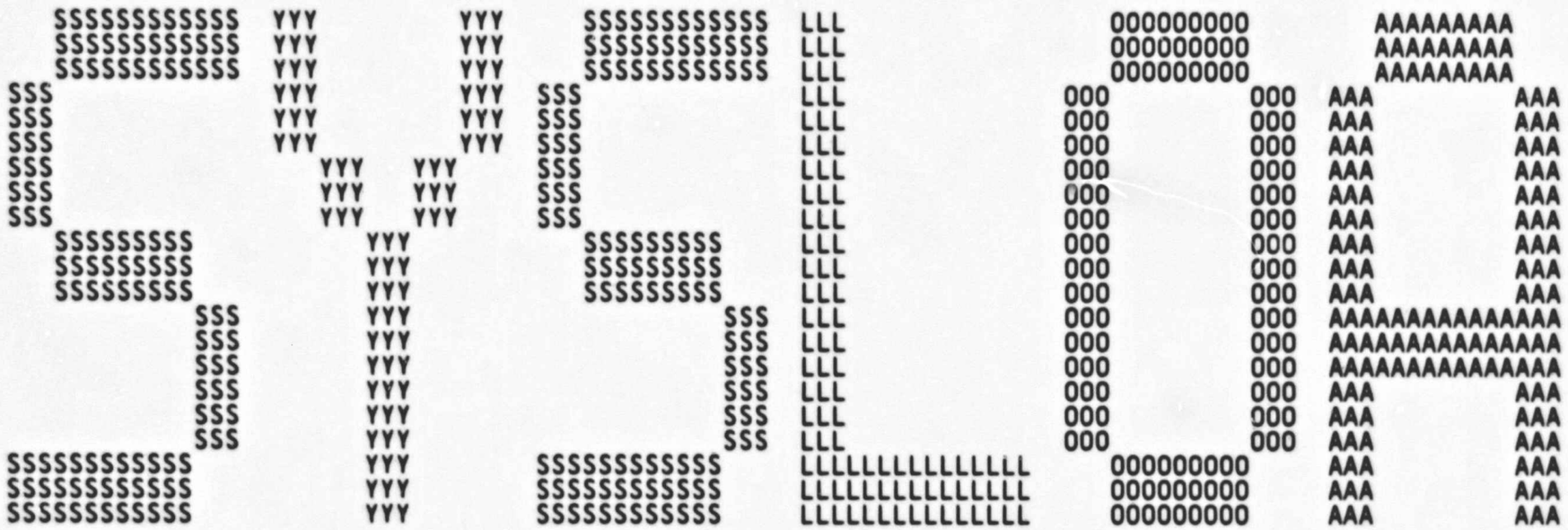
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL OOOOOOOOO AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL OOOOOOOOO AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL OOOOOOOOO AAAAAAAAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL OOO OOO AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL OOO OOO AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL OOO OOO AAA AAA
SSS YYY SSS LLL OOO OOO AAAAAAAAAAAAAAA
SSS YYY SSS LLL OOO OOO AAAAAAAAAAAAAAA
SSS YYY SSS LLL OOO OOO AAAAAAAAAAAAAAA
SSS YYY SSS LLL OOO OOO AAA AAA
SSS YYY SSS LLL OOO OOO AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL OOO OOO AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL OOOOOOOOO AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL OOOOOOOOO AAA AAA

```
  QQQQQQ    UU      UU    OOOOOO    RRRRRRRR    UU      UU  MM      MM
  QQQQQQ    UU      UU    OOOOOO    RRRRRRRR    UU      UU  MM      MM
QQ      QQ  UU      UU  OO      OO  RR      RR  UU      UU  MMMM  MMMM
QQ      QQ  UU      UU  OO      OO  RR      RR  UU      UU  MMMM  MMMM
QQ      QQ  UU      UU  OO      OO  RR      RR  UU      UU  MM  MM  MM
QQ      QQ  UU      UU  OO      OO  RR      RR  UU      UU  MM  MM  MM
QQ      QQ  UU      UU  OO      OO  RRRRRRRR    UU      UU  MM      MM
QQ      QQ  UU      UU  OO      OO  RRRRRRRR    UU      UU  MM      MM
QQ  QQ  QQ  UU      UU  OO      OO  RR  RR      UU      UU  MM      MM
QQ  QQ  QQ  UU      UU  OO      OO  RR  RR      UU      UU  MM      MM
QQ    QQ    UU      UU  OO      OO  RR    RR    UU      UU  MM      MM
QQ    QQ    UU      UU  OO      OO  RR    RR    UU      UU  MM      MM
  QQQQ  QQ  UUUUUUUUUU    OOOOOO    RR      RR  UUUUUUUUUU  MM      MM
  QQQQ  QQ  UUUUUUUUUU    OOOOOO    RR      RR  UUUUUUUUUU  MM      MM

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  QUORUM - DISK QUORUM MODULE
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28  ;++
0000    29  ; Facility: Executive, Cluster management
0000    30  ;
0000    31  ; Abstract:
0000    32  ;       This module contains the routines that implement the disk quorum
0000    33  ;       algorithm.
0000    34  ;
0000    35  ; Enviornment:
0000    36  ;       VMS Non Paged Exec - Kernel mode
0000    37  ;--
0000    38  ;
0000    39  ; Author:
0000    40  ;
0000    41  ;       R. Scott Hanna, CREATION DATE: 25-Jul-1983
0000    42  ;
0000    43  ; Modified by:
0000    44  ;
0000    45  ;       V03-008 WMC0003         Wayne Cardoza           16-Jul-1984
0000    46  ;               Call mount verification under some error conditions.
0000    47  ;               Clear CLUDCB$B_COUNTER on any entry to CLUSTER state.
0000    48  ;
0000    49  ;       V03-007 WMC0002         Wayne Cardoza           28-Jun-1984
0000    50  ;               Allow one error before calling CSP.
0000    51  ;
0000    52  ;       V03-006 WMC0001         Wayne Cardoza           31-May-1984
0000    53  ;               Make sure IRP$W_STS field is cleared.
0000    54  ;
0000    55  ;       V03-005 SSA0023         Stan Amway               6-Apr-1984
0000    56  ;               Decrement UCB device queue length when I/O completes
0000    57  ;               in READ_COMPLETE or WRITE_COMPLETE. This is required
```

```
0000     58 ;               because EXE$INSIOQ increments the length, but the IRP
0000     59 ;               does not go through the normal IOPOST code which does
0000     60 ;               the corresponding decrement.
0000     61 ;
0000     62 ;       V03-004 RSH0119         R. Scott Hanna          14-Mar-1984
0000     63 ;               Rewrite of module to use a new algorithm.
0000     64 ;
0000     65 ;       V03-003 RSH0085         R. Scott Hanna          23-Nov-1983
0000     66 ;               Remove clear of quorum file logical block number on
0000     67 ;               "connection" loss.
0000     68 ;
0000     69 ;       V03-002 RSH0078         R. Scott Hanna          10-Nov-1983
0000     70 ;               Changes in error handling to print error messages one
0000     71 ;               time only. Clear quorum file logical block number in
0000     72 ;               CLUDCB when "connection" is lost. Changes necessary due
0000     73 ;               to re-structured quorum block. Changes due to move of
0000     74 ;               QF_TRANS and QF_TIMEOUT from CLUB to CLUDCB.
0000     75 ;
0000     76 ;       V03-001 RSH0071         R. Scott Hanna          27-Sep-1983
0000     77 ;               Make sure CLUDCB$L_QBLAST and CLUDCB$L_QBBUF are
0000     78 ;               swapped on quorum file transition from inactive
0000     79 ;               regardless of the CLUB$V_QF_SKIP_READ bit.
0000     80 ;--
```

```
                                0000     82          .SBTTL  Declarations
                                0000     83 ;
                                0000     84 ;       Define Symbols
                                0000     85 ;
                                0000     86
                                0000     87          $CLUBDEF                                ; Cluster block
                                0000     88          $CLUDCBDEF                              ; Cluster quorum disk control block
                                0000     89          $CLUQFDEF                               ; Cluster quorum file
                                0000     90          $CSBDEF                                 ; Cluster system block
                                0000     91          $CSDDEF                                 ; Cluster server data
                                0000     92          $CSPDEF                                 ; CSP communication codes
                                0000     93          $DYNDEF                                 ; Dynamic data structure types
                                0000     94          $IODEF                                  ; I/O function codes
                                0000     95          $IPLDEF                                 ; Interrupt priority levels
                                0000     96          $IRPDEF                                 ; I/O request packet
                                0000     97          $SBDEF                                  ; System Block
                                0000     98          $TQEDEF                                 ; Time queue entry
                                0000     99          $UCBDEF                                 ; Unit control block
                                0000    100          $VADEF                                  ; Virtual address fields
                                0000    101
                                0000    102 ;
                                0000    103 ; The cycle count insures that we will not get burned by race conditions
                                0000    104 ; and not see another cluster through the quorum disk.
                                0000    105 ;
                     00000002   0000    106          CYCLE_COUNT = 2
                                0000    107
                                0000    108 ;
                                0000    109 ; The following assumptions are in effect for the entire module
                                0000    110 ;
                                0000    111          ASSUME  IPL$_TIMER EQ IPL$_SYNCH
                                0000    112          ASSUME  IPL$_TIMER EQ IPL$_SCS
                                0000    113          ASSUME  CLUDCB$S_BUFFER EQ CLUQF$K_LENGTH
                                0000    114          ASSUME  CLUQF$K_CHECK_LENGTH&3 EQ 0
                                0000    115
                                0000    116          .DEFAULT DISPLACEMENT,WORD
                                0000    117
                                0000    118 ;
                                0000    119 ; Own Storage
                                0000    120 ;
                            00000000    121          .PSECT  $$$060,LONG
                                0000    122
                                0000    123 CLUQF_IDENT_STRING:
45 4C 49 46 20 20 4D 55 52 4F 55 51     0000    124          .ASCII  /QUORUM  FILE/
                                000C    125          ASSUME  CLUQF$S_IDENT EQ .-CLUQF_IDENT_STRING
```

```
                     000C   127 .SBTTL  CNX$QUORUM_INIT - Quorum initialization
                     000C   128
                     000C   129 ;++
                     000C   130 ; CNX$QUORUM_INIT - Quorum initialization
                     000C   131 ;
                     000C   132 ; FUNCTIONAL DESCRIPTION:
                     000C   133 ;
                     000C   134 ;       This routine determines if a quorum disk has been specified,
                     000C   135 ;       and if so allocates and initializes the cluster quorum disk
                     000C   136 ;       control block (CLUDCB) and associated data structures.
                     000C   137 ;
                     000C   138 ; CALLING SEQUENCE:
                     000C   139 ;
                     000C   140 ;       JSB/BSBx CNX$QUORUM_INIT
                     000C   141 ;           IPL is 31
                     000C   142 ;
                     000C   143 ; INPUTS:
                     000C   144 ;
                     000C   145 ;       NONE
                     000C   146 ;
                     000C   147 ; OUTPUT:
                     000C   148 ;
                     000C   149 ;       NONE
                     000C   150 ;
                     000C   151 ; SIDE EFFECTS:
                     000C   152 ;
                     000C   153 ;       R0-R5 are destroyed
                     000C   154 ;--
                     000C   155
                 00000000   156         .PSECT  $$$002,LONG                     ; Initialization PSECT
                     0000   157
                     0000   158 CNX$QUORUM_INIT::
                     0000   159
             0FC0 8F    BB  0000   160         PUSHR   #^M<R6,R7,R8,R9,R10,R11>        ; Save registers
                     0004   161 ;
                     0004   162 ; Determine if we have a quorum file
                     0004   163 ;
          10    20    3A  0004   164         LOCC    #^A/ /,#CLUDCB$S_DISK_QUORUM,-  ; Locate end of quorum disk name
    00000000'GF         0007   165                 G^CLU$GB_QDISK
          10    50    D1  000C   166         CMPL    R0,#CLUDCB$S_DISK_QUORUM        ; Is there a disk name?
             03    12    000F   167         BNEQU   1$                              ; Br if yes
           00A7    31    0011   168         BRW     4$
                     0014   169 ;
                     0014   170 ; Allocate the CLUDCB
                     0014   171 ;
51   00000229 8F    D0  0014   172 1$:      MOVL    #CLUDCB$K_LENGTH,R1             ; CLUDCB size
    00000000'GF    16    001B   173         JSB     G^EXE$ALONONPAGED               ; Allocate CLUDCB
          25    50    E9  0021   174         BLBC    R0,2$                           ; Br if error
          56    51    7D  0024   175         MOVQ    R1,R6                           ; Save CLUDCB size and address
                     0027   176 ;
                     0027   177 ; Allocate the IRP
                     0027   178 ;
51   000000C4 8F    D0  0027   179         MOVL    #IRP$K_LENGTH,R1                ; IRP size
    00000000'GF    16    002E   180         JSB     G^EXE$ALONONPAGED               ; Allocate IRP
          12    50    E9  0034   181         BLBC    R0,2$                           ; Br if error
          58    51    7D  0037   182         MOVQ    R1,R8                           ; Save IRP size and address
                     003A   183 ;
```

```
                                  003A      ; Allocate the TQE
                                  003A      ;
                   51    30   D0  003A             MOVL    #TQE$K_LENGTH,R1              ; TQE size
            00000000'GF      16   003D             JSB     G^EXE$ALONONPAGED             ; Allocate TQE
                   5A    51   7D  0043             MOVQ    R1,R10                        ; Save TQE size and address
                   03    50   E8  0046             BLBS    R0,3$                         ; Br if success
                       0076   31  0049      2$:    BRW     5$
                                  004C      ;
                                  004C      ; Initialize the CLUDCB
                                  004C      ;
 67    56    00    6E    00   2C  004C      3$:    MOVC5   #0,(SP),#0,R6,(R7)            ; Zero the CLUDCB
             08 A7       56   B0  0052             MOVW    R6,CLUDCB$W_SIZE(R7)          ; Store size
          0A A7    65    8F   90  0056             MOVB    #DYN$C_CLU,CLUDCB$B_TYPE(R7)  ; Store type
                         05   90  005B             MOVB    #DYN$C_CLU_CLUDCB,-           ; Store subtype
                      0B A7       005D                     CLUDCB$B_SUBTYPE(R7)
             10 A7       59   D0  005F             MOVL    R9,CLUDCB$L_IRP(R7)           ; Store IRP address
             14 A7       5B   D0  0063             MOVL    R11,CLUDCB$L_TQE(R7)          ; Store TQE address
                         01   B0  0067             MOVW    #CLUDCB$M_QS_NOT_READY,-      ; Initial state is NOT_READY
                      20 A7       0069                     CLUDCB$W_STATE(R7)
                                  006B      ;
                                  006B      ; Initialize the IRP
                                  006B      ;
 69    58    00    6E    00   2C  006B             MOVC5   #0,(SP),#0,R8,(R9)            ; Zero the IRP
             08 A9       58   B0  0071             MOVW    R8,IRP$W_SIZE(R9)             ; Store size
             0A A9       0A   90  0075             MOVB    #DYN$C_IRP,IRP$B_TYPE(R9)     ; Store type
          23 A9    FF    8F   90  0079             MOVB    #^XFF,IRP$B_PRI(R9)           ; Store priority
                                  007E      ;
                                  007E      ; Initialize the TQE
                                  007E      ;
 6B    5A    00    6E    00   2C  007E             MOVC5   #0,(SP),#0,R10,(R11)          ; Zero the TQE
             08 AB       5A   D0  0084             MOVL    R10,TQE$W_SIZE(R11)           ; Store size
             0A AB       0F   90  0088             MOVB    #DYN$C_TQE,TQE$B_TYPE(R11)    ; Store type
             0B AB       05   90  008C             MOVB    #TQE$C_SSREPT,TQE$B_RQTYPE(R11) ; Store request type
       0C AB    0000'CF       9E  0090             MOVAB   QUORUM_TIMEOUT,TQE$L_FPC(R11) ; Set up timer request fork PC
             10 AB       57   D0  0096             MOVL    R7,TQE$L_FR3(R11)             ; Store fork register three
     54     00000000'GF       D0  009A             MOVL    G^CLU$GL_CLUB,R4              ; Get CLUB address
             14 AB       54   D0  00A1             MOVL    R4,TQE$L_FR4(R11)             ; Store fork register four
     52     00000000'GF       3C  00A5             MOVZWL  G^CLU$GW_QDSKINTERVAL,R2      ; Get timeout value. (in seconds)
 00    00989680 8F       52   7A  00AC             EMUL    R2,#10000000,#0,-             ; Convert timeout to 100ns units
                      20 AB       00B4                     TQE$Q_DELTA(R11)              ; ...and store in TQE
                                  00B6      ;
                                  00B6      ; Point CLUB to CLUDCB
                                  00B6      ;
          00B4 C4        57   D0  00B6             MOVL    R7,CLUB$L_CLUDCB(R4)          ; Store CLUDCB pointer in CLUB
                                  00BB
     50     00000000'8F       D0  00BB      4$:    MOVL    #SS$_NORMAL,R0                ; Return success
                  0FC0 8F     BA  00C2      5$:    POPR    #^M<R6,R7,R8,R9,R10,R11>      ; Restore registers
                         05       00C6             RSB
```

```
                                00C7   233 .SBTTL  QUORUM_TIMEOUT - Quorum timeout
                                00C7   234 ;++
                                00C7   235 ; QUORUM_TIMEOUT - Quorum timeout
                                00C7   236 ;
                                00C7   237 ; FUNCTIONAL DESCRIPTION:
                                00C7   238 ;
                                00C7   239 ;       This routine executes every n seconds as a fork process where n is
                                00C7   240 ;       determined by the sysgen parameter QDSKINTERVAL.
                                00C7   241 ;
                                00C7   242 ; CALLING SEQUENCE:
                                00C7   243 ;
                                00C7   244 ;       JSB QUORUM_TIMEOUT
                                00C7   245 ;
                                00C7   246 ; INPUTS:
                                00C7   247 ;
                                00C7   248 ;       R3 = address of CLUDCB
                                00C7   249 ;       R4 = address of CLUB
                                00C7   250 ;       R5 = address of TQE
                                00C7   251 ;
                                00C7   252 ; OUTPUT:
                                00C7   253 ;
                                00C7   254 ;       R0-R2 Destroyed
                                00C7   255 ;--
                                00C7   256
                            00000000   257         .PSECT  $$$100,LONG
                                0000   258
                                0000   259 QUORUM_TIMEOUT::
                                0000   260
                           56   DD  0000   261         PUSHL   R6                              ; Save R6
                                00  E0  0002   262         BBS     #CLUDCB$V_QF_TIM,-              ; Br if we already timed out the
                             2E 22  A3      0004   263                 CLUDCB$W_FLAGS(R3),5$           ; ...I/O in progress
                           56   25  A3  DE  0007   264         MOVAL   CLUDCB$T_BUFFER(R3),R6          ; Get buffer address
                                02  E1  000B   265         BBC     #CLUDCB$V_QF_WIP,-              ; Br if no write in progress
                             05 22  A3      000D   266                 CLUDCB$W_FLAGS(R3),1$
                                48  A6  96  0010   267         INCB    CLUQF$B_IGNORE(R6)              ; Invalidate buffer
                                05  11  0013   268         BRB     2$
                                01  E1  0015   269 1$:      BBC     #CLUDCB$V_QF_RIP,-              ; Br if no read in progress
                             0E 22  A3      0017   270                 CLUDCB$W_FLAGS(R3),3$
                                01  A8  001A   271 2$:      BISW2   #CLUDCB$M_QF_TIM,-              ; Set timeout bit
                             22  A3      001C   272                 CLUDCB$W_FLAGS(R3)
                           50   0000'CF  9E  001E   273         MOVAB   W^QDTIMOUT_MSG,R0               ; Point to timeout message
                                03AF  30  0023   274         BSBW    QUORUM_DISK_TIMEOUT             ; Process timeout error
                                0D  11  0026   275         BRB     5$
                                00  E1  0028   276 3$:      BBC     #CLUDCB$V_QS_NOT_READY,-        ; Br if we are in one of the
                             05 20  A3      002A   277                 CLUDCB$W_STATE(R3),4$           ; ...ready states
                                03E4  30  002D   278         BSBW    REQUEST_CSP
                                03  11  0030   279         BRB     5$
                                0004  30  0032   280 4$:      BSBW    READ_QUORUM_FILE                ; Queue a quorum file read request
                            56   8E  D0  0035   281 5$:      MOVL    (SP)+,R6                        ; Restore R6
                                    05  0038   282         RSB
```

```
                     0039   284 .SBTTL  READ_QUORUM_FILE - Queue a read to the quorum file
                     0039   285 ;++
                     0039   286 ; READ_QUORUM_FILE - Queue a read to the quorum file
                     0039   287 ;
                     0039   288 ; FUNCTIONAL DESCRIPTION:
                     0039   289 ;
                     0039   290 ;       This routine builds and queues an IRP to read the quorum file.
                     0039   291 ;
                     0039   292 ; CALLING SEQUENCE:
                     0039   293 ;
                     0039   294 ;       JSB/BSBx READ_QUORUM_FILE
                     0039   295 ;
                     0039   296 ; INPUTS:
                     0039   297 ;
                     0039   298 ;       R3 = address of CLUDCB
                     0039   299 ;       R6 = address of quorum file buffer
                     0039   300 ;
                     0039   301 ; OUTPUT:
                     0039   302 ;
                     0039   303 ;       R0-R2 destroyed
                     0039   304 ;--
                     0039   305
                     0039   306 READ_QUORUM_FILE:
                     0039   307
           38   BB  0039   308         PUSHR   #^M<R3,R4,R5>                   ; Save registers
           02   A8  003B   309         BISW    #CLUDCB$M_QF_RIP,-              ; Set read in progress bit
       22 A3           003D   310                 CLUDCB$W_FLAGS(R3)
     52   10 A3   D0  003F   311         MOVL    CLUDCB$L_IRP(R3),R2             ; Get IRP address
  0C A2   0097'CF   DE  0043   312         MOVAL   READ_COMPLETE,IRP$L_PID(R2)     ; Store completion address in PID
     55   0C A3   D0  0049   313         MOVL    CLUDCB$L_UCB(R3),R5             ; Get UCB address
     1C A2   55   D0  004D   314         MOVL    R5,IRP$L_UCB(R2)                ; Store UCB address
     20 A2   0C   B0  0051   315         MOVW    #IO$_READPBLK,IRP$W_FUNC(R2)    ; Store function code
        2A A2     B4  0055   316         CLRW    IRP$W_STS(R2)                   ; Mount verification bit may be set
  06 68 A5   02   E0  0058   317         BBS     #UCB$V_NOCNVRT,UCB$W_DEVSTS(R5),1$ ; Br if logical I/O
  2A A2   0100 8F   B0  005D   318         MOVW    #IRP$M_PHYSIO,IRP$W_STS(R2)     ; Set physical I/O flag in IRP
  32 A2   0204 8F   3C  0063   319 1$:     MOVZWL  #CLUQF$K_LENGTH,IRP$L_BCNT(R2)  ; Store byte count
30 A2   56   FE00 8F   AB  0069   320         BICW3   #^C<VA$M_BYTE>,R6,-             ; Store buffer start byte offset
                        0070   321                 IRP$W_BOFF(R2)
  51   56   15   09   EF  0070   322         EXTZV   #VA$V_VPN,#VA$S_VPN,R6,R1       ; Get buffer virtual page number
   50   00000000'GF   D0  0075   323         MOVL    G^MMG$GL_SPTBASE,R0             ; Get SPT base address
     2C A2   6041   DE  007C   324         MOVAL   (R0)[R1],IRP$L_SVAPTE(R2)       ; Store PTE address
        50   1C A3   D0  0081   325         MOVL    CLUDCB$L_QFLBN(R3),R0           ; Get logical block number
             53   52   D0  0085   326         MOVL    R2,R3                           ; Set up IRP address
        00000000'GF   16  0088   327         JSB     G^IOC$CVTLOGPHY                 ; Convert LBN to PBN
        00000000'GF   16  008E   328         JSB     G^EXE$INSIOQ                    ; Queue the request
                 38   BA  0094   329         POPR    #^M<R3,R4,R5>                   ; Restore registers
                      05  0096   330         RSB
```

```
                                0097   332 .SBTTL  READ_COMPLETE - Quorum file read complete
                                0097   333 ;++
                                0097   334 ; READ_COMPLETE - Quorum file read complete
                                0097   335 ;
                                0097   336 ; FUNCTIONAL DESCRIPTION:
                                0097   337 ;
                                0097   338 ;       This routine is called when the quorum file read completes.
                                0097   339 ;
                                0097   340 ; CALLING SEQUENCE:
                                0097   341 ;
                                0097   342 ;       JSB READ_COMPLETE
                                0097   343 ;
                                0097   344 ;           Called as a fork process by IOCIOPOST at IPL$_IOPOST
                                0097   345 ;
                                0097   346 ; INPUTS:
                                0097   347 ;
                                0097   348 ;       R5 = address of IRP
                                0097   349 ;
                                0097   350 ; OUTPUT:
                                0097   351 ;
                                0097   352 ;       R0-R5 destroyed
                                0097   353 ;--
                                0097   354
                                0097   355 READ_COMPLETE::
          00C0 8F      BB       0097   356         PUSHR   #^M<R6,R7>                      ; Save registers
       54   1C A5      D0       009B   357         MOVL    IRP$L_UCB(R5),R4                ; Get UCB address
            6A A4      B7       009F   358         DECW    UCB$W_QLEN(R4)                  ; Decrement device queue length
                               00A2   359         SETIPL  #IPL$_TIMER                     ; Raise IPL
    54  00000000'GF   D0       00A5   360         MOVL    G^CLU$GL_CLUB,R4                ; Get CLUB address
    53    00B4 C4     D0       00AC   361         MOVL    CLUB$L_CLUDCB(R4),R3            ; Get CLUDCB address
       56   25 A3     DE       00B1   362         MOVAL   CLUDCB$T_BUFFER(R3),R6          ; Get quorum file buffer
            02 A3     AA       00B5   363         BICW2   #CLUDCB$M_QF_RIP,-              ; Clear read in progress bit
                               00B7   364                 CLUDCB$W_FLAGS(R3)
    50  0000'CF       9E       00B9   365         MOVAB   W^QDRDERROR_MSG,R0              ; Assume read error
          00 E5       00BE   366         BBCC    #CLUDCB$V_QF_TIM,-              ; Br if read has not timed out
       13 22 A3               00C0   367                 CLUDCB$W_FLAGS(R3),10$
       50 38 A5       E8      00C3   368         BLBS    IRP$L_IOST1(R5),40$             ; Br if read was successful
             035C     30      00C7   369         BSBW    CHECK_ERROR                     ; Is error fatal?
          4A 50       E8      00CA   370         BLBS    R0,40$                          ; Continue
             01       B0      00CD   371         MOVW    #CLUDCB$M_QS_NOT_READY,-        ; Set state to not ready
          20 A3               00CF   372                 CLUDCB$W_STATE(R3)
            0340     30      00D1   373         BSBW    REQUEST_CSP
              41     11      00D4   374         BRB     40$
       15 38 A5       E8      00D6   375 10$:   BLBS    IRP$L_IOST1(R5),14$             ; Br if no read error
             0349     30      00DA   376         BSBW    CHECK_ERROR                     ; Is error fatal?
          37 50       E8      00DD   377         BLBS    R0,40$                          ; Continue
    50  0000'CF       9E      00E0   378         MOVAB   W^QDRDERROR_MSG,R0              ; Read error
             05       E2      00E5   379         BBSS    #CLUDCB$V_QF_FIRST_ERR,-        ; Is this first error
       15 22 A3               00E7   380                 CLUDCB$W_FLAGS(R3),20$
            02E8     30      00EA   381         BSBW    QUORUM_FILE_RETRY               ; Process error
              28     11      00ED   382         BRB     40$
             05       E5      00EF   383 14$:   BBCC    #CLUDCB$V_QF_FIRST_ERR,-        ; Clear any previous error
       00 22 A3               00F1   384                 CLUDCB$W_FLAGS(R3),15$
            0260     30      00F4   385 15$:   BSBW    VALIDATE_QUORUM_FILE
          0A 50       E8      00F7   386         BLBS    R0,30$                          ; Br if quorum file valid
    50  0000'CF       9E      00FA   387         MOVAB   W^QDINVDAT_MSG,R0               ; Point to invalid data message
            02D8     30      00FF   388 20$:   BSBW    QUORUM_FILE_ERROR               ; Process error
```

```
                13   11  0102   389          BRB     40$
          04    01   EA  0104   390 30$:     FFS     #CLUDCB$V_QS_READY,#4,-          ; Get relative state bit position
       50    20 A3       0107   391                  CLUDCB$W_STATE(R3),R0
     51    011F'CF   DE  010A   392          MOVAL   DISPATCH,R1                      ; Get dispatch table address
  51    011B'CF40   C0  010F   393          ADDL2   DISPATCH-4[R0],R1                ; Form routine address
                61   16  0115   394          JSB     (R1)                             ; Dispatch to routine
                         0117   395 40$:     SETIPL  #IPL$_IOPOST                     ; Restore IPL
           00C0 8F   BA  011A   396          POPR    #^M<R6,R7>                       ; Restore registers
                05   011E   397          RSB
                         011F   398
          00000010' 011F   399 DISPATCH:        .LONG   READ_COMPLETE_READY-DISPATCH
          0000004F' 0123   400                  .LONG   READ_COMPLETE_ACTIVE-DISPATCH
          00000087' 0127   401                  .LONG   READ_COMPLETE_CLUSTER-DISPATCH
          00000087' 012B   402                  .LONG   READ_COMPLETE_VOTE-DISPATCH
```

```
                     012F    404 .SBTTL  READ_COMPLETE_READY - Read complete processing for READY state
                     012F    405 ;++
                     012F    406 ; READ_COMPLETE_READY - Read complete processing for READY state
                     012F    407 ;
                     012F    408 ; FUNCTIONAL DESCRIPTION:
                     012F    409 ;
                     012F    410 ;       This routine performs the read complete processing specific
                     012F    411 ;       to the READY state.
                     012F    412 ;
                     012F    413 ; CALLING SEQUENCE:
                     012F    414 ;
                     012F    415 ;       JSB/BSBx READ_COMPLETE_READY
                     012F    416 ;
                     012F    417 ; INPUTS:
                     012F    418 ;
                     012F    419 ;       R3 = address of CLUDCB
                     012F    420 ;       R4 = address of CLUB
                     012F    421 ;       R6 = address of quorum file buffer
                     012F    422 ;
                     012F    423 ; OUTPUT:
                     012F    424 ;
                     012F    425 ;       R0-R2,R5 Destroyed
                     012F    426 ;--
                     012F    427
                     012F    428 READ_COMPLETE_READY:
                     012F    429
          04   B0    012F    430         MOVW    #CLUDCB$M_QS_ACTIVE,-           ; Set state to active
        20 A3          0131    431                 CLUDCB$W_STATE(R3)
          08   AA    0133    432         BICW    #CLUDCB$M_QF_ERROR,-            ; Clear error reported bit
        22 A3          0135    433                 CLUDCB$W_FLAGS(R3)
      0200 C6   D0    0137    434         MOVL    CLUQF$L_ACT_COUNT(R6),-         ; Save activity longword
        18 A3          013B    435                 CLUDCB$L_ACT_COUNT(R3)
  00C8 C4     00    D2    013D    436         MCOML   #0,CLUB$L_FOREIGN_CLUSTER(R4)   ; Fill shift register with 1's
          02   C8    0142    437         BISL    #CLUB$M_QF_ACTIVE,-             ; Set active bit
        1C A4          0144    438                 CLUB$L_FLAGS(R4)
  50  0000'CF   9E    0146    439         MOVAB   W^QDCON_MSG,R0                  ; Point to connect message
          55   D4    014B    440         CLRL    R5                              ; No CSB
        FEB0'   30    014D    441         BSBW    CNX$CONFIG_CHANGE               ; Output message
        FEAD'   30    0150    442         BSBW    CNX$DISK_CHANGE                 ; Let connection manager know
          00   E1    0153    443         BBC     #CLUB$V_CLUSTER,-               ; Br if local node not a
     15 1C A4          0155    444                 CLUB$L_FLAGS(R4),1$             ; ...cluster member
          08   B0    0158    445         MOVW    #CLUDCB$M_QS_CLUSTER,-          ; Set state to cluster
        20 A3          015A    446                 CLUDCB$W_STATE(R3)
        24 A3    94    015C    447         CLRB    CLUDCB$B_COUNTER(R3)            ; Clear counter
01000000 8F      CA    015F    448         BICL    #CLUB$M_QF_FAILED_NODE,-        ; Clear failout bit in CLUB
        1C A4          0165    449                 CLUB$L_FLAGS(R4)
        008B    30    0167    450         BSBW    BUILD_QUORUM_FILE               ; Build the owner & activity blocks
        00F2    30    016A    451         BSBW    WRITE_QUORUM_OWNACT             ; Write the owner & activity blocks
                05    016D    452 1$:     RSB
```

```
                     016E    454 .SBTTL  READ_COMPLETE_ACTIVE - Read complete processing for ACTIVE state
                     016E    455 ;++
                     016E    456 ; READ_COMPLETE_ACTIVE - Read complete processing for ACTIVE state
                     016E    457 ;
                     016E    458 ; FUNCTIONAL DESCRIPTION:
                     016E    459 ;
                     016E    460 ;       This routine performs the read complete processing specific
                     016E    461 ;       to the ACTIVE state.
                     016E    462 ;
                     016E    463 ; CALLING SEQUENCE:
                     016E    464 ;
                     016E    465 ;       JSB/BSBx READ_COMPLETE_ACTIVE
                     016E    466 ;
                     016E    467 ; INPUTS:
                     016E    468 ;
                     016E    469 ;       R3 = address of CLUDCB
                     016E    470 ;       R4 = address of CLUB
                     016E    471 ;       R6 = address of quorum file buffer
                     016E    472 ;
                     016E    473 ; OUTPUT:
                     016E    474 ;
                     016E    475 ;       R0-R2 Destroyed
                     016E    476 ;--
                     016E    477
                     016E    478 READ_COMPLETE_ACTIVE:
                     016E    479
          00   E1    016E    480         BBC     #CLUB$V_CLUSTER,-               ; Br if local node not a
 17 1C A4              0170    481                 CLUB$L_FLAGS(R4),1$             ; ...cluster member
          08   B0    0173    482         MOVW    #CLUDCB$M_QS_CLUSTER,-          ; Set state to cluster
    20 A3              0175    483                 CLUDCB$W_STATE(R3)
    24 A3        94    0177    484         CLRB    CLUDCB$B_COUNTER(R3)            ; Clear counter
01000000 8F      CA    017A    485         BICL    #CLUB$M_QF_FAILED_NODE,-        ; Clear failout bit in CLUB
       1C A4              0180    486                 CLUB$L_FLAGS(R4)
        0070      30    0182    487         BSBW    BUILD_QUORUM_FILE               ; Build the owner & activity blocks
        00D7      30    0185    488         BSBW    WRITE_QUORUM_OWNACT             ; Write the owner & activity blocks
          1B      11    0188    489         BRB     2$
00C8 C4   01      78    018A    490 1$:     ASHL    #1,CLUB$L_FOREIGN_CLUSTER(R4),- ; Assume no activity
    00C8 C4               018F    491                 CLUB$L_FOREIGN_CLUSTER(R4)
    0200 C6   D1    0192    492         CMPL    CLUQF$L_ACT_COUNT(R6),-         ; Activity longword change?
      18 A3               0196    493                 CLUDCB$L_ACT_COUNT(R3)
          0B    13    0198    494         BEQLU   2$                              ; Br if not
00C8 C4   01      C8    019A    495         BISL    #1,CLUB$L_FOREIGN_CLUSTER(R4)   ; We have seen a foreign cluster
    0200 C6   D0    019F    496         MOVL    CLUQF$L_ACT_COUNT(R6),-         ; Save activity longword
      18 A3               01A3    497                 CLUDCB$L_ACT_COUNT(R3)
          05    05    01A5    498 2$:     RSB
```

```
                     01A6   500 .SBTTL  READ_COMPLETE_CLUSTER/VOTE - Read complete processing for CLUSTER and VOTE s
                     01A6   501 ;++
                     01A6   502 ; READ_COMPLETE_CLUSTER - Read complete processing for CLUSTER state
                     01A6   503 ; READ_COMPLETE_VOTE - Read complete processing for VOTE state
                     01A6   504 ;
                     01A6   505 ; FUNCTIONAL DESCRIPTION:
                     01A6   506 ;
                     01A6   507 ;       This routine performs the read complete processing specific
                     01A6   508 ;       to the CLUSTER and VOTE states.
                     01A6   509 ;
                     01A6   510 ; CALLING SEQUENCE:
                     01A6   511 ;
                     01A6   512 ;       JSB/BSBx READ_COMPLETE_CLUSTER
                     01A6   513 ;       JSB/BSBx READ_COMPLETE_VOTE
                     01A6   514 ;
                     01A6   515 ; INPUTS:
                     01A6   516 ;
                     01A6   517 ;       R3 = address of CLUDCB
                     01A6   518 ;       R4 = address of CLUB
                     01A6   519 ;       R6 = address of quorum file buffer
                     01A6   520 ;
                     01A6   521 ; OUTPUT:
                     01A6   522 ;
                     01A6   523 ;       R0-R2,R5 Destroyed
                     01A6   524 ;--
                     01A6   525
                     01A6   526 READ_COMPLETE_CLUSTER:
                     01A6   527 READ_COMPLETE_VOTE:
                     01A6   528
          18   E5    01A6   529          BBCC    #CLUB$V_QF_FAILED_NODE,-        ; Br if node was not failed out
06 1C A4             01A8   530                  CLUB$L_FLAGS(R4),T$
          08   B0    01AB   531          MOVW    #CLUDCB$M_QS_CLUSTER,-          ; Set state to CLUSTER
     20 A3             01AD   532                  CLUDCB$W_STATE(R3)
          3A   11    01AF   533          BRB     4$
     48 A6   95    01B1   534 1$:       TSTB    CLUQF$B_IGNORE(R6)              ; Is data in quorum file stale?
          35   12    01B4   535          BNEQU   4$                              ; Br if yes
       01C1   30    01B6   536          BSBW    CHECK_OWNER                     ; Determine who owns quorum file
     08 50    E9    01B9   537          BLBC    R0,2$                           ; Br if not a member of my cluster
     24 A3    96    01BC   538          INCB    CLUDCB$B_COUNTER(R3)            ; Increment counter
      00AA    30    01BF   539          BSBW    WRITE_QUORUM_ACT                ; Write the activity block
          30   11    01C2   540          BRB     5$
50  0000'CF   9E    01C4   541 2$:       MOVAB   W^QDFORCLUS_MSG,R0              ; Point to foreign cluster message
          55   D4    01C9   542          CLRL    R5                              ; No CSB
      FE32'    30    01CB   543          BSBW    CNX$CONFIG_CHANGE               ; Output message
          00   E0    01CE   544          BBS     #CLUQF$V_QUORUM,-               ; Bugcheck if he has dynamic quorum
13 0E A6             01D0   545                  CLUQF$W_FLAGS(R6),3$
          1C   E0    01D3   546          BBS     #CLUB$V_QUORUM,-                ; Continue if we have dynamic quorum
13 1C A4             01D5   547                  CLUB$L_FLAGS(R4),4$
     36 A6   B1    01D8   548          CMPW    CLUQF$W_VOTES(R6),-             ; Does he have static quorum?
     34 A6             01DB   549                  CLUQF$W_QUORUM(R6)
          07   1E    01DD   550          BGEQU   3$                              ; Br if yes
     22 A4   B1    01DF   551          CMPW    CLUB$W_VOTES(R4),-              ; Do we have static quorum?
     20 A4             01E2   552                  CLUB$W_QUORUM(R4)
          05   1E    01E4   553          BGEQU   4$                              ; Br if yes
      FE17'    30    01E6   554 3$:       BSBW    CNX$BUGCHECK_CLUSTER            ; Cause all nodes to bugcheck
          09   11    01E9   555          BRB     5$
     24 A3    94    01EB   556 4$:       CLRB    CLUDCB$B_COUNTER(R3)            ; Clear counter
```

```
0004   30  01EE   557          BSBW    BUILD_QUORUM_FILE          ; Build the owner & activity blocks
006B   30  01F1   558          BSBW    WRITE_QUORUM_OWNACT        ; Write the owner & activity blocks
       05  01F4   559 5$:      RSB
```

```
                        01F5   561 .SBTTL  BUILD_QUORUM_FILE - Build the quorum file owner and activity blocks
                        01F5   562 ;++
                        01F5   563 ; BUILD_QUORUM_FILE - Build the quorum file owner and activity blocks
                        01F5   564 ;
                        01F5   565 ; FUNCTIONAL DESCRIPTION:
                        01F5   566 ;
                        01F5   567 ;       This routine builds the quorum file owner and activity blocks.
                        01F5   568 ;
                        01F5   569 ; CALLING SEQUENCE:
                        01F5   570 ;
                        01F5   571 ;       JSB/BSBx BUILD_QUORUM_FILE
                        01F5   572 ;
                        01F5   573 ; INPUTS:
                        01F5   574 ;
                        01F5   575 ;       R4 = address of CLUB
                        01F5   576 ;       R6 = address of quorum file buffer
                        01F5   577 ;
                        01F5   578 ; OUTPUT:
                        01F5   579 ;
                        01F5   580 ;       R0-R2 destroyed
                        01F5   581 ;--
                        01F5   582
                        01F5   583 BUILD_QUORUM_FILE:
                        01F5   584
                        01F5   585         ASSUME  CLUQF$K_VERSION EQ 2            ; Assume version 2 structure
                        01F5   586         ASSUME  SB$S_SYSTEMID EQ 6              ; Assume system ID is 6 bytes
        00B8 8F    BB   01F5   587         PUSHR   #^M<R3,R4,R5,R7>                ; Save registers
             0C    28   01F9   588         MOVC3   #CLUQF$S_IDENT,-                ; Store ID string
66      0000'CF          01FB   589                 CLUQF_IDENT_STRING,(R6)
        50    53   D0   01FF   590         MOVL    R3,R0                           ; R0 = current buffer pointer
        53    6E   7D   0202   591         MOVQ    (SP),R3                         ; Restore CLUDCB and CLUB pointers
        80    02   B0   0205   592         MOVW    #CLUQF$K_VERSION,(R0)+          ; Store QF version number
        80    01   B0   0208   593         MOVW    #CLUQF$M_QUORUM,(R0)+           ; Assume we have dynamic quorum
              1C   E0   020B   594         BBS     #CLUB$V_QUORUM,-                ; Br if we do have dynamic quorum
           03 1C   A4         020D   595                 CLUB$L_FLAGS(R4),1$
              FE   A0   B4   0210   596         CLRW    -2(R0)                      ; Fix the incorrect assumption
        80    2C   A4   7D   0213   597 1$:     MOVQ    CLUB$Q_FTIME(R4),(R0)+      ; Store FOU_TIME
        80    3C   A4   7D   0217   598         MOVQ    CLUB$Q_LST_TIME(R4),(R0)+   ; Store LST_TIME
80      00000000'GF      7D   021B   599         MOVQ    G^EXE$GQ_SYSTIME,(R0)+      ; Store QF_TIME
80      0000002C'GF      7D   0222   600         MOVQ    G^SCS$GA_LOCALSB+SB$Q_SWINCARN,(R0)+ ; Store SWINCARN
        80    60   A4   D0   0229   601         MOVL    CLUB$L_LOCAL_CSID(R4),(R0)+ ; Store CSID
        80    20   A4   B0   022D   602         MOVW    CLUB$W_QUORUM(R4),(R0)+     ; Store cluster quorum
        80    22   A4   B0   0231   603         MOVW    CLUB$W_VOTES(R4),(R0)+      ; Store cluster votes
80      00000018'GF      D0   0235   604         MOVL    G^SCS$GA_LOCALSB+SB$B_SYSTEMID,(R0)+ ; Store system ID
80      0000001C'GF      B0   023C   605         MOVW    G^SCS$GA_LOCALSB+SB$B_SYSTEMID+4,(R0)+
        80    26   A4   D0   0243   606         MOVL    CLUB$B_FSYSID(R4),(R0)+     ; Store FSYSID
        80    2A   A4   B0   0247   607         MOVW    CLUB$B_FSYSID+4(R4),(R0)+
                    60   D4   024B   608         CLRL    (R0)                        ; Initialize checksum
                 04 A0   94   024D   609         CLRB    4(R0)                       ; Zero the ignore flag
                   016F   30   0250   610         BSBW    CALCULATE_CHECKSUM          ; Calculate the owner block checksum
                 60    57   D0   0253   611         MOVL    R7,(R0)                     ; Store checksum
                 0200 C6    D6   0256   612         INCL    CLUQF$L_ACT_COUNT(R6)       ; Increment the activity counter
                 00B8 8F    BA   025A   613         POPR    #^M<R3,R4,R5,R7>            ; Restore registers
                            05   025E   614         RSB
```

```
                     025F   616 .SBTTL  Quorum file write routines
                     025F   617 ;++
                     025F   618 ; WRITE_QUORUM_OWNACT - Write the quorum file owner and activity blocks
                     025F   619 ; WRITE_QUORUM_ACT - Write the quorum file activity block
                     025F   620 ;
                     025F   621 ; FUNCTIONAL DESCRIPTION:
                     025F   622 ;
                     025F   623 ;       This routine builds and queues an IRP to write the owner and activity
                     025F   624 ;       block or just the activity block to the quorum file.
                     025F   625 ;
                     025F   626 ; CALLING SEQUENCE:
                     025F   627 ;
                     025F   628 ;       JSB/BSBx WRITE_QUORUM_OWNACT
                     025F   629 ;       JSB/BSBx WRITE_QUORUM_ACT
                     025F   630 ;
                     025F   631 ; INPUTS:
                     025F   632 ;
                     025F   633 ;       R3 = address of CLUDCB
                     025F   634 ;       R6 = address of quorum file buffer
                     025F   635 ;
                     025F   636 ; OUTPUT:
                     025F   637 ;
                     025F   638 ;       R0-R2 destroyed
                     025F   639 ;--
                     025F   640         .ENABLE LSB
                     025F   641
                     025F   642 WRITE_QUORUM_OWNACT:
                     025F   643
          0078 8F    BB  025F   644         PUSHR   #^M<R3,R4,R5,R6>                ; Save registers
               7E    D4  0263   645         CLRL    -(SP)                           ; Quorum file block 0
     7E   0204 8F    3C  0265   646         MOVZWL  #CLUQF$K_LENGTH,-(SP)           ; Byte count
               11    11  026A   647         BRB     1$
                         026C   648
                         026C   649 WRITE_QUORUM_ACT:
                         026C   650
          0078 8F    BB  026C   651         PUSHR   #^M<R3,R4,R5,R6>                ; Save registers
     56   0200 C6    DE  0270   652         MOVAL   CLUQF$L_ACT_COUNT(R6),R6        ; Get activity block address
                    66  D6  0275   653         INCL    (R6)                            ; Increment the activity counter
                    7E  01  9A  0277   654         MOVZBL  #1,-(SP)                        ; Quorum file block 1
                    7E  04  9A  027A   655         MOVZBL  #CLUQF$K_ACT_LENGTH,-(SP)       ; Byte count
                    04  A8  027D   656 1$:      BISW    #CLUDCB$M_QF_WIP,-              ; Set write in progress bit
                      22  A3  027F   657                 CLUDCB$W_FLAGS(R3)
                52  10  A3  D0  0281   658         MOVL    CLUDCB$L_IRP(R3),R2             ; Get IRP address
              0C A2 02DA'CF  DE  0285   659         MOVAL   WRITE_COMPLETE,IRP$L_PID(R2)    ; Store completion address in PID
                55  0C  A3  D0  028B   660         MOVL    CLUDCB$L_UCB(R3),R5             ; Get UCB address
                1C A2   55  D0  028F   661         MOVL    R5,IRP$L_UCB(R2)                ; Store UCB address
                20 A2   0B  B0  0293   662         MOVW    #IO$_WRITEPBLK,IRP$W_FUNC(R2)   ; Store function code
                  2A A2     B4  0297   663         CLRW    IRP$W_STS(R2)                   ; Mount verification bit may be set
             06 68 A5   02  E0  029A   664         BBS     #UCB$V_NOCNVRT,UCB$W_DEVSTS(R5),2$ ; Br if logical I/O
           2A A2   0100 8F  B0  029F   665         MOVW    #IRP$M_PHYSIO,IRP$W_STS(R2)     ; Set physical I/O flag in IRP
                32 A2   8E  D0  02A5   666 2$:      MOVL    (SP)+,IRP$L_BCNT(R2)            ; Store byte count
        30 A2   56   FE00 8F  AB  02A9   667         BICW3   #^C<VA$M_BYTE>,R6,-             ; Store buffer start byte offset
                                  02B0   668                 IRP$W_BOFF(R2)
           51   56   15   09  EF  02B0   669         EXTZV   #VA$V_VPN,#VA$S_VPN,R6,R1       ; Get buffer virtual page number
            50   00000000'GF  D0  02B5   670         MOVL    G^MMG$GL_SPTBASE,R0             ; Get SPT base address
                2C A2   6041  DE  02BC   671         MOVAL   (R0)[R1],IRP$L_SVAPTE(R2)       ; Store PTE address
              50   1C A3   8E  C1  02C1   672         ADDL3   (SP)+,CLUDCB$L_QFLBN(R3),R0     ; Get logical block number
```

```
                53    52  D0  02C6   673          MOVL    R2,R3                           ; Set up IRP address
00000000'GF         16  02C9   674          JSB     G^IOC$CVTLOGPHY                 ; Convert LBN to PBN
00000000'GF         16  02CF   675          JSB     G^EXE$INSIOQ                    ; Queue the request
              0078  8F  BA  02D5   676          POPR    #^M<R3,R4,R5,R6>                ; Restore registers
                      05  02D9   677          RSB
                          02DA   678
                          02DA   679          .DISABLE LSB
```

```
                                02DA   681 .SBTTL  WRITE_COMPLETE - Quorum file write complete
                                02DA   682 ;++
                                02DA   683 ; WRITE_COMPLETE - Quorum file write complete
                                02DA   684 ;
                                02DA   685 ; FUNCTIONAL DESCRIPTION:
                                02DA   686 ;
                                02DA   687 ;       This routine is called when a quorum file write completes.
                                02DA   688 ;
                                02DA   689 ; CALLING SEQUENCE:
                                02DA   690 ;
                                02DA   691 ;       JSB WRITE_COMPLETE
                                02DA   692 ;
                                02DA   693 ;           Called as a fork process by IOCIOPOST at IPL$_IOPOST
                                02DA   694 ;
                                02DA   695 ; INPUTS:
                                02DA   696 ;
                                02DA   697 ;       R5 = address of IRP
                                02DA   698 ;
                                02DA   699 ; OUTPUT:
                                02DA   700 ;
                                02DA   701 ;       R0-R4 destroyed
                                02DA   702 ;--
                                02DA   703
                                02DA   704 WRITE_COMPLETE::
          54    1C A5     D0    02DA   705          MOVL    IRP$L_UCB(R5),R4                ; Get UCB address
                6A A4     B7    02DE   706          DECW    UCB$W_QLEN(R4)                  ; Decrement device queue length
                                02E1   707          SETIPL  #IPL$_TIMER                     ; Raise IPL
       54    00000000'GF     D0    02E4   708          MOVL    G^CLU$GL_CLUB,R4                ; Get CLUB address
             53    00B4 C4     D0    02EB   709          MOVL    CLUB$L_CLUDCB(R4),R3            ; Get CLUDCB address
                     04     AA    02F0   710          BICW2   #CLUDCB$M_QF_WIP,-              ; Clear write in progress bit
                   22 A3           02F2   711                  CLUDCB$W_FLAGS(R3)
          50    0000'CF     9E    02F4   712          MOVAB   W^QDWRERROR_MSG,R0              ; Point to write error message
                     00     E5    02F9   713          BBCC    #CLUDCB$V_QF_TIM,-              ; Br if write has not timed out
                13 22 A3           02FB   714                  CLUDCB$W_FLAGS(R3),10$
                51 38 A5     E8    02FE   715          BLBS    IRP$L_IOST1(R5),30$             ; Br if write was successful
                   0121      30    0302   716          BSBW    CHECK_ERROR                     ; Is error fatal?
                   4B 50     E8    0305   717          BLBS    R0,30$                          ; Continue
                     01      B0    0308   718          MOVW    #CLUDCB$M_QS_NOT_READY,-        ; Set state to not ready
                   20 A3           030A   719                  CLUDCB$W_STATE(R3)
                   0105      30    030C   720          BSBW    REQUEST_CSP                     ; Request the CSP process
                     42      11    030F   721          BRB     30$
                1A 38 A5     E8    0311   722 10$:     BLBS    IRP$L_IOST1(R5),20$             ; Br if write success
                   010E      30    0315   723          BSBW    CHECK_ERROR                     ; Is error fatal?
                   38 50     E8    0318   724          BLBS    R0,30$                          ; Continue
          50    0000'CF     9E    031B   725          MOVAB   W^QDWRERROR_MSG,R0              ; Point to write error message
                     05     E2    0320   726          BBSS    #CLUDCB$V_QF_FIRST_ERR,-        ; Is this first error
                05 22 A3           0322   727                  CLUDCB$W_FLAGS(R3),15$
                   00AD      30    0325   728          BSBW    QUORUM_FILE_RETRY               ; Process error (retry)
                     29      11    0328   729          BRB     30$
                   00AD      30    032A   730 15$:     BSBW    QUORUM_FILE_ERROR               ; Process error
                     24      11    032D   731          BRB     30$
                             032F   732 20$:     ASSUME  CLUDCB$M_QF_WRL_ERR LE 255
                   40 8F     8A    032F   733          BICB    #CLUDCB$M_QF_WRL_ERR,-          ; Not write locked
                   22 A3           0332   734                  CLUDCB$W_FLAGS(R3)
                     04     E0    0334   735          BBS     #CLUDCB$V_QS_VOTE,-             ; Br if state = VOTE
                1A 20 A3           0336   736                  CLUDCB$W_STATE(R3),30$
                     02     91    0339   737          CMPB    #CYCLE_COUNT,-                  ; Have we cycled enough?
```

```
        24 A3       033B   738                   CLUDCB$B_COUNTER(R3)
           14    12 033D   739          BNEQU    30$                          ; Br if not
           18    E0 033F   740          BBS      #CLUB$V_QF_FAILED_NODE,-     ; Br if a node has been failed out
     0F 1C A4       0341   741                   CLUB$L_FLAGS(R4),30$
           10    B0 0344   742          MOVW     #CLUDCB$M_QS_VOTE,-          ; Set state to VOTE
        20 A3       0346   743                   CLUDCB$W_STATE(R3)
  40000000 8F    C8 0348   744          BISL     #CLUB$M_QF_DYNVOTE,-         ; Set dynamic vote bit in CLUB
        1C A4       034E   745                   CLUB$L_FLAGS(R4)
        FCAD'    30 0350   746          BSBW     CNX$DISK_CHANGE              ; Let connection manager know
                    0353   747 30$:     SETIPL   #IPL$_IOPOST                 ; Restore IPL
                 05 0356   748          RSB
```

```
                     0357   750 .SBTTL  VALIDATE_QUORUM_FILE - Validate quorum file
                     0357   751 ;++
                     0357   752 ; VALIDATE_QUORUM_FILE - Validate quorum file
                     0357   753 ;
                     0357   754 ; FUNCTIONAL DESCRIPTION:
                     0357   755 ;
                     0357   756 ;       This routine validates the contents of the quorum file.
                     0357   757 ;
                     0357   758 ; CALLING SEQUENCE:
                     0357   759 ;
                     0357   760 ;       JSB/BSBx VALIDATE_QUORUM_FILE
                     0357   761 ;
                     0357   762 ; INPUTS:
                     0357   763 ;
                     0357   764 ;       R6 = address of quorum file buffer
                     0357   765 ;
                     0357   766 ; OUTPUT:
                     0357   767 ;
                     0357   768 ;       R0 = status
                     0357   769 ;            0 - The block is invalid
                     0357   770 ;            1 - The block is valid
                     0357   771 ;
                     0357   772 ;       R1-R2 destroyed
                     0357   773 ;--
                     0357   774
                     0357   775 VALIDATE_QUORUM_FILE:
                     0357   776
0088 8F    BB        0357   777         PUSHR   #^M<R3,R7>                      ; Save CLUDCB
     7E    D4        035B   778         CLRL    -(SP)                           ; Assume invalid buffer
  0062     30        035D   779         BSBW    CALCULATE_CHECKSUM              ; Calculate quorum file checksum
     57    D5        0360   780         TSTL    R7                              ; Is checksum valid?
     11    12        0362   781         BNEQU   1$                              ; Br if not
     0C    29        0364   782         CMPC3   #CLUQF$S_IDENT,-                ; Validate ID area
     66              0366   783                 CLUQF$T_IDENT(R6),-
0000'CF              0367   784                 CLUQF_IDENT_STRING
     09    12        036A   785         BNEQU   1$                              ; Br if invalid
     02    B1        036C   786         CMPW    #CLUQF$K_VERSION,-              ; Is version correct?
   0C A6              036E   787                 CLUQF$W_VERSION(R6)
     03    12        0370   788         BNEQU   1$                              ; Br if not
6E   01    D0        0372   789         MOVL    #1,(SP)                         ; Indicate success
0089 8F    BA        0375   790 1$:      POPR    #^M<R0,R3,R7>                   ; Return status and restore register
           05        0379   791         RSB
```

```
                                ;
                                ; CHECK_OWNER - Check quorum file ownership
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine checks the quorum file owner block to see if it
                                ;       is owned by a member of this nodes cluster.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB/BSBx CHECK_OWNER
                                ;
                                ; INPUTS:
                                ;
                                ;       R4 = address of CLUB
                                ;       R6 = address of quorum file buffer
                                ;
                                ; OUTPUT:
                                ;
                                ;       R0 = Status
                                ;            0 - Quorum file is owned by a foreign cluster
                                ;            1 - Quorum file is owned by my cluster
                                ;
                                ;       R1-R2 Destroyed
                                ;--
                                
                                CHECK_OWNER:
                                
                                        PUSHL   R3                              ; Save CLUDCB
                                        CLRL    -(SP)                           ; Assume foreign cluster
    14 A6       30 A4       D1      CMPL    CLUB$Q_FTIME+4(R4), -               ; Same high order foundation times?
                                                CLUQF$Q_FOU_TIME+4(R6)
                                        BNEQU   1$                              ; Br if not
    10 A6       2C A4       D1      CMPL    CLUB$Q_FTIME(R4), -                 ; Same low order foundation times?
                                                CLUQF$Q_FOU_TIME(R6)
                                        BNEQU   1$                              ; Br if not
                                        CMPC3   #CLUQF$S_FSYSID,-               ; Same founding system ID's?
                                26 A4           CLUB$B_FSYSID(R4),-
                                3E A6           CLUQF$B_FSYSID(R6)
                                        BNEQU   1$                              ; Br if not
          51    30 A6       3C      MOVZWL  CLUQF$W_CSID_IDX(R6),R1             ; Get CSID index
00000000'GF     51          B1      CMPW    R1,G^CLU$GW_MAXINDEX                ; Is index in range?
                            1E      BGEQU   1$                                  ; Br if not
50  00000000'GF             D0      MOVL    G^CLU$GL_CLUSVEC,R0                 ; Get vector address
       50   6041            D0      MOVL    (R0)[R1],R0                         ; Get entry (should be CSB address)
                            11      BGEQ    1$                                  ; Br if no entry
                4C A0       D1      CMPL    CSB$L_CSID(R0),-                    ; Do CSID's match?
                30 A6               CLUQF$L_CSID(R6)
                            0A      BNEQ    1$                                  ; Br if not
                38 A0       D1      CMPL    CSB$Q_SWINCARN(R0),-                ; Incarnation numbers match?
                28 A6               CLUQF$Q_SWINCARN(R6)
                            03      BNEQU   1$                                  ; Br if not
             6E             01      MOVL    #1,(SP)                             ; Quorum file is owned by my cluster
                            09      POPR    #^M<R0,R3>                          ; Restore CLUDCB
                            05      RSB
```

```
                     03C2    850 .SBTTL  CALCULATE_CHECKSUM - Calculate the quorum file checksum
                     03C2    851 ;++
                     03C2    852 ; CALCULATE_CHECKSUM - Calculate the quorum file checksum
                     03C2    853 ;
                     03C2    854 ; FUNCTIONAL DESCRIPTION:
                     03C2    855 ;
                     03C2    856 ;       This routine calulates the checksum of the quorum owner block
                     03C2    857 ;       pointed to by R6. It includes the field CLUQF$L_CHECKSUM in the
                     03C2    858 ;       checksum calculation.
                     03C2    859 ;
                     03C2    860 ; CALLING SEQUENCE:
                     03C2    861 ;
                     03C2    862 ;       JSB/BSBx CALCULATE_CHECKSUM
                     03C2    863 ;
                     03C2    864 ; INPUTS:
                     03C2    865 ;
                     03C2    866 ;       R6 = address of quorum file buffer
                     03C2    867 ;
                     03C2    868 ; OUTPUT:
                     03C2    869 ;
                     03C2    870 ;       R7 = Quorum block checksum
                     03C2    871 ;--
                     03C2    872
                     03C2    873 CALCULATE_CHECKSUM:
                     03C2    874
      0C   BB        03C2    875         PUSHR   #^M<R2,R3>                      ; Save registers
52    12   D0        03C4    876         MOVL    #CLUQF$K_CHECK_LENGTH/4,R2      ; R2 = checksum longword count
53    56   D0        03C7    877         MOVL    R6,R3                           ; Copy buffer address
      57   D4        03CA    878         CLRL    R7                              ; Form checksum in R7
57    83   CC        03CC    879 1$:     XORL2   (R3)+,R7                        ; Accumulate checksum
  FA  52   F5        03CF    880         SOBGTR  R2,1$                           ; Br if more
      0C   BA        03D2    881         POPR    #^M<R2,R3>                      ; Restore registers
           05        03D4    882         RSB
```

```
                        03D5    884 .SBTTL  Quorum file error routines
                        03D5    885 ;++
                        03D5    886 ; QUORUM_DISK_TIMEOUT - Quorum disk timeout
                        03D5    887 ; QUORUM_FILE_ERROR - Quorum file error
                        03D5    888 ;
                        03D5    889 ; FUNCTIONAL DESCRIPTION:
                        03D5    890 ;
                        03D5    891 ;       This routine handles timeouts and other errors associated
                        03D5    892 ;       with the quorum disk.
                        03D5    893 ;
                        03D5    894 ; CALLING SEQUENCE:
                        03D5    895 ;
                        03D5    896 ;       JSB/BSBx QUORUM_DISK_TIMEOUT
                        03D5    897 ;       JSB/BSBx QUORUM_FILE_ERROR
                        03D5    898 ;
                        03D5    899 ; INPUTS:
                        03D5    900 ;
                        03D5    901 ;       R0 = address of error message
                        03D5    902 ;       R3 = address of CLUDCB
                        03D5    903 ;       R4 = address of CLUB
                        03D5    904 ;
                        03D5    905 ; OUTPUT:
                        03D5    906 ;
                        03D5    907 ;       R0-R2 destroyed
                        03D5    908 ;--
                        03D5    909         .ENABLE LSB
                        03D5    910
                        03D5    911 QUORUM_DISK_TIMEOUT:
                        03D5    912 QUORUM_FILE_RETRY:
                        03D5    913
 51    02    B0         03D5    914         MOVW    #CLUDCB$M_QS_READY,R1           ; The new state is READY
       0B    11         03D8    915         BRB     1$
                        03DA    916
                        03DA    917 QUORUM_FILE_ERROR:
                        03DA    918
            50    DD    03DA    919         PUSHL   R0                              ; Save error message address
          0035    30    03DC    920         BSBW    REQUEST_CSP                     ; Request the CSP process
       50    8E    D0  03DF    921         MOVL    (SP)+,R0                        ; Restore error message address
       51    01    B0  03E2    922         MOVW    #CLUDCB$M_QS_NOT_READY,R1       ; The new state is not ready
             20    BB  03E5    923 1$:     PUSHR   #^M<R5>                         ; Save R5
             55    D4  03E7    924         CLRL    R5                              ; No CSB (input to CNX$CONFIG_CHANGE
    7E    20 A3    3C  03E9    925         MOVZWL  CLUDCB$W_STATE(R3),-(SP)        ; Save current state
 20 A3    51    B0  03ED    926         MOVW    R1,CLUDCB$W_STATE(R3)           ; Update state
             03    E2  03F1    927         BBSS    #CLUDCB$V_QF_ERROR,-            ; Br if an error has already been re
       03 22 A3          03F3    928                 CLUDCB$W_FLAGS(R3),2$
          FC07'    30  03F6    929         BSBW    CNX$CONFIG_CHANGE               ; Output error message
       8E    03    D3  03F9    930 2$:     BITL    #<CLUDCB$M_QS_NOT_READY! -      ; Was state NOT_READY or READY?
                        03FC    931                   CLUDCB$M_QS_READY>,(SP)+
             13    12  03FC    932         BNEQU   3$                              ; Br if yes
                   CA  03FE    933         BICL    #<CLUB$M_QF_ACTIVE! -           ; Clear the CLUB bits
                        03FF    934                   CLUB$M_QF_DYNVOTE! -
                        03FF    935                   CLUB$M_QF_FAILED_NODE>,-
 1C A4   41000002 8F            03FF    936                 CLUB$L_FLAGS(R4)
       50    0000'CF    9E  0406    937         MOVAB   W^QDDISCON_MSG,R0               ; Point to quorum disk disconnect me
          FBF2'    30  040B    938         BSBW    CNX$CONFIG_CHANGE               ; Output message
          FBEF'    30  040E    939         BSBW    CNX$DISK_CHANGE                 ; Let connection manager know
             20    BA  0411    940 3$:     POPR    #^M<R5>                         ; Restore R5
```

```
05  0413   941          RSB
    0414   942
    0414   943          .DISABLE LSB
```

```
                     0414   945 .SBTTL  REQUEST_CSP - Request the CSP process
                     0414   946 ;++
                     0414   947 ; REQUEST_CSP - Request the CSP process
                     0414   948 ;
                     0414   949 ; FUNCTIONAL DESCRIPTION:
                     0414   950 ;
                     0414   951 ;       If it has not already been requested, this routine requests the
                     0414   952 ;       quorum thread of the CSP process.
                     0414   953 ;
                     0414   954 ; CALLING SEQUENCE:
                     0414   955 ;
                     0414   956 ;       JSB/BSBx REQUEST_CSP
                     0414   957 ;
                     0414   958 ; INPUTS:
                     0414   959 ;
                     0414   960 ;       R3 = address of CLUDCB
                     0414   961 ;
                     0414   962 ; OUTPUT:
                     0414   963 ;
                     0414   964 ;       R0-R2 destroyed
                     0414   965 ;--
                     0414   966
                     0414   967 REQUEST_CSP:
      18     BB      0414   968         PUSHR   #^M<R3,R4>                      ; Save CLUDCB and CLUB pointers
51    07     D0      0416   969         MOVL    #CSP$_LOCAL,R1                  ; Send to local CSP
      52     D4      0419   970         CLRL    R2                              ; No CSD pointer
      53     D4      041B   971         CLRL    R3                              ; Must be zero
54    07     D0      041D   972         MOVL    #CSD$K_QUORUM,R4                ; R4 = client code
   FBDD'     30      0420   973         BSBW    EXE$CSP_COMMAND                 ; Request CSP
      18     BA      0423   974         POPR    #^M<R3,R4>                      ; Restore CLUDCB and CLUB pointers
      05     0425   975         RSB
                     0426   976
```

```
                     0426   978 .SBTTL  CHECK_ERROR - Check to see if error is fatal
                     0426   979 ;++
                     0426   980 ; CHECK_ERROR - Check to see if error is fatal
                     0426   981 ;
                     0426   982 ; FUNCTIONAL DESCRIPTION:
                     0426   983 ;
                     0426   984 ;       This routine checks the error status to see if we should simply retry.
                     0426   985 ;       We then cause a cluster state change and also cause mount verification
                     0426   986 ;       to be invoked.  This is necessary because the "internal" IRP
                     0426   987 ;       format used by quorum I/Os does not trigger mount verification.
                     0426   988 ;
                     0426   989 ;       In the case of accidental write-lock, quorum I/O is retried.
                     0426   990 ;
                     0426   991 ; CALLING SEQUENCE:
                     0426   992 ;
                     0426   993 ;       JSB/BSBx CHECK_ERROR
                     0426   994 ;
                     0426   995 ; INPUTS:
                     0426   996 ;
                     0426   997 ;       R3 = address of CLUDCB
                     0426   998 ;       R4 = address of CLUB
                     0426   999 ;       R5 = address of UCB
                     0426  1000 ;
                     0426  1001 ; OUTPUT:
                     0426  1002 ;
                     0426  1003 ;       R0 = Status (low bit)
                     0426  1004 ;           0 - no recovery - normal error processing
                     0426  1005 ;           1 - non-fatal error
                     0426  1006 ;
                     0426  1007 ;--
                     0426  1008
                     0426  1009 CHECK_ERROR:
                     0426  1010
          3E   BB    0426  1011         PUSHR   #^M<R1,R2,R3,R4,R5>
                     0428  1012 ;
    51   38 A5   3C  0428  1013         MOVZWL  IRP$L_IOST1(R5),R1      ; Get the error status
                     042C  1014         ;
                     042C  1015         ; If the medium is offline, or the volume is
                     042C  1016         ; invalid, the error can be recovered from.
                     042C  1017         ;
    51   0000'8F   B1  042C  1018         CMPW    #SS$_MEDOFL,R1          ; Is the media (disk volume) offline?
          37   13    0431  1019         BEQL    40$                     ; Branch if true
    51   0000'8F   B1  0433  1020         CMPW    #SS$_VOLINV,R1          ; Is the volume invalid?
          30   13    0438  1021         BEQL    40$                     ; Branch if true
                     043A  1022         ;
                     043A  1023         ; If the volume has been writelocked, make sure that it was
                     043A  1024         ; an accidental writelock.  If the software writelock bit is
                     043A  1025         ; on, then the volume was mounted with the volume write protected.
                     043A  1026         ; If the bit is not set, then the volume has been mounted for
                     043A  1027         ; read/write access, and has since been (accidentally) write protected.
                     043A  1028         ; The first time through this code and any time we are in the cluster or
                     043A  1029         ; vote states, we put everything in mount verification and cause a
                     043A  1030         ; cluster state change and return to the active state.  All other times,
                     043A  1031         ; we remain in the same state and quietly return.  This saves many
                     043A  1032         ; trees.
                     043A  1033         ;
    51   0000'8F   B1  043A  1034         CMPW    #SS$_WRITLCK,R1         ; Is the device writelocked?
```

```
                05   13  043F  1035          BEQL    10$
           50   51   D0  0441  1036          MOVL    R1,R0                           ; Get an error code in R0
                21   11  0444  1037          BRB     30$                             ; Go back to treat it as real error
00000000'8F     E0   0446  1038 10$:         BBS     #DEV$V_SWL,-                    ; Branch if software writelocked
    18 38 A5         044C  1039                      UCB$L_DEVCHAR(R5),30$
                06   E3  044F  1040          BBCS    #CLUDCB$V_QF_WRL_ERR,-          ; See if this is the first time
    08 22 A3         0451  1041                      CLUDCB$W_FLAGS(R3),15$
       24 A3   94   0454  1042          CLRB    CLUDCB$B_COUNTER(R3)            ; Restart counter in case in cluster state
                04   E1  0457  1043          BBC     #CLUDCB$V_QS_VOTE,-             ; Is it a dangerous state
    08 20 A3         0459  1044                      CLUDCB$W_STATE(R3),25$          ; No - leave it there
50   0000'CF   9E   045C  1045 15$:         MOVAB   W^QDWRLERROR_MSG,R0             ; Point to write error message
        FF71   30   0461  1046 20$:         BSBW    QUORUM_FILE_RETRY               ; Go try again
           50   01   D0  0464  1047 25$:         MOVL    #1,R0                           ; Error recovery in progress
                3E   BA  0467  1048 30$:         POPR    #^M<R1,R2,R3,R4,R5>
                05   0469  1049          RSB
                     046A  1050 ;
00000000'GF     16   046A  1051 40$:         JSB     G^EXE$CLUTRANIO                 ; Get everyting in mount verification
                EF   11  0470  1052          BRB     20$
                     0472  1053
                     0472  1054
                     0472  1055          .END
```

```
BUILD_QUORUM_FILE                000001F5 R      04
CALCULATE_CHECKSUM               000003C2 R      04
CHECK_ERROR                      00000426 R      04
CHECK_OWNER                      0000037A R      04
CLU$GB_QDISK                     ********   X    03
CLU$GL_CLUB                      ********   X    03
CLU$GL_CLUSVEC                   ********   X    04
CLU$GW_MAXINDEX                  ********   X    04
CLU$GW_QDSKINTERVAL              ********   X    03
CLUB$B_FSYSID                  = 00000026
CLUB$L_CLUDCB                  = 000000B4
CLUB$L_FLAGS                   = 0000001C
CLUB$L_FOREIGN_CLUSTER         = 000000C8
CLUB$L_LOCAL_CSID              = 00000060
CLUB$M_QF_ACTIVE               = 00000002
CLUB$M_QF_DYNVOTE              = 40000000
CLUB$M_QF_FAILED_NODE          = 01000000
CLUB$Q_FTIME                   = 0000002C
CLUB$Q_LST_TIME                = 0000003C
CLUB$V_CLUSTER                 = 00000000
CLUB$V_QF_FAILED_NODE          = 00000018
CLUB$V_QUORUM                  = 0000001C
CLUB$W_QUORUM                  = 00000020
CLUB$W_VOTES                   = 00000022
CLUDCB$B_COUNTER               = 00000024
CLUDCB$B_SUBTYPE               = 0000000B
CLUDCB$B_TYPE                  = 0000000A
CLUDCB$K_LENGTH                = 00000229
CLUDCB$L_ACT_COUNT             = 00000018
CLUDCB$L_IRP                   = 00000010
CLUDCB$L_QFLBN                 = 0000001C
CLUDCB$L_TQE                   = 00000014
CLUDCB$L_UCB                   = 0000000C
CLUDCB$M_QF_ERROR              = 00000008
CLUDCB$M_QF_RIP                = 00000002
CLUDCB$M_QF_TIM                = 00000001
CLUDCB$M_QF_WIP                = 00000004
CLUDCB$M_QF_WRL_ERR            = 00000040
CLUDCB$M_QS_ACTIVE             = 00000004
CLUDCB$M_QS_CLUSTER            = 00000008
CLUDCB$M_QS_NOT_READY          = 00000001
CLUDCB$M_QS_READY              = 00000002
CLUDCB$M_QS_VOTE               = 00000010
CLUDCB$S_BUFFER                = 00000204
CLUDCB$S_DISK_QUORUM           = 00000010
CLUDCB$T_BUFFER                = 00000025
CLUDCB$V_QF_ERROR              = 00000003
CLUDCB$V_QF_FIRST_ERR          = 00000005
CLUDCB$V_QF_RIP                = 00000001
CLUDCB$V_QF_TIM                = 00000000
CLUDCB$V_QF_WIP                = 00000002
CLUDCB$V_QF_WRL_ERR            = 00000006
CLUDCB$V_QS_NOT_READY          = 00000000
CLUDCB$V_QS_READY              = 00000001
CLUDCB$V_QS_VOTE               = 00000004
CLUDCB$W_FLAGS                 = 00000022
CLUDCB$W_SIZE                  = 00000008
CLUDCB$W_STATE                 = 00000020
CLUQF$B_FSYSID                 = 0000C03E
CLUQF$B_IGNORE                 = 00000048
CLUQF$K_ACT_LENGTH             = 00000004
CLUQF$K_CHECK_LENGTH           = 00000048
CLUQF$K_LENGTH                 = 00000204
CLUQF$K_VERSION                = 00000002
CLUQF$L_ACT_COUNT              = 00000200
CLUQF$L_CSID                   = 00000030
CLUQF$M_QUORUM                 = 00000001
CLUQF$Q_FOU_TIME               = 00000010
CLUQF$Q_SWINCARN               = 00000028
CLUQF$S_FSYSID                 = 00000006
CLUQF$S_IDENT                  = 0000000C
CLUQF$T_IDENT                  = 00000000
CLUQF$V_QUORUM                 = 00000000
CLUQF$W_CSID_IDX               = 00000030
CLUQF$W_FLAGS                  = 0000000E
CLUQF$W_QUORUM                 = 00000034
CLUQF$W_VERSION                = 0000000C
CLUQF$W_VOTES                  = 00000036
CLUQF_IDENT_STRING               00000000 R      02
CNX$BUGCHECK_CLUSTER             ********   X    04
CNX$CONFIG_CHANGE                ********   X    04
CNX$DISK_CHANGE                  ********   X    04
CNX$QUORUM_INIT                  00000000 RG     03
CSB$L_CSID                     = 0000004C
CSB$Q_SWINCARN                 = 00000038
CSD$K_QUORUM                   = 00000007
CSP$_LOCAL                     = 00000007
CYCLE_COUNT                    = 00000002
DEV$V_SWL                        ********   X    04
DISPATCH                         0000011F R      04
DYN$C_CLU                      = 00000065
DYN$C_CLU_CLUDCB               = 00000005
DYN$C_IRP                      = 0000000A
DYN$C_TQE                      = 0000000F
EXE$ALONONPAGED                  ********   X    03
EXE$CLUTRANIO                    ********   X    04
EXE$CSP_COMMAND                  ********   X    04
EXE$GQ_SYSTIME                   ********   X    04
EXE$INSIOQ                       ********   X    04
IO$_READPBLK                   = 0000000C
IO$_WRITEPBLK                  = 0000000B
IOC$CVTLOGPHY                    ********   X    04
IPL$_IOPOST                    = 00000004
IPL$_SCS                       = 00000008
IPL$_SYNCH                     = 00000008
IPL$_TIMER                     = 00000008
IRP$B_PRI                      = 00000023
IRP$B_TYPE                     = 0000000A
IRP$K_LENGTH                   = 000000C4
IRP$L_BCNT                     = 00000032
IRP$L_IOST1                    = 00000038
IRP$L_PID                      = 0000000C
IRP$L_SVAPTE                   = 0000002C
IRP$L_UCB                      = 0000001C
```

```
IRP$M_PHYSIO                    =  00000100
IRP$W_BOFF                      =  00000030
IRP$W_FUNC                      =  00000020
IRP$W_SIZE                      =  00000008
IRP$W_STS                       =  0000002A
MMG$GL_SPTBASE                     ********    X   04
PR$_IPL                            ********    X   04
QDCON_MSG                          ********    X   04
QDDISCON_MSG                       ********    X   04
QDFORCLUS_MSG                      ********    X   04
QDINVDAT_MSG                       ********    X   04
QDRDERROR_MSG                      ********    X   04
QDTIMOUT_MSG                       ********    X   04
QDWRERROR_MSG                      ********    X   04
QDWRLERROR_MSG                     ********    X   04
QUORUM_DISK_TIMEOUT                000003D5 R      04
QUORUM_FILE_ERROR                  000003DA R      04
QUORUM_FILE_RETRY                  000003D5 R      04
QUORUM_TIMEOUT                     00000000 RG     04
READ_COMPLETE                      00000097 RG     04
READ_COMPLETE_ACTIVE               0000016E R      04
READ_COMPLETE_CLUSTER              000001A6 R      04
READ_COMPLETE_READY                0000012F R      04
READ_COMPLETE_VOTE                 000001A6 R      04
READ_QUORUM_FILE                   00000039 R      04
REQUEST_CSP                        00000414 R      04
SB$B_SYSTEMID                   =  00000018
SB$Q_SWINCARN                   =  0000002C
SB$S_SYSTEMID                   =  00000006
SCS$GA_LOCALSB                     ********    X   04
SS$_MEDOFL                         ********    X   04
SS$_NORMAL                         ********    X   03
SS$_VOLINV                         ********    X   04
SS$_WRITLCK                        ********    X   04
TQE$B_RQTYPE                    =  0000000B
TQE$B_TYPE                      =  0000000A
TQE$C_SSREPT                    =  00000005
TQE$K_LENGTH                    =  00000030
TQE$L_FPC                       =  0000000C
TQE$L_FR3                       =  00000010
TQE$L_FR4                       =  00000014
TQE$Q_DELTA                     =  00000020
TQE$W_SIZE                      =  00000008
UCB$L_DEVCHAR                   =  00000038
UCB$V_NOCNVRT                   =  00000002
UCB$W_DEVSTS                    =  00000068
UCB$W_QLEN                      =  0000006A
VA$M_BYTE                       =  000001FF
VA$S_VPN                        =  00000015
VA$V_VPN                        =  00000009
VALIDATE_QUORUM_FILE               00000357 R      04
WRITE_COMPLETE                     000002DA RG     04
WRITE_QUORUM_ACT                   0000026C R      04
WRITE_QUORUM_OWNACT                0000025F R      04
```

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
$$$060	0000000C (12.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$002	000000C7 (199.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$100	00000472 (1138.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.05	00:00:02.35
Command processing	137	00:00:00.46	00:00:03.48
Pass 1	420	00:00:10.39	00:00:36.94
Symbol table sort	0	00:00:01.64	00:00:07.11
Pass 2	188	00:00:02.44	00:00:10.00
Symbol table output	20	00:00:00.11	00:00:00.11
Psect synopsis output	3	00:00:00.02	00:00:00.51
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	805	00:00:15.11	00:01:00.50

The working set limit was 1950 pages.
90025 bytes (176 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1566 non-local and 44 local symbols.
1055 source lines were read in Pass 1, producing 21 object records in Pass 2.
23 pages of virtual memory were used to define 22 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1	3
_255DUA28:[SYS.OBJ]LIB.MLB;1	11
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	19

1637 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

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

MOUNTVER
LIS

OPDRVWS1
LIS

QUORUM
LIS

OPDRV790
LIS

OPDRIVER
CIS

REBLDLOCK
LIS