                           *
0000     9  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    10  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    11  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    12  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    13  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    14  ;*  TRANSFERRED.                                                              *
0000    15  ;*                                                                           *
0000    16  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    17  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    18  ;*  CORPORATION.                                                              *
0000    19  ;*                                                                           *
0000    20  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    21  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
0000    22  ;*                                                                           *
0000    23  ;*                                                                           *
0000    24  ;*****************************************************************************
0000    25
0000    26  ;++
0000    27  ; Facility:
0000    28  ;
0000    29  ;       Executive I/O system.
0000    30  ;
0000    31  ; Abstract:
0000    32  ;
0000    33  ;       This module contains most of the code necessary to implement
0000    34  ;       mount verification.  Support routines not in this module are
0000    35  ;       contained in other modules in the EXEC, notably IOSUBNPAG.
0000    36  ;
0000    37  ; Environment:
0000    38  ;
0000    39  ;       This code executes in KERNEL mode, at device IPL
0000    40  ;       or higher, and the code must therefore be resident.
0000    41  ;
0000    42  ; Author:
0000    43  ;
0000    44  ;       Steven T. Jeffreys
0000    45  ;
0000    46  ; Creation date:
0000    47  ;
0000    48  ;       June 10, 1981
0000    49  ;
0000    50  ; Update history:
0000    51  ;
0000    52  ;       V04-002 ROW0415         Ralph O. Weber          10-SEP-1984
0000    53  ;               Fix no-quorum branch destination after PAUSE to go to the
0000    54  ;               TIME_DELAY subroutine call.
0000    55  ;
0000    56  ;       V04-001 ROW0414         Ralph O. Weber           6-SEP-1984
0000    57  ;               Rework handling of MVTIMEOUT checking after PAUSE so that
```

```
0000     58 ;               /FOREIGN volumes -- which get into mount verification during
0000     59 ;               cluster state transitions -- will time out mount verification.
0000     60 ;
0000     61 ;       V03-022 ROW0410         Ralph O. Weber          6-AUG-1984
0000     62 ;               Setup use of UCB$V_CLUTRAN to guarantee that mount
0000     63 ;               verification always runs for all disks after a VAXcluster
0000     64 ;               state transition.
0000     65 ;
0000     66 ;       V03-021 ROW0407         Ralph O. Weber          25-JUL-1984
0000     67 ;               Add $GETDVI escape vector place holder, EXE$MNTVER_DVI_ASSIST.
0000     68 ;               Also move EXE$MNTVERSP2 back to RSB; EXE$CLUTRANIO is
0000     69 ;               correctly used by the connection manager now.
0000     70 ;
0000     71 ;       V03-020 ROW0403         Ralph O. Weber          22-JUL-1984
0000     72 ;               Change volume name checking so that if either the volume label
0000     73 ;               or the volume lock name is correct the volume is considered to
0000     74 ;               be correct.  This volume name checking technique is designed
0000     75 ;               to handle the cases where the volume is write locked or
0000     76 ;               becomes write locked such that the volume lock name cannot be
0000     77 ;               written into the SCB.
0000     78 ;
0000     79 ;       V03-019 ROW0385         Ralph O. Weber          7-JUL-1984
0000     80 ;               - Add GET_VCB, a generalized routine which gets the VCB
0000     81 ;                 address when possible and signals its legal absence.
0000     82 ;               - Add EXE$CLUTRANIO symbol which will eventually replace
0000     83 ;                 EXE$MNTVERSP2.
0000     84 ;               - Add symbol for EXE$UPDGNERNUM, the update a shadow set
0000     85 ;                 generation number routine.
0000     86 ;               - Change UCB$V_SUPMVMSG processing to clear that bit if a
0000     87 ;                 message which cannot be suppressed is ever broadcast.
0000     88 ;               - Change GET_BUFFER and FREE_BUFFER to save the address of the
0000     89 ;                 IRP currently using the mount verification work buffer.
0000     90 ;               - Change volume validation algorithm to check volume lock name
0000     91 ;                 stored in the SCB instead of the volume name stored in the
0000     92 ;                 home block.  This prevents incorrect failures in a
0000     93 ;                 VAXcluster where SET VOLUME /LABEL= has be used.
0000     94 ;
0000     95 ;       V03-018 ROW0372         Ralph O. Weber          30-MAY-1984
0000     96 ;               Correct setting of UCB$V_MNTVERIP and UCB$V_MNTVERPND bits in
0000     97 ;               EXE$MNTVERSP2 to have them set in UCB$L_STS not UCB$L_DEVCHAR2.
0000     98 ;
0000     99 ;       V03-017 ROW0366         Ralph O. Weber          18-MAY-1984
0000    100 ;               Cause QUORUM_LOOP to REMQUE the quorum disk IRP before passing
0000    101 ;               it off to the driver for processing.  This is useful because
0000    102 ;               the driver will eventually insert the IRP on some queue and
0000    103 ;               failing to remove it first causes queue tangles.
0000    104 ;
0000    105 ;       V03-016 WMC0004         Wayne Cardoza           15-May-1984
0000    106 ;               Class driver wants QUORLOST bit off after call.
0000    107 ;
0000    108 ;       V03-015 WMC0003         Wayne Cardoza           11-May-1984
0000    109 ;               More VCB fixes.
0000    110 ;
0000    111 ;       V03-014 WMC0002         Wayne Cardoza           10-May-1984
0000    112 ;               Fix problem for mount verification of disk without VCB.
0000    113 ;
0000    114 ;       V03-013 WMC0001         Wayne Cardoza           02-May-1984
```

```
0000   115 ;               Add support for loss of quorum stalling of I/O.
0000   116 ;
0000   117 ;       V03-012 ROW0343         Ralph O. Weber          10-APR-1984
0000   118 ;               Setup usage of UCB$V_SUPMVMSG to actually supress "normal"
0000   119 ;               mount verification messages.
0000   120 ;
0000   121 ;       V03-011 ROW0330         Ralph O. Weber          24-MAR-1984
0000   122 ;               Add EXE$MNTVERSP1 and EXE$MNTVERSP2, two spare mount
0000   123 ;               verification entry points.  Also move EXE$MNTVERSHDOL to a
0000   124 ;               place where its future implementation is less likely to break
0000   125 ;               branch displacements.
0000   126 ;
0000   127 ;       V03-010 ROW0326         Ralph O. Weber          20-MAR-1984
0000   128 ;               Add testing of SCB$Q_MOUNTTIME against VCB$Q_MOUNTTIME to
0000   129 ;               enhance wrong volume detection.  Eliminate mount verification
0000   130 ;               timeout for system disk.
0000   131 ;
0000   132 ;       V03-009 ROW0321         Ralph O. Weber           4-MAR-1984
0000   133 ;               Fix incorrect usage of self-relative queues while waiting for
0000   134 ;               mount verification work page.  Change to using absolute queues.
0000   135 ;
0000   136 ;       V03-008 ROW0314         Ralph O. Weber          28-FEB-1984
0000   137 ;               Setup definition and use of MVMSL structure for the list of
0000   138 ;               messages and other information relivant for using the
0000   139 ;               SEND_MESSAGE routine.
0000   140 ;
0000   141 ;       V03-007 ROW0311         Ralph O. Weber          26-FEB-1984
0000   142 ;               > Change handling of work page based upon use of new mount
0000   143 ;                 verification private work page instead of blackhole page.
0000   144 ;               > Setup no mount verification processing of any internal IRP.
0000   145 ;               > Fix bug in failure to allocate a mount verification IRP
0000   146 ;                 error path.
0000   147 ;               > Eliminate unneeded code in INIT_IRP and SEND_MESSAGE.
0000   148 ;
0000   149 ;       V03-006 ROW0308         Ralph O. Weber          21-FEB-1984
0000   150 ;               Fix message handling to be position independent.
0000   151 ;
0000   152 ;       V03-005 ROW0292         Ralph O. Weber           4-FEB-1984
0000   153 ;               o Make changes necessary to move this module to SYSLOAxxx.
0000   154 ;                 Make all address references position independent.
0000   155 ;               o Restore TIME_DELAY routine, but have it FORK_WAIT using the
0000   156 ;                 CDRP portion of the internal IRP.  Fix GET_BUFFER and
0000   157 ;                 FREE_BUFFER to fork on the CDRP as well.  This makes mount
0000   158 ;                 verfication compatible with the class drivers which fork on
0000   159 ;                 the UCB at random times.
0000   160 ;               o Add replace START_IO label with EXE$MNTVERSIO thus making
0000   161 ;                 that routine accessible to in-driver mount verification
0000   162 ;                 routines.
0000   163 ;               o Define EXE$MNTVERSHDOL as a NOP routine, for now.
0000   164 ;
0000   165 ;       V03-004 TCM0001         Trudy C. Matthews       09-Jun-1983
0000   166 ;               Set up register R4 before calling IOC$CVT_DEVNAM to get
0000   167 ;               node$ddcu name if a remote device and ddcu if a local device.
0000   168 ;
0000   169 ;       V03-003 ROW0177         Ralph O. Weber           5-APR-1983
0000   170 ;               Remove routine TIME_DELAY and convert all its callers to use
0000   171 ;               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    179          .SBTTL  Declarations
0000    180 ;
0000    181 ; Macro definitions
0000    182 ;
0000    183          $CADEF                          ; Define conditional assembly symbols
0000    184          $CDDBDEF                        ; Class driver data block
0000    185          $CDRPDEF                        ; Define Class Driver Req. Pkt. fields
0000    186          $CLUBDEF                        ; Cluster block
0000    187          $CLUDCBDEF                      ; Quorum disk control block
0000    188          $DCDEF                          ; Device classes
0000    189          $DDBDEF                         ; Define Device Data Block fields
0000    190          $DDTDEF                         ; Define Driver Dispatch Table fields
0000    191          $DEVDEF                         ; Define device characteristics bits
0000    192          $DYNDEF                         ; Define block type symbols
0000    193          $FKBDEF                         ; Define Fork Block fields
0000    194          $HM1DEF                         ; Define ODS-1 home block fields
0000    195          $HM2DEF                         ; Define ODS-2 home block fields
0000    196          $IODEF                          ; Define I/O function codes
0000    197          $IRPDEF                         ; Define I/O Request Packet fields
0000    198          $MSGDEF                         ; Define system message type codes
0000    199          $MVMSLDEF                       ; Define MV message list symbols
0000    200          $PRDEF                          ; Define processor registers
0000    201          $PTEDEF                         ; Define Page Table Entry fields
0000    202          $SCBDEF                         ; Define Storage Control Block offsets
0000    203          $SSDEF                          ; Define system status codes
0000    204          $UCBDEF                         ; Define Unit Control Block fields
0000    205          $VCBDEF                         ; Define Volume Control Block fields
0000    206
0000    207 ;+
0000    208 ; BUILD_TABLE
0000    209 ;
0000    210 ;       This macro builds entries in a table of mount verification messages.
0000    211 ;       The table is used to coordinate messages broadcast to OPA0 with
0000    212 ;       messages sent to OPCOM.  It existance allows both kinds of messages to
0000    213 ;       be referenced by a single value, the table index value.
0000    214 ;
0000    215 ;       This table is also used by mount verification routines in various
0000    216 ;       drivers.  The driver-specific mount verification routines are passed
0000    217 ;       the base address of the table.  From this they can determine the
0000    218 ;       address of the SEND_MESSAGE routine, and by scanning the table, they
0000    219 ;       can determine the table index to supply to SEND_MESSAGE.
0000    220 ;
0000    221 ; Parameters:
0000    222 ;
0000    223 ;       index   symbol to be assigned the index value for a given message
0000    224 ;       opcom   OPCOM message number for a given message
0000    225 ;       text    OPA0 broadcast text for a given message
0000    226 ;       flags   bits giving special properties of a given message
0000    227 ;               the allowable flags are:
0000    228 ;                 MVMSL$M_NOSUFFIX      OPA0 broadcast should not include
0000    229 ;                                         "Mount verification in progress."
0000    230 ;                 MVMSL$M_SUPRESS       message sending can be supressed by
0000    231 ;                                         UCB$V_SUPMVMSG
0000    232 ;--
0000    233
0000    234          .MACRO  BUILD_TABLE index, opcom, text, flags=0, ?l1
0000    235          .IIF    NDF TABLE_INDEX, TABLE_INDEX=0
```

```
                                                L 2
                     0000   236          .ENABLE LSB
                     0000   237          .SAVE
                     0000   238          .PSECT  WMOUNTVERMSG
                     0000   239 l1:       .ASCIC  text
                     0000   240          .RESTORE
                     0000   241          .WORD   opcom
                     0000   242          .WORD   flags
                     0000   243          .ADDRESS l1-EXE$AB_MVMSLBAS
                     0000   244          .DISABLE LSB
                     0000   245          index = TABLE_INDEX
                     0000   246          TABLE_INDEX = TABLE_INDEX + 1
                     0000   247          .ENDM   BUILD_TABLE
                     0000   248 ;
                     0000   249 ; Own storage (read only)
                     0000   250 ;
                     0000   251
                 00000000   252          .PSECT  WIONONPAGED LONG
                     0000   253
                     0000   254 ; The following is the MVMSL passed to driver specific mount verification
                     0000   255 ; support routines.
                     0000   256
                     0000   257          ASSUME  MVMSL$L_SNDMSGOFF EQ -4
                     0000   258          ASSUME  MVMSL$W_MSG_CODE  EQ  0
                     0000   259          ASSUME  MVMSL$W_FLAGS     EQ  2
                     0000   260          ASSUME  MVMSL$L_TEXTOFF   EQ  4
                     0000   261          ASSUME  MVMSL$K_LENGTH    EQ  8
                     0000   262
            00000402'  0000   263          .ADDRESS SEND_MESSAGE - EXE$AB_MVMSLBAS ; Offset to SEND_MESSAGE
                     0004   264 EXE$AB_MVMSLBAS::
                     0004   265          BUILD_TABLE     index=OFFLINE, opcom=MSG$_DEVOFFLINX, -
                     0004   266                          flags=MVMSL$M_SUPRESS, -
                     0004   267                          text=<" is offline.">
                     000C   268
                     000C   269          BUILD_TABLE     index=WRONGVOL, opcom=MSG$_WRONGVOL, -
                     000C   270                          text=<" contains the wrong volume.">
                     0014   271
                     0014   272          BUILD_TABLE     index=WRITELOCK, opcom=MSG$_DEVWRTLCK, -
                     0014   273                          text=<" has been write-locked.">
                     001C   274
                     001C   275          BUILD_TABLE     index=MVCOMPLETE, opcom=MSG$_MVCOMPLETE, -
                     001C   276                          flags=<MVMSL$M_NOSUFFIX ! MVMSL$M_SUPRESS>, -
                     001C   277                          text=<" has completed mount verification.">
                     0024   278
                     0024   279          BUILD_TABLE     index=MVABORTED, opcom=MSG$_MVABORTED, -
                     0024   280                          flags=MVMSL$M_NOSUFFIX, -
                     0024   281                          text=<" has aborted mount verification.">
                     002C   282 ;
                     002C   283 ; The following message text is part of every message, and does not
                     002C   284 ; require an entry in the message table.
                     002C   285 ;
                     002C   286
                 00000085   287          .PSECT  WMOUNTVERMSG
49 2D 4D 45 54 53 59 53 25 07 07 00'  0085   288 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   289 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    290
                                     00BE    291 ;
                                     00BE    292 ; The following literal is used as carriage control
                                     00BE    293 ; in the message sent to the operator's console.
                                     00BE    294 ;
                           00000A0D  00BE    295 CAR_CTRL        = ^X0A0D                ; <cr><lf> characters
                           00000002  00BE    296 CAR_CTRL_SIZE   = 2                     ; Size in bytes
```

```
00BE   298          .SBTTL  EXE$MOUNTVER - initial entry point
00BE   299 ;++
00BE   300 ; EXE$MOUNTVER
00BE   301 ;
00BE   302 ; Functional description:
00BE   303 ;
00BE   304 ;       Mount verification is the mechanism whereby Files-11 volumes are
00BE   305 ;       are brought back online after a catastrophic (but hopefully transient)
00BE   306 ;       hardware problem has rendered the volume unusable.
00BE   307 ;
00BE   308 ;       This is the main routine, and initial entry point of the
00BE   309 ;       mount verification code.  For a detailed discussion of
00BE   310 ;       the implementation, please see the section entitled "Design Notes".
00BE   311 ;
00BE   312 ; Input:
00BE   313 ;
00BE   314 ;       R0,R1     = I/O status
00BE   315 ;       R2        = scratch
00BE   316 ;       R3        = IRP address
00BE   317 ;       R4        = scratch
00BE   318 ;       R5        = UCB address
00BE   319 ;       0(SP)     = return address of caller
00BE   320 ;       4(SP)     = return address of caller's caller
00BE   321 ;
00BE   322 ; Output:
00BE   323 ;
00BE   324 ;       None.
00BE   325 ;
00BE   326 ; Side Effects:
00BE   327 ;
00BE   328 ;       Providing that whatever external event has caused a given
00BE   329 ;       volume to undergo mount verification has been corrected,
00BE   330 ;       normal I/O activity will resume on the device.  Otherwise,
00BE   331 ;       the volume will do no useful work until mount verification
00BE   332 ;       is complete, and will appear to be hung.
00BE   333 ;
00BE   334 ; Design Notes:
00BE   335 ;
00BE   336 ;       Mount verification (MV for short) is only done for FILES-11
00BE   337 ;       disk volumes.  Tapes are already covered by the magnetic
00BE   338 ;       tape ACP, and handling foreign disk volumes would open a
00BE   339 ;       security hole that would be difficult to close.
00BE   340 ;
00BE   341 ;       There are two error conditions that can cause a volume to
00BE   342 ;       undergo mount verification.  The first arises when a volume
00BE   343 ;       is somehow hardware writelocked sometime after it was mount
00BE   344 ;       write-enabled.  (The writelock recovery mechanism is covered
00BE   345 ;       in detail in the funtional description of the WRITLCK_HNDLR
00BE   346 ;       routine, and will not be covered here.)
00BE   347 ;
00BE   348 ;       The second, and more serious, situation is the result of the
00BE   349 ;       volume being declared software invalid.  The volume-valid
00BE   350 ;       bit in the volume's UCB is cleared by the driver when it
00BE   351 ;       detects a situation that warrants such action, such as the
00BE   352 ;       device issuing an ONLINE interrupt.  This implies that the
00BE   353 ;       volume has been spun down, and then spun back up, so the
00BE   354 ;       contents of the drive may not be the same.  It is MV's task
```

```
00BE    355 ;       to notify the operator of the situation, and to verify that
00BE    356 ;       the volume (if any) now in the drive is the same one that
00BE    357 ;       was there originally.
00BE    358 ;
00BE    359 ;       What follows is a series of one-liners and short paragraphs
00BE    360 ;       that are meant to explain various design decisions, and
00BE    361 ;       to explain obscure parts of the code.  There is no particular
00BE    362 ;       order to the notes.
00BE    363 ;
00BE    364 ;       The cell EXE$GL_SVAPTE maps the MV work page into S0 space,
00BE    365 ;       and is set up by INIT at system boot time.  MV used to use
00BE    366 ;       the blackhole page for its I/O operations.  However, the
00BE    367 ;       read-modify-write operations performed by shadow set state
00BE    368 ;       change processing prohibit this.  Permanent provision of a
00BE    369 ;       512 byte buffer for MV is preferable to dynamic allocation
00BE    370 ;       of the space because a pileup of MV requests might make
00BE    371 ;       allocation of the needed space impossible.
00BE    372 ;
00BE    373 ;       Broadcasting a message directly to the operator's console
00BE    374 ;       is done to ensure that someone is notified of MV being in
00BE    375 ;       progess.  OPCOM cannot be relied on since it may not be
00BE    376 ;       present, or the very nature of the problem may prevent it
00BE    377 ;       from operating (eg.  OPCOM is swapped out, and the system
00BE    378 ;       disk is undergoing MV).
00BE    379 ;
00BE    380 ;       The code is optimized for size over speed, since the code
00BE    381 ;       is infrequently executed, but must be resident.
00BE    382 ;
00BE    383 ;       Once MV starts, normal I/O activity on the device ceases until
00BE    384 ;       MV completes or the volume is dismounted.  Unfortunately, there
00BE    385 ;       is a deadlock problem with the ACP, because to dismount a volume
00BE    386 ;       an ACP I/O request must be done to the volume, and that can't
00BE    387 ;       happen since we're in MV, so the ACP hangs waiting for the I/O
00BE    388 ;       to complete.  Worse, requests will pile up at the ACP, and
00BE    389 ;       other processes will be hung by the hung ACP.  The situation
00BE    390 ;       clears itself up nicely when MV completes.
00BE    391 ;
00BE    392 ;       MV consists of the code in this module, support routines in
00BE    393 ;       other modules, and a driver-dependent piece of code that
00BE    394 ;       is pointed to by the driver's DDT.  The covention is that
00BE    395 ;       if R3 is nonzero, the routine should assume that R3 points
00BE    396 ;       to an IRP and requeue the IRP.  If R3 is zero, dequeue the
00BE    397 ;       first IRP and resume normal I/O activity.  The cell in the
00BE    398 ;       DDT is DDT$L_MNTVER, and it defaults to IOC$MNTVER.
00BE    399 ;
00BE    400 ;       MV is entered by the common I/O completion routine, IOC$REQCOM.
00BE    401 ;       This may be different for DISK_CLASS drivers.
00BE    402 ;--
```

```
                     0000002C   404          .PSECT  WIONONPAGED
                         002C   405          .ENABL LSB
                         002C   406
                         002C   407 EXE$MOUNTVER::                          ; Initial entry point
                         002C   408 ;
                         002C   409 ; See if an IRP was supplied.  If not, skip all the checks and I/O cleanup.
                         002C   410 ;
                   53    D5  002C   411          TSTL    R3
                   58    13  002E   412          BEQL    35$
                         0030   413 ;
                         0030   414 ; Determine if mount verification is possible and necessary.
                         0030   415 ;
                         0030   416          ;
                         0030   417          ; Ignore all internal IRPs (including MV IRPs).
                         0030   418          ;
                0C A3    D5  0030   419          TSTL    IRP$L_PID(R3)           ; Is this an internal IRP?
                   47    19  0033   420          BLSS    10$                     ; Branch to exit if internal IRP.
                         0035   421          ;
                         0035   422          ; The device must be file oriented, and contain a mounted
                         0035   423          ; volume that is not mounted /FOREIGN.  In addition, the volume
                         0035   424          ; must have a VCB with the
                         0035   425          ; The one time this is not true is if this is a cluster that is out
                         0035   426          ; out of quorum.  In this case, we must go into mount verification in
                         0035   427          ; order to stall all I/O
                         0035   428          ;
                      0E E1  0035   429          BBC     #DEV$V_FOD,-            ; Branch if device not file oriented
                42 38 A5         0037   430                  UCB$L_DEVCHAR(R5),10$ ;
                      05 E0  003A   431          BBS     #DEV$V_SQD,-            ; Is this a sequential device?
                3D 38 A5         003C   432                  UCB$L_DEVCHAR(R5),10$ ;
                    0527 30  003F   433          BSBW    QUORUM                  ; Is it out-of-quorum cluster
                   38 54 E8  0042   434          BLBS    R4,20$                  ; Yes
             33 64 A5   15 E4  0045   435          BBSC    #UCB$V_CLUTRAN, -       ; Branch if here due to a VAXcluster
                               004A   436                  UCB$L_STS(R5), 20$      ; state transition (and clear flag).
                      13 E1  004A   437          BBC     #DEV$V_MNT,-            ; Branch if device not mounted
                2D 38 A5         004C   438                  UCB$L_DEVCHAR(R5),10$ ;
                      18 E0  004F   439          BBS     #DEV$V_FOR,-            ; Is this a foreign device?
                28 38 A5         0051   440                  UCB$L_DEVCHAR(R5),10$ ;
                               0054   441          ;
                               0054   442          ; Check the VCB$V_MOUNTVER bit to ensure the volume
                               0054   443          ; is a candidate for mount verification.
                               0054   444          ;
                54    34 A5  D0  0054   445          MOVL    UCB$L_VCB(R5),R4        ; Get the VCB address
                         22 13  0058   446          BEQL    10$                     ; Exit if none
                         02 E1  005A   447          BBC     #VCB$V_MOUNTVER,-       ; Exit if bit not set
                1D 53 A4         005C   448                  VCB$B_STATUS2(R4),10$ ;
                               005F   449          ;
                               005F   450          ; If the medium is offline, or the volume is
                               005F   451          ; invalid, the error can be recovered from.
                               005F   452          ;
                50   01A4 8F  B1  005F   453          CMPW    #SS$_MEDOFL,R0          ; Is the media (disk volume) offline?
                         17 13  0064   454          BEQL    20$                     ; Branch if true
                50   0254 8F  B1  0066   455          CMPW    #SS$_VOLINV,R0          ; Is the volume invalid?
                         10 13  006B   456          BEQL    20$                     ; Branch if true
                               006D   457          ;
                               006D   458          ; If the volume has been writelocked, make sure that it was
                               006D   459          ; an accidental writelock.  If the software writelock bit is
                               006D   460          ; on, then the volume was mounted with the volume write protected.
```

```
                     006D   461          ; If the bit is not set, then the volume has been mounted for
                     006D   462          ; read/write access, and has since been (accidentally) write protected.
                     006D   463          ;
50   025C 8F    B1   006D   464          CMPW    #SS$_WRITLCK,R0          ; Is the device writelocked?
          08    12   0072   465          BNEQ    10$                      ; Branch if not
          19    E0   0074   466          BBS     #DEV$V_SWL,-             ; Branch if software writelocked
     03 38 A5        0076   467                  UCB$L_DEVCHAR(R5),10$    ;
        0366    31   0079   468          BRW     WRITLCK_HNDLR            ; Recover from accidental writelock
                05   007C   469 10$:     RSB                              ; Mount verifcation is not called for - exit
                     007D   470 ;
                     007D   471 ; The error can be recovered from.
                     007D   472 ;
          0D    E5   007D   473 20$:     BBCC    #IRP$V_MVIRP,-           ; Clear the MV bit in the IRP.
     00 2A A3        007F   474                  IRP$W_STS(R3),30$        ;
        0342    30   0082   475 30$:     BSBW    CLEANUP_IO               ; Clean up the I/O operation
                     0085   476          ;
                     0085   477          ; Perform any driver-specific initialization for mount verification.
                     0085   478          ; R5 points to the device UCB, and R3 points to the IRP.
                     0085   479          ;
        0352    30   0085   480          BSBW    DRIVER_CODE
          8E    D5   0088   481 35$:     TSTL    (SP)+                    ; Discard the return address
                     008A   482 ;
                     008A   483 ; This entry point is used to start mount verification without having an error
                     008A   484 ; IRP
                     008A   485 ;
                     008A   486 MNTVER_NOIRP:
          0E    E2   008A   487          BBSS    #UCB$V_MNTVERIP,-        ; Set mount verification in progress
     ED 64 A5        008C   488                  UCB$W_STS(R5),10$        ; (return to caller's caller if already set)
                     008F    9 ;
                     008F   490 ; Inform interested parties that the device needs attention.
                     008F   491 ;
          54    00   D0   008F   492          MOVL    #OFFLINE,R4              ; Set message code
        0371    30   0092   493          BSBW    SEND_MESSAGE             ; Send message
                     0095   494 ;
                     0095   495 ; Allocate an IRP.  If non exists, exit.
                     0095   496 ;
     51   00C4 8F    3C   0095   497          MOVZWL  #IRP$K_LENGTH,R1         ; Set IRP size
      00000000'GF    16   009A   498          JSB     G^EXE$ALONONPAGED        ; Allocate an IRP
          53   52    D0   00A0   499          MOVL    R2,R3                    ; Copy IRP address
        05 50    E8   00A3   500          BLBS    R0,40$                   ; Branch if success
           53    D4   00A6   501          CLRL    R3                       ; Signal the no IRP was allocated.
        0193    31   00A8   502          BRW     ERROR_EXIT               ; Exit if no IRP available
                     00AB   503 ;
                     00AB   504 ; Calculate the maximum time for which we will
                     00AB   505 ; continue to attempt mount verification.  Store
                     00AB   506 ; the value in the IRP.
                     00AB   507 ;
     50   00000000'GF    3C   00AB   508 40$:     MOVZWL  G^IOC$GW_MVTIMEOUT,R0    ; Get delta time
14 A3 50   00000000'GF    C1   00B2   509          ADDL3   G^EXE$GL_ABSTIM,R0,-     ; Add site-specific delta time to current ti
                     00BB   510                  IRP$L_ASTPRM(R3)         ; Fall through...
                     00BB   511
                     00BB   512 ;
                     00BB   513 ; Decide whether or not this is a cluster that is out of quorum.
                     00BB   514 ; From this point on the meaning of R4 is:
                     00BB   515 ;       0 -> normal mount verification
                     00BB   516 ;       1 -> out of quorum
                     00BB   517 ;       -1-> out of quorum and pack-ack completed
```

```
                                E 3
                                00BB    518 ;
                        04AB    30  00BB    519         BSBW    QUORUM                  ; Get indicator in R4
                                00BE    520
                                00BE    521         .DSABL  LSB
                                00BE    522
                                00BE    523 BUILD_PACKACK_IRP:
                        0202    30  00BE    524         BSBW    INIT_IRP                ; Set request-independent fields in the IRP
                                00C1    525         ;
                                00C1    526         ; Determine if the device supports a PACKACK function.
                                00C1    527         ; If it does, then issue a PACKACK request, else attempt
                                00C1    528         ; to read the volume's home block.
                                00C1    529         ;
                          0B    E2  00C1    530         BBSS    #UCB$V_VALID,-          ; Set volume valid
                     00 64 A5           00C3    531                 UCB$W_STS(R5),10$       ;
                51    0088 C5    D0    00C6    532 10$:    MOVL    UCB$L_DDT(R5),R1        ; Get address of DDT
                  51    08 A1    D0    00CB    533         MOVL    DDT$L_FDT(R1),R1        ; Get address of FDT masks
                      61    08    E1  00CF    534         BBC     #IO$_PACKACK,(R1),-     ; Branch if PACKACK not supported
                            16          00D2    535                 30$                     ;
                                00D3    536         ;
                                00D3    537         ; Set the request dependent fields in the IRP and start the I/O.
                                00D3    538         ;
                                00D3    539 20$:    ASSUME  IO$_PHYSICAL GE IO$_PACKACK
                        B0  00D3    540         MOVW    #<IRP$M_PHYSIO  ! -     ; Set physical I/O function
                                00D4    541                   IRP$M_MVIRP>,-        ; Mark this a mount verification IRP
                2A A3   2100 8F     00D4    542                 IRP$W_STS(R3)           ;
                        0808 8F     B0  00D9    543         MOVW    #<IO$_PACKACK ! IO$M_INHERLOG>,-
                          20 A3             00DD    544                 IRP$W_FUNC(R3)          ; Set function code, inhibit error logging
                           02B5     30  00DF    545         BSBW    EXE$MNTVERSIO           ; Start I/O request
                                00E2    546 ;
                                00E2    547 ; When the PACKACK I/O is done, control returns here.
                                00E2    548 ;
                         0484   30  00E2    549         BSBW    QUORUM                  ; Is it out-of-quorum cluster
                        38 A3   E9  00E5    550         BLBC    IRP$L_MEDIA(R3),-       ; If failure, try again
                            10          00E8    551                 PAUSE                   ;
                      54    54    CE  00E9    552 30$:    MNEGL   R4,R4                   ; Indicate pack-ack succeeded
                            39    12  00EC    553         BNEQ    QUORUM_LOOP             ; No quorum
                         0491   30  00EE    554         BSBW    GET_VCB                 ; Check for legally absent VCB.
                           03   13  00F1    555         BEQL    39$                     ; Branch if VCB legally absent.
                         00AC   31  00F3    556         BRW     BUILD_RDHOME_IRP        ; Otherwise, go check the volume.
                         0107   31  00F6    557 39$:    BRW     NORMAL_EXIT             ; Branch assist.
                                00F9    558
                                00F9    559 PAUSE:  ;
                                00F9    560         ; If out-of-quorum, skip all checks and just wait a while.  Else,
                                00F9    561         ; check for MVTIMEOUT expired on this request or VCB$V_MOUNTVER
                                00F9    562         ; clear (indicate abort mount verification).  If mount verification
                                00F9    563         ; still ok, wait.  Else, abort.
                                00F9    564         ;
                      20 54     E8  00F9    565         BLBS    R4, 15$                 ; If no quorum, skip checks.
                         0483   30  00FC    566         BSBW    GET_VCB                 ; Get VCB address.
            14 A3   00000000'GF     D1  00FF    567         CMPL    G^EXE$GL_ABSTIM, -      ; Have we run out of time?
                                0107    568                 IRP$L_ASTPRM(R3)
                          0A    1F  0107    569         BLSSU   10$                     ; Branch if we still have time.
                          50    D5  0109    570         TSTL    R0                      ; Out of time.  Is there a VCB?
                          17    13  010B    571         BEQL    20$                     ; Branch if no VCB to update.
                    53 A0   04    8A  010D    572         BICB    #<1@VCB$V_MOUNTVER>, -  ; Else, disable mount verification.
                                0111    573                 VCB$B_STATUS2(R0)
                          11    11  0111    574         BRB     20$                     ; Then abort.
```

```
                50   D5  0113   575 10$:    TSTL    R0                          ; Still have time.  Is there a VCB?
                05   13  0115   576         BEQL    15$                         ; Branch if no VCB to check.
        08 53 A0    02   E1  0117   577         BBC     #VCB$V_MOUNTVER, -          ; Abort MV if MV is now disabled.
                                011C   578                 VCB$B_STATUS2(R0), 20$
                018A   30  011C   579 15$:    BSBW    TIME_DELAY                  ; Pause for a bit
              05 54    E8  011F   580         BLBS    R4,QUORUM_LOOP              ; We are waiting for quorum
                 9A    11  0122   581         BRB     BUILD_PACKACK_IRP           ; Retry the packack
                0117   31  0124   582 20$:    BRW     ERROR_EXIT                  ; Exit
                                0127   583 ;
                                0127   584 ; Check for quorum IRP and issue it if found.
                                0127   585 ; Wait a while longer if still no quorum
                                0127   586 ;
                                0127   587 QUORUM_LOOP:
    51   00000000'GF    D0  0127   588         MOVL    G^CLU$GL_CLUB,R1
     51    00B4 C1    D0  012E   589         MOVL    CLUB$L_CLUDCB(R1),R1        ; Get quorum disk control block
                50   13  0133   590         BEQL    20$                         ; None
       51    10 A1    D0  0135   591         MOVL    CLUDCB$L_IRP(R1),R1         ; Quorum IRP
                4A   13  0139   592         BEQL    20$                         ; None
       50    4C A5    9E  013B   593         MOVAB   UCB$L_IOQFL(R5),R0          ; IO queue
          52    50    D0  013F   594         MOVL    R0,R2
          52    62    D0  0142   595 10$:    MOVL    (R2),R2                     ; Next IRP
          52    50    D1  0145   596         CMPL    R0,R2
                3B   13  0148   597         BEQL    20$                         ; End of queue
          51    52    D1  014A   598         CMPL    R2,R1                       ; Is this the quorum IRP
                F3   12  014D   599         BNEQ    10$                         ; Not an interesting IRP
          52    62    0F  014F   600         REMQUE  (R2),R2                     ; Dequeue the quorum IRP
    14 3C A5    05   E1  0152   601         BBC     #DEV$V_MSCP,UCB$L_DEVCHAR2(R5),15$ ; Standard disk
                0D   E2  0157   602         BBSS    #IRP$V_MVIRP,-              ; Let the I/O go through
             00 2A A2            0159   603                 IRP$W_STS(R2),12$       ; pretend it is MV IRP
                38   BB  015C   604 12$:    PUSHR   #^M<R3,R4,R5>
          53    52    D0  015E   605         MOVL    R2,R3                       ; Quorum IRP
       00000000'GF    16  0161   606         JSB     G^IOC$INITIATE              ; Go start the I/O
                38   BA  0167   607         POPR    #^M<R3,R4,R5>
                1A   11  0169   608         BRB     20$                         ; Continue
                                016B   609 ;
 64 A5   00080000 8F    C8  016B   610 15$:    BISL    #UCB$M_MNTVERPND,UCB$L_STS(R5) ; Get control back after I/O
             50   53    D0  0173   611         MOVL    R3,R0                       ; Mount verification IRP
       00000000'GF    16  0176   612         JSB     G^COM$DRVDEALMEM            ; Free it
             53   52    D0  017C   613         MOVL    R2,R3                       ; Quorum IRP
       00000000'GF    17  017F   614         JMP     G^IOC$INITIATE              ; Go do I/O
                                0185   615 ;
    52   00000000'GF    D0  0185   616 20$:    MOVL    G^CLU$GL_CLUB,R2
       0A 1C A2    1C   E0  018C   617         BBS     #CLUB$V_QUORUM,CLUB$L_FLAGS(R2),30$ ; At last we have quorum
                54   D5  0191   618         TSTL    R4
                03   19  0193   619         BLSS    25$                         ; Go wait some more
              FF26   31  0195   620         BRW     BUILD_PACKACK_IRP           ; Pack-ack not done yet
              FF5E   31  0198   621 25$:    BRW     PAUSE
                54   D4  019B   622 30$:    CLRL    R4                          ; Indicate we now have quorum
              03E2   30  019D   623         BSBW    GET_VCB                     ; Check for legally absent VCB.
                5E   13  01A0   624         BEQL    NORMAL_EXIT                 ; Branch if VCB is legally absent.
                                01A2   625                                         ; Else, test volume validity.
                                01A2   626
                                01A2   627 ;
                                01A2   628 ; Validate that the correct volume is still in the drive.
                                01A2   629 ;
                                01A2   630 ;       The validation is performed in two steps.  First the home block is
                                01A2   631 ;       read and checked against fields stored in the VCB.  If that test
```

```
                          01A2    632 ;       succeeds, the storage control block is read and checked against fields
                          01A2    633 ;       stored in the VCB.  If either test fails, wrong volume mount
                          01A2    634 ;       verification is entered.
                          01A2    635 ;
                          01A2    636 BUILD_RDHOME_IRP:
                          01A2    637
            00B4    30    01A2    638         BSBW    GET_BUFFER              ; Obtain ownership of MV work buffer.
                          01A5    639
                          01A5    640         ; Read and validate the home block.
                          01A5    641
            011B    30    01A5    642         BSBW    INIT_IRP                ; Initialize IRP.
      50  34 A5    D0    01A8    643         MOVL    UCB$L_VCB(R5), R0       ; Get VCB address.
              03    12    01AC    644         BNEQ    10$
            008B    31    01AE    645         BRW     FREE_BUF_ERRXIT         ; Branch if no VCB address.
      50  24 A0    D0    01B1    646 10$:    MOVL    VCB$L_HOMELBN(R0), R0   ; Get home block LBN.
            012B    30    01B5    647         BSBW    INIT_IRP_READ           ; Setup IRP for home block read.
            01DC    30    01B8    648         BSBW    EXE$MNTVERSIO           ; Do the read.
            03AB    30    01BB    649         BSBW    QUORUM                  ; Is it out-of-quorum cluster
          04 54     E9    01BE    650         BLBC    R4,20$                  ; Yes
             50     D4    01C1    651         CLRL    R0                      ; Error status
             54     11    01C3    652         BRB     VALIDATE_FAILED         ; WE don't want to succeed without quorum
                          01C5    653 20$:
        50 38 A3    E9    01C5    654         BLBC    IRP$L_MEDIA(R3), -      ; Branch if error occured on the read.
                          01C9    655                 VALIDATE_FAILED
            0137    30    01C9    656         BSBW    VALIDATE_HOME           ; Validate the home block.
          4A 50     E9    01CC    657         BLBC    R0, VALIDATE_FAILED     ; Branch if home blk. validation failed.
                          01CF    658
                          01CF    659         ; Read and validate the storage control block.
                          01CF    660
            00F1    30    01CF    661         BSBW    INIT_IRP                ; Initialize IRP.
      50  34 A5    D0    01D2    662         MOVL    UCB$L_VCB(R5), R0       ; Get VCB address.
              64    13    01D6    663         BEQL    FREE_BUF_ERRXIT         ; Branch if no VCB address.
50    34 A0   01    C3    01D8    664         SUBL3   #1, -                   ; Get storage control block LBN.
                          01DD    665                 VCB$L_SBMAPLBN(R0), R0
            0103    30    01DD    666         BSBW    INIT_IRP_READ           ; Setup IRP for home block read.
            01B4    30    01E0    667         BSBW    EXE$MNTVERSIO           ; Do the read.
            0383    30    01E3    668         BSBW    QUORUM                  ; Is it out-of-quorum cluster
          04 54     E9    01E6    669         BLBC    R4,30$                  ; Yes
             50     D4    01E9    670         CLRL    R0                      ; Error status
             2C     11    01EB    671         BRB     VALIDATE_FAILED         ; WE don't want to succeed without quorum
                          01ED    672 30$:
        28 38 A3    E9    01ED    673         BLBC    IRP$L_MEDIA(R3), -      ; Branch if error occured on the read.
                          01F1    674                 VALIDATE_FAILED
            0148    30    01F1    675         BSBW    VALIDATE_SCB            ; Validate the storage block.
          22 50     E9    01F4    676         BLBC    R0, VALIDATE_FAILED     ; Branch if SCB validation failed.
                          01F7    677
                          01F7    678         ; Mount verification has succeeded.
                          01F7    679         ; Release the MV work buffer, inform the world that this volume has
                          01F7    680         ; returned to the living, and drop through to a normal exit.
                          01F7    681
            0089    30    01F7    682         BSBW    FREE_BUFFER             ; Release MV work buffer.
          54  03    D0    01FA    683         MOVL    #MVCOMPLETE, R4         ; Set success message code.
             0206   30    01FD    684         BSBW    SEND_MESSAGE            ; Signal the world.
                          0200    685
                          0200    686 ;
                          0200    687 ; This is the common exit path for mount verification.
                          0200    688 ; Return all resources, clear MNTVERIP, and resume normal
```

```
                                0200   689 ; I/O activity for the device.
                                0200   690 ;
                                0200   691 NORMAL_EXIT:
                     50   53    D0  0200   692          MOVL    R3,R0                   ; Copy IRP address
                          06    13  0203   693          BEQL    10$                     ; Branch if no IRP was allocated.
                00000000'GF     16  0205   694          JSB     G^COM$DRVDEALMEM        ; Deallocate the IRP
             64 A5  0004C000 8F   CA  020B   695 10$:     BICL    #<UCB$M_MNTVERIP ! -    ; Clear MNTVERIP and WRONGVOL
                                     0213   696                    UCB$M_WRONGVOL ! -    ; and SUPMVMSG
                                     0213   697                    UCB$M_SUPMVMSG>, -    ;
                                     0213   698                    UCB$L_STS(R5)         ;
                          53    D4  0213   699          CLRL    R3                      ; Clear pointer to IRP
                        01C2    30  0215   700          BSBW    DRIVER_CODE             ; Do driver-specific clean-up
                                05  0218   701          RSB                             ; Exit
                                    0219   702 ;
                                    0219   703 ; Attempt to recover from an error encountered while reading
                                    0219   704 ; the home block or while validating the volume.
                                    0219   705 ;
                                    0219   706 VALIDATE_FAILED:
                          68    10  0219   707          BSBB    FREE_BUFFER             ; Release MV work buffer.
                    50    010C 8F   B1  021B   708          CMPW    #SS$_INCVOLLABEL,R0     ; Is this the correct volume?
                          08    13  0220   709          BEQL    20$                     ; Branch if not
                          0F    E4  0222   710          BBSC    #UCB$V_WRONGVOL,-       ; Clear WRONGVOL bit
                       00 64 A5       0224   711                  UCB$W_STS(R5),10$       ;
                           FECF   31  0227   712 10$:     BRW     PAUSE                   ; Retry mount verification
                                      022A   713          ;
                                      022A   714          ; The wrong volume is in the drive.  If this is the first time
                                      022A   715          ; for this volume, inform all interested parties of the event.
                                      022A   716          ;
                          0F    E2  022A   717 20$:     BBSS    #UCB$V_WRONGVOL,-       ; Branch if not the first time
                    F8 64 A5       022C   718                  UCB$W_STS(R5),10$       ;
                          54    DD  022F   719          PUSHL   R4                      ; Save quorum indicator
                    54    01    D0  0231   720          MOVL    #WRONGVOL,R4            ; Set message code
                       01CF   30  0234   721          BSBW    SEND_MESSAGE            ; Send message to console
                          54 8ED0  0237   722          POPL    R4
                          EB    11  023A   723          BRB     10$                     ; Try again
                                    023C   724 ;
                                    023C   725 ; This is the error exit path for mount verification.
                                    023C   726 ; Since the operation never completed, clear volume valid
                                    023C   727 ; to prevent the volume from being used.
                                    023C   728 ;
                                    023C   729 FREE_BUF_ERRXIT:
                          45    10  023C   730          BSBB    FREE_BUFFER             ; Free MV work buffer, then error exit.
                                    023E   731 ERROR_EXIT:
                    54    04    D0  023E   732          MOVL    #MVABORTED,R4           ; Set message code
                       01C2   30  0241   733          BSBW    SEND_MESSAGE            ; Send message to console
                          0B    E5  0244   734          BBCC    #UCB$V_VALID,-          ; Clear volume valid and join common code
                       64 A5          0246   735                  UCB$W_STS(R5),-         ;
                          B7          0248   736                  NORMAL_EXIT             ;
                          B5    11  0249   737          BRB     NORMAL_EXIT             ; Branch to common exit code
                                    024B   738
```

```
                                                I 3
                     024B   740          .SBTTL  GET_BUFFER - allocate an I/O buffer
                     024B   741 ;++
                     024B   742 ; GET_BUFFER
                     024B   743 ;
                     024B   744 ; Functional description:
                     024B   745 ;
                     024B   746 ;       This routine will attempt to allocate the MV work page
                     024B   747 ;       for tc the caller.  If the page is not busy, the page is
                     024B   748 ;       marked busy and control returns to the caller.  If the page
                     024B   749 ;       is busy, the caller's context is folded up into the CDRP
                     024B   750 ;       attached to the input internal IRP and the CDRP is put on
                     024B   751 ;       the blakhole page wait queue.  When the wait queue entry is
                     024B   752 ;       processed, input context is restored and control is return
                     024B   753 ;       to the caller, who now owns the MV work page.
                     024B   754 ;
                     024B   755 ; Input:
                     024B   756 ;
                     024B   757 ;       R3      = Mount verification IRP address (CDRP used as fork block)
                     024B   758 ;       R5      = Device UCB address
                     024B   759 ;       (SP)    = Return address for caller
                     024B   760 ;      4(SP)    = Return address for caller's caller
                     024B   761 ;
                     024B   762 ; Output:
                     024B   763 ;
                     024B   764 ;       R3..R5  are preserved.
                     024B   765 ;--
                     024B   766
                     024B   767 ; Define cells used to control access to MV work page.
                     024B   768
                     024B   769          .ALIGN  LONG
            00000000 024C   770 EXE$AL_WRKWQFL: .LONG 0                     ; Work page wait queue.
            00000000 0250   771 EXE$AL_WRKWQBL: .LONG 0
            00000000 0254   772 EXE$AL_BSYIRP: .LONG 0                      ; Current buffer owner IRP.
                  00 0258   773 EXE$AB_MVWORK: .BYTE 0                      ; Byte of storage belonging to the
                     0259   774                                            ; MV work buffer owner.
                     0259   775
                     0259   776 GET_BUFFER:
                     0259   777
          F8 AF    D5 0259   778          TSTL    B^EXE$AL_BSYIRP          ; Is the work buffer currently owned?
             05    12 025C   779          BNEQ    10$                      ; Branch if work buffer is owned.
       F2 AF    53  D0 025E   780          MOVL    R3, B^EXE$AL_BSYIRP      ; Else, setup address of current owner.
                05  05 0262   781          RSB                              ; Return to caller
                    0263   782          ;
                    0263   783          ; The page is busy. Put the caller in the wait queue.
                    0263   784          ;
       55   60 A3   9E 0263   785 10$:     MOVAB   IRP$L_FQFL(R3), R5       ; Get CDRP for use as fork block.
       10 A5   53   7D 0267   786          MOVQ    R3, CDRP$L_FR3(R5)       ; Save R3 and R4.
          0C A5 8ED0  026B   787          POPL    CDRP$L_FPC(R5)           ; Save return address.
          DA AF    D5 026F   788          TSTL    B^EXE$AL_WRKWQFL         ; Is work page wait queue initialized?
             0A    12 0272   789          BNEQ    20$                      ; Branch if queue initialized.
D3 AF    D5 AF    DE 0274   790          MOVAL   B^EXE$AL_WRKWQFL, -      ; Else, initalize queue neader.
                    0279   791                  B^EXE$AL_WRKWQFL
D2 AF    D0 AF    DE 0279   792          MOVAL   B^EXE$AL_WRKWQFL, -
                    027E   793                  B^EXE$AL_WRKWQBL
   CE BF    65   0E 027E   794 20$:     INSQUE  (R5), @B^EXE$AL_WRKWQBL ; Put CDRP on the work page wait queue.
                05 0282   795          RSB                              ; Return to caller's caller.
```

```
                                0283   797          .SBTTL  FREE_BUFFER - release an I/O buffer
                                0283   798 ;++
                                0283   799 ; FREE_BUFFER
                                0283   800 ;
                                0283   801 ; Functional description:
                                0283   802 ;
                                0283   803 ;       This routine is called by the current owner of the MV work page
                                0283   804 ;       to make the page available.  If the MV work page wait queue is
                                0283   805 ;       not empty, a fork process will be created for the waiting device,
                                0283   806 ;       and the MV work page will be given to that process.
                                0283   807 ;
                                0283   808 ; Input:
                                0283   809 ;
                                0283   810 ;       None.
                                0283   811 ;
                                0283   812 ; Output:
                                0283   813 ;
                                0283   814 ;       R0, R3...R5 are preserved.
                                0283   815 ;--
                                0283   816
                                0283   817 FREE_BUFFER:                            ; Release an I/O buffer
                                0283   818         ;
                                0283   819         ; Remove first entry from wait queue.  If empty, then return.
                                0283   820         ;
      C6 AF    D5               0283   821         TSTL    B^EXE$AL_WRKWQFL        ; Is work page wait queue initialized?
         06    13               0286   822         BEQL    10$                     ; Branch if Q not init'ed (ie. unused).
51    C1 BF    0F               0288   823         REMQUE  @B^EXE$AL_WRKWQFL, R1   ; Remove first entry from wait queue.
         04    1C               028C   824         BVC     20$                     ; Branch if somebody was waiting.
      C3 AF    D4               028E   825 10$:    CLRL    B^EXE$AL_BSYIRP         ; Mark the MV work page as unowned.
               05               0291   826         RSB                             ; Return to caller.
                                0292   827         ;
                                0292   828         ; Restore the waiting fork process's context
                                0292   829         ; and call it at its return address.
                                0292   830         ;
         39    BB               0292   831 20$:    PUSHR   #^M<R0,R3,R4,R5>        ; Save current fork context and R0.
   55    51    D0               0294   832         MOVL    R1, R5                  ; Copy fork block address.
53    10 A5    7D               0297   833         MOVQ    CDRP$L_FR3(R5), R3      ; Restore saved R3 and R4.
55    1C A3    D0               029B   834         MOVL    IRP$L_UCB(R3), R5       ; Restore UCB address.
B1 AF    53    D0               029F   835         MOVL    R3, B^EXE$AL_BSYIRP     ; Save address of work page owner.
                                02A3   836         ASSUME  IRP$L_FPC EQ <IRP$L_FQFL + CDRP$L_FPC>
      6C B3    16               02A3   837         JSB     @IRP$L_FPC(R3)          ; Call waiting fork process.
         39    BA               02A6   838         POPR    #^M<R0,R3,R4,R5>        ; Restore input context.
               05               02A8   839         RSB                             ; Return.
```

```
                        02A9    841          .SBTTL  TIME_DELAY - Put mount verification into a wait state
                        02A9    842 ;+
                        02A9    843 ; TIME_DELAY
                        02A9    844 ;
                        02A9    845 ; Functional description:
                        02A9    846 ;
                        02A9    847 ;       This routine will put the specified mount verification
                        02A9    848 ;       request into a wait state for a short while.  This is
                        02A9    849 ;       accomplished by performing a FORK_WAIT using the CDRP
                        02A9    850 ;       in the IRP/CDRP pair of the input internal IRP.
                        02A9    851 ;
                        02A9    852 ; Input:
                        02A9    853 ;
                        02A9    854 ;       R3 = internal irp address
                        02A9    855 ;       R5 = device ucb address
                        02A9    856 ;
                        02A9    857 ; Output:
                        02A9    858 ;
                        02A9    859 ;       R3..R5  are preserved
                        02A9    860 ;-
                        02A9    861
                        02A9    862 TIME_DELAY:                             ; Pause for a moment
        78 A3 8ED0      02A9    863          POPL    IRP$L_SAVD_RTN(R3)     ; Save caller's return address.
6B A3   0B A5   90      02AD    864          MOVB    UCB$B_FIPL(R5), -      ; Copy fork IPL to CDRP at tail of
                        02B2    865                  IRP$B_FIPL(R3)         ; IRP.
   55   60 A3   9E      02B2    866          MOVAB   IRP$L_FQFL(R3), R5     ; Get CDRP address.
                        02B6    867          FORK_WAIT                      ; Wait with CDRP as fork block.
   55   1C A3   D0      02BC    868          MOVL    IRP$L_UCB(R3), R5      ; Restore UCB address.
        78 B3   17      02C0    869          JMP     @IRP$L_SAVD_RTN(R3)    ; Return to original caller.
```

```
                                02C3    871          .SBTTL  INIT_IRP - set request independent fields of the IRP
                                02C3    872 ;++
                                02C3    873 ; INIT_IRP
                                02C3    874 ;
                                02C3    875 ; Functional description:
                                02C3    876 ;
                                02C3    877 ;       Zero an IRP and then set certain request-independent
                                02C3    878 ;       fields to their initial values.
                                02C3    879 ;
                                02C3    880 ; Input:
                                02C3    881 ;
                                02C3    882 ;       R3      = IRP address
                                02C3    883 ;       R5      = UCB address
                                02C3    884 ;
                                02C3    885 ; Output:
                                02C3    886 ;
                                02C3    887 ;       R0,R1,R2  are overwritten
                                02C3    888 ;       R4      preserved
                                02C3    889 ;       R3      = IRP address
                                02C3    890 ;       R5      = UCB address
                                02C3    891 ;--
                                02C3    892
                                02C3    893 INIT_IRP:                               ; Set request-independent fields of IRP
                                02C3    894 ;
                                02C3    895 ; Zero the IRP.
                                02C3    896 ;
                     14 A3   DD 02C3    897          PUSHL   IRP$L_ASTPRM(R3)       ; Save the MV timeout time
                        38   BB 02C6    898          PUSHR   #^M<R3,R4,R5>          ; Save IRP and UCB address from MOVC5
63   00C4 8F   00   60   00   2C 02C8    899          MOVC5   #0,(R0),#0,#IRP$K_LENGTH,(R3)   ; Zero the IRP
                        38   BA 02D0    900          POPR    #^M<R3,R4,R5>          ; Restore IRP and UCB address
                     14 A3 8ED0 02D2    901          POPL    IRP$L_ASTPRM(R3)       ; Restore the MV timeout time
                                02D6    902 ;
                                02D6    903 ; Set the request-independent fields of the IRP.
                                02D6    904 ;
                                02D6    905          ;
                                02D6    906          ; Set the size, type and access mode fields.
                                02D6    907          ;
                                02D6    908          ASSUME  IRP$W_SIZE  EQ 8
                                02D6    909          ASSUME  IRP$B_TYPE  EQ 10
                                02D6    910          ASSUME  IRP$B_RMOD  EQ 11
08 A3   000A00C4 8F   D0 02D6    911          MOVL    #<<DYN$C_IRP@16>!IRP$K_LENGTH>, IRP$W_SIZE(R3)
                                02DE    912          ;
                                02DE    913          ; Set the UCB address.  All other fields remain zero for now.
                                02DE    914          ;
                     1C A3   55 D0 02DE    915          MOVL    R5,IRP$L_UCB(R3)       ; Set UCB address
                           05   02E2    916          RSB
```

```
                                02E3    918          .SBTTL  INIT_IRP_READ - setup IRP for a one block read
                                02E3    919 ;++
                                02E3    920 ; INIT_IRP_READ
                                02E3    921 ;
                                02E3    922 ; Functional description:
                                02E3    923 ;
                                02E3    924 ;       Setup IRP fields required for a read operation.  Assume a one
                                02E3    925 ;       block read of the LBN specified in R0.
                                02E3    926 ;
                                02E3    927 ; Input:
                                02E3    928 ;
                                02E3    929 ;       R0      LBN to read
                                02E3    930 ;       R3      IRP address
                                02E3    931 ;       R5      UCB address
                                02E3    932 ;
                                02E3    933 ; Output:
                                02E3    934 ;
                                02E3    935 ;       R0 ... R2 overwritten
                                02E3    936 ;       All other registers preserved.
                                02E3    937 ;--
                                02E3    938
                                02E3    939 INIT_IRP_READ:
                                02E3    940
      20 A3    080C 8F    B0   02E3    941          MOVW    #<IO$_READPBLK -        ; Set function: read block
                                02E9    942                   !IO$M_INHERLOG>, -    ;                 w/o error logging
                                02E9    943                   IRP$W_FUNC(R3)
                                02E9    944          ASSUME  IO$_READPBLK LE IO$_PHYSICAL
      2A A3    2002 8F    A8   02E9    945          BISW    #<IRP$M_FUNC -          ; Set req. status:      read function
                                02EF    946                   !IRP$M_MVIRP>, -      ;                       MV request
                                02EF    947                   IRP$W_STS(R3)
   2C A3    00000000'GF    D0   02EF    948          MOVL    G^EXE$GL_SVAPTE, -      ; Set transfer SVAPTE for MV work
                                02F7    949                   IRP$L_SVAPTE(R3)      ; page.
      32 A3    0200 8F    3C   02F7    950          MOVZWL  #512, IRP$L_BCNT(R3)    ; Set transfer byte count of 1 block.
           00000000'GF    17   02FD    951          JMP     G^IOC$CVTLOGPHY         ; Convert LBN (in R0) to PBN, and
                                0303    952                                        ; return to caller.
```

```
                        0303   954          .SBTTL  VALIDATE_HOME
                        0303   955 ;++
                        0303   956 ; VALIDATE_HOME
                        0303   957 ;
                        0303   958 ;       This routine will check to see if the home block that was just
                        0303   959 ;       read contains valid data, and if so it will check to see if the
                        0303   960 ;       home block belongs to same volume as the one described in the
                        0303   961 ;       VCB for this device.  If both checks succeed, return success.
                        0303   962 ;       If either check fails, return failure.
                        0303   963 ;
                        0303   964 ; Input:
                        0303   965 ;
                        0303   966 ;       R3      = IRP address
                        0303   967 ;       R5      = UCB address
                        0303   968 ;
                        0303   969 ; Output:
                        0303   970 ;
                        0303   971 ;       R0              LBS ==> volume is correct
                        0303   972 ;                       LBC ==> some type of error
                        0303   973 ;       EXE$AB_MVWORK   is zero if the volume label comparison failed
                        0303   974 ;                       and one if it succeeded
                        0303   975 ;
                        0303   976 ;       R3, R4 and R5 are preserved.
                        0303   977 ;
                        0303   978 ; Side effects:
                        0303   979 ;
                        0303   980 ;       If the volume now physicaly mounted in the device is not
                        0303   981 ;       the same volume as the one described in the VCB, then the
                        0303   982 ;       interested parties will be informed, and the mount verification
                        0303   983 ;       will be restarted.
                        0303   984 ;--
                        0303   985
                        0303   986 VALIDATE_HOME:
                        0303   987
                  54    DD    0305   988          PUSHL   R4                      ; Save quorum indicator
                  7B    10    0305   989          BSBB    GET_BUF_ADDR            ; Get virtual address of MV buffer.
                        0307   990
                        0307   991          ;
                        0307   992          ; Compare the volume serial number and volume name in the VCB
                        0307   993          ; with those stored on the stack.  If they are identical, then
                        0307   994          ; this is the correct volume.  The following assumptions must
                        0307   995          ; be true if we are to avoid special-casing ODS-1 and ODS-2 volumes.
                        0307   996          ;
                        0307   997          ASSUME  HM2$L_SERIALNUM EQ HM1$L_SERIALNUM
                        0307   998          ASSUME  HM2$T_VOLNAME   EQ HM1$T_VOLNAME2
                        0307   999          ;
              50  34 A5   D0    0307  1000          MOVL    UCB$L_VCB(R5), R0       ; Get VCB address.
                  2B    13    030B  1001          BEQL    VALIDATE_EXIT           ; Branch if no VCB address.
           64 A0  01C8 C4   D1    030D  1002          CMPL    HM2$L_SERIALNUM(R4), -  ; Check the volume serial number.
                        0313  1003                  VCB$L_SERIALNUM(R0)
                  19    12    0313  1004          BNEQ    VALIDATE_ERROR          ; Branch if serial nos. not equal.
                FF3F CF   94    0315  1005          CLRB    W^EXE$AB_MVWORK         ; Assume label comparison will fail.
                  38    BB    0319  1006          PUSHR   #^M<R3,R4,R5>           ; Save registers.
        14 A0   01D8 C4   0C    29    031B  1007          CMPC3   #VCB$S_VOLNAME, -       ; Check the volume name.
                        0322  1008                  HM2$T_VOLNAME(R4), -
                        0322  1009                  VCB$T_VOLNAME(R0)
                  38    BA    0322  1010          POPR    #^M<R3,R4,R5>           ; Restore registers.
```

```
                04   12  0324  1011          BNEQ    10$                             ; Branch if not the same volume.
            FF2E CF   96  0326  1012          INCB    W^EXE$AB_MVWORK                 ; Else, set same volume flag.
                42   10  032A  1013 10$:      BSBB    CHECKSUM                        ; Test home block checksum, and
                0A   11  032C  1014          BRB     VALIDATE_EXIT
                         032E  1015
                         032F  1016 VALIDATE_ERROR:
                         032E  1017
                         032E  1018          ; A difference has been detected between the block in the MV work
                         032E  1019          ; buffer (either the home block or the storage control block) and the
                         032E  1020          ; VCB.  If the block checksum agrees with the caculated checksum, then
                         032E  1021          ; the volume is incorrect.  If the checksums do not agree, then the
                         032E  1022          ; preceived difference may be due to a corrupted disk block.  In such
                         032E  1023          ; cases the read must be retried.
                         032E  1024
                3E   10  032E  1025          BSBB    CHECKSUM                        ; Checksum the MV work buffer.
              05 50  E9  0330  1026          BLBC    R0, VALIDATE_EXIT               ; Branch if checksums differ.
50          010C 8F  3C  0333  1027          MOVZWL  #SS$_INCVOLLABEL, R0            ; Else, set incorrect volume status.
                         0338  1028
                         0338  1029 VALIDATE_EXIT:
                54 8ED0  0338  1030          POPL    R4
                    05   033B  1031          RSB                                     ; Return.
```

```
                     033C  1033          .SBTTL  VALIDATE_SCB
                     033C  1034 ;++
                     033C  1035 ; VALIDATE_SCB
                     033C  1036 ;
                     033C  1037 ;       This routine verifies the checksum of the storage control block in
                     033C  1038 ;       the MV work buffer.  If the checksum is ok, the mount time in the SCB
                     033C  1039 ;       is compared to the equivalent value in the VCB.  If the volume label
                     033C  1040 ;       tested in VALIDATE_HOME did not match, the volume-lock-names in the
                     033C  1041 ;       VCB and SCB are also compared.  If all tested values are equal,
                     033C  1042 ;       success is returned.  Otherwise, SS$_INCVOLLABEL is returned.
                     033C  1043 ;
                     033C  1044 ; Input:
                     033C  1045 ;
                     033C  1046 ;       R5              UCB address
                     033C  1047 ;       EXE$AB_MVWORK   is zero if the volume label comparison in
                     033C  1048 ;                       VALIDATE_VOLUME failed and one if it succeeded
                     033C  1049 ;
                     033C  1050 ; Output:
                     033C  1051 ;
                     033C  1052 ;       R0      LBS ==> volume is correct
                     033C  1053 ;               LBC ==> some type of error
                     033C  1054 ;
                     033C  1055 ;       R1 and R2 are overwritten.
                     033C  1056 ;       All other registers are preserved.
                     033C  1057 ;--
                     033C  1058
                     033C  1059 VALIDATE_SCB:
                 54  DD   033C  1060          PUSHL   R4                      ; Save quorum indicator
                 42  10   033E  1061          BSBB    GET_BUF_ADDR            ; Get the MV work buffer VA.
          50  34 A5  D0   0340  1062          MOVL    UCB$L_VCB(R5), R0       ; Get VCB address.
                 F2  13   0344  1063          BEQL    VALIDATE_EXIT           ; Branch if no VCB.
      0090 C0   2E A4  D1   0346  1064          CMPL    SCB$Q_MOUNTTIME(R4), -  ; Compare mount times in the SCB
                     034C  1065                  VCB$Q_MOUNTTIME(R0)     ; and in the VCB.
                 E0  12   034C  1066          BNEQ    VALIDATE_ERROR          ; Take error exit if times don't
      0094 C0    32 A4  D1   034E  1067          CMPL    SCB$Q_MOUNTTIME+4(R4), -; match.
                     0354  1068                  VCB$Q_MOUNTTIME+4(R0)
                 D8  12   0354  1069          BNEQ    VALIDATE_ERROR
            0D FEFE CF   E8   0356  1070          BLBS    W^EXE$AB_MVWORK, 50$    ; Branch if VALIDATE_VOLUME succeeded.
                 38  BB   035B  1071          PUSHR   #^M<R3,R4,R5>           ; Save registers.
  0080 C0   22 A4  0C   29   035D  1072          CMPC3   #VCB$S_VOLCKNAM, -      ; Check the volume-lock-name.
                     0364  1073                  SCB$T_VOLOCKNAME(R4), -
                     0364  1074                  VCB$T_VOLCKNAM(R0)
                 38  BA   0364  1075          POPR    #^M<R3,R4,R5>           ; Restore registers.
                 C6  12   0366  1076          BNEQ    VALIDATE_ERROR          ; Branch if not the same volume.
                 04  10   0368  1077 50$:     BSBB    CHECKSUM                ; Test SCB checksum, and
                 54 8ED0   036A  1078          POPL    R4
                     05   036D  1079          RSB
```

```
                     036E  1081          .SBTTL  CHECKSUM - compute FILES-11 structure block checksum
                     036E  1082 ;++
                     036E  1083 ; CHECKSUM
                     036E  1084 ;
                     036E  1085 ; Functional description:
                     036E  1086 ;
                     036E  1087 ;       This routine computes the FILES-11 structure block checksum and
                     036E  1088 ;       compares it to the checksum stored in the FILES-11 structure block.
                     036E  1089 ;       If the checksums match, a success status is returned.
                     036E  1090 ;
                     036E  1091 ; Input:
                     036E  1092 ;
                     036E  1093 ;       R4      = FILES-11 structure block buffer address
                     036E  1094 ;
                     036E  1095 ; Ouput:
                     036E  1096 ;
                     036E  1097 ;       R0      = status value
                     036E  1098 ;       R3 and R5 are preserved
                     036E  1099 ;--
                     036E  1100
                     036E  1101 CHECKSUM:                               ; Compute FILES-11 block checksum
                     036E  1102
                     036E  1103         ; Note: the FILES-11 structure block checksum is stored in the last
                     036E  1104         ; word of the structure.  After the checksum loop completes, R4 points
                     036E  1105         ; to the stored checksum and R1 contains the computed checksum.  The
                     036E  1106         ; following assumptions must be true for this technique to work.
                     036E  1107
                     036E  1108         ASSUME  HM2$W_CHECKSUM2 EQ HM1$W_CHECKSUM2
                     036E  1109         ASSUME  HM2$W_CHECKSUM2/2 EQ 255
                     036E  1110
          50   7C    036E  1111         CLRQ    R0                      ; Assume failure; clear total
      52  FF 8F    9A 0370  1112         MOVZBL  #255,R2                 ; Set loop counter
        51   84    A0 0374  1113 10$:    ADDW2   (R4)+,R1                ; Sum adjacent words
          FA 52    F5 0377  1114         SOBGTR  R2,10$                  ; Branch if more to go
        64   51    B1 037A  1115         CMPW    R1,(R4)                 ; Compare checksums
             02    12 037D  1116         BNEQ    13$                     ; Branch if not equal
             50    D6 037F  1117         INCL    R0                      ; Set success status
                   05 0381  1118 13$:    RSB                             ; Return
```

```
                                0382  1120          .SBTTL  GET_BUF_ADDR
                                0382  1121 ;++
                                0382  1122 ; GET_BUF_ADDR
                                0382  1123 ;
                                0382  1124 ;       This routine returns the virtual address of the MV work buffer in R4.
                                0382  1125 ;
                                0382  1126 ; Input:
                                0382  1127 ;
                                0382  1128 ;       None.
                                0382  1129 ;
                                0382  1130 ;
                                0382  1131 ; Output:
                                0382  1132 ;
                                0382  1133 ;       R4      system virtual address of the MV work buffer
                                0382  1134 ;
                                0382  1135 ;       All other registers are preserved.
                                0382  1136 ;--
                                0382  1137
                                0382  1138 GET_BUF_ADDR:
                                0382  1139
00000000'GF   00000000'GF   C3  0382  1140          SUBL3   G^MMG$GL_SPTBASE, -     ; Calculate byte offset to MV work
                        54      038D  1141                  G^EXE$GL_SVAPTE, R4     ; SPTE.
                  54    54  07  038E  1142          ASHL    #9-2, R4, R4            ; Shift offset making VA minus 80000000.
                    00  54  1F  E2  0392  1143          BBSS    #PTE$V_VALID, R4, 10$   ; Fix the 80000000 part.
                            05  0396  1144 10$:     RSB                             ; Return.
```

```
                     0397  1146          .SBTTL  EXE$MNTVERSIO
                     0397  1147 ;++
                     0397  1148 ; EXE$MNTVERSIO
                     0397  1149 ;
                     0397  1150 ; Functional description:
                     0397  1151 ;
                     0397  1152 ;       This routine is called to initiate an I/O to a device.  The
                     0397  1153 ;       request is packed in an 'internal' IRP.  This means that the
                     0397  1154 ;       PID field of the IRP contains the address of a routine to be
                     0397  1155 ;       called by IOPOST to finish processing the I/O request.  Note
                     0397  1156 ;       that the PID field will be set to the address of a generic
                     0397  1157 ;       post-processor, which will in turn branch to the actual post
                     0397  1158 ;       processing routine.
                     0397  1159 ;
                     0397  1160 ; Input:
                     0397  1161 ;
                     0397  1162 ;       R0,R1,R2,R4 are scratch registers
                     0397  1163 ;       R3      = IRP address
                     0397  1164 ;       R5      = UCB address
                     0397  1165 ;        (SP)   = address of action routine to post-process the I/O
                     0397  1166 ;
                     0397  1167 ; Output:
                     0397  1168 ;
                     0397  1169 ;       None.
                     0397  1170 ;--
                     0397  1171
                     0397  1172 EXE$MNTVERSIO::                             ; Start an internal I/O request
                     0397  1173
 0C A3   AC'AF   9E  0397  1174          MOVAB   B^END_IO,IRP$L_PID(R3)      ; Set I/O post address
       10 A3 8ED0    039C  1175          POPL    IRP$L_AST(R3)               ; Set action routine address
                     03A0  1176                                              ; Note that the stack is now clean.
                     03A0  1177          ;
                     03A0  1178          ; Begin optional I/O performance measurement.
                     03A0  1179          ;
                     03A0  1180          .IF     DF      CA$_MEASURE_IOT
 00000000'GF   16    03A0  1181          JSB     G^PMS$START_RQ              ; Start I/O request measurement
                     03A6  1182          .ENDC
                     03A6  1183
 00000000'GF   17    03A6  1184          JMP     G^IOC$INITIATE              ; Start the I/O immediately
```

```
                        03AC  1186          .SBTTL  END_IO
                        03AC  1187 ;++
                        03AC  1188 ; END_IO
                        03AC  1189 ;
                        03AC  1190 ; Functional description:
                        03AC  1191 ;
                        03AC  1192 ;       This routine performs some common I/O post-processing
                        03AC  1193 ;       before dispatching to the specific post-processing routine,
                        03AC  1194 ;       whose address is stored in IRP$L_AST(R3).  This routine is
                        03AC  1195 ;       called by IOPOST, at IPL 4, so it is necessary to get into
                        03AC  1196 ;       the proper driver fork context before continuing.
                        03AC  1197 ;
                        03AC  1198 ; Input:
                        03AC  1199 ;
                        03AC  1200 ;       R5      = IRP address
                        03AC  1201 ;       IPL     = IPL$_POST
                        03AC  1202 ;
                        03AC  1203 ; Output:
                        03AC  1204 ;
                        03AC  1205 ;       R5      = UCB address
                        03AC  1206 ;       R3      = IRP address
                        03AC  1207 ;
                        03AC  1208 ;--
                        03AC  1209
                        03AC  1210 END_IO:                                      ; End of I/O request
                        03AC  1211          ;
                        03AC  1212          ; Complete I/O performance measurement.
                        03AC  1213          ;
                        03AC  1214          .IF     DF      CA$_MEASURE_IOT
00000000'GF     16      03AC  1215          JSB     G^PMS$END_RQ                ; Gather performance data
                        03B2  1216          .ENDC
                        03B2  1217          ;
    53    55    D0      03B2  1218          MOVL    R5,R3                       ; Copy IRP addres
 55    1C A3    D0      03B5  1219          MOVL    IRP$L_UCB(R3),R5            ; Get UCB address
                        03B9  1220          DSBINT  UCB$B_FIPL(R5)              ; Raise IPL to driver FORK level
       10 B3    16      03C0  1221          JSB     @IRP$L_AST(R3)              ; Dispatch to post processing routine
                        03C3  1222          ENBINT                              ; Restore IPL
                  05    03C6  1223          RSB                                 ; Return
```

```
                                                H 4
                     03C7  1225          .SBTTL  CLEANUP_IO
                     03C7  1226 ;++
                     03C7  1227 ; CLEANUP_IO
                     03C7  1228 ;
                     03C7  1229 ; Functional description:
                     03C7  1230 ;
                     03C7  1231 ;       This routine is called to perform any necessary cleanup on the
                     03C7  1232 ;       I/O request that has come into the main entry point of mount
                     03C7  1233 ;       verification.
                     03C7  1234 ;
                     03C7  1235 ; Input:
                     03C7  1236 ;
                     03C7  1237 ;       R0,R1   = Status of the I/O operation
                     03C7  1238 ;       R3      = IRP address
                     03C7  1239 ;       R5      = UCB address
                     03C7  1240 ;
                     03C7  1241 ; Output:
                     03C7  1242 ;
                     03C7  1243 ;       All registers are preserved.
                     03C7  1244 ;--
                     03C7  1245
                     03C7  1246 CLEANUP_IO:                             ; Finish processing an I/O packet
                     03C7  1247          ;
                     03C7  1248          ; If I/O performance measurement is enabled, call the proper routine
                     03C7  1249          ; to gather the performance data.  Note that the performance routine
                     03C7  1250          ; assumes that IRP$L_MEDIA and IRP$L_MEDIA+4 contain the I/O status.
                     03C7  1251          ; However, these longwords currently contain some information that will
                     03C7  1252          ; be needed to later restart the request.  Therefore, the contents of
                     03C7  1253          ; IRP$L_MEDIA and IRP$L_MEDIA+4 must be saved and later restored.
                     03C7  1254          ;
                     03C7  1255          .IF     DF      CA$_MEASURE_IOT
   7E    38 A3   7D  03C7  1256          MOVQ    IRP$L_MEDIA(R3),-(SP)     ; Save info from IRP
 38 A3     50   7D  03CB  1257          MOVQ    R0,IRP$L_MEDIA(R3)        ; Set I/O status
00000000'GF    16  03CF  1258          JSB     G^PMS$END_IO              ; Take performance measurement
 38 A3     8E   7D  03D5  1259          MOVQ    (SP)+,IRP$L_MEDIA(R3)     ; Restore info to IRP
                     03D9  1260          .ENDC
                  05 03D9  1261          RSB                               ; Return
```

```
                                                03DA  1263          .SBTTL  DRIVER_CODE - Driver specific code
                                                03DA  1264 ;++
                                                03DA  1265 ; DRIVER_CODE
                                                03DA  1266 ;
                                                03DA  1267 ; Functional description:
                                                03DA  1268 ;
                                                03DA  1269 ;       This routine will call the driver specific routine that is necessary
                                                03DA  1270 ;       to begin and end mount verification.  The address of the routine is
                                                03DA  1271 ;       stored in the cell DDT$L_MNTVER.  If a driver does not specify the
                                                03DA  1272 ;       contents of DDT$L_MNTVER, it defaults to the address of IOC$MNTVER.
                                                03DA  1273 ;
                                                03DA  1274 ; Input:
                                                03DA  1275 ;
                                                03DA  1276 ;       R3      = IRP address or 0
                                                03DA  1277 ;       R5      = UCB address
                                                03DA  1278 ;
                                                03DA  1279 ; Output:
                                                03DA  1280 ;
                                                03DA  1281 ;       None.
                                                03DA  1282 ;
                                                03DA  1283 ; Side effects:
                                                03DA  1284 ;
                                                03DA  1285 ;       Driver specific, but the contents of R0,R1,R2 and R4 may be overwritten.
                                                03DA  1286 ;--
                                                03DA  1287
                                                03DA  1288 DRIVER_CODE:                                 ; Call driver specific code
50   0088 C5   D0                              03DA  1289          MOVL    UCB$L_DDT(R5),R0           ; Get DDT address
       20 B0   17                              03DF  1290          JMP     @DDT$L_MNTVER(R0)          ; Branch to driver specified routine
```

```
                                J 4
                                .SBTTL  WRITLCK_HNDLR
                        ;++
                        ; WRITLCK_HNDLR
                        ;
                        ; Functional description:
                        ;
                        ;       This routine will allow the file system to recover from the
                        ;       accidental hardware writelocking of a FILES-11 volume.  The
                        ;       method used is to mark the current IRP as a mount verification
                        ;       IRP, and try the request over again.  If the request then
                        ;       succeeds, nothing more need be done.  If the request fails again,
                        ;       keep trying until it succeeds.  If this is the first time this
                        ;       request has failed (the MVIRP bit will be clear), then inform
                        ;       all interested parties that the device has been writelocked.
                        ;
                        ;       Note that writelock recovery is not interlocked by the MVNVERIP
                        ;       bit.  This will allow recovery of a device offline error to supersede
                        ;       writelock recovery.  If this were not done, it would be possible to
                        ;       become deadlocked by an offline error occurring after a writelock
                        ;       error.
                        ;
                        ; Input:
                        ;
                        ;       R0,R1   = I/O status
                        ;       R3      = IRP address
                        ;       R5      = UCB address
                        ;
                        ; Output:
                        ;
                        ;       None.
                        ;--
                        ;
                        WRITLCK_HNDLR:                          ; Recover from accidental writelock
8E   D5                 TSTL    (SP)+                           ; Remove return address from stack
E1   10                 BSBB    CLEANUP_IO                      ; Clean up current I/O operation
                        ;
                        ; To prevent the error log to become saturated with entries
                        ; due the repeated failure of this request, inhibit error logging
                        ; for this operation.  Note that if error logging had previously
                        ; been enabled, then the error log will already contain an entry
                        ; for this request.
                        ;
0B   E2                 BBSS    #IO$V_INHERLOG,-                ; Inhibit error logging
00 20 A3                        IRP$W_FUNC(R3),10$              ;
                        ;
                        ; If mount verification is already in progress, then
                        ; requeue this request to the device and try again later.
                        ;
0E   E1                 BBC     #UCB$V_MNTVERIP,-               ; Branch if mount verification not in progre
03 64 A5                        UCB$W_STS(R5),20$               ;
E8   10                 BSBB    DRIVER_CODE                     ; Requeue the IRP to the driver
05                      RSB                                     ; Return to caller's caller
                        ;
                        ; Mark this IRP as a mount verification IRP.  If it already is,
                        ; then retry the I/O immediately.
                        ;
0D   E2                 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
```

```
                     0406  1362          .SBTTL  SEND_MESSAGE
                     0406  1363 ;++
                     0406  1364 ; SEND_MESSAGE
                     0406  1365 ;
                     0406  1366 ; Functional description:
                     0406  1367 ;
                     0406  1368 ;       This routine is used to inform all interested parties of a mount
                     0406  1369 ;       verification related event.  A message is sent to OPCOM, which will
                     0406  1370 ;       in turn send the message to all operators enabled to receive DEVICE
                     0406  1371 ;       or DISK messages.
                     0406  1372 ;
                     0406  1373 ;       Since there is a possibility that OPCOM will not be able to relay
                     0406  1374 ;       the message to the operator, also send a message that is targeted
                     0406  1375 ;       explicitly at the operator's console.
                     0406  1376 ;
                     0406  1377 ; Input:
                     0406  1378 ;
                     0406  1379 ;       R4      = MV message list index for message
                     0406  1380 ;       R5      = UCB address
                     0406  1381 ;
                     0406  1382 ; Output:
                     0406  1383 ;
                     0406  1384 ;       None.  Contents of R0 .. R2 are destroyed
                     0406  1385 ;
                     0406  1386 ;--
                     0406  1387
00000000             0406  1388 SAVED_R1 =      0                           ;.
00000004             0406  1389 SAVED_R2 =      4                           ; .
00000008             0406  1390 SAVED_R3 =      8                           ;  . Offsets to saved registers
0000000C             0406  1391 SAVED_R4 =      12                          ; .
00000010             0406  1392 SAVED_R5 =      16                          ;.
00000014             0406  1393 DEVNAM_SIZE     = 20                        ; Allow 20 character device names
0000000A             0406  1394 POOL_OVERHEAD   = 2+IRP$W_SIZE              ; Allow for listhead and size fields
00000022             0406  1395 MSG_OVERHEAD    = POOL_OVERHEAD + DEVNAM_SIZE + <2 * CAR_CTRL_SIZE>
0000000A             0406  1396 MSG_START       = POOL_OVERHEAD             ; First usuable byte in pool
                     0406  1397
                     0406  1398 SEND_MESSAGE:                              ; Send message to interested parties
                     0406  1399 ;
                     0406  1400 ; Send message to OPCOM.
                     0406  1401 ;
                  3E   BB 0406  1402          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save registers
                00AF   30 0408  1403          BSBW    GET_MSG_ID              ; Put address of message Id in R4
08 02 A4         01   E1 040B  1404          BBC     #MVMSL$V_SUPRESS, -     ; Branch if MVMSL does not indicate
                     0410  1405                  MVMSL$W_FLAGS(R4), 15$  ; that this message is supressable.
07 64 A5         12   E1 0410  1406          BBC     #UCB$V_SUPMVMSG, -      ; Branch if UCB does not indicate that
                     0415  1407                  UCB$L_STS(R5), 17$      ; normal MV messages are supressed.
                009F   31 0415  1408          BRW     90$                     ; Else, supress this message.
                     0418  1409 15$:     ASSUME  UCB$V_SUPMVMSG GE 16
   66 A5         04   8A 0418  1410          BICB    #<UCB$M_SUPMVMSG@-16>, -; For must print messages, clear the
                     041C  1411                  UCB$L_STS+2(R5)         ; supress mount verification msgs. bit.
      54         64   3C 041C  1412 17$:     MOVZWL  MVMSL$W_MSG_CODE(R4),R4 ; Get the OPCOM message number.
53 00000000'GF      9E 041F  1413          MOVAB   G^SYS$GL_OPRMBX,R3      ; Get operator mailbox UCB address
   00000000'GF      16 0426  1414          JSB     G^EXE$SNDEVMSG          ; Mail message to OPCOM; ignore failure
                     042C  1415 ;
                     042C  1416 ; Send message to _OPA0:.
                     042C  1417 ;
                     042C  1418          ;
```

```
                                042C  1419          ; Calculate the total message size and allocate a block
                                042C  1420          ; of nonpaged pool in which to build the message.
                                042C  1421          ; SIZE = prefix_size + message_size + suffix_size + pool_header_size + devic
                                042C  1422          ;
                    008B    30  042C  1423          BSBW    GET_MSG_ID                  ; Get message description and id.
                  51    61    9A  042F  1424          MOVZBL  (R1), R1                    ; Setup allocation size accumulator.
                51  0085'CF    80  0432  1425          ADDB    W^PREFIX, R1                ; Add prefix size to message size.
                51  009C'CF    80  0437  1426          ADDB    W^SUFFIX, R1                ; Add suffix size to accumulated size.
                  51    22    80  043C  1427          ADDB    #MSG_OVERHEAD, R1           ; Add message overhead to acc. size.
             00000000'GF    16  043F  1428          JSB     G^EXE$ALONONPAGED           ; Allocate the necessary pool
                  6F    50    E9  0445  1429          BLBC    R0, 90$                     ; If no pool available, give up.
                                0448  1430          ;
                                0448  1431          ; Build the message in the pool just allocated.
                                0448  1432          ; The entire message consists of the prefix, followed by
                                0448  1433          ; the formatted device name, followed by the message text,
                                0448  1434          ; followed by the sufix.
                                0448  1435          ;
                                0448  1436          ; R1 = actual pool size
                                0448  1437          ; R2 = address of pool
                                0448  1438          ;
                08  A2    51    B0  0448  1439          MOVW    R1,IRP$W_SIZE(R2)           ; Save size of pool in the block itself
                04  AE    52    D0  044C  1440          MOVL    R2,SAVED_R2(SP)             ; Save address of pool
                53    0A  A2    9E  0450  1441          MOVAB   MSG_START(R2),R3            ; Get address of message buffer.
              83    0A0D  8F    B0  0454  1442          MOVW    #CAR_CTRL,(R3)+             ; Insert leading <cr><lf>
              51    0085'CF    9E  0459  1443          MOVAB   W^PREFIX,R1                 ; Get address of prefix ASCIC string
                          6C    10  045E  1444          BSBB    COPY_ASCIC                  ; Copy string to buffer (R5 destroyed)
                55    10  AE    D0  0460  1445          MOVL    SAVED_R5(SP),R5             ; Restore R5
                    50    14    D0  0464  1446          MOVL    #DEVNAM_SIZE,R0             ; Set device name size
                    51    53    D0  0467  1447          MOVL    R3,R1                       ; Set buffer address
                          54    DD  046A  1448          PUSHL   R4                          ; Save R4
                    54    01    CE  046C  1449          MNEGL   #1,R4                       ; Signal get node+devnam for remote devs
                 00000000'GF    16  046F  1450          JSB     G^IOC$CVT_DEVNAM            ; Format the device name (R1 = result length
                          54  8ED0  0475  1451          POPL    R4                          ; Restore R4
                    53    51    C0  0478  1452          ADDL    R1,R3                       ; Point to next byte in msg buffer
                          3D    10  047B  1453          BSBB    GET_MSG_ID                  ; R4 = address of message identifier
                          4D    10  047D  1454          BSBB    COPY_ASCIC                  ; Copy message text to buffer.
                          39    10  047F  1455          BSBB    GET_MSG_ID                  ; Get message text descriptor
          07  02  A4      00    E0  0481  1456          BBS     #MVMSL$V_NOSUFFIX, -        ; Branch if suffix not requested.
                                0486  1457                  MVMSL$W_FLAGS(R4), 30$
              51    009C'CF    9E  0486  1458          MOVAB   W^SUFFIX,R1                 ; Get address of suffix ASCIC string
                          3F    10  048B  1459          BSBB    COPY_ASCIC                  ; Copy suffix to the buffer
              83    0A0D  8F    B0  048D  1460 30$:     MOVW    #CAR_CTRL, (R3)+            ; Insert trailing <CR><LF>.
                                0492  1461          ;
                                0492  1462          ; Send the message to _OPA0:
                                0492  1463          ;
                54    04  AE    D0  0492  1464          MOVL    SAVED_R2(SP),R4             ; Restore pool address
                52    0A  A4    9E  0496  1465          MOVAB   MSG_START(R4),R2            ; Get start of message address
              51    53    52    C3  049A  1466          SUBL3   R2,R3,R1                    ; Calculate the length of the message
          55   00000000'GF    9E  049E  1467          MOVAB   G^OPA$UCB0,R5               ; Get console terminal UCB address
               00000000'GF    16  04A5  1468          JSB     G^IOC$BROADCAST             ; Send the message
                    50    54    D0  04AB  1469          MOVL    R4,R0                       ; Copy pool address
               00000000'GF    16  04AE  1470          JSB     G^COM$DRVDEALMEM            ; Deallocate the pool
                    50    01    3C  04B4  1471          MOVZWL  #SS$_NORMAL,R0              ; Set success status
                          3E    BA  04B7  1472 90$:     POPR    #^M<R1,R2,R3,R4,R5>         ; Restore R1 ... R5.
                                05  04B9  1473          RSB                                 ; Return
                                04BA  1474
                                04BA  1475 ;+
```

```
                     04BA  1476 ; This is a local subroutine to index into the message
                     04BA  1477 ; identifier table and return the address of the correct
                     04BA  1478 ; entry in R4.  Note that the stack depth has changed due
                     04BA  1479 ; to the return address on the stack.
                     04BA  1480 ;-
                     04BA  1481 GET_MSG_ID:                                  ; Return address of message identifier
  54    10 AE    D0  04BA  1482         MOVL    4+SAVED_R4(SP),R4           ; Get the table index number
51    FB42 CF    9E  04BE  1483         MOVAB   W^EXE$AB_MVMSLBAS, R1       ; Get base of message info table.
                     04C3  1484         ASSUME  MVMSL$K_LENGTH EQ 8
  54    6144    7E  04C3  1485         MOVAQ   (R1)[R4], R4                ; Get address of entry for message id.
  51    04 A4   C0  04C7  1486         ADDL    MVMSL$L_TEXTOFF(R4), R1     ; Get base of counted string address.
                05  04CB  1487         RSB                                 ; Return
                     04CC  1488
                     04CC  1489
                     04CC  1490 ;+
                     04CC  1491 ; This routine is a spcial case of COPY_STRING.  On input, R1 is
                     04CC  1492 ; assumed to contain the address of an ASCIC string.  Execution
                     04CC  1493 ; drops through to COPY_STRING, and will return from there to the caller.
                     04CC  1494 ;-
                     04CC  1495 COPY_ASCIC:                                  ; Special case of COPY_STRING
    50      81   9A  04CC  1496         MOVZBL  (R1)+,R0                    ; Get length of string and advance the point
                     04CF  1497
                     04CF  1498 ;+
                     04CF  1499 ; This routine is used to copy a string to a buffer.
                     04CF  1500 ;
                     04CF  1501 ; Input:
                     04CF  1502 ;
                     04CF  1503 ;       R0 = string length
                     04CF  1504 ;       R1 = string address
                     04CF  1505 ;       R3 = buffer address
                     04CF  1506 ;
                     04CF  1507 ; Output:
                     04CF  1508 ;
                     04CF  1509 ;       R3 = address of next byte in the buffer.
                     04CF  1510 ;            (This is a fortuitous side-effect of the MOVC3 instruction.)
                     04CF  1511 ;-
                     04CF  1512 COPY_STRING:                                 ; Copy a string to a buffer
63    61    50   28  04CF  1513         MOVC3   R0,(R1),(R3)                ; Go to it!
                05  04D3  1514         RSB                                 ; Return
```

```
                04D4  1516          .SBTTL  EXE$UPDGNERNUM - update shadow set generation number
                04D4  1517          .SBTTL  EXE$MNTVERSHDOL - bring a shadow set member online
                04D4  1518          .SBTTL  EXE$MNTVERSP1 - spare mount verification entry point
                04D4  1519          .SBTTL  EXE$MNTVERSP2 - spare mount verification entry point
                04D4  1520          .SBTTL  EXE$MNTVER_DVI_ASSIST - $GETDVI escape transfer vector
                04D4  1521 ;++
                04D4  1522 ; EXE$UPDGNERNUM - update shadow set generation number
                04D4  1523 ; EXE$MNTVERSHDOL - bring a shadow set member online
                04D4  1524 ; EXE$MNTVERSP1 - spare mount verification entry point
                04D4  1525 ; EXE$MNTVERSP2 - spare mount verification entry point
                04D4  1526 ; EXE$MNTVER_DVI_ASSIST - $GETDVI escape transfer vector
                04D4  1527 ;
                04D4  1528 ; Functional description:
                04D4  1529 ;
                04D4  1530 ;       These mount verification entry points do not yet have any code written
                04D4  1531 ;       for them.  There are defined here to hold their places for possibly /
                04D4  1532 ;       probable implementation during the life-time of Version 4.
                04D4  1533 ;--
                04D4  1534
                04D4  1535 EXE$UPDGNERNUM::
                04D4  1536 EXE$MNTVERSHDOL::
                04D4  1537 EXE$MNTVERSP1::
                04D4  1538 EXE$MNTVERSP2::
                04D4  1539
          05    04D4  1540          RSB
                04D5  1541
                04D5  1542 EXE$MNTVER_DVI_ASSIST::
                04D5  1543
    60    17    04D5  1544          JMP     (R0)
```

```
                                04D7  1546          .SBTTL  EXE$CLUTRANIO - VAXcluster State Change I/O Blocking
                                04D7  1547 ;++
                                04D7  1548 ; EXE$CLUTRANIO - VAXcluster State Change I/O Blocking
                                04D7  1549 ;
                                04D7  1550 ; Functional description:
                                04D7  1551 ;
                                04D7  1552 ;       This routine is used to place all MSCP or cluster accessible disks into
                                04D7  1553 ;       mount verification.  The intended use is to ensure that no I/O is
                                04D7  1554 ;       issued when a cluster may be out of quorum.
                                04D7  1555 ;
                                04D7  1556 ; Input:
                                04D7  1557 ;
                                04D7  1558 ;       None
                                04D7  1559 ;
                                04D7  1560 ; Output:
                                04D7  1561 ;
                                04D7  1562 ;       None.  Contents of all registers preserved.
                                04D7  1563 ;
                                04D7  1564 ;--
                                04D7  1565
                                04D7  1566 EXE$CLUTRANIO::
                    007F 8F BB  04D7  1567          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6>
                                04DB  1568 ;
                                04DB  1569 ; First look at all local devices.
                                04DB  1570 ; If the UCB is not busy, it is marked busy and mount verifiction is started.
                                04DB  1571 ; If busy, volume valid is cleared and the mount verification pending bit is
                                04DB  1572 ; set.
                                04DB  1573 ;
                56 00000000'GF  D0  04DB  1574          MOVL    G^IOC$GL_DEVLIST,R6     ; Start of DDB list
                     55   04 A6  D0  04E2  1575 10$:     MOVL    DDB$L_UCB(R6),R5        ; Get a UCB
                             49  13  04E6  1576          BEQL    50$                     ; None
                     01   40 A5  91  04E8  1577          CMPB    UCB$B_DEVCLASS(R5),#DC$_DISK
                             43  12  04EC  1578          BNEQ    50$                     ; Not a disk
                   3E 38 A5      0E  E1  04EE  1579          BBC     #DEV$V_FOD,UCB$L_DEVCHAR(R5),50$ ; Not file oriented
                   39 3C A5      05  E0  04F3  1580          BBS     #DEV$V_MSCP,UCB$L_DEVCHAR2(R5),50$ ; Not interested in MSCP now
                         34 A5   D5  04F8  1581 20$:     TSTL    UCB$L_VCB(R5)           ; Check the VCB - consistency check only
                             2E  13  04FB  1582          BEQL    40$                     ; Next UCB if none
                   29 64 A5      0E  E0  04FD  1583          BBS     #UCB$V_MNTVERIP,UCB$W_STS(R5),40$ ; Already have it
                   24 3C A5      00  E1  0502  1584          BBC     #DEV$V_CLU,UCB$L_DEVCHAR2(R5),40$ ; Not cluster accessible
                   00 64 A5      12  E2  0507  1585          BBSS    #UCB$V_SUPMVMSG,UCB$L_STS(R5),25$ ; No messages
                   10 64 A5      08  E1  050C  1586 25$:     BBC     #UCB$V_BSY,UCB$W_STS(R5),30$ ; UCB not busy
                64 A5    0800 8F  AA  0511  1587          BICW    #UCB$M_VALID,UCB$W_STS(R5) ; Clear vol-valid to stop I/Os
            64 A5   00284000 8F  C8  0517  1588          BISL    #<UCB$M_MNTVERIP -      ; Pending when I/O completes
                                051F  1589                   !UCB$M_MNTVERPND -     ; initiate mount verification for
                                051F  1590                   !UCB$M_CLUTRAN>, -     ; for a VAXcluster state transition.
                                051F  1591                  UCB$L_STS(R5)
                             0A  11  051F  1592          BRB     40$
                64 A5    0100 8F  A8  0521  1593 30$:     BISW    #UCB$M_BSY,UCB$W_STS(R5); Make UCB busy to stall I/O
                         FB5F CF  16  0527  1594          JSB     MNTVER_NOIRP            ; Put it in mount verification
                      55   30 A5  D0  052B  1595 40$:     MOVL    UCB$L_LINK(R5),R5       ; Next UCB
                             C7  12  052F  1596          BNEQ    20$                     ; Go look at it
                         56   66  D0  0531  1597 50$:     MOVL    DDB$L_LINK(R6),R6       ; next DDB
                             AC  12  0534  1598          BNEQ    10$                     ; Go look at it
                                0536  1599 ;
                                0536  1600 ; Now handle the MSCP disks.  It is only necessary to set the CDDB bit to
                                0536  1601 ; indicate out-of-quorum and then call the revalidate routine.
                                0536  1602 ;
```

```
                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  1620          .SBTTL  QUORUM
                                0569  1621 ;++
                                0569  1622 ; QUORUM
                                0569  1623 ;
                                0569  1624 ; Functional description:
                                0569  1625 ;
                                0569  1626 ;       This routine is used to determine if we are in a cluster that is
                                0569  1627 ;       out of quorum.
                                0569  1628 ;
                                0569  1629 ; Input:
                                0569  1630 ;
                                0569  1631 ;       None
                                0569  1632 ;
                                0569  1633 ; Output:
                                0569  1634 ;
                                0569  1635 ;       R4  0 -> not an out of quorum cluster
                                0569  1636 ;           1 -> out of quorum cluster
                                0569  1637 ;
                                0569  1638 ;--
                                0569  1639
                     54   D4   0569  1640 QUORUM: CLRL    R4                              ; Assume no cluster
52   00000000'GF        D0   056B  1641          MOVL    G^CLU$GL_CLUB,R2
                        0D   13   0572  1642          BEQL    50$                             ; No CLUB, so no cluster
   08 1C A2             00   E1   0574  1643          BBC     #CLUB$V_CLUSTER,CLUB$L_FLAGS(R2),50$ ; Not a cluster member
   03 1C A2             1C   E0   0579  1644          BBS     #CLUB$V_QUORUM,CLUB$L_FLAGS(R2),50$ ; Not out of quorum
         54             01   D0   057E  1645          MOVL    #1,R4                           ; Indicate no quorum
                        05   0581  1646 50$:     RSB
                             0582  1647
```

```
                        0582  1649          .SBTTL  GET_VCB - Obtain VCB address
                        0582  1650 ;++
                        0582  1651 ;
                        0582  1652 ; GET_VCB - Obtain VCB address
                        0582  1653 ;
                        0582  1654 ; Functional Description:
                        0582  1655 ;
                        0582  1656 ;       Because mount verification is used for more than just fixing mistaken
                        0582  1657 ;       hardware changes, there are several conditions in which mount
                        0582  1658 ;       verification can be entered legally without there being a  FILES-11
                        0582  1659 ;       VCB.  Such cases are:
                        0582  1660 ;
                        0582  1661 ;         1. The volume being processed is mounted /FOREIGN.
                        0582  1662 ;
                        0582  1663 ;         2. The volume being processed is the system disk and it has not yet
                        0582  1664 ;            been properly mounted.
                        0582  1665 ;
                        0582  1666 ;         3. The volume being processed is the quorum disk and it has not yet
                        0582  1667 ;            been properly mounted.
                        0582  1668 ;
                        0582  1669 ;       This routine returns control if a VCB is present or legally absent.
                        0582  1670 ;       The condition code is NEQ if the VCB is present or EQL if it is
                        0582  1671 ;       legally absent.  Otherwise, mount verification is aborted.
                        0582  1672 ;
                        0582  1673 ; Inputs:
                        0582  1674 ;
                        0582  1675 ;       R3      MVIRP address
                        0582  1676 ;       R5      UCB address
                        0582  1677 ;
                        0582  1678 ; Outputs:
                        0582  1679 ;
                        0582  1680 ;       Condition codes:
                        0582  1681 ;
                        0582  1682 ;       NEQ     VCB is present
                        0582  1683 ;       EQL     VCB is legally absent
                        0582  1684 ;
                        0582  1685 ;       R0      VCB address
                        0582  1686 ;
                        0582  1687 ;       All other registers are preserved.
                        0582  1688 ;--
                        0582  1689
                        0582  1690 GET_VCB:
                        0582  1691
   25 38 A5    18   E0  0582  1692          BBS     #DEV$V_FOR, -           ; Branch if /FOREIGN disk.
                        0587  1693                  UCB$L_DEVCHAR(R5), 60$
                        0587  1694
      50    34 A5   D0  0587  1695          MOVL    UCB$L_VCB(R5), R0       ; Get VCB address.
                  21   12  058B  1696          BNEQ    70$                     ; Branch if VCB is present.
                        058D  1697
55    00000000'8F   D1  058D  1698          CMPL    #SYS$GL_BOOTUCB, R5     ; Is this the system disk?
                  16   13  0594  1699          BEQL    60$                     ; Branch if system disk.
                        0596  1700
50    00000000'GF   D0  0596  1701          MOVL    G^CLU$GL_CLUB, R0       ; Get CLUB address.
                  10   13  059D  1702          BEQL    80$                     ; Branch if no CLUB.
   50    00B4 C0   D0  059F  1703          MOVL    CLUB$L_CLUDCB(R0), R0   ; Get quorum disk control block addr.
                  09   13  05A4  1704          BEQL    80$                     ; Branch if no CLUDCB.
     55    0C A0   D1  05A6  1705          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
BUILD_RDHOME_IRP                000001A2 R      02
CAR_CTRL                    =   00000A0D
CAR_CTRL_SIZE               =   00000002
CDDB$L_CDDBLINK             =   00000058
CDDB$L_DDB                  =   0000001C
CDDB$M_QUORLOST             =   00000200
CDDB$W_STATUS               =   00000012
CDRP$L_FPC                  =   0000000C
CDRP$L_FR3                  =   00000010
CHECKSUM                        0000036E R      02
CLEANUP_IO                      000003C7 R      02
CLU$GL_CLUB                     ******** X      02
CLUB$L_CLUDCB               =   000000B4
CLUB$L_FLAGS                =   0000001C
CLUB$V_CLUSTER              =   00000000
CLUB$V_QUORUM               =   0000001C
CLUDCB$L_IRP                =   00000010
CLUDCB$L_UCB                =   0000000C
COM$DRVDEALMEM                  ******** X      02
COPY_ASCIC                      000004CC R      02
COPY_STRING                     000004CF R      02
DC$_DISK                    =   00000001
DDB$L_DDT                   =   0000000C
DDB$L_LINK                  =   00000000
DDB$L_UCB                   =   00000004
DDT$L_FDT                   =   00000008
DDT$L_MNTVER                =   00000020
DDT$L_UNSOLINT              =   00000004
DEV$V_CLU                   =   00000000
DEV$V_FOD                   =   0000000E
DEV$V_FOR                   =   00000018
DEV$V_MNT                   =   00000013
DEV$V_MSCP                  =   00000005
DEV$V_SQD                   =   00000005
DEV$V_SWL                   =   00000019
DEVNAM_SIZE                 =   00000014
DRIVER_CODE                     000003DA R      02
DYN$C_IRP                   =   0000000A
END_IO                          000003AC R      02
ERROR_EXIT                      0000023E R      02
EXE$AB_MVMSLBAS                 00000004 RG     02
EXE$AB_MVWORK                   00000258 R      02
EXE$ALONONPAGED                 ******** X      02
EXE$AL_BSYIRP                   00000254 R      02
EXE$AL_WRKWQBL                  00000250 R      02
EXE$AL_WRKWQFL                  0000024C R      02
EXE$CLOTRANIO                   000004D7 RG     02
EXE$FORK_WAIT                   ******** X      02
EXE$GL_ABSTIM                   ******** X      02
EXE$GL_SVAPTE                   ******** X      02
EXE$MNTVERSHDOL                 000004D4 RG     02
EXE$MNTVERSIO                   00000397 RG     02
EXE$MNTVERSP1                   000004D4 RG     02
EXE$MNTVERSP2                   000004D4 RG     02
EXE$MNTVER_DVI_ASSIST           000004D5 RG     02
EXE$MOUNTVER                    0000002C RG     02
EXE$SNDEVMSG                    ********   X    02
EXE$UPDGNERNUM                  000004D4 RG     02
FREE_BUFFER                     00000283 R      02
FREE_BUF_ERRXIT                 0000023C R      02
GET_BUFFER                      00000259 R      02
GET_BUF_ADDR                    00000382 R      02
GET_MSG_ID                      000004BA R      02
GET_VCB                         00000582 R      02
HM1$L_SERIALNUM             =   000001C8
HM1$T_VOLNAME2              =   000001D8
HM1$W_CHECKSUM2             =   000001FE
HM2$L_SERIALNUM             =   000001C8
HM2$T_VOLNAME               =   000001D8
HM2$W_CHECKSUM2             =   000001FE
INIT_IRP                        000002C3 R      02
INIT_IRP_READ                   000002E3 R      02
IO$M_INHERLOG               =   00000800
IO$V_INHERLOG               =   0000000B
IO$_PACKACK                 =   00000008
IO$_PHYSICAL                =   0000001F
IO$_READPBLK                =   0000000C
IOC$BROADCAST                   ********   X    02
IOC$CVTLOGPHY                   ********   X    02
IOC$CVT_DEVNAM                  ********   X    02
IOC$GL_DEVLIST                  ********   X    02
IOC$GL_DU_CDDB                  ********   X    02
IOC$GW_MVTIMEOUT                ********   X    02
IOC$INITIATE                    ********   X    02
IRP$B_FIPL                  =   0000006B
IRP$B_RMOD                  =   0000000B
IRP$B_TYPE                  =   0000000A
IRP$K_LENGTH                =   000000C4
IRP$L_AST                   =   00000010
IRP$L_ASTPRM                =   00000014
IRP$L_BCNT                  =   00000032
IRP$L_FPC                   =   0000006C
IRP$L_FQFL                  =   00000060
IRP$L_MEDIA                 =   00000038
IRP$L_PID                   =   0000000C
IRP$L_SAVD_RTN              =   00000078
IRP$L_SVAPTE                =   0000002C
IRP$L_UCB                   =   0000001C
IRP$M_FUNC                  =   00000002
IRP$M_MVIRP                 =   00002000
IRP$M_PHYSIO                =   00000100
IRP$V_MVIRP                 =   0000000D
IRP$W_FUNC                  =   00000020
IRP$W_SIZE                  =   00000008
IRP$W_STS                   =   0000002A
MMG$GL_SPTBASE                  ********   X    02
MNTVER_NOIRP                    0000008A R      02
MSG$_DEVOFFLINX             =   00000050
MSG$_DEVWRTLCK              =   00000052
MSG$_MVABORTED              =   00000055
MSG$_MVCOMPLETE             =   00000054
MSG$_WRONGVOL               =   00000051
MSG_OVERHEAD                =   00000022
```

```
MSG_START                       =  0000000A
MVABORTED                       =  00000004
MVCOMPLETE                      =  00000003
MVMSL$K_LENGTH                  =  00000008
MVMSL$L_SNDMSGOFF               =  FFFFFFFC
MVMSL$L_TEXTOFF                 =  00000004
MVMSL$M_NOSUFFIX                =  00000001
MVMSL$M_SUPRESS                 =  00000002
MVMSL$V_NOSUFFIX                =  00000000
MVMSL$V_SUPRESS                 =  00000001
MVMSL$W_FLAGS                   =  00000002
MVMSL$W_MSG_CODE                =  00000000
NORMAL_EXIT                        00000200 R      02
OFFLINE                         =  00000000
OPA$UCB0                           ********   X    02
PAUSE                              000000F9 R      02
PMS$END_IO                         ********   X    02
PMS$END_RQ                         ********   X    02
PMS$START_RQ                       ********   X    02
POOL_OVERHEAD                   =  0000000A
PR$_IPL                         =  00000012
PREFIX                             00000085 R      03
PTE$V_VALID                     =  0000001F
QUORUM                             00000569 R      02
QUORUM_LOOP                        00000127 R      02
SAVED_R1                        =  00000000
SAVED_R2                        =  00000004
SAVED_R3                        =  00000008
SAVED_R4                        =  0000000C
SAVED_R5                        =  00000010
SCB$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
UCB$M_VALID                     =  00000800
UCB$M_WRONGVOL                  =  00008000
UCB$V_BSY                       =  00000008
UCB$V_CLUTRAN                   =  00000015
UCB$V_MNTVERIP                  =  0000000E
UCB$V_SUPMVMSG                  =  00000012
UCB$V_VALID                     =  0000000B
UCB$V_WRONGVOL                  =  0000000F
UCB$W_STS                       =  00000064
VALIDATE_ERROR                     0000032E R      02
VALIDATE_EXIT                      00000338 R      02
VALIDATE_FAILED                    00000219 R      02
VALIDATE_HOME                      00000303 R      02
VALIDATE_SCB                       0000033C R      02
VCB$B_STATUS2                   =  00000053
VCB$L_HOMELBN                   =  00000024
VCB$L_SBMAPLBN                  =  00000034
VCB$L_SERIALNUM                 =  00000064
VCB$Q_MOUNTTIME                 =  00000090
VCB$S_VOLCKNAM                  =  0000000C
VCB$S_VOLNAME                   =  0000000C
VCB$T_VOLCKNAM                  =  00000080
VCB$T_VOLNAME                   =  00000014
VCB$V_MOUNTVER                  =  00000002
WRITELOCK                       =  00000002
WRITLCK_HNDLR                      000003E2 R      02
WRONGVOL                        =  00000001
```

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
_255DUA28:[SYS.OBJ]LIB.MLB;1	20
_255DUA28:[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
```

MOUNTVER
LIS

OPDRVWS1
LIS

QUORUM
LIS

OPDRV790
LIS

OPDRIVER
CIS

REBLDLOCK
LIS