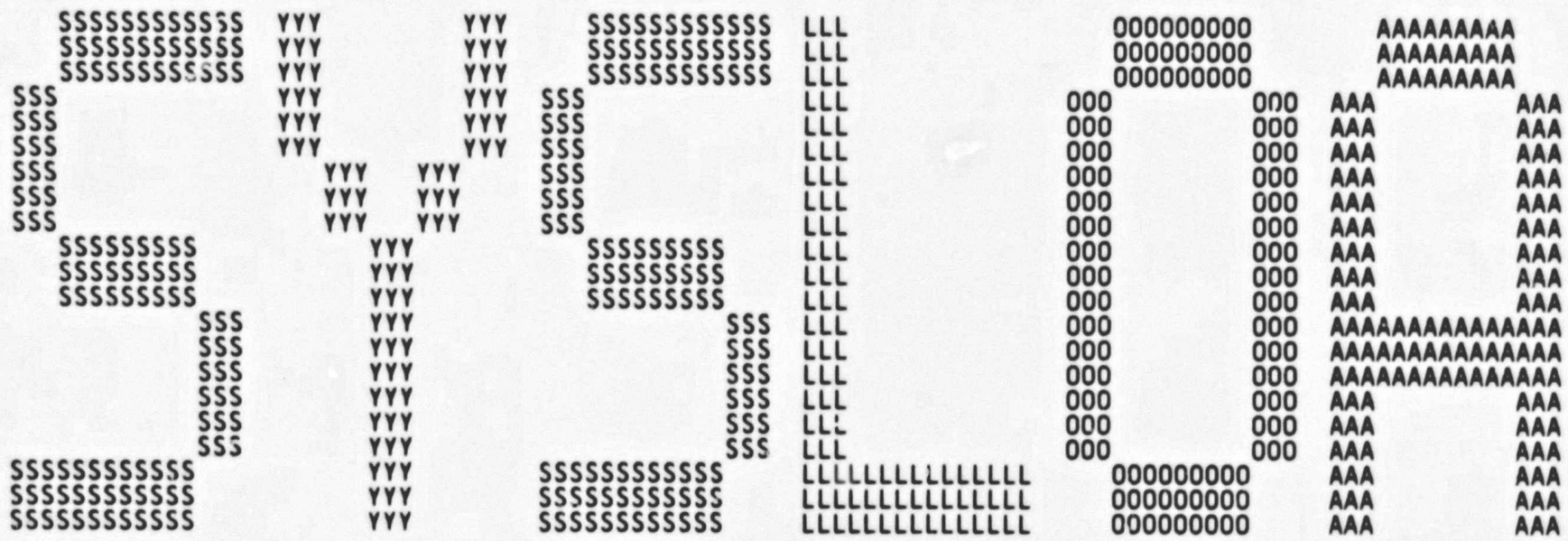
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL OOOOOOOOO AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL OOOOOOOOO AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL OOOOOOOOO AAAAAAAAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO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

```
  CCCCCCCC    SSSSSSSS  PPPPPPPP
  CCCCCCCC    SSSSSSSS  PPPPPPPP
CC          SS          PP      PP
CC          SS          PP      PP
CC          SS          PP      PP
CC          SS          PP      PP
CC            SSSSSS    PPPPPPPP
CC            SSSSSS    PPPPPPPP
CC                  SS  PP
CC                  SS  PP
CC                  SS  PP                ....
CC                  SS  PP                ....
  CCCCCCCC  SSSSSSSS    PP                ....
  CCCCCCCC  SSSSSSSS    PP                ....

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
```

```
;     1          0001  0 !
;     2          0002  0 ! - CSP$$WAIT does not return status!  What if LIB$GET_VM fails?
;     3          0003  0 !
;     4          0004  0
;     5          0005  0 MODULE  CSP
;001 !DYC0001    0006  0                 (IDENT = 'V04-001'
;    7-1         0007  0                 ,MAIN  = CLUSTER_SERVER
;     8          0008  0                 ,LANGUAGE (BLISS32)
;     9          0009  0                 ,ADDRESSING_MODE (EXTERNAL=GENERAL)
;    10          0010  0                 ) =
;    11          0011  1 BEGIN
;    12          0012  1 %TITLE  'Cluster Server Process - Main Routine'
;    13          0013  1 !
;    14          0014  1 !*****************************************************************************
;    15          0015  1 !*                                                                           *
;    16          0016  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    17          0017  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    18          0018  1 !*  ALL RIGHTS RESERVED.                                                     *
;    19          0019  1 !*                                                                           *
;    20          0020  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    21          0021  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    22          0022  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    23          0023  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    24          0024  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    25          0025  1 !*  TRANSFERRED.                                                             *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    28          0028  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    29          0029  1 !*  CORPORATION.                                                             *
;    30          0030  1 !*                                                                           *
;    31          0031  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    32          0032  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    33          0033  1 !*                                                                           *
;    34          0034  1 !*                                                                           *
;    35          0035  1 !*****************************************************************************
;    36          0036  1 !++
;    37          0037  1 !
;    38          0038  1 !  FACILITY:     Cluster Server Process
;    39          0039  1 !
;    40          0040  1 !  ABSTRACT:     Process context for coordinating the actions of cluster
;    41          0041  1 !                 servers and event logging in a VAXcluster.
;    42          0042  1 !
;    43          0043  1 !  AUTHOR:       Paul R. Beck
;    44          0044  1 !
;    45          0045  1 !  DATE:         03-MAR-1983     Last Edit:  24-AUG-1983 10:32
;    46          0046  1 !
;    47          0047  1 !  REVISION HISTORY:
;001 !DYC0002    0048  1 !        V04-002 DYC0002         Dennis Y. Chan          20-Dec-1984
;002 !DYC0002    0049  1 !                Changed last modification (V04-001) to create a thread to check
;003 !DYC0002    0050  1 !                software version and send OPCOM message to eliminate startup
;004 !DYC0002    0051  1 !                sequence dependency between CSP and OPCOM.
;005 !DYC0002    0052  1 !
;006 !DYC0002    0053  1 !        V04-001 DYC0001         Dennis Y. Chan          26-Nov-1984
;007 !DYC0002    0054  1 !                Added check for different versions of VMS in cluster and
;008 !DYC0002    0055  1 !                output informational message to opcom.
;    48          0056  1 !
;    49          0057  1 !        V03-010 ADE0003         Alan D. Eldridge        24-Apr-1984
```

```
;    50          0058  1 !                Cleaned up some error paths.  Use CSP$GL_CURCTX instead
;    51          0059  1 !                of CLX where appropriate.
;    52          0060  1 !
;    53          0061  1 !        V03-009 RSH0125         R. Scott Hanna          22-Mar-1984
;    54          0062  1 !                Remove call to CSP$QUORUM_INIT.
;    55          0063  1 !
;    56          0064  1 !        V03-008 ADE0002         Alan D. Eldridge        28-Feb-1983
;    57          0065  1 !                Change name of CSP$QUORUM to CSP$QUORUM_INIT.  Change name
;    58          0066  1 !                of CSP$$OPCOM to CSP$TELL_OPCOM.
;    59          0067  1 !
;    60          0068  1 !        V03-007 ADE0001         Alan D. Eldridge        1-Dec-1983
;    61          0069  1 !                Use the ACKMSG service of the connection manager instead
;    62          0070  1 !                of DECnet for inter-node communication.
;    63          0071  1 !
;    64          0072  1 !        V03-006 PRB0248         Paul Beck               8-Sep-1983
;    65          0073  1 !                Fix problem with the way CSP waits for DECnet availability.
;    66          0074  1 !
;    67          0075  1 !        V03-005 RSH0060         R. Scott Hanna          24-AUG-1983
;    68          0076  1 !                Add call to CSP$QUORUM during initialization.
;    69          0077  1 !
;    70          0078  1 !        V03-004 PRB0226         Paul Beck               19-AUG-1983 18:48
;    71          0079  1 !                Start up DECnet object asynchronously from the rest of the
;    72          0080  1 !                initialization, so the scheduler can be running before
;    73          0081  1 !                DECnet has started. Also, remove some excess baggage which
;    74          0082  1 !                would only be needed if DECnet were the permanent mechanism
;    75          0083  1 !                instead of just a holding action.
;    76          0084  1 !
;    77          0085  1 !        V03-003 PRB0226         Paul Beck               9-JUL-1983 16:39
;    78          0086  1 !                Get requests from CLUB$L_CSPFL, and allow for nonpaged pool
;    79          0087  1 !                (system addresses) therein. Get CSPDEF from LIB$.
;    80          0088  1 !
;    81          0089  1 !        V03-002 PRB0214         Paul Beck               24-JUN-1983 14:34
;    82          0090  1 !                Change SRC$:CSPDEF to SHRLIB$:CSPDEF
;    83          0091  1 !
;    84          0092  1 !        V03-001 PRB0200         Paul Beck               6-JUN-1983 21:05
;    85          0093  1 !                Change CTX$ symbols to CLX$ to prevent conflict with RCP.
;    86          0094  1 !--
```

```
;    88          0095  1 !
;    89          0096  1 ! External References:
;    90          0097  1 !
;    91          0098  1
;    92          0099  1
;    93          0100  1 REQUIRE
;    94          0101  1     'LIB$:CSPDEF.R32' ;                    ! CSP common definitions
;    95          0295  1 LIBRARY
;    96          0296  1     'SYS$LIBRARY:LIB' ;
;    97          0297  1
;   101          0301  1 SWITCHES LIST (SOURCE) ;
;   102          0302  1
;   103          0303  1 MACRO
;   104          0304  1     !
;   105          0305  1     !  Define offsets for lock_status.
;   106          0306  1     !
;   107          0307  1     LKSB$W_STATUS       = 0,0,16,0 %,          ! completion status
;   108          0308  1     LKSB$W_RESERVED     = 2,0,16,0 %,          !
;   109          0309  1     LKSB$L_LKID         = 4,0,32,0 %,          ! lock identification
;   110          0310  1     LKSB$AB_VALBLK      = 8,0,0,0  %,          ! value block
;   111          0311  1     LKSB$L_VALBLK_KEY   = 8,0,32,0 % ;         ! storage cell for random key
;   112          0312  1
;   113          0313  1
;   114          0314  1 !
;   115          0315  1 !  Linkages used
;   116          0316  1 !
;   117          0317  1 LINKAGE
;   118          0318  1     NO_REGISTERS        = CALL: NOPRESERVE (0,1,2,3,4,5,6,7,8,9,10,11),
;   119          0319  1     JSB_2               = JSB (REGISTER=2),
;   120          0320  1     JSB_12              = JSB (REGISTER=1,REGISTER=2),
;   121          0321  1     JSB_LINKAGE         = JSB,
;   122          0322  1     LINKDEANONPAGED     = JSB (REGISTER=0): NOPRESERVE(0,1,2,3) ;
;   123          0323  1
;   124          0324  1 BUILTIN
;   125          0325  1     INSQUE,
;   126          0326  1     REMQUE,
;   127          0327  1     CALLG,
;   128          0328  1     TESTBITCC,
;   129          0329  1     TESTBITCS,
;   130          0330  1     TESTBITSC,
;   131          0331  1     TESTBITSS ;
;   132          0332  1
```

```
;  134          0333  1 !
;  135          0334  1 !  External References:
;  136          0335  1 !
;  137          0336  1 EXTERNAL ROUTINE
;  138          0337  1     LIB$FREE_VM,                        ! Free virtual memory
;  139          0338  1     LIB$GET_VM,                         ! Get virtual memory
;  140          0339  1     LIB$GET_EF,                         ! Get event flag
;  141          0340  1     EXE$DEANONPAGED     : LINKDEANONPAGED, ! deallocated nonpaged pool
;  142          0341  1     EXE$CSP_COMMAND     : JSB_12,       ! Issue commands to loadable CSP code
;  143          0342  1     CSP$$CALL_ACTION    : JSB_2,        ! Action Routine dispatcher
;  144          0343  1     CSP$$WAIT,                          ! Wait for completion AST
;  145          0344  1     CSP$$CREATE_CTX     : JSB_LINKAGE,  ! Create new context block
;  146          0345  1     CSP$$DELETE_CTX     : JSB_LINKAGE,  ! Delete current context block
;  147          0346  1     CSP$$SAVE_STACK     : JSB_LINKAGE,  ! Save current stack in context block
;001 !DYC0002  0347  1     CSP$$RESUME         : NOVALUE,      ! Internal completion AST completion
;002 !DYC0002  0348  1     CSP$$FORK;                          ! Create new thread
;  149-1        0349  1
;  150          0350  1 !
;  151          0351  1 ! Forward References:
;  152          0352  1 !
;  153          0353  1 FORWARD ROUTINE
;  154          0354  1     EXIT_HANDLER        : NOVALUE,      ! Exit handler
;  155          0355  1     KERNEL_INIT,                        ! kernel mode initialization
;  156          0356  1     KERNEL_CLEANUP      : NOVALUE,      ! kernel mode exit handler cleanup
;  157          0357  1     SCHEDULE            : NO_REGISTERS, ! Scheduler for multithreading
;  158          0358  1     REMQUE_CSD,                         ! Get local request from CLUB, if any
;  159          0359  1     DEANONPAGED,                        ! Kernel routine to call EXE$DEANONPAGED
;  160          0360  1     CSP$TELL_OPCOM,                     ! Report error to OPCOM
;  161          0361  1     RESUME_THREAD,                      ! Resume execution of thread
;  162          0362  1     NEW_REQUEST         : JSB_LINKAGE NOVALUE, ! Process new client request
;  163          0363  1     REPLY,                              ! Tell loadable Exec code to reply
;  164          0364  1     KERNEL_ENQW,                        ! $ENQW call from kernel mode
;  165          0365  1     CSP$$CRASH,                         ! Report bug
;001 !DYC0001  0366  1     CHECK_SWVERS,                       ! Check for different VMS versions in cluster
;  166          0367  1     MUMBLE ;
;  167          0368  1
```

```
;  169     0369  1 !
;  170     0370  1 ! Fixed storage
;  171     0371  1 !
;  172     0372  1 GLOBAL
;  173     0373  1     CSP$GL_BASE_FP: LONG,                   ! save base FP for scheduler
;  174     0374  1     CSP$GL_CSPQ   : LONG,                   ! addr of CLUB$GL_CSPFL queue
;  175     0375  1     CSP$GQ_RESUME : VECTOR [2,LONG]         ! queue of scheduled context blocks
;  176     0376  1                     INITIAL (CSP$GQ_RESUME
;  177     0377  1                             ,CSP$GQ_RESUME
;  178     0378  1                             ),
;  179     0379  1     CSP$GQ_WAIT   : VECTOR [2,LONG]         ! queue of suspended context blocks
;  180     0380  1                     INITIAL (CSP$GQ_WAIT
;  181     0381  1                             ,CSP$GQ_WAIT
;  182     0382  1                             ),
;  183     0383  1     CSP$GL_CURCTX : REF BLOCK [,BYTE] ; ! Current context block
;  184     0384  1
;  185     0385  1 OWN
;  186     0386  1     CSP$Q_CLX_CSD: VECTOR [2,LONG]          ! queue of free CLX/CSD blocks
;  187     0387  1                     INITIAL (CSP$Q_CLX_CSD
;  188     0388  1                             ,CSP$Q_CLX_CSD
;  189     0389  1                             ),
;  190     0390  1     EXIT_REASON   : LONG,                   ! reason returned to exit handler
;  191     0391  1     EXIT_HANDLER_BLOCK
;  192     0392  1                   : VECTOR [4,LONG]         ! define exit handler
;  193     0393  1                     INITIAL (0,EXIT_HANDLER
;  194     0394  1                             ,1,EXIT_REASON
;  195     0395  1                             ),
;  196     0396  1     LOCK_STATUS   : BLOCK  [24,BYTE],       ! lock status block plus value block
;  197     0397  1     LOCK_BUFFER   : VECTOR [31,BYTE],       ! text of lock resource name
;  198     0398  1     LOCK_NAME     : VECTOR [2,LONG]         ! working descriptor for lock_buffer
;  199     0399  1                     INITIAL (0,LOCK_BUFFER),
;  200     0400  1     LOCK_NAME_DESC: VECTOR [2,LONG]         ! initial descriptor for lock_buffer
;  201     0401  1                     INITIAL (31,LOCK_BUFFER),
;  202     0402  1     STARTUP_TIME  : VECTOR [2,LONG],        ! system time for value block
;  203     0403  1     BASE_IOSB     : VECTOR [2,LONG],        ! IOSB for CSP's QIOs
;  204     0404  1     BASE_EFN      : BYTE ;                  ! allocated event flag for use by CSP
;  205     0405  1
;  206     0406  1
;  207     0407  1 !
;  208     0408  1 ! Macro to issue call with arguments from kernel mode.
;  209     0409  1 !
;  210   M 0410  1 MACRO   KRNL_CALL (K_ROUTINE) =
;  211   M 0411  1     BEGIN
;  212   M 0412  1     EXTERNAL ROUTINE  SYS$CMKRNL    : ADDRESSING_MODE (ABSOLUTE) ;
;  213   M 0413  1     BUILTIN SP  ;
;  214   M 0414  1
;  215   M 0415  1     SYS$CMKRNL (K_ROUTINE , .SP , %LENGTH - 1
;  216   M 0416  1                 %IF %LENGTH GTR 1 %THEN , %REMAINING %FI)
;  217     0417  1     END% ;
;  218     0418  1
;  219     0419  1 MACRO   ELSEIF = ELSE IF % ;
;  220     0420  1
;  221     0421  1 ROUTINE FLUSH_LISTING : NOVALUE =       ! Force output to .LIS file during
;  222     0422  1 RETURN ;                                ! compile

                                                .TITLE  CSP Cluster Server Process - Main Routine
```

```
                                        .IDENT  \V04-001\

                                        .PSECT  $OWN$,NOEXE,2

00000000' 00000000' 00000 CSP$Q_CLX_CSD:
                                        .ADDRESS CSP$Q_CLX_CSD, CSP$Q_CLX_CSD           ;
                           00008 EXIT_REASON:
                                        .BLKB   4
                  00000000  0000C EXIT_HANDLER_BLOCK:
                                        .LONG   0
                  00000000V 00010       .ADDRESS EXIT_HANDLER                           ;
                  00000001  00014       .LONG   1                                       ;
                  00000000' 00018       .ADDRESS EXIT_REASON                            ;
                            0001C LOCK_STATUS:
                                        .BLKB   24
                            00034 LOCK_BUFFER:
                                        .BLKB   31
                            00053       .BLKB   1
                  00000000  00054 LOCK_NAME:
                                        .LONG   0
                  00000000' 00058       .ADDRESS LOCK_BUFFER                            ;
                  0000001F  0005C LOCK_NAME_DESC:
                                        .LONG   31
                  00000000' 00060       .ADDRESS LOCK_BUFFER                            ;
                            00064 STARTUP_TIME:
                                        .BLKB   8
                            0006C BASE_IOSB:
                                        .BLKB   8
                            00074 BASE_EFN:
                                        .BLKB   1

                                        .PSECT  $GLOBAL$,NOEXE,2

                           00000 CSP$GL_BASE_FP::
                                        .BLKB   4
                           00004 CSP$GL_CSPQ::
                                        .BLKB   4
00000000' 00000000' 00008 CSP$GQ_RESUME::
                                        .ADDRESS CSP$GQ_RESUME, CSP$GQ_RESUME           ;
00000000' 00000000' 00010 CSP$GQ_WAIT::
                                        .ADDRESS CSP$GQ_WAIT, CSP$GQ_WAIT               ;
                           00018 CSP$GL_CURCTX::
                                        .BLKB   4

                                        .EXTRN  LIB$FREE_VM, LIB$GET_VM
                                        .EXTRN  LIB$GET_EF, EXE$DEANONPAGED
                                        .EXTRN  EXE$CSP_COMMAND
                                        .EXTRN  CSP$$CALL_ACTION
                                        .EXTRN  CSP$$WAIT, CSP$$CREATE_CTX
                                        .EXTRN  CSP$$DELETE_CTX
                                        .EXTRN  CSP$$SAVE_STACK
                                        .EXTRN  CSP$$RESUME, CSP$$FORK

                                        .PSECT  $CODE$,NOWRT,2

                      0000 00000 FLUSH_LISTING:
                                        .WORD   Save nothing                            ; 0421
```

```
                                        04 00002        RET                                     ; 0422

; Routine Size:  3 bytes,    Routine Base:  $CODE$ + 0000

;  223          0423  1
```

```
;   225           0424  1 ROUTINE CLUSTER_SERVER: NOVALUE =
;   226           0425  2 BEGIN
;   227           0426  2
;   228           0427  2 LOCAL
;   229           0428  2     STATUS,                                   ! All-purpose completion status
;   230           0429  2     CLX     : REF BLOCK [,BYTE] ;             ! Ptr to CLX block
;   231           0430  2
;001  !DYC0002    0431  2 BUILTIN
;002  !DYC0002    0432  2     FP;
;   232           0433  2 !
;   233           0434  2 !  The maximum number of requests and the maximum CSD size for each
;   234           0435  2 !  request are each fixed values.  Therefore, allocate and queue one
;   235           0436  2 !  CSD per request.
;   236           0437  2 !
;   237           0438  2 INCR I FROM 1 TO CSP$K_MAX_FLWCTL
;   238           0439  2 DO
;   239           0440  2     IF (CLX = CSP$$CREATE_CTX ())  EQL  0
;   240           0441  2     THEN
;   241           0442  3         RETURN (SS$_INSFMEM)
;   242           0443  2     ELSE
;   243           0444  3         IF (STATUS = LIB$GET_VM (%REF (CSP$K_MAX_CSDLNG), CLX [CLX$A_PO_CSD]))
;   244           0445  2         THEN
;   245           0446  2             INSQUE (.CLX, .CSP$Q_CLX_CSD)
;   246           0447  2         ELSE
;   247           0448  2             RETURN (.STATUS) ;
;   248           0449  2
;   249           0450  2 !
;   250           0451  2 !  Perform kernel-mode initialization as needed.
;   251           0452  2 !
;   252           0453  3 IF  NOT KRNL_CALL (KERNEL_INIT)
;   253           0454  2 THEN
;001  !DYC0002    0455  2
;002  !DYC0002    0456  3     BEGIN
;003  !DYC0002    0457  3 !
;004  !DYC0002    0458  3 !   This message will never make it because OPCOM has not started yet.
;005  !DYC0002    0459  3 !   Leave it there for now.   DYC 12/20/84
;006  !DYC0002    0460  3 !
;007  !DYC0002    0461  3     CSP$TELL_OPCOM
;008  !DYC0002    0462  3         (%ASCID '%CSP-E-NOCLUSTER, Cluster Server Process exiting: no cluster') ;
;009  !DYC0002    0463  3
;010  !DYC0002    0464  3     $EXIT (CODE = SS$_ABORT) ;
;011  !DYC0002    0465  3
;   258-4         0466  2     END ;
;   259           0467  2
;   260           0468  2 !
;   261           0469  2 !  Grab an event flag
;   262           0470  2 !
;   263           0471  2 IF NOT (STATUS = LIB$GET_EF (BASE_EFN) ) THEN RETURN .STATUS ;
;   264           0472  2
;   265           0473  2 !
;   266           0474  2 !  Declare an exit handler to deal with emergencies. In particular, it should
;   267           0475  2 !  empty the queue CLUB$GL_CSPFL and restore the blocks to nonpaged pool.
;   268           0476  2 !
;   269           0477  3 IF NOT (STATUS = $DCLEXH (DESBLK = EXIT_HANDLER_BLOCK))
;   270           0478  2 THEN
;   271           0479  3     RETURN .STATUS ;
;   272           0480  2
```

```
;  273          0481  2 !
;  274          0482  2 !  Set up a condition handler to deal with problems. (Needed?)
;  275          0483  2 !
;  276          0484  2 !MUMBLE ;
;  277          0485  2
;  278          0486  2 !
;  279          0487  2 !  Request notification of cluster events.
;  280          0488  2 !
;  281          0489  2 !MUMBLE ;
;  282          0490  2
;  283          0491  2 !
:001  !DYC0002  0492  2 !  Set up a new thread to check for different versions of VMS in cluster
:002  !DYC0002  0493  2 !
:003  !DYC0002  0494  2 CSP$GL_BASE_FP = .FP;          ! Set address for stack save
:004  !DYC0002  0495  2
:005  !DYC0002  0496  2 IF CSP$$FORK() EQL 0 THEN      ! Parent
:006  !DYC0002  0497  2
:007  !DYC0002  0498  2 !
:008  !DYC0002  0499  2 !  Wait for a request. A request will arrive as an incoming connect
:009  !DYC0002  0500  2 !  request, which is validated, followed by a buffer of data. This
:010  !DYC0002  0501  2 !  data is passed along to the server associated with the connect
:011  !DYC0002  0502  2 !  request.
:012  !DYC0002  0503  2 !
:013  !DYC0002  0504  2     WHILE 1 DO SCHEDULE()
:014  !DYC0002  0505  2
:015  !DYC0002  0506  2 ELSE                          ! New thread
:016  !DYC0002  0507  2
:017  !DYC0002  0508  3     BEGIN
:018  !DYC0002  0509  3         IF NOT CHECK_SWVERS() THEN      ! Check software version
:019  !DYC0002  0510  3
:020  !DYC0002  0511  3 !       Send message to consoles through OPCOM and if it is not there yet
:021  !DYC0002  0512  3 !       try every 10 seconds until the message is sent or an error has occured.
:022  !DYC0002  0513  3 !
:023  !DYC0002  0514  4             WHILE (CSP$TELL_OPCOM(%ASCID '%CSP-I-DIFSWVER, different versions of VMS exist in cluster')
:024  !DYC0002  0515  3                     EQL OPC$_NOPERATOR) DO
:025  !DYC0002  0516  4                 BEGIN
:026  !DYC0002  0517  4                 LOCAL
:027  !DYC0002  0518  4                     INTERVAL : VECTOR[2,LONG];      ! Q-word for delta time
:028  !DYC0002  0519  4
:029  !DYC0002 P 0520  5                 IF (STATUS = $BINTIM(TIMBUF=%ASCID '0000 00:00:10.00',
:030  !DYC0002  0521  4                                      TIMADR=INTERVAL)) NEQ SS$_NORMAL THEN
:031  !DYC0002  0522  4                     EXITLOOP
:032  !DYC0002 P 0523  5                 ELSE IF (STATUS = $SETIMR(DAYTIM=INTERVAL,
:033  !DYC0002 P 0524  5                                           ASTADR=CSP$$RESUME,
:034  !DYC0002  0525  4                                           REQIDT=.CSP$GL_CURCTX)) NEQ SS$_NORMAL THEN
:035  !DYC0002  0526  4                     EXITLOOP
:036  !DYC0002  0527  4                 ELSE
:037  !DYC0002  0528  4                     CSP$$WAIT();    ! Wait 10 seconds
:038  !DYC0002  0529  3                 END;
:039  !DYC0002  0530  3             CSP$$DELETE_CTX();              ! Delete own clx
:040  !DYC0002  0531  3             FP = .CSP$GL_BASE_FP;           ! Return to SCHEDULE
:041  !DYC0002  0532  2     END;
;  290-6        0533  1 END ;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2
```

```
45 54 53 55 4C 43 4F 4E 2D 45 2D 50 53 43 25  00000 P.AAB:  .ASCII  \%CSP-E-NOCLUSTER, Cluster Server Process\
76 72 65 53 20 72 65 74 73 75 6C 43 20 2C 52  0000F
                73 73 65 63 6F 72 50 20 72 65  0001E
6C 63 20 6F 6E 20 3A 67 6E 69 74 69 78 65 20  00028          .ASCII  \ exiting: no cluster\
                                  72 65 74 73 75  00037
                                  010E003C  0003C P.AAA:  .LONG   17694780
                                  00000000'  00040        .ADDRESS P.AAB
52 45 56 57 53 46 49 44 2D 49 2D 50 53 43 25  00044 P.AAD:  .ASCII  \%CSP-I-DIFSWVER, different versions of V\
72 65 76 20 74 6E 65 72 65 66 66 69 64 20 2C  00053
                56 20 66 6F 20 73 6E 6F 69 73  00062
75 6C 63 20 6E 69 20 74 73 69 78 65 20 53 4D  0006C          .ASCII  \MS exist in cluster\<0>
                                  00 72 65 74 73  0007B
                                  010E003B  00080 P.AAC:  .LONG   17694779
                                  00000000'  00084        .ADDRESS P.AAD
30 2E 30 31 3A 30 30 3A 30 30 20 30 30 30 30  00088 P.AAF:  .ASCII  \0000 00:00:10.00\
                                  30  00097
                                  010E0010  00098 P.AAE:  .LONG   17694736
                                  00000000'  0009C        .ADDRESS P.AAF

                                                          .EXTRN  SYS$CMKRNL, SYS$EXIT
                                                          .EXTRN  SYS$DCLEXH, SYS$BINTIM
                                                          .EXTRN  SYS$SETIMR

                                                          .PSECT  $CODE$,NOWRT,2

                                      0FFC 00000 CLUSTER_SERVER:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 0424
                    5E                10  C2 00002         SUBL2   #16, SP
                    53                01  D0 00005         MOVL    #1, I                                       ; 0438
                      00000000G       00  16 00008 1$:      JSB     CSP$$CREATE_CTX                             ; 0440
                    52                50  D0 0000E         MOVL    R0, CLX
                                      01  12 00011         BNEQ    2$
                                          04 00013         RET
                          10        A2  9F 00014 2$:      PUSHAB  16(CLX)                                     ; 0444
                      08  AE      1000  8F 3C 00017         MOVZWL  #4096, 8(SP)
                          08    AE  9F 0001D               PUSHAB  8(SP)
                00000000G 00            02  FB 00020         CALLS   #2, LIB$GET_VM
                          6E            50  D0 00027         MOVL    R0, STATUS
                          4C            6E  E9 0002A         BLBC    STATUS, 4$
                    0000' DF            62  0E 0002D         INSQUE  (CLX), @CSP$Q_CLX_CSD                       ; 0446
              D2        53            08  F3 00032         AOBLEQ  #8, I, 1$                                   ; 0440
                                      7E  D4 00036         CLRL    -(SP)                                       ; 0453
                                      5E  DD 00038         PUSHL   SP
                          0000V       CF  9F 0003A         PUSHAB  KERNEL_INIT
                00000000G 9F            03  FB 0003E         CALLS   #3, @#SYS$CMKRNL
                          12            50  E8 00045         BLBS    R0, 3$
                          0000'       CF  9F 00048         PUSHAB  P.AAA                                       ; 0462
                    0000V CF            01  FB 0004C         CALLS   #1, CSP$TELL_OPCOM
                                      2C  DD 00051         PUSHL   #44                                         ; 0464
                00000000G 00            01  FB 00053         CALLS   #1, SYS$EXIT
                          0000'       CF  9F 0005A 3$:      PUSHAB  BASE_EFN                                    ; 0471
                00000000G 00            01  FB 0005E         CALLS   #1, LIB$GET_EF
                          6E            50  D0 00065         MOVL    R0, STATUS
                          0E            6E  E9 00068         BLBC    STATUS, 4$
                          0000'       CF  9F 0006B         PUSHAB  EXIT_HANDLER_BLOCK                          ; 0477
                00000000G 00            01  FB 0006F         CALLS   #1, SYS$DCLEXH
```

```
                 6E              50   D0 00076          MOVL    R0, STATUS                                      ;
                 79              6E   E9 00079 4$:      BLBC    STATUS, 9$                                      ;
          0000'  CF              5D   D0 0007C          MOVL    FP, CSP$GL_BASE_FP                              ; 0494
00000000G  00                    00   FB 00081          CALLS   #0, CSP$$FORK                                   ; 0496
                                 50   D5 00088          TSTL    R0
                                 07   12 0008A          BNEQ    6$
          0000V  CF              00   FB 0008C 5$:      CALLS   #0, SCHEDULE                                    ; 0504
                                 F9   11 00091          BRB     5$
          0000V  CF              00   FB 00093 6$:      CALLS   #0, CHECK_SWVERS                                ; 0509
                 4F              50   E8 00098          BLBS    R0, 8$
                          0000'  CF   9F 0009B 7$:      PUSHAB  P.AAC                                           ; 0514
          0000V  CF              01   FB 0009F          CALLS   #1, CSP$TELL_OPCOM
00058061   8F                    50   D1 000A4          CMPL    R0, #360545                                     ; 0515
                                 3D   12 000AB          BNEQ    8$
                            08   AE   9F 000AD          PUSHAB  INTERVAL                                        ; 0521
                          0000'  CF   9F 000B0          PUSHAB  P.AAE
00000000G  00                    02   FB 000B4          CALLS   #2, SYS$BINTIM
                 6E              50   D0 000BB          MOVL    R0, STATUS
                 01              6E   D1 000BE          CMPL    STATUS, #1
                                 27   12 000C1          BNEQ    8$
                          0000'  CF   DD 000C3          PUSHL   CSP$GL_CURCTX                                   ; 0525
                      00000000G  00   9F 000C7          PUSHAB  CSP$$RESUME
                             10  AE   9F 000CD          PUSHAB  INTERVAL
                                 7E   D4 000D0          CLRL    -(SP)
00000000G  00                    04   FB 000D2          CALLS   #4, SYS$SETIMR
                 6E              50   D0 000D9          MOVL    R0, STATUS
                 01              6E   D1 000DC          CMPL    STATUS, #1
                                 09   12 000DF          BNEQ    8$
00000000G  00                    00   FB 000E1          CALLS   #0, CSP$$WAIT                                   ; 0528
                                 B1   11 000E8          BRB     7$                                              ; 0514
                   00000000G     00   16 000EA 8$:      JSB     CSP$$DELETE_CTX                                 ; 0530
                 5D      0000'   CF   D0 000F0          MOVL    CSP$GL_BASE_FP, FP                              ; 0531
                                 04   000F5 9$:         RET                                                     ; 0533

; Routine Size:  246 bytes,    Routine Base:  $CODE$ + 0003

;  291          0534  1
```

```
;   293     0535  1 %SBTTL 'SCHEDULE - schedule new requests, resume suspended threads'
;   294     0536  1 ROUTINE SCHEDULE: NO_REGISTERS =
;   295     0537  1
;   296     0538  1 !++
;   297     0539  1 !
;   298     0540  1 !  SCHEDULE is the thread scheduler.  It hibernates when there is nothing to do.
;   299     0541  1 !  When wakened, it removes items from one of the following queues:
;   300     0542  1 !
;   301     0543  1 !       1. New Requests         - containing buffers which are  sent off to the
;   302     0544  1 !                                 appropriate servers.
;   303     0545  1 !       2. Thread Resumptions   - containing context blocks to be resumed.
;   304     0546  1 !
;   305     0547  1 !  It continues to service these queues until both are empty, then
;   306     0548  1 !  hibernates once more.
;   307     0549  1 !
;   308     0550  1 !  CALLING SEQUENCE:
;   309     0551  1 !       CALL
;   310     0552  1 !
;   311     0553  1 !  FORMAL PARAMETERS:
;   312     0554  1 !       None.
;   313     0555  1 !
;   314     0556  1 !  COMPLETION CODES:
;   315     0557  1 !       None. SCHEDULE runs ad nauseum.
;   316     0558  1 !--
;   317     0559  2 BEGIN
;   318     0560  2 LOCAL
;   319     0561  2     CSD         : REF BLOCK [,BYTE],    ! new local request
;   320     0562  2     STATUS ;
;   321     0563  2
;   322     0564  2 BUILTIN  FP ;
;   323     0565  2 !
;   324     0566  2 !  Save the frame pointer, enabling the scheduler to be reentered from
;   325     0567  2 !  the context save routines.
;   326     0568  2 !
;   327     0569  2 CSP$GL_BASE_FP = .FP ;
;   328     0570  2
;   329     0571  2 !
;   330     0572  2 !  Try for something to do, and hibernate if there's nothing.
;   331     0573  2 !  If a thread is active (CSP$GL_CURCTX non-zero) then there's a bug.
;   332     0574  2 !
;   333     0575  2 WHILE .CSP$GL_CURCTX EQL 0
;   334     0576  2 DO
;   335     0577  2     IF  NOT REMQUE (.CSP$GQ_RESUME, CSP$GL_CURCTX)  THEN
;   336     0578  2         !
;   337     0579  2         !  Resume a suspended thread. Context block has been placed in the
;   338     0580  2         !  grant queue by an AST.
;   339     0581  2         !
;   340     0582  2         !       **   Note that RESUME_THREAD does not return in-line   **
;   341     0583  2         !       **   the scheduler will be reentered from the top      **
;   342     0584  2         !
;   343     0585  2         IF TESTBITCC (CSP$GL_CURCTX [CLX$V_QUEUED])
;   344     0586  2         THEN CSP$$CRASH (SS$_NOPRIVSTRT + T6)
;   345     0587  2         ELSE RESUME_THREAD ()
;   346     0588  2     ELSE
;   347     0589  2         !
;   348     0590  2         !  If we have a free process space CSD then service a new request
;   349     0591  2         !
```

```
;  350          0592  2          IF  KRNL_CALL (REMQUE_CSD) THEN
;  351          0593  2              NEW_REQUEST ()
;  352          0594  2          ELSE
;  353          0595  2              $HIBER ;
;  354          0596  2
;  355          0597  2  CSP$$CRASH (SS$_BADPARAM);
;  356          0598  2  RETURN 0 ;
;  357          0599  1  END ;


                                                        .EXTRN  SYS$HIBER

                                              0000 00000 SCHEDULE:
                                                        .WORD   Save nothing                                    ; 0536
                    0000'  CF                      5D  D0 00002        MOVL    FP, CSP$GL_BASE_FP              ; 0569
                                          0000'  CF 05  D5 00007 1$:    TSTL    CSP$GL_CURCTX                   ; 0575
                                                 46 12  0000B         BNEQ    5$                              ;
                    0000'  CF             0000'  DF  0F  0000D         REMQUE  @CSP$GQ_RESUME, CSP$GL_CURCTX   ; 0577
                                                 1D 1D  00014         BVS     3$                              ;
                           50             0000'  CF  D0  00016         MOVL    CSP$GL_CURCTX, R0               ; 0585
                 0C          0B          A0          00  E4  0001B         BBSC    #0, 11(R0), 2$                  ;
                           7E             2810        8F  3C  00020         MOVZWL  #10256, -(SP)                   ; 0586
                    0000V  CF                      01  FB  00025         CALLS   #1, CSP$$CRASH                  ;
                                                   DB  11  0002A         BRB     1$                              ;
                    0000V  CF                      00  FB  0002C 2$:     CALLS   #0, RESUME_THREAD               ; 0587
                                                   D4  11  00031         BRB     1$                              ; 0585
                                                   7E  D4  00033 3$:     CLRL    -(SP)                           ; 0592
                                                   5E  DD  00035         PUSHL   SP                              ;
                                          0000V    CF  9F  00037         PUSHAB  REMQUE_CSD                      ;
                                00000000G  9F          03  FB  0003B         CALLS   #3, @#SYS$CMKRNL                ;
                                           05          50  E9  00042         BLBC    R0, 4$                          ;
                                                  0000V 30  00045         BSBW    NEW_REQUEST                     ; 0593
                                                   BD  11  00048         BRB     1$                              ;
                                00000000G  00          00  FB  0004A 4$:     CALLS   #0, SYS$HIBER                   ; 0594
                                                   B4  11  00051         BRB     1$                              ; 0577
                                                   14  DD  00053 5$:     PUSHL   #20                             ; 0597
                                     0000V  CF         01  FB  00055         CALLS   #1, CSP$$CRASH                  ;
                                                   50  D4  0005A         CLRL    R0                              ; 0598
                                                       04  0005C         RET                                     ; 0599

; Routine Size:  93 bytes,    Routine Base:  $CODE$ + 00F9

;  358          0600  1
```

```
;  360          0601  1 %SBTTL  'NEW_REQUEST - process a new request'
;  361          0602  1 ROUTINE  NEW_REQUEST : JSB_LINKAGE NOVALUE =
;  362          0603  1 !++
;  363          0604  1 !
;  364          0605  1 !  Dispatch a new execution thread .
;  365          0606  1 !
;  366          0607  1 !
;  367          0608  1 !  CALLING SEQUENCE:
;  368          0609  1 !       JSB
;  369          0610  1 !
;  370          0611  1 !  FORMAL PARAMETERS:
;  371          0612  1 !       None
;  372          0613  1 !
;  373          0614  1 !--
;  374          0615  2 BEGIN
;  375          0616  2 LOCAL
;  376          0617  2     CSD         : REF BLOCK [,BYTE],          ! Context block
;  377          0618  2     STATUS ;
;  378          0619  2
;  379          0620  2 CSD = .CSP$GL_CURCTX [CLX$A_PO_CSD] ;         ! Get the PO space CSD
;  380          0621  2
;  381          0622  2 !
;  382          0623  2 !  Relocate pointers in CSD, dispatch to action routine for this client,
;  383          0624  2 !  respond to EXE$CSP_COMMAND when done.
;  384          0625  2 !
;  385          0626  2 CSD [CSD$L_SENDOFF] = .CSD [CSD$L_SENDOFF] + .CSD ;
;  386          0627  2 CSD [CSD$L_RECVOFF] = .CSD [CSD$L_RECVOFF] + .CSD ;
;  387          0628  2
;  388          0629  2 STATUS = CSP$$CALL_ACTION (.CSD) ;
;  389          0630  2
;  390          0631  2 KRNL_CALL (REPLY) ;
;  391          0632  2
;  392          0633  2 INSQUE (.CSP$GL_CURCTX, CSP$Q_CLX_CSD) ;
;  393          0634  2 CSP$GL_CURCTX = 0 ;
;  394          0635  2
;  395          0636  2 RETURN
;  396          0637  1 END ;

                                    52  DD 00000 NEW_REQUEST:
                                                 PUSHL   R2                                  ; 0602
                          50    0000'   CF  D0 00002       MOVL    CSP$GL_CURCTX, R0          ; 0620
                          52      10    A0  D0 00007       MOVL    16(R0), CSD                ;
                    16    A2            52  C0 0000B       ADDL2   CSD, 22(CSD)               ; 0626
                    1E    A2            52  C0 0000F       ADDL2   CSD, 30(CSD)               ; 0627
                              00000000G 00  16 00013       JSB     CSP$$CALL_ACTION           ; 0629
                                        7E  D4 00019       CLRL    -(SP)                      ; 0631
                                        5E  DD 0001B       PUSHL   SP                         ;
                                  0000V CF  9F 0001D       PUSHAB  REPLY                      ;
                00000000G         9F        03  FB 00021   CALLS   #3, @#SYS$CMKRNL           ;
                    0000'         CF  0000' DF  0E 00028   INSQUE  @CSP$GL_CURCTX, CSP$Q_CLX_CSD ; 0633
                                      0000' CF  D4 0002F   CLRL    CSP$GL_CURCTX              ; 0634
                                        04  BA 00033       POPR    #^M<R2>                    ; 0637
                                        05  00035       RSB                                 ;
```

```
; Routine Size:  54 bytes,    Routine Base:  $CODE$ + 0156

;  397          0638  1
```

```
;   399     0639  1 %SBTTL  'REPLY  - Call loadable Exec code to finish block transfer'
;   400     0640  1 ROUTINE  REPLY =
;   401     0641  1 !++
;   402     0642  1 !
;   403     0643  1 !  Client is done.  Copy the response (if any) and give then SO space CSD back
;   404     0644  1 !  to the loadable Exec code.
;   405     0645  1 !
;   406     0646  1 !
;   407     0647  1 ! CALLING SEQUENCE:
;   408     0648  1 !       CALL in kernel mode at IPL 0
;   409     0649  1 !
;   410     0650  1 ! FORMAL PARAMETERS:
;   411     0651  1 !       None
;   412     0652  1 !
;   413     0653  1 !--
;   414     0654  2 BEGIN
;   415     0655  2 BIND    SO_CSD = .CSP$GL_CURCTX [CLX$A_SO_CSD] : BLOCK [,BYTE],
;   416     0656  2         PO_CSD = .CSP$GL_CURCTX [CLX$A_PO_CSD] : BLOCK [,BYTE] ;
;   417     0657  2
;   418     0658  2 LOCAL   SIZE, COMMAND ;
;   419     0659  2
;   420     0660  2 SIZE = .PO_CSD [CSD$L_RECVLEN] ;
;   421     0661  2
;   422     0662  2 !
;   423     0663  2 !
;   424     0664  2 !  SO_CSD [CSD$L_RECVOFF] contains a real offset, but PO_CSD [CSD$L_RECVOFF]
;   425     0665  2 !  has been converted to a pointer.
;   426     0666  2 !
;   427     0667  2 !  Determine the proper response code and move whatever data needs moving.
;   428     0668  2 !
;   429     0669  2 !
;   430     0670  2 IF  (.SIZE + .SO_CSD [CSD$L_RECVOFF])  GTRU  .SO_CSD [CSD$W_SIZE]  THEN
;   431     0671  3     BEGIN
;   432     0672  3     SIZE    = .SO_CSD [CSD$W_SIZE] - .SO_CSD [CSD$L_RECVOFF] ;
;   433     0673  3     COMMAND = CSP$_BADCSD ;
;   434     0674  3     END
;   435     0675  3
;   436     0676  2 ELSEIF .PO_CSD [CSD$L_RECVLEN] NEQ 0  THEN
;   437     0677  2     COMMAND = CSP$_REPLY
;   438     0678  2 ELSE
;   439     0679  2     COMMAND = CSP$_DONE ;
;   440     0680  2
;   441     0681  2 CH$MOVE (.SIZE
;   442     0682  2         ,.PO_CSD [CSD$L_RECVOFF]                        ! Already a pointer
;   443     0683  2         ,.SO_CSD [CSD$L_RECVOFF] + .SO_CSD              ! Calculate pointer
;   444     0684  2         ) ;
;   445     0685  2 !
;   446     0686  2 !
;   447     0687  2 !  Pass the CSD back to the loadable Exec CSD code and erase the CSD pointer
;   448     0688  2 !
;   449     0689  2 !
;   450     0690  2 EXE$CSP_COMMAND (.COMMAND, .CSP$GL_CURCTX [CLX$A_SO_CSD]) ;
;   451     0691  2
;   452     0692  2 CSP$GL_CURCTX [CLX$A_SO_CSD] = 0 ;
;   453     0693  2
;   454     0694  2 RETURN 1 ;
;   455     0695  1 END ;
```

```
                                                 OFFC 00000 REPLY:   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 0640
                                  56      0000'  CF   D0 00002       MOVL    CSP$GL_CURCTX, R6                           ; 0655
                                  51        0C   A6   7D 00007       MOVQ    12(R6), R1                                  ;
                                  50        1A   A2   D0 0000B       MOVL    26(R2), SIZE                                ; 0660
                            53    50        1E   A1   C1 0000F       ADDL3   30(R1), SIZE, R3                            ; 0670
                  53        08    A1        10        00   ED 00014  CMPZV   #0, #16, 8(R1), R3                         ;
                                                      0D   1E 0001A  BGEQU   1$                                          ;
                                  50        08   A1   3C 0001C       MOVZWL  8(R1), SIZE                                 ; 0672
                                  50        1E   A1   C2 00020       SUBL2   30(R1), SIZE                                ;
                                  57             03   D0 00024       MOVL    #3, COMMAND                                 ; 0673
                                                 0D   11 00027       BRB     3$                                          ; 0670
                                            1A   A2   D5 00029 1$:    TSTL    26(R2)                                      ; 0676
                                                 05   13 0002C       BEQL    2$                                          ;
                                  57             05   D0 0002E       MOVL    #5, COMMAND                                 ; 0677
                                                 03   11 00031       BRB     3$                                          ;
                                  57             04   D0 00033 2$:    MOVL    #4, COMMAND                                 ; 0679
                            51    1E   A1        61   C1 00036 3$:    ADDL3   (R1), 30(R1), R1                            ; 0683
                            61    1E   B2        50   28 0003B       MOVC3   SIZE, @30(R2), (R1)                         ;
                                  52        0C   A6   D0 00040       MOVL    12(R6), R2                                  ; 0690
                                  51             57   D0 00044       MOVL    COMMAND, R1                                 ;
                                       00000000G 00   16 00047       JSB     EXE$CSP_COMMAND                             ;
                                  50      0000'  CF   D0 0004D       MOVL    CSP$GL_CURCTX, R0                           ; 0692
                                            0C   A0   D4 00052       CLRL    12(R0)                                      ;
                                  50             01   D0 00055       MOVL    #1, R0                                      ; 0694
                                                      04 00058       RET                                                 ; 0695

; Routine Size:  89 bytes,    Routine Base:  $CODE$ + 018C

;  456          0696  1
```

```
;  458          0697  1 %SBTTL  'RESUME_THREAD - resume execution of suspended thread'
;  459          0698  1 ROUTINE  RESUME_THREAD =
;  460          0699  1
;  461          0700  1 !++
;  462          0701  1 !  Restore the context of a thread and resume its execution.  The context
;  463          0702  1 !  consists of the stack and registers.  The top of the saved stack contains a
;  464          0703  1 !  call frame which points to the resume PC and contains the thread's registers.
;  465          0704  1 !  So by restoring the stack and "return"ing, we resume the thread.
;  466          0705  1 !
;  467          0706  1 !  CALLING SEQUENCE:
;  468          0707  1 !       CALL (from SCHEDULE only!)
;  469          0708  1 !
;  470          0709  1 !  FORMAL PARAMETERS:
;  471          0710  1 !       None
;  472          0711  1 !
;  473          0712  1 !  COMPLETION CODES:
;  474          0713  1 !       NA, R0 is restored from its saved state.
;  475          0714  1 !
;  476          0715  1 !--
;  477          0716  2 BEGIN
;  478          0717  2
;  479          0718  2 REGISTER
;  480          0719  2     SAVE_R0,SAVE_R1,            ! temp save registers R0,R1
;  481          0720  2     STATUS ;                    ! completion status
;  482          0721  2
;  483          0722  2 BUILTIN
;  484          0723  2     R0,R1,FP,SP ;
;  485          0724  2
;  486          0725  2 !
;  487          0726  2 !  Thread resumption consists of simply restoring the exact stack as it existed
;  488          0727  2 !  for the suspended thread, by copying it from the context block. Then kill the
;  489          0728  2 !  context block and restore R0,R1. The other registers are restored by the
;  490          0729  2 !  RETURN, which is logically a RETURN from CSP$$WAIT.
;  491          0730  2 !
;  492          0731  2 !
;  493          0732  2
;  494          0733  2 SAVE_R0 = .CSP$GL_CURCTX [CLX$L_R0] ;
;  495          0734  2 SAVE_R1 = .CSP$GL_CURCTX [CLX$L_R1] ;
;  496          0735  2
;  497          0736  2 SP = .CSP$GL_BASE_FP - .CSP$GL_CURCTX [CLX$L_STACKSIZE] ;
;  498          0737  2 FP = .SP ;
;  499          0738  2
;  500          0739  2 CH$MOVE (.CSP$GL_CURCTX [CLX$L_STACKSIZE], .CSP$GL_CURCTX [CLX$A_STACK], .FP) ;
;  501          0740  2
;  502          0741  2 !
;  503          0742  2 !  Deallocate the saved stack, then restore the saved context by
;  504          0743  2 !  simply returning to it.
;  505          0744  2 !
;  506          0745  2 IF  TESTBITCC (CSP$GL_CURCTX [CLX$V_LOCAL_STACK]) THEN
;  507          0746  2     LIB$FREE_VM (CSP$GL_CURCTX [CLX$L_STACKSIZE], CSP$GL_CURCTX [CLX$A_STACK]) ;
;  508          0747  2
;  509          0748  2 R0 = .SAVE_R0 ;
;  510          0749  2 R1 = .SAVE_R1 ;
;  511          0750  2
;  512          0751  2 RETURN (.R0) ;                  ! This restores reg's and resumes thread
;  513          0752  1 END ;
```

```
                                                01FC 00000 RESUME_THREAD:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8                ; 0698
                                58      0000'   CF  D0 00002       MOVL    CSP$GL_CURCTX, R8                ; 0733
                                56        28    A8  7D 00007       MOVQ    40(R8), SAVE_R0                  ;
                      5E  0000' CF        38    A8  C3 0000B       SUBL3   56(R8), CSP$GL_BASE_FP, SP       ; 0736
                                5D              5E  D0 00012       MOVL    SP, FP                           ; 0737
                      6D    3C  B8        38    A8  28 00015       MOVC3   56(R8), @60(R8), (FP)            ; 0739
                      0D    0B  A8              03  E4 0001B       BBSC    #3, 11(R8), 1$                   ; 0745
                                          3C    A8  9F 00020       PUSHAB  60(R8)                           ; 0746
                                          38    A8  9F 00023       PUSHAB  56(R8)                           ;
                          00000000G   00        02  FB 00026       CALLS   #2, LIB$FREE_VM                  ;
                                      50              56  7D 0002D 1$:     MOVQ    SAVE_R0, R0              ; 0748
                                                      04  00030       RET                                  ; 0752

; Routine Size:  49 bytes,    Routine Base:  $CODE$ + 01E5

;  514          0753  1
```

```
;  516          0754  1
;  517          0755  1 %SBTTL  'KERNEL_ENQW - $ENQW call from kernel mode'
;  518          0756  1
;  519          0757  1 GLOBAL ROUTINE KERNEL_ENQW =
;  520          0758  1
;  521          0759  1 !++
;  522          0760  1 !
;  523          0761  1 !  Issues $ENQW call from kernel mode.
;  524          0762  1 !
;  525          0763  1 !  CALLING SEQUENCE:
;  526          0764  1 !       CALL
;  527          0765  1 !
;  528          0766  1 !  FORMAL PARAMETERS:
;  529          0767  1 !       Call parameters identical to those for $ENQW;
;  530          0768  1 !       the argument list is simply passed on.
;  531          0769  1 !
;  532          0770  1 !  COMPLETION CODES:
;  533          0771  1 !       As from $ENQW, plus SS$_NOPRIV if process lacks CMKRNL.
;  534          0772  1 !
;  535          0773  1 !--
;  536          0774  2 BEGIN
;  537          0775  2 EXTERNAL ROUTINE  SYS$ENQW ;
;  538          0776  2 BUILTIN           AP ;
;  539          0777  2
;  540          0778  2 !RETURN $CMKRNL ( ROUTIN = SYS$ENQW, ARGLST = .AP ) ;
;  541          0779  2 RETURN CALLG (.AP, SYS$ENQW) ;
;  542          0780  1 END ;

                                                        .EXTRN  SYS$ENQW

                                              0000 00000        .ENTRY  KERNEL_ENQW, Save nothing                ; 0757
                00000000G  00           6C    FA 00002        CALLG   (AP), SYS$ENQW                           ; 0779
                                              04 00009        RET                                              ; 0780

; Routine Size:  10 bytes,    Routine Base:  $CODE$ + 0216

;  543          0781  1
```

```
;  545          0782  1 %SBTTL  'CSP$TELL_OPCOM - operator communications'
;  546          0783  1
;  547          0784  1 GLOBAL ROUTINE CSP$TELL_OPCOM
;  548          0785  1                 (MESSAGE : REF VECTOR [2,LONG]) =
;  549          0786  1 !++
;  550          0787  1 !
;  551          0788  1 !  Send indicated message to operator.
;  552          0789  1 !
;  553          0790  1 !  CALLING SEQUENCE:
;  554          0791  1 !       CALL
;  555          0792  1 !
;  556          0793  1 !  FORMAL PARAMETERS:
;  557          0794  1 !       P1 =    Address of descriptor of message to send.
;  558          0795  1 !
;  559          0796  1 !  COMPLETION CODES:
;  560          0797  1 !       N.A.
;  561          0798  1 !--
;  562          0799  2 BEGIN
;  563          0800  2 OWN
;  564          0801  2     OP_BUF      : BLOCK [128,BYTE]                  ! buffer for output
;  565          0802  2                   PRESET ( [OPC$B_MS_TYPE]    = OPC$_RQ_RQST,
;  566          0803  2                            [OPC$B_MS_TARGET]  = OPC$M_NM_CENTRL,
;  567          0804  2                            [OPC$L_MS_RQSTID]  = 0 ),
;  568          0805  2     OP_DESC     : VECTOR [2,LONG]                   ! descriptor of op_buf
;  569          0806  2                   INITIAL (128,OP_BUF) ;
;  570          0807  2
;  571          0808  2 !
;  572          0809  2 !  Copy the message into the message buffer.
;  573          0810  2 !
;  574          0811  2 CH$MOVE (.MESSAGE [0], .MESSAGE [1], OP_BUF [OPC$L_MS_TEXT]) ;
;  575          0812  2
;  576          0813  2 !
;  577          0814  2 !  Adjust the descriptor according to the size of the message
;  578          0815  2 !
;  579          0816  2 OP_DESC [0] = .MESSAGE [0] + (OP_BUF [OPC$L_MS_TEXT] - OP_BUF) ;
;  580          0817  2
;  581          0818  2 !
;  582          0819  2 !  Send the message to the operator. Status of $SNDOPR is returned to caller.
;  583          0820  2 !
;  584          0821  2 RETURN $SNDOPR ( MSGBUF = OP_DESC ) ;
;  585          0822  1 END ;


                                                .PSECT  $OWN$,NOEXE,2

                                        00075           .BLKB   3
                                  01    03  00078 OP_BUF: .BYTE   3,1                                                   ;
                                        00#   0007A       .BYTE   0[2]                                                    ;
                                  00000000    0007C       .LONG   0                                                       ;
                                        00080           .BLKB   120
                                  00000080    00F8 OP_DESC:.LONG  128                                                     ;
                                  00000000'   00FC        .ADDRESS OP_BUF                                                 ;

                                                .EXTRN  SYS$SNDOPR

                                                .PSECT  $CODE$,NOWRT,2
```

```
                                                 007C 00000          .ENTRY  CSP$TELL_OPCOM, Save R2,R3,R4,R5,R6          ; 0784
                                          56     04   AC   D0 00002  MOVL    MESSAGE, R6                                  ; 0811
                0000'  CF        04       B6        66   28 00006  MOVC3   (R6), @4(R6), OP_BUF+8                       ;
                0000'  CF                 66        08   C1 0000D  ADDL3   #8, (R6), OP_DESC                            ; 0816
                                                    7E   D4 00013  CLRL    -(SP)                                        ; 0821
                                                0000'  CF   9F 00015  PUSHAB  OP_DESC                                      ;
                          00000000G      00           02   FB 00019  CALLS   #2, SYS$SNDOPR                              ;
                                                                04 00020  RET                                              ; 0822

; Routine Size:  33 bytes,    Routine Base:  $CODE$ + 0220

;  586          0823  1
```

```
;  588          0824  1 %SBTTL  'EXIT_HANDLER'
;  589          0825  1 ROUTINE  EXIT_HANDLER : NOVALUE =
;  590          0826  1
;  591          0827  1 !++
;  592          0828  1 !
;  593          0829  1 !  Image exit handler; return excess items in CLUB$GL_CSPFL to nonpaged pool.
;  594          0830  1 !  Also report the problem.
;  595          0831  1 !
;  596          0832  1 !  CALLING SEQUENCE:
;  597          0833  1 !
;  598          0834  1 !  FORMAL PARAMETERS:
;  599          0835  1 !
;  600          0836  1 !  COMPLETION CODES:
;  601          0837  1 !
;  602          0838  1 !--
;  603          0839  2 BEGIN
;  604          0840  2 LOCAL
;  605          0841  2     CSD ;
;  606          0842  2
;  607          0843  2 CSP$TELL_OPCOM ( %ASCID 'CSP-E-EXIT, Cluster Server Process exiting' ) ;
;  608          0844  2 KRNL_CALL ( KERNEL_CLEANUP ) ;
;  609          0845  2
;  610          0846  2 RETURN
;  611          0847  1 END ;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2
75 6C 43 20 2C 54 49 58 45 2D 45 2D 50 53 43  000A0 P.AAH:  .ASCII  \CSP-E-EXIT, Cluster Server Process exiti\  ;  ;
6F 72 50 20 72 65 76 72 65 53 20 72 65 74 73  000AF                                                          ;  ;
                69 74 69 78 65 20 73 73 65 63  000BE                                                          ;  ;
                                   00 00 67 6E  000C8          .ASCII  \ng\<0><0>                             ;  ;
                                          010E002A  000CC P.AAG:  .LONG   17694762                           ;  ;
                                          00000000'  000D0        .ADDRESS P.AAH                             ;  ;


                                                        .PSECT  $CODE$,NOWRT,2

                                                  0000 00000 EXIT_HANDLER:
                                                             .WORD   Save nothing                            ;  0825
                                        0000'  CF  9F 00002          PUSHAB  P.AAG                          ;  0843
                                     D5 AF        01  FB 00006          CALLS   #1, CSP$TELL_OPCOM             ;
                                                  7E  D4 0000A          CLRL    -(SP)                          ;  0847
                                                  5E  DD 0000C          PUSHL   SP                             ;
                                        0000V  CF  9F 0000E          PUSHAB  KERNEL_CLEANUP                 ;
                                00000000G  9F        03  FB 00012          CALLS   #3, @#SYS$CMKRNL           ;
                                                     04  00019          RET                                    ;  0847

; Routine Size:  26 bytes,    Routine Base:  $CODE$ + 0241

;  612          0848  1
```

```
;  614          0849  1 %SBTTL  'KERNEL_INIT'
;  615          0850  1 ROUTINE  KERNEL_INIT =
;  616          0851  1
;  617          0852  1 !++
;  618          0853  1 !
;  619          0854  1 !  Perform kernel mode initialization:
;  620          0855  1 !
;  621          0856  1 !       Initialize queue in CLUB structure, fill in CLUB$GL_CSPIPID.
;  622          0857  1 !
;  623          0858  1 !  CALLING SEQUENCE:
;  624          0859  1 !       $CMKRNL ( KERNEL_INIT )
;  625          0860  1 !
;  626          0861  1 !  FORMAL PARAMETERS:
;  627          0862  1 !       none
;  628          0863  1 !
;  629          0864  1 !  COMPLETION CODES:
;  630          0865  1 !
;  631          0866  1 !--
;  632          0867  2 BEGIN
;  633          0868  2 EXTERNAL
;  634          0869  2     CLU$GL_CLUB : REF BLOCK [,BYTE],          ! cluster block
;  635          0870  2     SCH$GL_CURPCB
;  636          0871  2                 : REF BLOCK [,BYTE] ;         ! current PCB.
;  637          0872  2
;  638          0873  2 IF .CLU$GL_CLUB NEQ 0 THEN
;  639          0874  3     BEGIN
;  640          0875  3     !
;  641          0876  3     !  Init the queue
;  642          0877  3     !
;  643          0878  3     CSP$GL_CSPQ =
;  644          0879  3                 CLU$GL_CLUB [CLUB$L_CSPFL] =
;  645          0880  3                 CLU$GL_CLUB [CLUB$L_CSPBL] =
;  646          0881  3                                 CLU$GL_CLUB [CLUB$L_CSPFL] ;
;  647          0882  3
;  648          0883  3     CLU$GL_CLUB [CLUB$L_CSPIPID] = .SCH$GL_CURPCB [PCB$L_PID] ;
;  649          0884  3
;  650          0885  3     RETURN 1 ;
;  651          0886  3     END
;  652          0887  2 ELSE
;  653          0888  2     !
;  654          0889  2     !  Not in the cluster
;  655          0890  2     !
;  656          0891  2     RETURN 0 ;
;  657          0892  1 END ;


                                                .EXTRN  CLU$GL_CLUB, SCH$GL_CURPCB

                                    0000 00000 KERNEL_INIT:
                                                .WORD   Save nothing                    ; 0850
                    51 00000000G    00  D0 00002        MOVL    CLU$GL_CLUB, R1         ; 0873
                                    25  13 00009        BEQL    1$                      ;
                    50      0088    C1  9E 0000B        MOVAB   136(R1), R0             ; 0881
              008C  C1              50  D0 00010        MOVL    R0, 140(R1)             ;
              0088  C1              50  D0 00015        MOVL    R0, 136(R1)             ; 0880
              0000' CF              50  D0 0001A        MOVL    R0, CSP$GL_CSPQ         ; 0879
```

```
                        50 00000000G   00  D0 0001F          MOVL    SCH$GL_CURPCB, R0                       ; 0883
                0090    C1       60    A0  D0 00026          MOVL    96(R0), 144(R1)                         ;
                        50             01  D0 0002C          MOVL    #1, R0                                  ; 0891
                                           C4 0002F          RET                                             ;
                                       50  D4 00030 1$:       CLRL    R1                                      ;
                                           04 00032          RET                                             ; 0892

; Routine Size:  51 bytes,    Routine Base:  $CODE$ + 025B

;  658          0893  1
```

```
;  660          0894  1 %SBTTL  'KERNEL_CLEANUP'
;  661          0895  1 ROUTINE  KERNEL_CLEANUP : NOVALUE =
;  662          0896  1
;  663          0897  1 !++
;  664          0898  1 !
;  665          0899  1 !  Perform kernel-mode exit handler stuff: reset the CLUB so that
;  666          0900  1 !  it's clear the CSP has disappeared.
;  667          0901  1 !
;  668          0902  1 !  CALLING SEQUENCE:
;  669          0903  1 !
;  670          0904  1 !  FORMAL PARAMETERS:
;  671          0905  1 !
;  672          0906  1 !  COMPLETION CODES:
;  673          0907  1 !
;  674          0908  1 !--
;  675          0909  2 BEGIN
;  676          0910  2 EXTERNAL
;  677          0911  2     CLU$GL_CLUB         : REF BLOCK [,BYTE] ;
;  678          0912  2
;  679          0913  2 LOCAL
;  680          0914  2     CSD ;
;  681          0915  2
;  682          0916  2 CLU$GL_CLUB [CLUB$L_CSPIPID] = 0 ;
;  683          0917  2 UNTIL REMQUE (.CLU$GL_CLUB [CLUB$L_CSPFL], CSD)
;  684          0918  2 DO
;  685          0919  2     EXE$CSP_COMMAND (CSP$_ABORT, .CSD) ;
;  686          0920  2
;  687          0921  2 RETURN ;
;  688          0922  1 END ;


                                OFFC 00000 KERNEL_CLEANUP:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 0895
                53 00000000G    00  9E 00002       MOVAB   CLU$GL_CLUB, R3                       ;
                50              63  D0 00009       MOVL    CLU$GL_CLUB, R0                       ; 0916
                     0090       C0  D4 0000C       CLRL    144(R0)                               ;
                50              63  D0 00010 1$:   MOVL    CLU$GL_CLUB, R0                       ; 0917
                52     0088     D0  0F 00013       REMQUE  @136(R0), CSD                         ;
                                0B  1D 00018       BVS     2$                                    ;
                51              02  D0 0001A       MOVL    #2, R1                                ; 0919
                    00000000G   00  16 0001D       JSB     EXE$CSP_COMMAND                       ;
                                EB  11 00023       BRB     1$                                    ;
                                    04 00025 2$:   RET                                           ; 0922

; Routine Size:  38 bytes,    Routine Base:  $CODE$ + 028E

;  689          0923  1
```

```
;  691     0924  1 %SBTTL  'REMQUE_CSD'
;  692     0925  1 ROUTINE  REMQUE_CSD =
;  693     0926  1
;  694     0927  1 !++
;  695     0928  1 !
;  696     0929  1 !  Go to kernel mode and read an entry from the queue CLUB$GL_CSPFL in the
;  697     0930  1 !  system CLUB structure. If we get one, copy it to the CSD structure and
;  698     0931  1 !  deallocate the nonpaged pool.
;  699     0932  1 !
;  700     0933  1 !  CALLING SEQUENCE:
;  701     0934  1 !       STATUS = REMQUE_CSD ()
;  702     0935  1 !
;  703     0936  1 !  FORMAL PARAMETERS:
;  704     0937  1 !       None
;  705     0938  1 !
;  706     0939  1 !  IMPLICIT OUTPUTS:
;  707     0940  1 !       CSP$GL_CURCTX   Address of CLX block if work was found
;  708     0941  1 !                       0 if there's nothing to do
;  709     0942  1 !
;  710     0943  1 !  COMPLETION CODES:
;  711     0944  1 !       0       = queue was empty (should not happen).
;  712     0945  1 !       1       = got an entry
;  713     0946  1 !
;  714     0947  1 !--
;  715     0948  2 BEGIN
;  716     0949  2 LOCAL   QUIT    : INITIAL (0),
;  717     0950  2         SO_CSD  : REF BLOCK [,BYTE] ;           ! Nonpaged pool CSD ptr
;  718     0951  2
;  719     0952  2 IF ..CSP$GL_CSPQ EQL 0 THEN RETURN 0 ;          ! Not in cluster yet
;  720     0953  2
;  721     0954  2 IF NOT REMQUE (.CSP$Q_CLX_CSD, CSP$GL_CURCTX)
;  722     0955  3 THEN BEGIN
;  723     0956  3      UNTIL .QUIT
;  724     0957  3      DO IF   REMQUE (..CSP$GL_CSPQ, SO_CSD)
;  725     0958  3         THEN QUIT = 1
;  726     0959  3         ELSE IF .SO_CSD [CSD$B_TYPE] NEQ DYN$C_CLU
;  727     0960  3              OR .SO_CSD [CSD$W_SIZE] GTRU CSP$K_MAX_CSDLNG
;  728     0961  3              THEN EXE$CSP_COMMAND (CSP$_BADCSD, .SO_CSD)
;  729     0962  4              ELSE BEGIN
;  730     0963  4                   CSP$GL_CURCTX [CLX$A_SO_CSD] = .SO_CSD ;
;  731     0964  4                   CSP$GL_CURCTX [CLX$B_FLAGS]  = 0 ;
;  732     0965  4                   CH$MOVE (.SO_CSD [CSD$W_SIZE]
;  733     0966  4                            ,.SO_CSD
;  734     0967  4                            ,.CSP$GL_CURCTX [CLX$A_PO_CSD]
;  735     0968  4                            ) ;
;  736     0969  4                   RETURN 1 ;
;  737     0970  3                   END ;
;  738     0971  3
;  739     0972  3      INSQUE (.CSP$GL_CURCTX, CSP$Q_CLX_CSD) ;
;  740     0973  2      END ;
;  741     0974  2
;  742     0975  2 CSP$GL_CURCTX = 0 ;
;  743     0976  2 RETURN 0 ;
;  744     0977  1 END ;
```

```
                                        0FFC 00000 REMQUE_CSD:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                ; 0925
                              58      0000'  CF  9E 00002          MOVAB   CSP$GL_CURCTX, R8
                                             57  D4 00007          CLRL    QUIT                                       ;
                                                                                                                       ; 0948
                                        EC   B8  D5 00009          TSTL    @CSP$GL_CSPQ                               ;
                                                                                                                       ; 0952
                                             52  13 0000C          BEQL    7$                                         ;
                              68      0000'  DF  0F 0000E          REMQUE  @CSP$Q_CLX_CSD, CSP$GL_CURCTX              ;
                                                                                                                       ; 0954
                                             49  1D 00013          BVS     6$                                         ;
                              40             57  E8 00015 1$:      BLBS    QUIT, 5$                                   ;
                                                                                                                       ; 0956
                              50        EC   B8  9E 00018          MOVAB   @CSP$GL_CSPQ, R0                           ;
                                                                                                                       ; 0957
                              56        00   B0  0F 0001C          REMQUE  @0(R0), S0_CSD                             ;
                                             05  1C 00020          BVC     2$                                         ;
                              57             01  D0 00022          MOVL    #1, QUIT                                   ;
                                                                                                                       ; 0958
                                             EE  11 00025          BRB     1$                                         ;
                            65  8F           0A  A6 91 00027 2$:    CMPB    10(S0_CSD), #101                           ;
                                                                                                                       ; 0959
                                             08  12 0002C          BNEQ    3$                                         ;
                          1000  8F           08  A6 B1 0002E          CMPW    8(S0_CSD), #4096                         ;
                                                                                                                       ; 0960
                                             0E  1B 00034          BLEQU   4$                                         ;
                              52             56  D0 00036 3$:      MOVL    S0_CSD, R2                                 ;
                                                                                                                       ; 0961
                              51             03  D0 00039          MOVL    #3, R1                                     ;
                                    00000000G    00  16 0003C          JSB     EXE$CSP_COMMAND                        ;
                                             D1  11 00042          BRB     1$                                         ;
                              50             68  D0 00044 4$:      MOVL    CSP$GL_CURCTX, R0                          ;
                                                                                                                       ; 0963
                            0C  A0           56  D0 00047          MOVL    S0_CSD, 12(R0)                             ;
                                        0B   A0  94 0004B          CLRB    11(R0)                                     ;
                                                                                                                       ; 0964
                10  B0        66        08   A6  28 0004E          MOVC3   8(S0_CSD), (S0_CSD), @16(R0)               ;
                                                                                                                       ; 0967
                              50             01  D0 00054          MOVL    #1, R0                                     ;
                                                                                                                       ; 0969
                                             04  00057          RET                                                ;
                                                                                                                       ;
                          0000'  CF        00   B8  0E 00058 5$:      INSQUE  @CSP$GL_CURCTX, CSP$Q_CLX_CSD          ;
                                                                                                                       ; 0972
                                             68  D4 0005E 6$:      CLRL    CSP$GL_CURCTX                              ;
                                                                                                                       ; 0975
                                             50  D4 00060 7$:      CLRL    R0                                         ;
                                                                                                                       ; 0977
                                             04  00062          RET                                                ;
                                                                                                                       ;

; Routine Size:  99 bytes,    Routine Base:  $CODE$ + 02B4

;  745          0978  1
```

```
;  747          0979  1 %SBTTL  'DEANONPAGED - call EXE$DEANONPAGED from kernel mode'
;  748          0980  1 ROUTINE  DEANONPAGED (BUFFER) =
;  749          0981  1
;  750          0982  1 !++
;  751          0983  1 !
;  752          0984  1 !  Call EXE$DEANONPAGED from kernel mode
;  753          0985  1 !
;  754          0986  1 !  CALLING SEQUENCE:
;  755          0987  1 !       KRNL_CALL (DEANONPAGED, BUF) in Kernel mode
;  756          0988  1 !
;  757          0989  1 !  FORMAL PARAMETERS:
;  758          0990  1 !       BUF     = address of buffer to deallocate (SIZE field indicates how big)
;  759          0991  1 !
;  760          0992  1 !  COMPLETION CODES:
;  761          0993  1 !
;  762          0994  1 !--
;  763          0995  1 RETURN EXE$DEANONPAGED ( .BUFFER ) ;



                                        0FFC 00000 DEANONPAGED:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 0980
                                 50        04     AC  D0 00002    MOVL    BUFFER, R0                            ; 0995
                                    00000000G   00  16 00006    JSB     EXE$DEANONPAGED                       ;
                                                    04 0000C    RET                                           ;

; Routine Size:  13 bytes,    Routine Base:  $CODE$ + 0317
```

```
:002 !DYC0002    0996  1 %SBTTL  'CHECK_SWVERS - Check for different versions of VMS'
:003 !DYC0002    0997  1 ROUTINE  CHECK_SWVERS =
:004 !DYC0002    0998  1
:005 !DYC0002    0999  1 !++
:006 !DYC0002    1000  1 !
:007 !DYC0002    1001  1 !  Check for different VMS versions in cluster.
:008 !DYC0002    1002  1 !
:009 !DYC0002    1003  1 !
:010 !DYC0002    1004  1 !  CALLING SEQUENCE:
:011 !DYC0002    1005  1 !       CHECK_SWVERS()
:012 !DYC0002    1006  1 !
:013 !DYC0002    1007  1 !  FORMAL PARAMETERS:
:014 !DYC0002    1008  1 !       None
:015 !DYC0002    1009  1 !
:016 !DYC0002    1010  1 !  COMPLETEION CODES:
:017 !DYC0002    1011  1 !       0 if different versions exist
:018 !DYC0002    1012  1 !       1 if no other version
:019 !DYC0002    1013  1 !
:020 !DYC0002    1014  1 !--
:021 !DYC0002    1015  1
:022 !DYC0002    1016  2 BEGIN
:023 !DYC0002    1017  2
:024 !DYC0002    1018  2 LOCAL
:025 !DYC0002    1019  2     ITEM_LIST: $ITMLST_DECL(ITEMS=1),
:026 !DYC0002    1020  2     MY_VER,                         ! Own version number
:027 !DYC0002    1021  2     VER,                            ! Other's version number
:028 !DYC0002    1022  2     RET_LEN,
:029 !DYC0002    1023  2     CSID,
:030 !DYC0002    1024  2     STATUS;
:031 !DYC0002    1025  2
:032 !DYC0002    1026  2 !
:033 !DYC0002    1027  2 !  Set up item list for SYS$GETSYI to get software versions
:034 !DYC0002    1028  2 !
:035 !DYC0002 P  1029  2 $ITMLST_INIT(ITMLST = ITEM_LIST,
:036 !DYC0002 P  1030  2                (ITMCOD = SYI$_NODE_SWVERS,
:037 !DYC0002 P  1031  2                 BUFADR = MY_VER,
:038 !DYC0002    1032  2                 RETLEN = RET_LEN));
:039 !DYC0002    1033  2
:040 !DYC0002    1034  2 !
:041 !DYC0002    1035  2 !  Get my own VMS version number by calling $GETSYI
:042 !DYC0002    1036  2 !
:043 !DYC0002    1037  2 IF NOT $GETSYIW(ITMLST=ITEM_LIST) THEN
:044 !DYC0002    1038  2     RETURN 1 ;                      ! Ignore error and return
:045 !DYC0002    1039  2
:046 !DYC0002    1040  2 VER = .MY_VER ;                     ! Initialize other's version to be the same
:047 !DYC0002    1041  2
:048 !DYC0002    1042  2 !
:049 !DYC0002    1043  2 !  Check software version of nodes in cluster by calling SYS$GETSYIW repeatedly
:050 !DYC0002    1044  2 !  and compare it with own version until no more node.
:051 !DYC0002    1045  2 !
:052 !DYC0002    1046  2 CSID = -1 ;                         ! Wild card for all nodes
:053 !DYC0002    1047  2 ITEM_LIST[1,ITM$L_BUFADR] = VER ;
:054 !DYC0002    1048  2
:055 !DYC0002    1049  2 WHILE (STATUS = $GETSYIW(CSIDADR=CSID, ITMLST=ITEM_LIST)) NEQ SS$_NOMORENODE DO
:056 !DYC0002    1050  2
:057 !DYC0002    1051  2     IF NOT .STATUS THEN
:058 !DYC0002    1052  2         RETURN 1                    ! Ignore error and return
```

```
:059 !DYC0002    1053  2
:060 !DYC0002    1054  2      ELSE IF .VER NEQ .MY_VER THEN
:061 !DYC0002    1055  2          RETURN 0 ;                  ! Different versions of VMS
:062 !DYC0002    1056  2
:063 !DYC0002    1057  2 RETURN 1 ;                           ! All have same version
:064 !DYC0002    1058  1 END ;
; INFO#250                 L1:1040
; Referenced LOCAL symbol MY_VER is probably not initialized


                                                        .EXTRN  SYS$GETSYIW

                                        0004 00000 CHECK_SWVERS:
                                                        .WORD   Save R2                                 ; 0997
                    52 00000000G    00  9E 00002         MOVAB   SYS$GETSYIW, R2                         ;
                    5E              20  C2 00009         SUBL2   #32, SP                                 ;
                    50        10    AE  9E 0000C         MOVAB   ITEM_LIST, $$ITMBLKPTR                  ; 1032
                    80 10D60004     8F  D0 00010         MOVL    #282460164, ($$ITMBLKPTR)+              ;
                    80              6E  9E 00017         MOVAB   MY_VER, ($$ITMBLKPTR)+                  ;
                    80        04    AE  9E 0001A         MOVAB   RET_LEN, ($$ITMBLKPTR)+                 ;
                                    80  D4 0001E         CLRL    ($$ITMBLKPTR)+                          ;
                                    7E  7C 00020         CLRQ    -(SP)                                   ; 1037
                                    7E  D4 00022         CLRL    -(SP)                                   ;
                              1C    AE  9F 00024         PUSHAB  ITEM_LIST                               ;
                                    7E  7C 00027         CLRQ    -(SP)                                   ;
                                    7E  D4 00029         CLRL    -(SP)                                   ;
                    62              07  FB 0002B         CALLS   #7, SYS$GETSYIW                         ;
                    31              50  E9 0002E         BLBC    R0, 2$                                  ;
              08    AE              6E  D0 00031         MOVL    MY_VER, VER                             ; 1040
              0C    AE              01  CE 00035         MNEGL   #1, CSID                                ; 1046
                    6D        08    AE  9E 00039         MOVAB   VER, ITEM_LIST+16                       ; 1047
                                    7E  7C 0003D 1$:     CLRQ    -(SP)                                   ; 1049
                                    7E  D4 0003F         CLRL    -(SP)                                   ;
                              1C    AE  9F 00041         PUSHAB  ITEM_LIST                               ;
                                    7E  D4 00044         CLRL    -(SP)                                   ;
                              20    AE  9F 00046         PUSHAB  CSID                                    ;
                                    7E  D4 00049         CLRL    -(SP)                                   ;
                    62              07  FB 0004B         CALLS   #7, SYS$GETSYIW                         ;
         00000A00   8F              50  D1 0004E         CMPL    STATUS, #2560                           ;
                                    0B  13 00055         BEQL    2$                                      ;
                    08              50  E9 00057         BLBC    STATUS, 2$                              ; 1051
                    6E        08    AE  D1 0005A         CMPL    VER, MY_VER                             ; 1054
                                    DD  13 0005E         BEQL    1$                                      ;
                                    04  11 00060         BRB     3$                                      ; 105?
                    50              01  D0 00062 2$:     MOVL    #1, R0                                  ; 1057
                                    04  00065            RET                                             ;
                                    50  D4 00066 3$:     CLRL    R0                                      ; 1058
                                    04  00068            RET                                             ;

; Routine Size:  105 bytes,    Routine Base:  $CODE$ + 0324

;  764          1059  1
```

```
;  766          1060  1 %SBTTL  'MUMBLE - Dummy routine'
;  767          1061  1
;  768          1062  1 GLOBAL ROUTINE MUMBLE = RETURN 1 ;



                                                0000 00000          .ENTRY  MUMBLE, Save nothing                                    ; 1062
                                     50      01  D0 00002          MOVL    #1, R0                                                  ;
                                                 04 00005          RET                                                             ;

; Routine Size:  6 bytes,    Routine Base:  $CODE$ + 038D

;  769          1063  1
;  770          1064  1
;  771          1065  1
;  772          1066  1
;  773          1067  1 %SBTTL  'CSP$$CRASH - Report bug'
;  774          1068  1
;  775          1069  1 GLOBAL ROUTINE CSP$$CRASH (CODE) =
;  776          1070  1
;  777          1071  2 BEGIN
;  778          1072  2 $EXIT (CODE = .CODE) ;
;  779          1073  2 RETURN .CODE ;
;  780          1074  1 END ;



                                                0000 00000          .ENTRY  CSP$$CRASH, Save nothing                                ; 1069
                                     04  AC  DD 00002          PUSHL   CODE                                                    ; 1072
                         00000000G   00      01  FB 00005          CALLS   #1, SYS$EXIT                                            ;
                                     50  04  AC  D0 0000C          MOVL    CODE, R0                                                ; 1073
                                                 04 00010          RET                                                             ; 1074

; Routine Size:  17 bytes,    Routine Base:  $CODE$ + 0393

;  781          1075  1
;  782          1076  1
;  783          1077  1
;  784          1078  1 END
;  785          1079  0 ELUDOM



;                                       PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
;  $GLOBAL$                                28  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;  $OWN$                    256  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                   932  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                   212  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                              Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA18:[SYSLIB]LIB.L32;1          18619        39         0        1000        00:01.9



; Information:  1
; Warnings:     0
; Errors:       0



;                                    COMMAND QUALIFIERS

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

;  786            1080  0
; Size:           932 code + 496 data bytes
; Run Time:         00:22.8
; Elapsed Time:     00:51.9
; Lines/CPU Min:     2837
; Lexemes/CPU-Min:  19232
; Memory Used:  141 pages
; Compilation Complete
```

0448 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

CSP
LIS

TTYCHARI
LIS

TTDRVR

OPDRVWS
LIS

TTDRIVER
MAP

YCDRIVER
MAP

CSPQUORUM
LIS

TTYCHARO
LIS