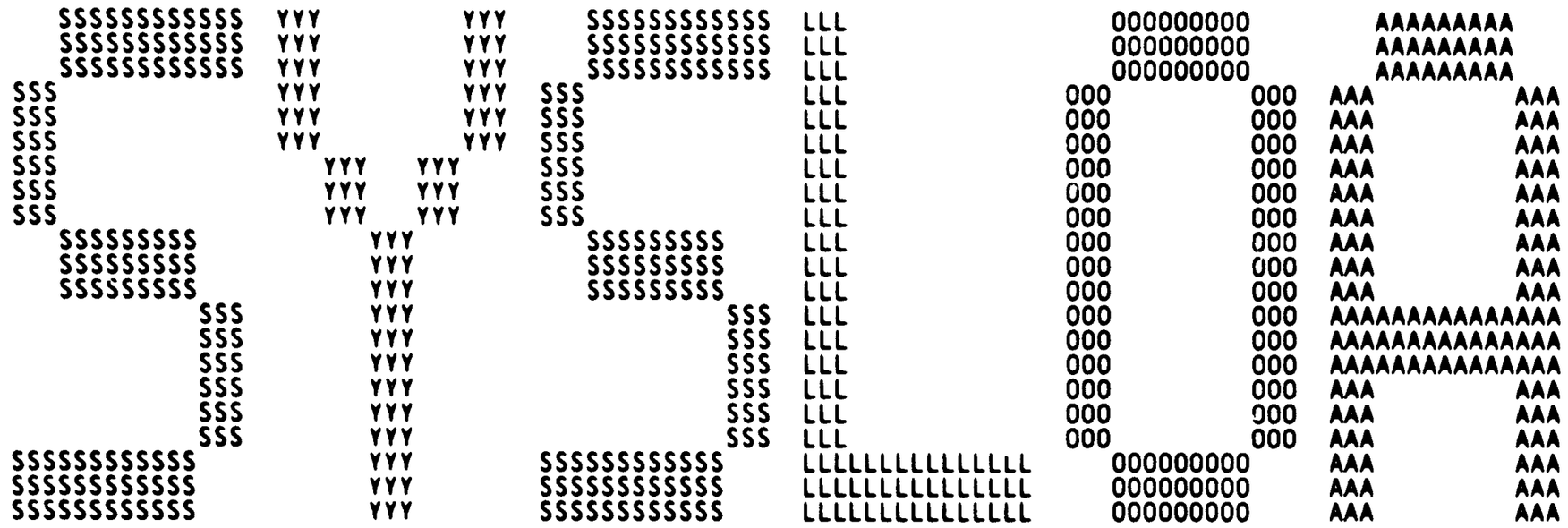
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 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

FILEID**CONUTIL

```
  CCCCCCCC    000000    NN      NN  UU      UU  TTTTTTTTTT    IIIIII    LL
  CCCCCCCC    000000    NN      NN  UU      UU  TTTTTTTTTT    IIIIII    LL
CC           00      00  NN      NN  UU      UU      TT        II      LL
CC           00      00  NN      NN  UU      UU      TT        II      LL
CC           00      00  NNNN    NN  UU      UU      TT        II      LL
CC           00      00  NNNN    NN  UU      UU      TT        II      LL
CC           00      00  NN  NN  NN  UU      UU      TT        II      LL
CC           00      00  NN  NN  NN  UU      UU      TT        II      LL
CC           00      00  NN    NNNN  UU      UU      TT        II      LL
CC           00      00  NN    NNNN  UU      UU      TT        II      LL
CC           00      00  NN      NN  UU      UU      TT        II      LL
CC           00      00  NN      NN  UU      UU      TT        II      LL
  CCCCCCCC    000000    NN      NN  UUUUUUUUUU      TT      IIIIII    LLLLLLLLLL
  CCCCCCCC    000000    NN      NN  UUUUUUUUUU      TT      IIIIII    LLLLLLLLLL


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  CONUTIL - Cluster Configuration Manager Utilities
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;****************************************************************************
0000     5  ;*                                                                          *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                    *
0000     9  ;*                                                                          *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15  ;*  TRANSFERRED.                                                            *
0000    16  ;*                                                                          *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                            *
0000    20  ;*                                                                          *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    23  ;*                                                                          *
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 utility routines used by cluster configuration
0000    33  ;       management and failover sequencing.
0000    34  ;
0000    35  ; ENVIRONMENT: VAX/VMS
0000    36  ;
0000    37  ; AUTHOR: David W. Thiel,        CREATION DATE: 4-Apr-1983
0000    38  ;
0000    39  ; MODIFIED BY:
0000    40  ;
0000    41  ;       V03-012 DWT0239         David W. Thiel          30-Aug-1984
0000    42  ;               Tie off CNX$BUGCHECK_CLUSTER to be a plain bugcheck,
0000    43  ;               since it doesn't work.
0000    44  ;
0000    45  ;       V03-011 DWT0225         David W. Thiel          11-Jul-1984
0000    46  ;               Change call to temporary name EXE$MNTVERSP2 to
0000    47  ;               real name EXE$CLUTRANIO.
0000    48  ;
0000    49  ;       V03-010 DWT0219         David W. Thiel          08-May-1984
0000    50  ;               Add synchronization of I/O cluster and lock cluster.
0000    51  ;
0000    52  ;       V03-009 DWT0210         David W. Thiel          07-Apr-1984
0000    53  ;               Use BUSY bit in CLUBFKB to track activity in this
0000    54  ;               block instead of a bit in the CLUB$L_FLAGS field.
0000    55  ;
0000    56  ;       V03-008 DWT0195         David W. Thiel          23-Mar-1984
0000    57  ;               Add notion of dynamic quorum for quorum disk.
```

```
0000     58 ;                  Move computation of dynamic quorum from
0000     59 ;                  CNX$QUORUM_CALC to COMPUTE_DYNAMIC_QUORUM.
0000     60 ;                  Correct dynamic and static quorum computation.
0000     61 ;
0000     62 ;          V03-007 DWT0174         David W. Thiel          21-Feb-1984
0000     63 ;                  Minimize quorum disk votes against value in CLUB
0000     64 ;                  when computing quorum.
0000     65 ;
0000     66 ;          V03-006 DWT0152         David W. Thiel          29-Dec-1983
0000     67 ;                  Replace lock manager directory system logic with
0000     68 ;                  new code to support directory system vector.
0000     69 ;                  Add common routine to compute quorum and support a
0000     70 ;                  variable number of votes for the quorum disk.
0000     71 ;                  Delete CNX$FAILO_ALLOW_JOIN routine.
0000     72 ;
0000     73 ;          V03-005 DWT0146         David W. Thiel          14-Nov-1983
0000     74 ;                  Block activity when out of quorum using a fork block
0000     75 ;                  in the CLUB rather than a wired-in fork block.
0000     76 ;                  Add routines CNX$DIRVEC_ADJ and CNX$DIRVEC_FILL to
0000     77 ;                  manipulate the lock manager directory system vector.
0000     78 ;
0000     79 ;          V03-004 DWT0128         David W. Thiel          30-Aug-1983
0000     80 ;                  Clean up lock manager directory system CSID
0000     81 ;                  management.
0000     82 ;
0000     83 ;          V03-003 DWT0119         David W. Thiel          11-Aug-1983
0000     84 ;                  Add support for quorum disk.  Change CONFIG_CHANGE
0000     85 ;                  to CNX$CONFIG_CHANGE.
0000     86 ;
0000     87 ;          V03-002 DWT0110         David W. Thiel          28-Jul-1983
0000     88 ;                  Add CNX$SCAN_CSBS_FORK and CNX$CLUB_FORK routines.
0000     89 ;                  Convert failover routines to JSB interface.
0000     90 ;                  Add routine to loop at IPL 4 while the cluster is out
0000     91 ;                  of quorum.
0000     92 ;
0000     93 ;          V03-001 DWT0106         David W. Thiel          21-Jun-1983
0000     94 ;                  Coorect CNX$SET_DIRSYS, add CNX$FORM_DIRSYS entry point.
0000     95 ;                  Add failover routine to allow nodes to join cluster.
0000     96 ;                  Enable code to fix EPIDs.
0000     97 ;
0000     98 ;--
0000     99
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; INCLUDE FILES:
                ;
                        $CDRPDEF                        ; CDRP offsets
                        $CDTDEF                         ; CDT Offsets
                        $CLMDRSDEF                      ; Cluster disconnect/reject codes
                        $CLSMSGDEF                      ; Cluster message definitions
                        $CLUBDEF                        ; CLUster Block offsets
                        $CSBDEF                         ; CSB Offsets
                        $DYNDEF                         ; Data structure type codes
                        $FKBDEF                         ; ForK Block offsets
                        $IPLDEF                         ; IPL definitions
                        $IRPDEF                         ; IRP offsets
                        $PCBDEF                         ; Process Control Block offsets
                        $SBDEF                          ; System Block offsets
                        $SSDEF                          ; Status code definitions
                        $TQEDEF                         ; TQE offsets

                ;*****************************************************************************
                ;
                ; NOTE: The following assumptions are in effect for this entire module.
                ;
                ;*****************************************************************************

                        ASSUME  IPL$_SYNCH  EQ  IPL$_SCS
                        ASSUME  IPL$_SYNCH  EQ  IPL$_TIMER
                        ASSUME  IPL$_SYNCH  GT  IPL$_IOPOST

00000000                .PSECT  $$$100,LONG

                        .DEFAULT        DISPLACEMENT,WORD
```

```
                        0000   134          .SBTTL  CNX$RESP_FORGET - Respond to a message and forget about it
                        0000   135          .SBTTL  CNX$SEND_FORGET - Send message and forget about it
                        0000   136
                        0000   137 ;++
                        0000   138 ;
                        0000   139 ; FUNCTIONAL DESCRIPTION:
                        0000   140 ;
                        0000   141 ;       Queue a message or response to be sent and forget about it.
                        0000   142 ;       When the acknowledgement is received or the connection breaks,
                        0000   143 ;       the CDRP is deallocated.
                        0000   144 ;
                        0000   145 ; CALLING SEQUENCE:
                        0000   146 ;
                        0000   147 ;       JSB     CNX$SEND_FORGET
                        0000   148 ;       JSB     CNX$RESP_FORGET
                        0000   149 ;       IPL is IPL$_SCS
                        0000   150 ;
                        0000   151 ; INPUT PARAMETERS:
                        0000   152 ;
                        0000   153 ;       R2:     Address of message buffer (CNX$RESP_FORGET only)
                        0000   154 ;       R3:     Address of CSB
                        0000   155 ;       R5:     Address of fully initialized CDRP
                        0000   156 ;
                        0000   157 ; OUTPUT PARAMETERS:
                        0000   158 ;
                        0000   159 ;       NONE
                        0000   160 ;
                        0000   161 ; COMPLETION CODES:
                        0000   162 ;
                        0000   163 ;       NONE
                        0000   164 ;
                        0000   165 ; SIDE EFFECTS:
                        0000   166 ;
                        0000   167 ;       NONE
                        0000   168 ;
                        0000   169 ;--
                        0000   170
                        0000   171 CNX$RESP_FORGET::
    1C A5    52    D0   0000   172          MOVL    R2,CDRP$L_MSG_BUF(R5)   ; Address of message buffer
58 A5    04 A2    D0   0004   173          MOVL    CLSMSG$L_RSPID(R2), -   ; Store RSPID to return in CDRP
                        0009   174                  CDRP$L_RETRSPID(R5)
                        0009   175
                        0009   176 CNX$SEND_FORGET::
          FFF4'  30     0009   177          BSBW    CNX$SEND_MSG_CSB        ; Send message
                        000C   178 ;
                        000C   179 ;       We are resumed here when the message is acknowledged.
                        000C   180 ;       Registers contain:
                        000C   181 ;               R0:     Status
                        000C   182 ;               R3:     Address of CSB
                        000C   183 ;               R4:     Address of PDT
                        000C   184 ;               R5:     Address of CDRP
                        000C   185 ;
      50    55    D0   000C   186          MOVL    R5,R0                   ; Address of CDRP
00000000'GF   17    000F   187          JMP     G^EXE$DEANONPAGED       ; Deallocate CDRP and return
```

```
0015   189          .SBTTL  CNX$BUGCHECK_CLUSTER - Bugcheck local cluster
0015   190
0015   191 ;++
0015   192 ;
0015   193 ; FUNCTIONAL DESCRIPTION:
0015   194 ;
0015   195 ;       This routine is called to request that all members of the local
0015   196 ;       cluster bugcheck.  A typical use is when two clusters detect each
0015   197 ;       other and one (or both) decide to bow out gracefully.
0015   198 ;
0015   199 ;       *** TEMPORARY ***
0015   200 ;       Since there was a fatal flaw in the approach, this has been replaced
0015   201 ;       by a plain old CLUEXIT bugcheck to minimize the confusion added to
0015   202 ;       the dump.
0015   203 ;
0015   204 ; CALLING SEQUENCE:
0015   205 ;
0015   206 ;       JSB     CNX$BUGCHECK_CLUSTER
0015   207 ;       IPL is IPL$_SCS
0015   208 ;
0015   209 ; INPUT PARAMETERS:
0015   210 ;
0015   211 ;       NONE
0015   212 ;
0015   213 ; OUTPUT PARAMETERS:
0015   214 ;
0015   215 ;       NONE
0015   216 ;
0015   217 ; COMPLETION CODES:
0015   218 ;
0015   219 ;       NONE
0015   220 ;
0015   221 ; SIDE EFFECTS:
0015   222 ;
0015   223 ;       Cluster Holocaust
0015   224 ;--
0015   225
0015   226 CNX$BUGCHECK_CLUSTER::
0015   227          BUG_CHECK       CLUEXIT,FATAL   ; We've can't shut down the world, so just
0019   228                                          ;  shut down self
0019   229
```

```
                     0019   231          .SBTTL  CNX$INIT_CSBS - Initialize CSBs for new transition
                     0019   232
                     0019   233 ;++
                     0019   234 ;
                     0019   235 ; FUNCTIONAL DESCRIPTION:
                     0019   236 ;
                     0019   237 ;       This routine initializes all CSBs for a new transition.
                     0019   238 ;
                     0019   239 ; CALLING SEQUENCE:
                     0019   240 ;
                     0019   241 ;       JSB     CNX$INIT_CSBS
                     0019   242 ;       IPL is IPL$_SCS
                     0019   243 ;
                     0019   244 ; INPUT PARAMETERS:
                     0019   245 ;
                     0019   246 ;       NONE
                     0019   247 ;
                     0019   248 ; OUTPUT PARAMETERS:
                     0019   249 ;
                     0019   250 ;       NONE
                     0019   251 ;
                     0019   252 ; COMPLETION CODES:
                     0019   253 ;
                     0019   254 ;       NONE
                     0019   255 ;
                     0019   256 ; SIDE EFFECTS:
                     0019   257 ;
                     0019   258 ;       R0 and R1 are destroyed
                     0019   259 ;
                     0019   260 ;--
                     0019   261
                     0019   262 CNX$INIT_CSBS::
          18   BB    0019   263          PUSHR   #^M<R3,R4>              ; Save registers
        000C   30    001B   264          BSBW    CNX$SCAN_CSBS           ; Iterate over all CSBs
      06 50  E9      001E   265          BLBC    R0,20$                  ; Branch when done
00 60 A3  11   E5    0021   266          BBCC    #CSB$V_SELECTED, -      ; Clear selected flag
                     0026   267                  CSB$L_STATUS(R3),10$
               05    0026   268 10$:     RSB
                     0027   269
          18   BA    0027   270 20$:     POPR    #^M<R3,R4>              ; Restore registers
               05    0029   271          RSB
```

```
                        002A   273          .SBTTL  CNX$SCAN_CSBS - Scan all CSB's
                        002A   274
                        002A   275 ;++
                        002A   276 ;
                        002A   277 ; FUNCTIONAL DESCRIPTION:
                        002A   278 ;
                        002A   279 ;       This routine scans all CSB's.
                        002A   280 ;       It does a co-routine call-back for each CSB.
                        002A   281 ;       When all CSB's have been scanned, it returns failure status.
                        002A   282 ;
                        002A   283 ; CALLING SEQUENCE:
                        002A   284 ;
                        002A   285 ;       JSB     CNX$SCAN_CSBS
                        002A   286 ;       IPL is IPL$_SCS
                        002A   287 ;
                        002A   288 ; INPUT PARAMETERS:
                        002A   289 ;
                        002A   290 ;       NONE
                        002A   291 ;
                        002A   292 ; OUTPUT PARAMETERS:
                        002A   293 ;
                        002A   294 ;       R4 is CLUB address
                        002A   295 ;
                        002A   296 ; COMPLETION CODES:
                        002A   297 ;
                        002A   298 ;       R0 = even:      end of scan
                        002A   299 ;       R0 = odd:       co-routine call-back
                        002A   300 ;                       R3 is address of CSB
                        002A   301 ;                       R4 is address of CLUB
                        002A   302 ;               On co-routine return:
                        002A   303 ;                       R3 is address of CSB following which
                        002A   304 ;                           the scan is to resume.
                        002A   305 ;
                        002A   306 ; SIDE EFFECTS:
                        002A   307 ;
                        002A   308 ;       R3 is destroyed
                        002A   309 ;
                        002A   310 ;--
                        002A   311
                        002A   312          .ENABLE LSB
                        002A   313
                        002A   314 CNX$SCAN_CSBS::
53   00000000'GF    D0  002A   315          MOVL    G^CLU$GL_CLUB,R3        ; Get CLUB address
                        0031   316          ASSUME  CLUB$L_CSBQFL EQ 0
                        0031   317          ASSUME  CSB$L_SYSQFL EQ 0
54   00000000'GF    D0  0031   318 110$:    MOVL    G^CLU$GL_CLUB,R4        ; Get CLUB address
     53    63       D0  0038   319          MOVL    CSB$L_SYSQFL(R3),R3     ; Advance to next CSB
     53    54       D1  003B   320          CMPL    R4,R3                   ; End of list?
           12       13  003E   321          BEQL    130$                    ; Branch if no more CSB's
     50    01       D0  0040   322 120$:    MOVL    S^#SS$_NORMAL,R0        ; Set success value
                        0043   323 ;
                        0043   324 ; Call back caller:
                        0043   325 ;
                        0043   326 ;       04(SP):         Original return address
                        0043   327 ;       00(SP):         Return address to resume scan
                        0043   328 ;
                        0043   329 ;       R0:             Status = SS$_NORMAL
```

```
                     0043   330 ;       R3:             Address of CSB scanned
                     0043   331 ;       R4:             Address of CLUB
                     0043   332 ;
00 BE    16          0043   333         JSB     @(SP)                   ; Return with CSB
                     0046   334 ;
                     0046   335 ;       R3 :            Address of CSB following which scan resumes
                     0046   336 ;       Others:         Don't care
                     0046   337 ;
   E9    11          0046   338         BRB     110$                    ; Go around again
                     0048   339
                     0048   340         .SBTTL  CNX$SCAN_CSBS_EXIT - Request immediate exit from CSB Scan
                     0048   341
                     0048   342 ;++
                     0048   343 ;
                     0048   344 ; FUNCTIONAL DESCRIPTION:
                     0048   345 ;
                     0048   346 ;       This code provides an immediate exit from the CSB scan loop.
                     0048   347 ;
                     0048   348 ; CALLING SEQUENCE:
                     0048   349 ;
                     0048   350 ;       JMP     CNX$SCAN_CSBS_EXIT
                     0048   351 ;       IPL is IPL$_SCS
                     0048   352 ;
                     0048   353 ; INPUT PARAMETERS:
                     0048   354 ;
                     0048   355 ;       R3 is CSB address
                     0048   356 ;
                     0048   357 ; OUTPUT PARAMETERS:
                     0048   358 ;
                     0048   359 ;       Return is from call to CNX$SCAN_CSBS.
                     0048   360 ;       R0 is 0
                     0048   361 ;       R4 is CLUB address
                     0048   362 ;
                     0048   363 ; COMPLETION CODES:
                     0048   364 ;
                     0048   365 ;       Return is from call to CNX$SCAN_CSBS
                     0048   366 ;
                     0048   367 ; SIDE EFFECTS:
                     0048   368 ;
                     0048   369 ;       R0 and R4 are destroyed
                     0048   370 ;
                     0048   371 ;--
                     0048   372
                     0048   373 CNX$SCAN_CSBS_EXIT::
        5E    04     C0   0048   374         ADDL    #4,SP                   ; Clear co-routine return from stack
54  00000000'GF     D0   004B   375         MOVL    G^CLU$GL_CLUB,R4        ; Get CLUB address
             50     D4   0052   376 130$:   CLRL    R0                      ; Set failure value
                    05   0054   377         RSB
                     0055   378
                     0055   379         .SBTTL  CNX$SCAN_CSBS_RETRY - Delay, then Contine CSB Scan
                     0055   380
                     0055   381 ;++
                     0055   382 ;
                     0055   383 ; FUNCTIONAL DESCRIPTION:
                     0055   384 ;
                     0055   385 ;       This routine handles the case where a thread must wait for
                     0055   386 ;       some resource before it may proceed during a CSB scan.
```

```
                     0055   387 ;       All context for the thread must reside in the R3 CSB address.
                     0055   388 ;       After the delay, CNX$SCAN_CSBS returns with the same CSB as on
                     0055   389 ;       the iteration during which this routine was called.
                     0055   390 ;
                     0055   391 ; CALLING SEQUENCE:
                     0055   392 ;
                     0055   393 ;       JMP     CNX$SCAN_CSBS_RETRY
                     0055   394 ;       IPL is IPL$_SCS
                     0055   395 ;
                     0055   396 ; INPUT PARAMETERS:
                     0055   397 ;
                     0055   398 ;       R3 is CSB address
                     0055   399 ;
                     0055   400 ; OUTPUT PARAMETERS:
                     0055   401 ;
                     0055   402 ;       Return is from call to CNX$SCAN_CSBS.
                     0055   403 ;       R3 is CSB address
                     0055   404 ;       R4 is CLUB address
                     0055   405 ;
                     0055   406 ; COMPLETION CODES:
                     0055   407 ;
                     0055   408 ;       Return is from call to CNX$SCAN_CSBS
                     0055   409 ;
                     0055   410 ; SIDE EFFECTS:
                     0055   411 ;
                     0055   412 ;       R0, R1, R2, R4, R5 are destroyed
                     0055   413 ;
                     0055   414 ;--
                     0055   415
                     0055   416 CNX$SCAN_CSBS_RETRY::
         5E   04     C0  0055   417         ADDL    #4,SP                   ; Know this value
54   00000000'GF     D0  0058   418         MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
   55    00CC C4     9E  005F   419         MOVAB   CLUB$B_FORK_BLOCK(R4),R5 ; Address of fork block
16 1C A5    00     E2  0064   420         BBSS    #CLUBFKB$V_FKB_BUSY, -  ; Branch if fork block is
                     0069   421                 CLUBFKB$L_STATUS(R5),300$ ;  not busy
             10     BA  0069   422         POPR    #^M<R4>                 ; Save return address
                     006B   423         ASSUME  CLUBFKB$B_FORK_BLOCK EQ 0
                     006B   424         ASSUME  CLUBFKB$S_FORK_BLOCK GE FKB$K_LENGTH
                     006B   425         FORK_WAIT                       ; Wait for the next clock tick
             54     DD  0071   426         PUSHL   R4                      ; Restore return address
   54    FF34 C5     9E  0073   427         MOVAB   -CLUB$B_FORK_BLOCK(R5),R4 ; Address of CLUB
02 1C A5    00     E5  0078   428         BBCC    #CLUBFKB$V_FKB_BUSY, -  ; Clear fork block busy indicator
                     007D   429                 CLUBFKB$L_STATUS(R5),300$
             C1     11  007D   430         BRB     120$                    ; Repeat co-routine call
                     007F   431
                     007F   432 300$:    BUG_CHECK       CNXMGRERR,FATAL ; Double use of fork block
                     0083   433
                     0083   434         .SBTTL  CNX$SCAN_CSBS_FORK - Fork, then Contine CSB Scan
                     0083   435
                     0083   436 ;++
                     0083   437 ;
                     0083   438 ; FUNCTIONAL DESCRIPTION:
                     0083   439 ;
                     0083   440 ;       This routine handles the case where a thread must momentarily
                     0083   441 ;       release control before it may proceed during a CSB scan.
                     0083   442 ;       All context for the thread must reside in the R3 CSB address.
                     0083   443 ;       After the delay, CNX$SCAN_CSBS returns with the same CSB as on
```

```
                                                     0083   444 ;       the iteration during which this routine was called.
                                                     0083   445 ;
                                                     0083   446 ; CALLING SEQUENCE:
                                                     0083   447 ;
                                                     0083   448 ;       JMP     CNX$SCAN_CSBS_FORK
                                                     0083   449 ;       IPL is IPL$_SCS
                                                     0083   450 ;
                                                     0083   451 ; INPUT PARAMETERS:
                                                     0083   452 ;
                                                     0083   453 ;       R3 is CSB address
                                                     0083   454 ;
                                                     0083   455 ; OUTPUT PARAMETERS:
                                                     0083   456 ;
                                                     0083   457 ;       Return is from call to CNX$SCAN_CSBS.
                                                     0083   458 ;       R3 is CSB address
                                                     0083   459 ;       R4 is CLUB address
                                                     0083   460 ;
                                                     0083   461 ; COMPLETION CODES:
                                                     0083   462 ;
                                                     0083   463 ;       Return is from call to CNX$SCAN_CSBS
                                                     0083   464 ;
                                                     0083   465 ; SIDE EFFECTS:
                                                     0083   466 ;
                                                     0083   467 ;       R0, R1, R2, R4, R5 are destroyed
                                                     0083   468 ;
                                                     0083   469 ;--
                                                     0083   470
                                                     0083   471 CNX$SCAN_CSBS_FORK::
                                5E   04   C0        0083   472         ADDL    #4,SP                   ; Know this value
                          54  00000000'GF   D0        0086   473         MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
                              55   00CC   C4   9E   008D   474         MOVAB   CLUB$B_FORK_BLOCK(R4),R5 ; Address of fork block
                          1B 1C A5   00   E2        0092   475         BBSS    #CLUBFKB$V_FKB_BUSY, -  ; Branch if fork block is
                                                     0097   476                 CLUBFKB$L_STATUS(R5),400$ ;  not busy
                             0B A5   08   90   0097   477         MOVB    #IPL$_SCS,FKB$B_FIPL(R5) ;Initialize fork IPL
                                     10   BA   009B   478         POPR    #^M<R4>                 ; Save return address
                                                     009D   479         ASSUME  CLUBFKB$B_FORK_BLOCK EQ 0
                                                     009D   480         ASSUME  CLUBFKB$S_FORK_BLOCK GE FKB$K_LENGTH
                                00000000'GF   16   009D   481         JSB     G^EXE$FORK              ; Wait until the fork list cycles
                                         54   DD   00A3   482         PUSHL   R4                      ; Restore return address
                                   54   FF34   C5   9E   00A5   483         MOVAB   -CLUB$B_FORK_BLOCK(R5),R4 ; Address of CLUB
                                03 1C A5   00   E5   00AA   484         BBCC    #CLUBFKB$V_FKB_BUSY, -  ; Clear fork block busy indicator
                                                     00AF   485                 CLUBFKB$L_STATUS(R5),400$
                                       FF7F   31   00AF   486         BRW     110$                    ; Advance to next CSB
                                                     00B2   487
                                                     00B2   488 400$:   BUG_CHECK       CNXMGRERR,FATAL ; Double use of fork block
                                                     00B6   489
                                                     00B6   490         .DISABLE        LSB
```

```
                                00B6   492          .SBTTL  CNX$ASSIGN_CSID - Assign a tentative CSID to a CSB
                                00B6   493 ;++
                                00B6   494 ;
                                00B6   495 ; FUNCTIONAL DESCRIPTION:
                                00B6   496 ;
                                00B6   497 ;       Allocate a CSID slot and form the CSID.
                                00B6   498 ;       This is all done using the tentative CSID allocation context
                                00B6   499 ;       found in CLUB$W_NEXT_CSID.
                                00B6   500 ;
                                00B6   501 ; CALLING SEQUENCE:
                                00B6   502 ;
                                00B6   503 ;       JSB     CNX$ASSIGN_CSID
                                00B6   504 ;       IPL is IPL$_SCS
                                00B6   505 ;
                                00B6   506 ; INPUT PARAMETERS:
                                00B6   507 ;
                                00B6   508 ;       R3:     CSB for which a CSID is to be allocated
                                00B6   509 ;       CLUB$W_NEXT_CSID contains first CSID slot to consider
                                00B6   510 ;
                                00B6   511 ; OUTPUT PARAMETERS:
                                00B6   512 ;
                                00B6   513 ;       R4:     CLUB address
                                00B6   514 ;
                                00B6   515 ; COMPLETION CODES:
                                00B6   516 ;
                                00B6   517 ;       R0:     success/failure status
                                00B6   518 ;
                                00B6   519 ; SIDE EFFECTS:
                                00B6   520 ;
                                00B6   521 ;       CSB$L_CSID(R3) receives the allocated CSID.
                                00B6   522 ;       CLUB$W_NEXT_CSID is updated.
                                00B6   523 ;       R1 and R2 are destroyed.
                                00B6   524 ;
                                00B6   525 ;--
                                00B6   526
                                00B6   527 CNX$ASSIGN_CSID::
      54    64 A3    D0   00B6   528          MOVL    CSB$L_CLUB(R3),R4       ; Address of CLUB
50    00000000'GF    3C   00BA   529          MOVZWL  G^CLU$GW_MAXINDEX,R0    ; Number of potentially available slots
      52    64 A4    3C   00C1   530          MOVZWL  CLUB$W_NEXT_CSID(R4),R2 ; First slot to consider
                09    13   00C5   531          BEQL    20$                     ; Branch to avoid slot 0
00000000'GF   52    B1   00C7   532 10$:     CMPW    R2,G^CLU$GW_MAXINDEX    ; Wrap slot index around yet?
                03    1F   00CE   533          BLSSU   30$                     ; Not yet
          52    01    D0   00D0   534 20$:     MOVL    #1,R2                   ; Do wrap around
51    00000000'GF    D0   00D3   535 30$:     MOVL    G^CLU$GL_CLUSVEC,R1     ; Address of CSB vector
        51    6142    D0   00DA   536          MOVL    (R1)[R2],R1             ; Look at CSB vector entry
                06    18   00DE   537          BGEQ    40$                     ; Branch if slot is free
                52    D6   00E0   538          INCL    R2
             E2 50    F5   00E2   539          SOBGTR  R0,10$                  ; Iterate over all possible slots
                05    00E5   540          RSB                             ; Return with failure status
                           00E6   541
 64 A4    52    01    A1   00E6   542 40$:     ADDW3   #1,R2, -                ; Update next CSID
                           00EB   543                   CLUB$W_NEXT_CSID(R4)
 4E A3    51    01    A1   00EB   544          ADDW3   #1,R1,CSB$W_CSID_SEQ(R3) ; CSID sequence number
       4C A3    52    B0   00F0   545          MOVW    R2,CSB$W_CSID_IDX(R3)   ; CSID index
          50    01    D0   00F4   546          MOVL    #1,R0                   ; Return success
                05    00F7   547          RSB
```

```
                     00F8   549          .SBTTL  CNX$MARK_LOCKED - Mark Node Locked
                     00F8   550 ;++
                     00F8   551 ;
                     00F8   552 ; FUNCTIONAL DESCRIPTION:
                     00F8   553 ;
                     00F8   554 ;       Mark the specified node as LOCKED.
                     00F8   555 ;
                     00F8   556 ; CALLING SEQUENCE:
                     00F8   557 ;
                     00F8   558 ;       JSB     CNX$MARK_LOCKED
                     00F8   559 ;       IPL is IPL$_SCS
                     00F8   560 ;
                     00F8   561 ; INPUT PARAMETERS:
                     00F8   562 ;
                     00F8   563 ;       R3:     CSB of the node to be marked locked
                     00F8   564 ;
                     00F8   565 ; OUTPUT PARAMETERS:
                     00F8   566 ;
                     00F8   567 ;       NONE
                     00F8   568 ;
                     00F8   569 ; COMPLETION CODES:
                     00F8   570 ;
                     00F8   571 ;       NONE
                     00F8   572 ;
                     00F8   573 ; SIDE EFFECTS:
                     00F8   574 ;
                     00F8   575 ;       R0-R1 are destroyed.
                     00F8   576 ;
                     00F8   577 ;--
                     00F8   578
                     00F8   579 CNX$MARK_LOCKED::
          6C A3      96  00F8   580          INCB    CSB$B_REF_CNT(R3)       ; Bump reference count to nail down CSB
01 60 A3    10      E2  00FB   581          BBSS    #CSB$V_LOCKED, -        ; Lock and branch if already locked
                     0100   582                  CSB$L_STATUS(R3),90$
                  05 0100   583          RSB
                     0101   584
                     0101   585 90$:     BUG_CHECK       CNXMGRERR,FATAL ; Consistency check
```

```
                     0105   587          .SBTTL  CNX$MARK_UNLOCKED - Mark Node Unlocked
                     0105   588 ;++
                     0105   589 ;
                     0105   590 ; FUNCTIONAL DESCRIPTION:
                     0105   591 ;
                     0105   592 ;       Mark the specified node as unLOCKED.
                     0105   593 ;
                     0105   594 ; CALLING SEQUENCE:
                     0105   595 ;
                     0105   596 ;       JSB     CNX$MARK_UNLOCKED
                     0105   597 ;       IPL is IPL$_SCS
                     0105   598 ;
                     0105   599 ; INPUT PARAMETERS:
                     0105   600 ;
                     0105   601 ;       R3:     CSB of the node to be marked unlocked
                     0105   602 ;
                     0105   603 ; OUTPUT PARAMETERS:
                     0105   604 ;
                     0105   605 ;       R3:     If CSB deleted, then contents of back link pointer
                     0105   606 ;               If CSB not deleted, then CSB address
                     0105   607 ;
                     0105   608 ; COMPLETION CODES:
                     0105   609 ;
                     0105   610 ;       NONE
                     0105   611 ;
                     0105   612 ; SIDE EFFECTS:
                     0105   613 ;
                     0105   614 ;       R0-R1 are destroyed.
                     0105   615 ;
                     0105   616 ;--
                     0105   617
                     0105   618 CNX$MARK_UNLOCKED::
            30   BB  0105   619          PUSHR   #^M<R4,R5>                      ; Save some registers
       54   64 A3   D0  0107   620          MOVL    CSB$L_CLUB(R3),R4               ; Address of CLUB
18 1C A4    1D   E1  010B   621          BBC     #CLUB$V_TRANSITION, -           ; Branch if no transition in progress
                     0110   622                  CLUB$L_FLAGS(R4),90$
5C A4    10 A4   D1  0110   623          CMPL    CLUB$L_LOCAL_CSB(R4), -         ; Is this node the coordinator?
                     0115   624                  CLUB$L_COORD(R4)
            0E   12  0115   625          BNEQ    10$                             ; Branch if no
0C 60 A3    10   E5  0117   626          BBCC    #CSB$V_LOCKED, -                ; Unlock and branch if not locked
                     011C   627                  CSB$L_STATUS(R3),90$
         55   53   D0  011C   628          MOVL    R3,R5                           ; Move CSB address
           FEDE'  30  011F   629          BSBW    CNX$DECREFCNT                   ; Decrement reference count -- if CSB
         53   55   D0  0122   630          MOVL    R5,R3                           ;   deleted, return previous list entry
            30   BA  0125   631 10$:       POPR    #^M<R4,R5>                      ; Restore registers
                 05  0127   632          RSB
                     0128   633
                     0128   634 90$:       BUG_CHECK       CNXMGRERR,FATAL ; Consistency check
```

```
012C    636          .SBTTL  CNX$DIRVEC_ADJ - Adjust size of lock manager directory system vector
012C    637 ;++
012C    638 ;
012C    639 ; FUNCTIONAL DESCRIPTION:
012C    640 ;
012C    641 ;       Adjust the length of the lock manager directory system vector to accomodate
012C    642 ;       nodes that may be added to the cluster by the state transition now in
012C    643 ;       progress.  This routine is also used to initially create the lock manager
012C    644 ;       directory system vector.  This routine never changes the contents of the
012C    645 ;       vector -- it only adjusts the size.
012C    646 ;
012C    647 ;       The purpose of the lock manager directory system vector is to encode the
012C    648 ;       node serving as the directory for each resource.  Resource names are hashed
012C    649 ;       and the directory system vector indicates the CSID of the node serving as
012C    650 ;       the directory for that resource.  The local node always appears as a 0
012C    651 ;       entry.  Each node may have 0, 1, or more entries depending upon the setting
012C    652 ;       of the LCK$GB_LCKDIRWT system parameter.
012C    653 ;
012C    654 ; CALLING SEQUENCE:
012C    655 ;
012C    656 ;       JSB     CNX$DIRVEC_ADJ
012C    657 ;       IPL is IPL$_SCS
012C    658 ;
012C    659 ; INPUT PARAMETERS:
012C    660 ;
012C    661 ;       LCK$GL_DIRVEC is 0 or points to the current directory vector.
012C    662 ;
012C    663 ; OUTPUT PARAMETERS:
012C    664 ;
012C    665 ;       LCK$GL_DIRVEC points to the adjusted directory vector.
012C    666 ;
012C    667 ; COMPLETION CODES:
012C    668 ;
012C    669 ;       R0 : Success/Failure
012C    670 ;               Succeeds if vector is big enough or can be expanded as required.
012C    671 ;               Fails if vector cannot be expanded as required.
012C    672 ;
012C    673 ; SIDE EFFECTS:
012C    674 ;
012C    675 ;       R1-R2 are destroyed.
012C    676 ;
012C    677 ; DATA STRUCTURES:
012C    678 ;
012C    679 ;       The following is a picture of the DIRVEC structure
012C    680 ;
012C    681 ;        31           24 23           16 15           08 07           00
012C    682 ;       +---------------+---------------+---------------+---------------+
012C    683 ;       |         Number of valid entries (lognwords)                   |
012C    684 ;       +---------------+---------------+---------------+---------------+
012C    685 ;       |         Number of allocated entries (longwords)               |
012C    686 ;       +---------------+---------------+---------------+---------------+
012C    687 ;       |DYN$C_CLU_LCKDIR DYN$C_CLU     |               SIZE            |
012C    688 ;       +---------------+---------------+---------------+---------------+
012C    689 ;       |         First entry (CSID or 0 if local system)               |
012C    690 ;       +---------------+---------------+---------------+---------------+
012C    691 ;       .                                                               .
012C    692 ;       .                                                               .
```

```
                        012C   693 ;
                        012C   694 ;       +---------------+---------------+---------------+---------------+
                        012C   695 ;       |         Last entry (CSID or 0 if local system)                |
                        012C   696 ;       +---------------+---------------+---------------+---------------+
                        012C   697 ;
                        012C   698 ;
                        012C   699 ;--
                        012C   700
                        012C   701 CNX$DIRVEC_ADJ::
                  38    BB 012C   702          PUSHR   #^M<R3,R4,R5>           ; Save registers
                  51    D4 012E   703          CLRL    R1                      ; Clear node counter
                  55    D4 0130   704          CLRL    R5                      ; Clear weight accumulator
              FEF5      30 0132   705          BSBW    CNX$SCAN_CSBS           ; Iterate over all CSBS
             0F 50      E9 0135   706          BLBC    R0,20$                  ; Branch when done
       09 60 A3   11    E1 0138   707          BBC     #CSB$V_SELECTED, -      ; Branch if node is not selected
                        013D   708                  CSB$L_STATUS(R3),10$
                  51    D6 013D   709          INCL    R1                      ; Count the node
          50  54 A3    3C 013F   710          MOVZWL  CSB$W_LCKDIRWT(R3),R0   ; Node weight
             55   50    C0 0143   711          ADDL2   R0,R5                   ; Count weight of the node
                        05 0146   712 10$:     RSB                             ; End of iteration
                        0147   713
                  55    D5 0147   714 20$:     TSTL    R5                      ; Any weight seen?
                  03    12 0149   715          BNEQ    30$                     ; Weight seen, use it
             55   51    D0 014B   716          MOVL    R1,R5                   ; Use number of nodes
       54  00000000'GF    D0 014E   717 30$:     MOVL    G^LCK$GL_DIRVEC,R4      ; Address of current directory vector
                  0E    13 0155   718          BEQL    40$                     ; No directory at present
          54  F4 A4    9E 0157   719          MOVAB   -12(R4),R4              ; Point to base of block
             55   64    D1 015B   720          CMPL    0(R4),R5                ; Compare present and future needs
                  5C    13 015E   721          BEQL    60$                     ; No change needed
                  03    1F 0160   722          BLSSU   40$                     ; Branch if present < future
                     55 64    D0 0162   723          MOVL    0(R4),R5        ; Maximum of present and future
                  51 55 04    C5 0165   724 40$:     MULL3   #4,R5,R1        ; Convert to byte count
                     51 0C    C0 0169   725          ADDL2   #3*4,R1         ; Add overhead
                00000000'GF    16 016C   726          JSB     G^EXE$ALONONPAGED       ; Allocate space for new vector
                        4A 50  E9 0172   727          BLBC    R0,70$          ; Can't allocate memory
          00000000'GF   0C A2  9E 0175   728          MOVAB   12(R2),G^LCK$GL_DIRVEC  ; Store address of new dirvec
                        62     D4 017D   729          CLRL    (R2)            ; Clear divisor longword
                     04 A2  55 D0 017F   730          MOVL    R5,4(R2)        ; Number of slots allocated
                     08 A2  51 B0 0183   731          MOVW    R1,8(R2)        ; Store size of new vector
                  0A A2    65 8F 90 0187   732          MOVB    #DYN$C_CLU,10(R2)       ; Fill in block type
                     0B A2  07 90 018C   733          MOVB    #DYN$C_CLU_LCKDIR,11(R2) ; Fill in block sub-type
                        54  D5 0190   734          TSTL    R4              ; Was there a previous block?
                        03  12 0192   735          BNEQ    50$             ; Branch if yes
                     54 52  D0 0194   736          MOVL    R2,R4           ; Pretend new is old
                     62 64  D0 0197   737 50$:     MOVL    0(R4),0(R2)     ; Copy divisor
                  50 04 64  C5 019A   738          MULL3   0(R4),#4,R0     ; Size of block to copy
               51 04  04 A2  C5 019E   739          MULL3   4(R2),#4,R1     ; Size of new block to fill or zero
                        52  DD 01A3   740          PUSHL   R2              ; Save address of new block
                        54  DD 01A5   741          PUSHL   R4              ; Save address of old block
0C A2  51  00  0C A4    50  2C 01A7   742          MOVC5   R0,12(R4),#0,R1,12(R2)  ; Copy data and zero the remainder of the ne
                        01  BA 01AF   743          POPR    #^M<R0>         ; Restore address of ?old? block
                     8E 50  D1 01B1   744          CMPL    R0,(SP)+        ; Test for old=new
                        06  13 01B4   745          BEQL    60$             ; Branch if no previous block to delete
                00000000'GF    16 01B6   746          JSB     G^EXE$DEANONPAGED       ; Deallocate old LCKDIR block
                     50 01  D0 01BC   747 60$:     MOVL    S^#SS$_NORMAL,R0        ; Mark success
                        38  BA 01BF   748 70$:     POPR    #^M<R3,R4,R5>           ; Restore register
                        05  01C1   749          RSB
```

```
                                01C2   751          .SBTTL  CNX$DIRVEC_FILL - Update contents of lock manager directory system v
                                01C2   752 ;++
                                01C2   753 ;
                                01C2   754 ; FUNCTIONAL DESCRIPTION:
                                01C2   755 ;
                                01C2   756 ;       Update the contents of the lock manager directory system vector to accuratel
                                01C2   757 ;       reflect the current state of the cluster.  It is assumed that the vector
                                01C2   758 ;       has previously been adjusted so that it can contain the data that will be
                                01C2   759 ;       stored.
                                01C2   760 ;
                                01C2   761 ; CALLING SEQUENCE:
                                01C2   762 ;
                                01C2   763 ;       JSB     CNX$DIRVEC_FILL
                                01C2   764 ;       IPL is IPL$_SCS
                                01C2   765 ;
                                01C2   766 ; INPUT PARAMETERS:
                                01C2   767 ;
                                01C2   768 ;       LCK$GL_DIRVEC contains address of directory system vector
                                01C2   769 ;
                                01C2   770 ; OUTPUT PARAMETERS:
                                01C2   771 ;
                                01C2   772 ;       LCK$GL_DIRVEC contains address of updated directory system vector
                                01C2   773 ;
                                01C2   774 ; COMPLETION CODES:
                                01C2   775 ;
                                01C2   776 ;       NONE
                                01C2   777 ;
                                01C2   778 ; SIDE EFFECTS:
                                01C2   779 ;
                                01C2   780 ;       R0-R2 are destroyed.
                                01C2   781 ;
                                01C2   782 ;--
                                01C2   783
                                01C2   784 CNX$DIRVEC_FILL::
                    38    BB    01C2   785          PUSHR   #^M<R3,R4,R5>           ; Save registers
                                01C4   786 ;
                                01C4   787 ; Fill in DIRVEC, ignoring zero-weight nodes
                                01C4   788 ;
54  00000000'GF       D0        01C4   789          MOVL    G^LCK$GL_DIRVEC,R4      ; Address of first DIRVEC entry
    53    F4 A4       9E        01CB   790          MOVAB   -12(R4),R3              ; Pointer to true base of DIRVEC
                63    D4        01CF   791          CLRL    0(R3)                   ; Initialize divisor
52  00000000'GF       D0        01D1   792          MOVL    G^CLU$GL_CLUSVEC,R2     ; Address of CLUSVEC
55  00000000'GF       3C        01D8   793          MOVZWL  G^CLU$GW_MAXINDEX,R5    ; Number of entries in CLUSVEC
      51    82        D0        01DF   794 10$:     MOVL    (R2)+,R1                ; Address of CSB for cluster member?
                08    18        01E2   795          BGEQ    30$                     ; Branch if slot unused
    50  54 A1        3C        01E4   796          MOVZWL  CSB$W_LCKDIRWT(R1),R0   ; Weight of this node
                02    13        01E8   797          BEQL    30$                     ; Ignore zero-weight nodes
                2A    10        01EA   798          BSBB    100$                    ; Call processing routine
          F0 55       F5        01EC   799 30$:     SOBGTR  R5,10$                  ; Iterate over all CLUSVEC entries
                                01EF   800 ;
                                01EF   801 ; See if any non-zero weight nodes were found
                                01EF   802 ;
00000000'GF     54      D1      01EF   803          CMPL    R4,G^LCK$GL_DIRVEC      ; Were any entries fill in
                1B      12      01F6   804          BNEQ    70$                     ; Branch if yes
                                01F8   805 ;
                                01F8   806 ; No non-zero weight nodes were found.
                                01F8   807 ; Give every node a weight of one.
```

```
                                01F8   808 ;
52   00000000'GF    D0  01F8   809          MOVL    G^CLU$GL_CLUSVEC,R2      ; Address of CLUSVEC
55   00000000'GF    3C  01FF   810          MOVZWL  G^CLU$GW_MAXINDEX,R5     ; Number of entries in CLUSVEC
       51    82    D0  0206   811 40$:      MOVL    (R2)+,R1                 ; Address of CSB for cluster member?
             05    18  0209   812          BGEQ    50$                      ; Branch if slot unused
       50    01    D0  020B   813          MOVL    #1,R0                    ; Use weight of 1
             06    10  020E   814          BSBB    100$                     ; Use standard processing routine
         F3  55    F5  0210   815 50$:      SOBGTR  R5,40$                   ; Iterate over all CLUSVEC entries
                           0213   816
             38    BA  0213   817 70$:      POPR    #^M<R3,R4,R5>            ; Restore register
             05    05  0215   818          RSB
                           0216   819 ;
                           0216   820 ; Processing a cluster member.
                           0216   821 ;       R0: weight of node
                           0216   822 ;       R3: address of base of directory vector
                           0216   823 ;       R4: next available slot in directory vector
                           0216   824 ;       R0,R2 are destroyed
                           0216   825 ;
             52    DD    0216   826 100$:    PUSHL   R2                       ; Save register
       63    50    C0    0218   827          ADDL2   R0,0(R3)                 ; Increment divisor
    04 A3    63    D1    021B   828          CMPL    0(R3),4(R3)              ; Check for overflow
             14    14    021F   829          BGTR    190$                     ; Branch if no slots left
    52   4C A1    D0    0221   830          MOVL    CSB$L_CSID(R1),R2        ; CSID of this system
 02 60 A1    18    E1    0225   831          BBC     #CSB$V_LOCAL,-           ; Branch if this is the local node
                           022A   832                  CSB$L_STATUS(R1),110$
             52    D4    022A   833          CLRL    R2                       ; Use zero instead of local system CSID
       84    52    D0    022C   834 110$:    MOVL    R2,(R4)+                 ; Store entry in DIRVEC
         FA  50    F5    022F   835          SOBGTR  R0,110$                  ; Store <weight> entries
             04    BA    0232   836          POPR    #^M<R2>                  ; Restore register
             05    05    0234   837          RSB                              ; Return
                           0235   838
                           0235   839 190$:    BUG_CHECK       CNXMGRERR,FATAL ; DIRVEC is not long enough -- consistency c
                           0239   840
```

```
                     0239   842          .SBTTL  CNX$CLUB_WAIT - Do CLUB-Based 1 Second Wait
                     0239   843
                     0239   844 ;++
                     0239   845 ;
                     0239   846 ; FUNCTIONAL DESCRIPTION:
                     0239   847 ;
                     0239   848 ;       Do a 0-1 second timeout using the CLUB$B_FORK_BLOCK and
                     0239   849 ;       the FORK_WAIT service.
                     0239   850 ;       An immediate return is made to the caller's caller.
                     0239   851 ;       When the timout occurs, a return is made to the caller.
                     0239   852 ;
                     0239   853 ; CALLING SEQUENCE:
                     0239   854 ;
                     0239   855 ;       JSB     CNX$CLUB_WAIT
                     0239   856 ;       IPL is IPL$_SCS
                     0239   857 ;
                     0239   858 ; INPUT PARAMETERS:
                     0239   859 ;
                     0239   860 ;       NONE
                     0239   861 ;
                     0239   862 ; OUTPUT PARAMETERS:
                     0239   863 ;
                     0239   864 ;       R4 is address of CLUB
                     0239   865 ;
                     0239   866 ; COMPLETION CODES:
                     0239   867 ;
                     0239   868 ;       NONE
                     0239   869 ;
                     0239   870 ; SIDE EFFECTS:
                     0239   871 ;
                     0239   872 ;       R0-R2,R5 are destroyed
                     0239   873 ;
                     0239   874 ;--
                     0239   875
                     0239   876 CNX$CLUB_WAIT::
54   00000000'GF    D0  0239   877          MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
   55    00CC  C4    9E  0240   878          MOVAB   CLUB$B_FORK_BLOCK(R4),R5 ; Address of fork block
15 1C A5    00    E2  0245   879          BBSS    #CLUBFKB$V_FKB_BUSY, -  ; Branch if fork block is
                     024A   880                  CLUBFKB$L_STATUS(R5),10$ ;  not busy
                10  BA  024A   881          PUSHR   #^M<R4>                 ; Save return address
                     024C   882          ASSUME  CLUBFKB$B_FORK_BLOCK EQ 0
                     024C   883          ASSUME  CLUBFKB$S_FORK_BLOCK GE FKB$K_LENGTH
                     024C   884          FORK_WAIT                       ; Wait for the next clock tick
                54  DD  0252   885          POPL    R4                      ; Restore return address
   54    FF34  C5    9E  0254   886          MOVAB   -CLUB$B_FORK_BLOCK(R5),R4 ; Address of CLUB
01 1C A5    00    E5  0259   887          BBCC    #CLUBFKB$V_FKB_BUSY, -  ; Clear fork block busy indicator
                     025E   888                  CLUBFKB$L_STATUS(R5),10$
                05  025E   889          RSB                             ; Terminate the thread
                     025F   890
                     025F   891 10$:     BUG_CHECK       CNXMGRERR,FATAL ; Double use of fork block
```

```
                     0263   893          .SBTTL  CNX$CLUB_FORK - Do CLUB-Based Fork
                     0263   894
                     0263   895 ;++
                     0263   896 ;
                     0263   897 ; FUNCTIONAL DESCRIPTION:
                     0263   898 ;
                     0263   899 ;       Momentarily release control using the CLUB$B_FORK_BLOCK
                     0263   900 ;       fork block.
                     0263   901 ;       An immediate return is made to the caller's caller.
                     0263   902 ;       When the fork list cycles, a return is made to the caller.
                     0263   903 ;
                     0263   904 ; CALLING SEQUENCE:
                     0263   905 ;
                     0263   906 ;       JSB     CNX$CLUB_FORK
                     0263   907 ;       IPL is IPL$_SCS
                     0263   908 ;
                     0263   909 ; INPUT PARAMETERS:
                     0263   910 ;
                     0263   911 ;       NONE
                     0263   912 ;
                     0263   913 ; OUTPUT PARAMETERS:
                     0263   914 ;
                     0263   915 ;       R4 is address of CLUB
                     0263   916 ;
                     0263   917 ; COMPLETION CODES:
                     0263   918 ;
                     0263   919 ;       NONE
                     0263   920 ;
                     0263   921 ; SIDE EFFECTS:
                     0263   922 ;
                     0263   923 ;       R0-R2,R5 are destroyed
                     0263   924 ;
                     0263   925 ;--
                     0263   926
                     0263   927 CNX$CLUB_FORK::
54   00000000'GF     D0      0263   928          MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
   55   00CC C4     9E      026A   929          MOVAB   CLUB$B_FORK_BLOCK(R4),R5 ; Address of fork block
19 1C A5    00     E2      026F   930          BBSS    #CLUBFKB$V_FKB_BUSY, -  ; Branch if fork block is
                            0274   931                  CLUBFKB$L_STATUS(R5),10$         ;  not busy
     0B A5    08     90      0274   932          MOVB    #IPL$_SCS,FKB$B_FIPL(R5) ;Initialize fork IPL
              10     BA      0278   933          POPR    #^M<R4>                 ; Save return address
                            027A   934          ASSUME  CLUBFKB$B_FORK_BLOCK EQ 0
                            027A   935          ASSUME  CLUBFKB$S_FORK_BLOCK GE FKB$K_LENGTH
     00000000'GF     16      027A   936          JSB     G^EXE$FORK              ; Wait until the fork list cycles
              54     DD      0280   937          PUSHL   R4                      ; Restore return address
   54   FF34 C5     9E      0282   938          MOVAB   -CLUB$B_FORK_BLOCK(R5),R4 ; Address of CLUB
01 1C A5    00     E5      0287   939          BBCC    #CLUBFKB$V_FKB_BUSY, -  ; Clear fork block busy indicator
                            028C   940                  CLUBFKB$L_STATUS(R5),10$
              05     05      028C   941          RSB                             ; Terminate the thread
                            028D   942
                            028D   943 10$:     BUG_CHECK       CNXMGRERR,FATAL ; Double use of fork block
```

```
                                0291   945          .SBTTL  CNX$FIX_EPID - Add node specifier to EPIDs
                                0291   946
                                0291   947 ;++
                                0291   948 ;
                                0291   949 ; FUNCTIONAL DESCRIPTION:
                                0291   950 ;
                                0291   951 ;       This routine is called when a node joins a cluster to add the node
                                0291   952 ;       identification to the EPIDs of existing processes.  This assumes that
                                0291   953 ;       very little has happened up to this point and that therefore, only
                                0291   954 ;       fields within the PCB need to be updated.
                                0291   955 ;
                                0291   956 ; CALLING SEQUENCE:
                                0291   957 ;
                                0291   958 ;       JSB     CNX$FIX_EPID
                                0291   959 ;       IPL is IPL$_SYNC
                                0291   960 ;
                                0291   961 ; INPUT PARAMETERS:
                                0291   962 ;
                                0291   963 ;       NONE
                                0291   964 ;
                                0291   965 ; OUTPUT PARAMETERS:
                                0291   966 ;
                                0291   967 ;       NONE
                                0291   968 ;
                                0291   969 ; SIDE EFFECTS:
                                0291   970 ;
                                0291   971 ;       R0-R1 may be destroyed
                                0291   972 ;
                                0291   973 ;--
                                0291   974
                                0291   975 CNX$FIX_EPID::
                          1C    BB    0291   976          PUSHR   #^M<R2,R3,R4>                   ; Save registers
                                0293   977 ;
                                0293   978 ; Set the contents of the cell SCH$GW_LOCALNODE to the compressed form of
                                0293   979 ; the local node CSID.  Update the extended PID cells of all processes on
                                0293   980 ; the system.  (Null process will be updated multiple times, but no harm done)
                                0293   981 ;
                                0293   982          ASSUME  PCB$S_EPID_NODE_IDX EQ 8        ; Index is one byte
                                0293   983          ASSUME  PCB$S_EPID_NODE_SEQ EQ 2        ; Seqno is two bits
                                0293   984          ASSUME  PCB$V_EPID_NODE_SEQ EQ -        ; Seqno is right after index
                                0293   985                  <PCB$V_EPID_NODE_IDX + PCB$S_EPID_NODE_IDX>
                          0000000A    0293   986 NODE_WIDTH = PCB$S_EPID_NODE_IDX + PCB$S_EPID_NODE_SEQ
                                0293   987
  54    00000000'GF    D0    0293   988          MOVL    G^CLU$GL_CLUB,R4                ; Address of CLUB
        51    10 A4    D0    029A   989          MOVL    CLUB$L_LOCAL_CSB(R4),R1         ; Local CSB address
  53    00000000'GF    3E    029E   990          MOVAW   G^SCH$GW_LOCALNODE,R3           ; Get pointer to the localnode
        83    4C A1    90    02A5   991          MOVB    CSB$W_CSID_IDX(R1),(R3)+        ; Store the index
83    4E A1    FC 8F    88    02A9   992          BICB3   #^XFC,CSB$W_CSID_SEQ(R1),(R3)+  ; Store the sequence, clearing high
           53    73    3C    02AF   993          MOVZWL  -(R3),R3                        ; Put localnode in a register
  52    00000000'GF    3C    02B2   994          MOVZWL  G^SCH$GL_MAXPIX,R2              ; Get index of highest PCB
  50    00000000'GF    D0    02B9   995          MOVL    G^SCH$GL_PCBVEC,R0              ; Get address of pcb vector
        54    80    D0    02C0   996 10$:      MOVL    (R0)+,R4                        ; Get address of PCB
        15    53    F0    02C3   997          INSV    R3,#PCB$V_EPID_NODE_IDX,-       ; Set the node bits in the
     64 A4    0A          02C6   998                  #NODE_WIDTH,PCB$L_EPID(R4)      ;   extended PID field
           68 A4    D5    02C9   999          TSTL    PCB$L_EOWNER(R4)                ; Is it a subprocess?
              06    13    02CC  1000          BEQL    20$                             ; Not subprocess, leave it alone
        15    53    F0    02CE  1001          INSV    R3,#PCB$V_EPID_NODE_IDX,-       ; Set the node bits in the
```

```
                68 A4    0A      02D1  1002                        #NODE_WIDTH,PCB$L_EOWNER(R4)    ;   subprocess owner field
                    E9 52    F4  02D4  1003 20$:     SOBGEQ      R2,10$                          ; Try next PCB
                        1C   BA  02D7  1004          POPR        #^M<R2,R3,R4>                   ; Restore registers
                             05  02D9  1005          RSB
```

```
                                                                .SBTTL  CNX$CHECK_QUORUM - Check Presence of Quorum and Hang if Absent

                                                        ;++
                                                        ; FUNCTIONAL DESCRIPTION:
                                                        ;
                                                        ;       Test for the presence of a quorum of "live" nodes.  If a quorum is
                                                        ;       absent, loop at IPL 4 until a quorum is restored.  This serves to
                                                        ;       block activity that might use shared resources for which the locks
                                                        ;       may not longer be valid (since this node may have been failed out
                                                        ;       by someone else).  A fork loop at IPL 4 is used because this blocks
                                                        ;       all process-based activity while allowing timer-based activity and
                                                        ;       SCS/connection management functions to continue uninterrupted.
                                                        ;
                                                        ; CALLING SEQUENCE:
                                                        ;
                                                        ;       JSB/BSBW/BSBB   CNX$CHECK_QUORUM
                                                        ;       IPL is IPL$_SCS
                                                        ;
                                                        ; INPUT PARAMETERS:
                                                        ;
                                                        ;       NONE
                                                        ;
                                                        ; OUTPUT PARAMETERS:
                                                        ;
                                                        ;       NONE
                                                        ;
                                                        ; COMPLETION CODES:
                                                        ;
                                                        ;       NONE
                                                        ;
                                                        ; SIDE EFFECTS:
                                                        ;
                                                        ;       R0 and R1 are destroyed.
                                                        ;
                                                        ;--

                                                CNX$CHECK_QUORUM::
                3C    BB    02DA                        PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
                68    10    02DC                        BSBB    COMPUTE_DYNAMIC_QUORUM  ; Is dynamic quorum present?
              1C 50   E8    02DE                        BLBS    R0,10$                  ; Branch if present
17 1C A4      00      E1    02E1                        BBC     #CLUB$V_CLUSTER, -      ; Branch if not cluster member
                            02E6                                CLUB$L_FLAGS(R4),10$    ;   and return
12 1C A4      1C      E5    02E6                        BBCC    #CLUB$V_QUORUM, -       ; Mark quorum absent, branch if
                            02EB                                CLUB$L_FLAGS(R4),10$    ;   already absent
  00000000'GF   16      02EB                        JSB     G^EXE$CLUTRANIO         ; Throw disks into mount verification
                            02F1                                                        ; to inhibit further I/O initiations
 50   0000'CF   9E      02F1                        MOVAB   W^LOSEQUORUM_MSG,R0     ; Quorum lost message
                55      D4    02F6                        CLRL    R5                      ; No CSB address
              FD05'     30    02F8                        BSBW    CNX$CONFIG_CHANGE       ; Log event
                28      10    02FB                        BSBB    200$                    ; Fork on I/O Post queue
                3C      BA    02FD                10$:    POPR    #^M<R2,R3,R4,R5>        ; Restore registers
                05      02FF                        RSB                             ; Return to caller
                            0300

                                                        ;
                                                        ; I/O Post fork routine.
                                                        ; Re-fork until a dynamic quorum is present
```

```
                                0300  1064 ;
                                0300  1065         ASSUME  IPL$_SCS GT IPL$_IOPOST
                                0300  1066 100$:    DSBINT  #IPL$_SCS                       ; Raise IPL
                          3E    0306  1067         BSBB    COMPUTE_DYNAMIC_QUORUM          ; Is dynamic quorum present?
                       04 50    0308  1068         BLBS    R0,110$                         ; Branch if present
                          18    030B  1069         BSBB    200$                            ; Fork on same block
                          12    030D  1070         BRB     120$                            ; Branch to common exit
                                030F  1071
                10000000 8F     030F  1072 110$:    BISL2   #CLUB$M_QUORUM, -               ; Mark quorum present
                                0317  1073                 CLUB$L_FLAGS(R4)
                     0000'CF    0317  1074         MOVAB   W^GAINQUORUM_MSG,R0             ; Quorum lost message
                          55    031C  1075         CLRL    R5                              ; No CSB address
                        FCDF'   031E  1076         BSBW    CNX$CONFIG_CHANGE               ; Log event
                                0321  1077 120$:    ENBINT                                  ; Restore IPL
                          05    0324  1078         RSB
                                0325  1079
                                0325  1080 ;
                                0325  1081 ; Set up and queue fork block onto I/O Post processing queue
                                0325  1082 ;
                     0174 C4    0325  1083 200$:    MOVAB   CLUB$B_HANG_FKB(R4),R5          ; Address of fork block
                  0A A5    08   032A  1084         MOVB    #DYN$C_FRK, -                   ; Block type
                                032E  1085                 FKB$B_TYPE(R5)
                  0B A5    04   032E  1086         MOVB    #IPL$_IOPOST, -                 ; IPL = 4
                                0332  1087                 FKB$B_FIPL(R5)
               0C A5    CB AF   0332  1088         MOVAB   B^100$,FKB$L_FPC(R5)            ; Restart PC address
            50  00000000'GF    0337  1089         MOVAQ   G^IOC$GL_PSFL,R0                ; Address of queue header
                  04 B0    65   033E  1090         INSQUE  (R5),@4(R0)                     ; Queue to I/O Post-processing queue
                                0342  1091         SOFTINT #IPL$_IOPOST                    ; Wake up dispatcher
                          05    0345  1092         RSB
```

```
                                0346  1094          .SBTTL  COMPUTE_DYNAMIC_QUORUM - Calculate dynamic quorum presence
                                0346  1095
                                0346  1096 ;++
                                0346  1097 ;
                                0346  1098 ; FUNCTIONAL DESCRIPTION:
                                0346  1099 ;
                                0346  1100 ;       Determine whether a quorum is dynamically present, i.e., whether
                                0346  1101 ;       there are live connections to a quorum of nodes.  In addition to
                                0346  1102 ;       calculating based on available nodes, add extra votes if the
                                0346  1103 ;       quorum file is a "member" of the cluster and is dynamically
                                0346  1104 ;       accessible to the local node.
                                0346  1105 ;
                                0346  1106 ; CALLING SEQUENCE:
                                0346  1107 ;
                                0346  1108 ;       JSB     COMPUTE_DYNAMIC_QUORUM
                                0346  1109 ;       IPL is IPL$_SCS
                                0346  1110 ;
                                0346  1111 ; INPUT PARAMETERS:
                                0346  1112 ;
                                0346  1113 ;       CSB'S in list with CSB$M_MEMBER bit set
                                0346  1114 ;
                                0346  1115 ; OUTPUT PARAMETERS:
                                0346  1116 ;
                                0346  1117 ;       R4 is address of CLUB
                                0346  1118 ;
                                0346  1119 ; COMPLETION CODES:
                                0346  1120 ;
                                0346  1121 ;       R0 = TRUE indicates that a quorum is dynamically present
                                0346  1122 ;       R0 = FALSE indicates that a quorum is not dynamically present
                                0346  1123 ;
                                0346  1124 ; SIDE EFFECTS:
                                0346  1125 ;
                                0346  1126 ;       NONE
                                0346  1127 ;--
                                0346  1128
                                0346  1129 COMPUTE_DYNAMIC_QUORUM:
                          0E    BB    0346  1130          PUSHR   #^M<R1,R2,R3>           ; Save registers
                          52    D4    0348  1131          CLRL    R2                      ; Votes to be included
                       FCDD     30    034A  1132          BSBW    CNX$SCAN_CSBS
                     1D 50      E9    034D  1133          BLBC    R0,30$                  ; All done
          17 60 A3     01      E1    0350  1134          BBC     #CSB$V_MEMBER, -        ; Branch if node is not cluster member
                                     0355  1135                  CSB$L_STATUS(R3),20$
          12 60 A3     00      E0    0355  1136          BBS     #CSB$V_LONG_BREAK, -    ; Branch if long break seen
                                     035A  1137                  CSB$L_STATUS(R3),20$
          06 60 A3     18      E0    035A  1138          BBS     #CSB$V_LOCAL, -         ; Branch if local node
                                     035F  1139                  CSB$L_STATUS(R3),10$    ;   and include votes
             01    43 A3      91    035F  1140          CMPB    CSB$B_STATE(R3), -      ; Is connection open?
                                     0363  1141                  #CSB$K_OPEN
                          07    12    0363  1142          BNEQ    20$                     ; Branch if not open
             50    50 A3      3C    0365  1143 10$:      MOVZWL  CSB$W_VOTES(R3),R0      ; Sum votes
                52    50      C0    0369  1144          ADDL2   R0,R2
                          05    036C  1145 20$:      RSB                             ; Continue scan
                                     036D  1146
          0D 1C A4     19      E1    036D  1147 30$:      BBC     #CLUB$V_QF_VOTE, -      ; Branch if disk is not "member"
                                     0372  1148                  CLUB$L_FLAGS(R4),40$
          08 1C A4     1E      E1    0372  1149          BBC     #CLUB$V_QF_DYNVOTE, -   ; Branch if connection can't be trusted yet
                                     0377  1150                  CLUB$L_FLAGS(R4),40$
```

```
                50   00AE C4   3C   0377  1151          MOVZWL  CLUB$W_QDVOTES(R4),R0   ; Votes for quorum disk
                     52   50   C0   037C  1152          ADDL2   R0,R2
                  51    20 A4   3C   037F  1153 40$:     MOVZWL  CLUB$W_QUORUM(R4),R1    ; Cluster quorum
                          50   D4   0383  1154          CLRL    R0                      ; Assume no dynamic quorum
                     51   52   D1   0385  1155          CMPL    R2,R1                   ; Check for presence of quorum
                          03   1F   0388  1156          BLSSU   50$                     ; Branch if quorum is not dynamically presen
                     50   01   D0   038A  1157          MOVL    #1,R0                   ; Quorum is dynamically present
                          0E   BA   038D  1158 50$:     POPR    #^M<R1,R2,R3>           ; Restore registers
                               05   038F  1159          RSB
```

```
                     0390  1161          .SBTTL  CNX$QUORUM_CALC - Calculate Quorum and related data
                     0390  1162
                     0390  1163 ;++
                     0390  1164 ;
                     0390  1165 ; FUNCTIONAL DESCRIPTION:
                     0390  1166 ;
                     0390  1167 ;       Calculate the Quorum, Votes, and Number of nodes for the selected
                     0390  1168 ;       nodes.  Quorum is maximized with the value in CLUB$W_QUORUM.
                     0390  1169 ;       Quorum disk votes is minimized with the value in CLUB$W_QDVOTES.
                     0390  1170 ;       Quorum is ratcheted up to be greater than half of the number of votes.
                     0390  1171 ;
                     0390  1172 ; CALLING SEQUENCE:
                     0390  1173 ;
                     0390  1174 ;       JSB     CNX$QUORUM_CALC
                     0390  1175 ;       IPL is IPL$_SCS
                     0390  1176 ;
                     0390  1177 ; INPUT PARAMETERS:
                     0390  1178 ;
                     0390  1179 ;       CSB'S in list
                     0390  1180 ;
                     0390  1181 ; OUTPUT PARAMETERS:
                     0390  1182 ;
                     0390  1183 ;       R4 is address of CLUB
                     0390  1184 ;       R2 is number of nodes in cluster
                     0390  1185 ;       R1 is maximum of quorum in selected CSBs and CLUB$W_QUORUM
                     0390  1186 ;       R0 is sum of the votes of the selected CSBs
                     0390  1187 ;
                     0390  1188 ; COMPLETION CODES:
                     0390  1189 ;
                     0390  1190 ;       NONE
                     0390  1191 ;
                     0390  1192 ; SIDE EFFECTS:
                     0390  1193 ;
                     0390  1194 ;       CLUB$W_NEWQUORUM contains calculated quorum
                     0390  1195 ;       CLUB$W_NEWQDVOTES contains calculated quorum disk votes
                     0390  1196 ;       CLUB$V_QF_NEWVOTE indicates whether or not quorum disk should be a "member"
                     0390  1197 ;
                     0390  1198 ;--
                     0390  1199
                     0390  1200 CNX$QUORUM_CALC::
         00E8 8F     BB  0390  1201          PUSHR   #^M<R3,R5,R6,R7>        ; Save registers
54 00000000'GF       D0  0394  1202          MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
   51    20 A4       3C  0398  1203          MOVZWL  CLUB$W_QUORUM(R4),R1    ; R1 = quorum
             52       D4  039F  1204          CLRL    R2                      ; R2 = Number of nodes
             55       D4  03A1  1205          CLRL    R5                      ; R5 = votes
   56    00AE C4     3C  03A3  1206          MOVZWL  CLUB$W_QDVOTES(R4),R6   ; R6 = Min quorum disk votes
         57    00    D2  03A8  1207          MCOML   #0,R7                   ; Turn all quorum disk bits on
            FC7C     30  03AB  1208          BSBW    CNX$SCAN_CSBS
           2A 50     E9  03AE  1209          BLBC    R0,100$                 ; All done
24 60 A3    11       E1  03B1  1210          BBC     #CSB$V_SELECTED, -      ; Branch if not selected
                         03B6  1211                  CSB$L_STATUS(R3),40$
   50    60 A3       D2  03B6  1212          MCOML   CSB$L_STATUS(R3),R0     ; Get complement of QF_ACTIVE and QF_SAME
         57    50    CA  03BA  1213          BICL2   R0,R7                   ; Cumulate QF_ACTIVE and QF_SAME bits
             52       D6  03BD  1214          INCL    R2                      ; Count another node
   51    52 A3       B1  03BF  1215          CMPW    CSB$W_QUORUM(R3),R1     ; Maximize quorum
             04       1B  03C3  1216          BLEQU   30$                     ; OK
   51    52 A3       3C  03C5  1217          MOVZWL  CSB$W_QUORUM(R3),R1     ; New maximum
```

```
                50    50 A3    3C   03C9  1218 30$:     MOVZWL  CSB$W_VOTES(R3),R0       ; Sum votes
                55    50 C0    CD   03CD  1219           ADDL2   R0,R5
                56    56 A3    B1   03D0  1220           CMPW    CSB$W_QDVOTES(R3),R6     ; Minimize quorum disk votes over all nodes
                      04 1E    03D4  1221                BGEQU   40$                      ; Branch if no new minimum
                56    56 A3    3C   03D6  1222           MOVZWL  CSB$W_QDVOTES(R3),R6     ; New minimum
                      05 03DA  1223 40$:     RSB                                      ; Continue scan
                         03DB  1224
                   50    55 D0    03DB  1225 100$:     MOVL    R5,R0                    ; Number of votes
             00 1C A4    1A E5    03DE  1226           BBCC    #CLUB$V_QF_NEWVOTE, -    ; Assume no vote for quorum disk
                         03E3  1227                    CLUB$L_FLAGS(R4),110$
                0C 57    03 E1    03E3  1228 110$:     BBC     #CSB$V_QF_SAME,R7,130$   ; Branch if same disk not specified by all
                08 57    09 E1    03E7  1229           BBC     #CSB$V_QF_ACTIVE,R7,130$ ; Branch if any disk is inactive
                   50    56 C0    03EB  1230 120$:     ADDL2   R6,R0                    ; Count quorum disk vote in total
             00 1C A4    1A E2    03EE  1231           BBSS    #CLUB$V_QF_NEWVOTE, -    ; Tag the quorum disk as contributing a vote
                         03F3  1232                    CLUB$L_FLAGS(R4),130$
             53    50    02 C1    03F3  1233 130$:     ADDL3   #2,R0,R3                 ; Votes + 2
                   53    02 C6    03F7  1234           DIVL2   #2,R3                    ; (Votes + 2) / 2
                   51    53 D1    03FA  1235           CMPL    R3,R1                    ; (Votes + 2)/2 < quorum?
                         03 1F    03FD  1236           BLSSU   140$                     ; Branch if yes
                   51    53 D0    03FF  1237           MOVL    R3,R1                    ; Up the quorum
             00A4  C4    51 B0    0402  1238 140$:     MOVW    R1,CLUB$W_NEWQUORUM(R4)  ; Computed quorum
               46  A4    56 B0    0407  1239           MOVW    R6, -                    ; Quorum disk votes
                         040B  1240                    CLUB$W_NEWQDVOTES(R4)
                   00E8 8F    BA   040B  1241           POPR    #^M<R3,R5,R6,R7>         ; Restore registers
                         05 040F  1242           RSB                                      ; Return
                            0410  1243
```

```
0410  1245          .SBTTL  CNX$DISPATCH - Second level input dispatcher for CNX
0410  1246
0410  1247  ;++
0410  1248  ;
0410  1249  ; FUNCTIONAL DESCRIPTION:
0410  1250  ;
0410  1251  ;       This routine is called by the first level input message dispatcher
0410  1252  ;       whenver the facility code in the incoming message is CLSMSG$K_FAC_CNX.
0410  1253  ;
0410  1254  ; CALLING SEQUENCE:
0410  1255  ;
0410  1256  ;       JSB     CNX$DISPATCH
0410  1257  ;       IPL is IPL$_SCS
0410  1258  ;
0410  1259  ; INPUT PARAMETERS:
0410  1260  ;
0410  1261  ;       R2 is address of message
0410  1262  ;       R3 is address of CSB
0410  1263  ;       R4 is address of PDT
0410  1264  ;       R5 :
0410  1265  ;               If CLSMSG$L_RSPID(R2) is non-zero, then R5 is the address of
0410  1266  ;               a noninitialized CDRP.
0410  1267  ;
0410  1268  ; OUTPUT PARAMETERS:
0410  1269  ;
0410  1270  ;       NONE
0410  1271  ;
0410  1272  ; SIDE EFFECTS:
0410  1273  ;
0410  1274  ;       R0-R5 may be destroyed
0410  1275  ;
0410  1276  ;--
0410  1277
0410  1278  CNX$DISPATCH::
0410  1279          DISPATCH        CLSMSG$B_FUNC(R2),TYPE=B,PREFIX=CLMCNX$K_FNC_, -
0410  1280                  < -
0410  1281                  <STATUS,CNX$RCVD_STATUS>, -     ; Status message
0410  1282                  <ENTER,CNX$RCVD_ENTER>, -       ; Cluster membership request message
0410  1283                  <LOCK,CNX$RCVD_LOCK>, -         ; Coordinator lock request
0410  1284                  <DESC,CNX$RCVD_DESC>, -         ; Node description
0410  1285                  <VEC,CNX$RCVD_VEC>, -           ; Cluster vector slot description
0410  1286                  <FORM,CNX$RCVD_FORM>, -         ; New cluster proposal
0410  1287                  <RECONFIG,CNX$RCVD_RECONFIG>, - ; Reconfiguration proposal
0410  1288                  <JOIN,CNX$RCVD_JOIN>, -         ; Proposal to add node
0410  1289                  <UNLOCK,CNX$RCVD_UNLOCK>, -     ; Unlock/Undo request
0410  1290                  <PH2,CNX$RCVD_PH2>, -           ; Phase 2 message
0410  1291                  <READY,CNX$RCVD_READY>, -       ; Ready for failover step
0410  1292                  <DOSTEP,CNX$RCVD_DOSTEP>, -     ; Do failover step
0410  1293                  <QUORUM,CNX$RCVD_QUORUM>, -     ; Quorum change message
0410  1294                  <TRNSTS,CNX$RCVD_TRNSTS>, -     ; Transition status request
0410  1295                  <TOPOLOGY,CNX$RCVD_TOPOLOGY> -  ; Topology exchange request
0410  1296                  >
0433  1297          BUG_CHECK       CNXMGRERR,FATAL         ; Invalid function code
0437  1298
0437  1299
0437  1300          .END
```

```
$$BASE                          = 00000001
$$DISPL                         = 00000010
$$GENSW                         = 00000001
$$HIGH                          = 0000000F
$$LIMIT                         = 0000000E
$$LOW                           = 00000001
$$MNSW                          = 00000001
$$MXSW                          = 00000001
BUG$_CLUEXIT                      ********  X   02
BUG$_CNXMGRERR                    ********  X   02
CDRP$L_MSG_BUF                  = 0000001C
CDRP$L_RETRSPID                 = 00000058
CLMCNX$K_FNC_DESC               = 00000005
CLMCNX$K_FNC_DOSTEP             = 0000000C
CLMCNX$K_FNC_ENTER              = 00000002
CLMCNX$K_FNC_FORM               = 00000007
CLMCNX$K_FNC_JOIN               = 00000009
CLMCNX$K_FNC_LOCK               = 00000003
CLMCNX$K_FNC_PH2                = 0000000A
CLMCNX$K_FNC_QUORUM             = 0000000D
CLMCNX$K_FNC_READY              = 0000000B
CLMCNX$K_FNC_RECONFIG           = 00000008
CLMCNX$K_FNC_STATUS             = 00000001
CLMCNX$K_FNC_TOPOLOGY           = 0000000F
CLMCNX$K_FNC_TRNSTS             = 0000000E
CLMCNX$K_FNC_UNLOCK             = 00000004
CLMCNX$K_FNC_VEC                = 00000006
CLSMSG$B_FUNC                   = 00000009
CLSMSG$L_RSPID                  = 00000004
CLU$GL_CLUB                       ********  X   02
CLU$GL_CLUSVEC                    ********  X   02
CLU$GW_MAXINDEX                   ********  X   02
CLUB$B_FORK_BLOCK               = 000000CC
CLUB$B_HANG_FKB                 = 00000174
CLUB$L_COORD                    = 0000005C
CLUB$L_CSBQFL                   = 00000000
CLUB$L_FLAGS                    = 0000001C
CLUB$L_LOCAL_CSB                = 00000010
CLUB$M_QUORUM                   = 10000000
CLUB$V_CLUSTER                  = 00000000
CLUB$V_QF_DYNVOTE               = 0000001E
CLUB$V_QF_NEWVOTE               = 0000001A
CLUB$V_QF_VOTE                  = 00000019
CLUB$V_QUORUM                   = 0000001C
CLUB$V_TRANSITION               = 0000001D
CLUB$W_NEWQDVOTES               = 00000046
CLUB$W_NEWQUORUM                = 000000A4
CLUB$W_NEXT_CSID                = 00000064
CLUB$W_QDVOTES                  = 000000AE
CLUB$W_QUORUM                   = 00000020
CLUBFKB$B_FORK_BLOCK            = 00000000
CLUBFKB$L_STATUS                = 0000001C
CLUBFKB$S_FORK_BLOCK            = 00000018
CLUBFKB$V_FKB_BUSY              = 00000000
CNX$ASSIGN_CSID                   000000B6  RG    02
CNX$BUGCHECK_CLUSTER              00000015  RG    02
CNX$CHECK_QUORUM                  000002DA  RG    02
CNX$CLUB_FORK                     00000263  RG    02
CNX$CLUB_WAIT                     00000239  RG    02
CNX$CONFIG_CHANGE                 ********    X   02
CNX$DECREFCNT                     ********    X   02
CNX$DIRVEC_ADJ                    0000012C  RG    02
CNX$DIRVEC_FILL                   000001C2  RG    02
CNX$DISPATCH                      00000410  RG    02
CNX$FIX_EPID                      00000291  RG    02
CNX$INIT_CSBS                     00000019  RG    02
CNX$MARK_LOCKED                   000000F8  RG    02
CNX$MARK_UNLOCKED                 00000105  RG    02
CNX$QUORUM_CALC                   00000390  RG    02
CNX$RCVD_DESC                     ********    X   02
CNX$RCVD_DOSTEP                   ********    X   02
CNX$RCVD_ENTER                    ********    X   02
CNX$RCVD_FORM                     ********    X   02
CNX$RCVD_JOIN                     ********    X   02
CNX$RCVD_LOCK                     ********    X   02
CNX$RCVD_PH2                      ********    X   02
CNX$RCVD_QUORUM                   ********    X   02
CNX$RCVD_READY                    ********    X   02
CNX$RCVD_RECONFIG                 ********    X   02
CNX$RCVD_STATUS                   ********    X   02
CNX$RCVD_TOPOLOGY                 ********    X   02
CNX$RCVD_TRNSTS                   ********    X   02
CNX$RCVD_UNLOCK                   ********    X   02
CNX$RCVD_VEC                      ********    X   02
CNX$RESP_FORGET                   00000000  RG    02
CNX$SCAN_CSBS                     0000002A  RG    02
CNX$SCAN_CSBS_EXIT                00000048  RG    02
CNX$SCAN_CSBS_FORK                00000083  RG    02
CNX$SCAN_CSBS_RETRY               00000055  RG    02
CNX$SEND_FORGET                   00000009  RG    02
CNX$SEND_MSG_CSB                  ********    X   02
COMPUTE_DYNAMIC_QUORUM            00000346  R     02
CSB$B_REF_CNT                   = 0000006C
CSB$B_STATE                     = 00000043
CSB$K_OPEN                      = 00000001
CSB$L_CLUB                      = 00000064
CSB$L_CSID                      = 0000004C
CSB$L_STATUS                    = 00000060
CSB$L_SYSQFL                    = 00000000
CSB$V_LOCAL                     = 00000018
CSB$V_LOCKED                    = 00000010
CSB$V_LONG_BREAK                = 00000000
CSB$V_MEMBER                    = 00000001
CSB$V_QF_ACTIVE                 = 00000009
CSB$V_QF_SAME                   = 00000003
CSB$V_SELECTED                  = 00000011
CSB$W_CSID_IDX                  = 0000004C
CSB$W_CSID_SEQ                  = 0000004E
CSB$W_LCKDIRWT                  = 00000054
CSB$W_QDVOTES                   = 00000056
CSB$W_QUORUM                    = 00000052
CSB$W_VOTES                     = 00000050
DYN$C_CLU                       = 00000065
DYN$C_CLU_LCKDIR                = 00000007
```

```
DYN$C_FRK                       = 00000008
EXE$ACONONPAGED                   ********    X   02
EXE$CLUTRANIO                     ********    X   02
EXE$DEANONPAGED                   ********    X   02
EXE$FORK                          ********    X   02
EXE$FORK_WAIT                     ********    X   02
FKB$B_FIPL                      = 0000000B
FKB$B_TYPE                      = 0000000A
FKB$K_LENGTH                    = 00000018
FKB$L_FPC                       = 0000000C
GAINQUORUM_MSG                    ********    X   02
IOC$GL_PSFC                       ********    X   02
IPL$_IOPOST                     = 00000004
IPL$_SCS                        = 00000008
IPL$_SYNCH                      = 00000008
IPL$_TIMER                      = 00000008
LCK$GL_DIRVEC                     ********    X   02
LOSEQUORUM_MSG                    ********    X   02
NODE_WIDTH                      = 0000000A
PCB$C_EOWNER                    = 00000068
PCB$L_EPID                      = 00000064
PCB$S_EPID_NODE_IDX             = 00000008
PCB$S_EPID_NODE_SEQ             = 00000002
PCB$V_EPID_NODE_IDX             = 00000015
PCB$V_EPID_NODE_SEQ             = 00C0001D
PR$_IPL                           ********    X   02
PR$_SIRR                          ********    X   02
SCH$GL_MAXPIX                     ********    X   02
SCH$GL_PCBVEC                     ********    X   02
SCH$GW_LOCALNODE                  ********    X   02
SS$_NORMAL                      = 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
$$$100	00000437 (1079.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.05	00:00:01.78
Command processing	113	00:00:00.42	00:00:02.79
Pass 1	512	00:00:15.09	00:01:05.22
Symbol table sort	0	00:00:01.91	00:00:07.88
Pass 2	226	00:00:03.28	00:00:12.67
Symbol table output	18	00:00:00.12	00:00:00.39
Psect synopsis output	1	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	901	00:00:20.89	00:01:30.84

```
The working set limit was 1800 pages.
118851 bytes (233 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1826 non-local and 51 local symbols.
1300 source lines were read in Pass 1, producing 19 object records in Pass 2.
32 pages of virtual memory were used to define 30 macros.
```

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

Macro library name	Macros defined
_255DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	17
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	24

```
1954 GETS were required to define 24 macros.

There were no errors, warnings or information messages.

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

0393 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CSPCALL.
LIS

CSPCJFMAS.
LIS

CSP
LIS

CSPBRKTHR.
LIS

CONUTIL.
LIS

CSPCALLACT.
LIS

CONSUBS.
LIS