C 8

```
   OOOOOOOOO     PPPPPPPPPPPP        CCCCCCCCCCCC     OOOOOOOOO     MMM         MMM
   OOOOOOOOO     PPPPPPPPPPPP        CCCCCCCCCCCC     OOOOOOOOO     MMM         MMM
   OOOOOOOOO     PPPPPPPPPPPP        CCCCCCCCCCCC     OOOOOOOOO     MMM         MMM
OOO         OOO  PPP         PPP  CCC              OOO         OOO  MMMMMM   MMMMMM
OOO         OOO  PPP         PPP  CCC              OOO         OOO  MMMMMM   MMMMMM
OOO         OOO  PPP         PPP  CCC              OOO         OOO  MMMMMM   MMMMMM
OOO         OOO  PPP         PPP  CCC              OOO         OOO  MMM   MMM   MMM
OOO         OOO  PPP         PPP  CCC              OOO         OOO  MMM   MMM   MMM
OOO         OOO  PPP         PPP  CCC              OOO         OOO  MMM   MMM   MMM
OOO         OOO  PPPPPPPPPPPP     CCC              OOO         OOO  MMM         MMM
OOO         OOO  PPPPPPPPPPPP     CCC              OOO         OOO  MMM         MMM
OOO         OOO  PPPPPPPPPPPP     CCC              OOO         OOO  MMM         MMM
OOO         OOO  PPP              CCC              OOO         OOO  MMM         MMM
OOO         OOO  PPP              CCC              OOO         OOO  MMM         MMM
OOO         OOO  PPP              CCC              OOO         OOO  MMM         MMM
OOO         OOO  PPP              CCC              OOO         OOO  MMM         MMM
OOO         OOO  PPP              CCC              OOO         OOO  MMM         MMM
OOO         OOO  PPP              CCC              OOO         OOO  MMM         MMM
   OOOOOOOOO     PPP                 CCCCCCCCCCCC     OOOOOOOOO     MMM         MMM
   OOOOOOOOO     PPP                 CCCCCCCCCCCC     OOOOOOOOO     MMM         MMM
   OOOOOOOOO     PPP                 CCCCCCCCCCCC     OOOOOOOOO     MMM         MMM
```

```
_$2

Sym
---
ALL

ASC

BOD
BOD
BOD
BOD
BOD
BOD
BOD
BOD
BUG
BYP
CAN
CAN
CAN
CHE
CHE

CLU
CLU
CLU
CLU
CLU
CLU
CLU

CLU
CLU
CLU
CLU
CLU
CLU
CLU

CLU
CLU
```

```
**FILE**ID**CLUSCOMM                                  M  2



  CCCCCCCC  LL          UU      UU    SSSSSSSS    CCCCCCCC    OOOOOO    MM      MM  MM      MM
  CCCCCCCC  LL          UU      UU    SSSSSSSS    CCCCCCCC    OOOOOO    MM      MM  MM      MM
CC          LL          UU      UU  SS          CC          OO      OO  MMMM  MMMM  MMMM  MMMM
CC          LL          UU      UU  SS          CC          OO      OO  MMMM  MMMM  MMMM  MMMM
CC          LL          UU      UU  SS          CC          OO      OO  MM  MM  MM  MM  MM  MM
CC          LL          UU      UU  SS          CC          OO      OO  MM  MM  MM  MM  MM  MM
CC          LL          UU      UU    SSSSSS    CC          OO      OO  MM      MM  MM      MM
CC          LL          UU      UU    SSSSSS    CC          OO      OO  MM      MM  MM      MM
CC          LL          UU      UU          SS  CC          OO      OO  MM      MM  MM      MM
CC          LL          UU      UU          SS  CC          OO      OO  MM      MM  MM      MM
CC          LL          UU      UU          SS  CC          OO      OO  MM      MM  MM      MM
CC          LL          UU      UU          SS  CC          OO      OO  MM      MM  MM      MM
CC          LL          UU      UU          SS  CC          OO      OO  MM      MM  MM      MM      ....
CC          LL          UU      UU          SS  CC          OO      OO  MM      MM  MM      MM      ....
  CCCCCCCC  LLLLLLLLLL  UUUUUUUUUU  SSSSSSSS      CCCCCCCC    OOOOOO    MM      MM  MM      MM      ....
  CCCCCCCC  LLLLLLLLLL  UUUUUUUUUU  SSSSSSSS      CCCCCCCC    OOOOOO    MM      MM  MM      MM      ....

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

```
0 MODULE  OPC$CLUSCOMM    (
0                             LANGUAGE (BLISS32),
0                             IDENT = 'V04-000'
0                            ) =
0
0 !*************************************************************************
0 !*                                                                       *
0 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
0 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.               *
0 !*  ALL RIGHTS RESERVED.                                                 *
0 !*                                                                       *
0 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
0 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
0 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
0 !*  TRANSFERRED.                                                         *
0 !*                                                                       *
0 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
0 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
0 !*  CORPORATION.                                                         *
0 !*                                                                       *
0 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
0 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.              *
0 !*                                                                       *
0 !*                                                                       *
0 !*************************************************************************
0
0 !++
0 ! FACILITY:
0 !
0 !       OPCOM
0 !
0 ! ABSTRACT:
0 !
0 !       This module contains communications routines used by cluster functions within OPCOM.
0 !
0 ! Environment:
0 !
0 !       VAX/VMS operating system.
0 !
0 ! Author:
0 !
0 !       CW Hobbs
0 !
0 ! Creation date:
0 !
0 !       14 July 1983
0 !
0 ! Revision history:
0 !
0 !       V03-004 CWH3004         CW Hobbs                        18-May-1984
0 !               Reduce csp messages to two total, one per node to avoid
0 !               a temporary problem with port overloads.
0 !
0 !       V03-003 CWH3169         CW Hobbs                        5-May-1984
0 !               Second pass for cluster-wide OPCOM:
```

```
0 !                     - Perform a fairly liberal rewrite of this module using
0 !                       kernel-ast driven, parallel calls to CSP so that
0 !                       performance can be much better.
0 !                     - Return SS$_NOSUCHNODE status if the target node does
0 !                       not exist at the present time.
0 !
0 !     V03-002 CWH3002         CW Hobbs                16-Sep-1983
0 !             Clean up kernel handler and error messages
0 !
0 !
0 !--
```

```
  1 BEGIN                                                 ! Start of CLUSCOMM
  1
  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
  1 LIBRARY 'LIB$:OPCOMLIB';
  1 REQUIRE 'SHRLIB$:CSPDEF';
  1
  1 FORWARD ROUTINE
  1         CLUSCOMM_COD_ALLOCATE,                        ! Allocate a cluster output descriptor
  1         CLUSCOMM_COD_ERROR : NOVALUE,                 ! Handle an error described by a cod
  1         CLUSCOMM_COD_ERROR_AST : NOVALUE,             ! User mode ast routine for a cod error
  1         CLUSCOMM_DECLARE_KERNEL_AST,                  ! Declare kernel AST to start things moving
  1         CLUSCOMM_OUTPUT_KERNEL_AST : NOVALUE,         ! Handle ast from CSP
  1         CLUSCOMM_SEND,                                ! Jacket routine to send message to remote node(s)
  1         CLUSCOMM_SEND_ONE,                            ! Send message to single remote node
  1         CLUSCOMM_TARGET_IN_QUEUE;                     ! Count number of times target node in queue
  1
  1 EXTERNAL ROUTINE
  1         CLUSUTIL_FIND_NOD_BY_CSID,
  1         CLUSUTIL_NODE_MESSAGE,
  1         DUMP_LOG_FILE,
  1         SHARE_FAO_BUFFER,
  1         WRITE_LOG_FILE;
  1
  1 GLOBAL                                                ! Global so that SDA can find them easily
  1         COD_ALLOCATED,                                ! Count of CODs created
  1         COD_BUSY_COUNT,                               ! Current count of i/os pending
  1         COD_BUSY_MAX    : INITIAL (2),                ! Maximum number of EXE$CSP_CALLs pending
  1         COD_BUSY_NODE   : INITIAL (1),                ! Maximum number of EXE$CSP_CALLs pending to single node
  1         COD_ERRORS,                                   ! Count of requests with errors
  1         COD_FLUSHED,                                  ! Count of requests flushed (also count as errors)
  1         COD_REQUESTS,                                 ! Count of requests made
  1         COD_QUEUED,                                   ! Count of requests queued
  1         COD_BUSY_QUEUE  : VECTOR [2, LONG]            ! Queue of CODs pending for I/O
  1               INITIAL (REP 2 OF (COD_BUSY_QUEUE)),
  1         COD_FREE_QUEUE  : VECTOR [2, LONG]            ! Queue of cods available for use
  1               INITIAL (REP 2 OF (COD_FREE_QUEUE)),
  1         COD_WAIT_QUEUE  : VECTOR [2, LONG]            ! Queue of cods waiting for actual EXE$CSP_CALL to be queued
  1               INITIAL (REP 2 OF (COD_WAIT_QUEUE)),
  1         COD_GARBAGE_QUEUE : VECTOR [2, LONG]          ! Pointer to list of virtual memory to deallocate
  1               INITIAL (REP 2 OF (COD_GARBAGE_QUEUE));
  1 !
  1 ! A macro to put virtual memory back on the queue of garbage to be deallocated
  1 !
  1 MACRO
M 1         COLLECT_GARBAGE (INP_DESC) =
M 1                     BEGIN
M 1                     BIND
M 1                         desc = (INP_DESC) : VECTOR [, LONG],
M 1                         garbage = .desc [1] : VECTOR [, LONG];
M 1                     garbage [2] = .desc [0];                ! Store length as second longword in block
M 1                     $queue_insert_tail (garbage, cod_garbage_queue);
  1                     END %;
```

```
1 GLOBAL ROUTINE cluscomm_cod_allocate =
1
1 !++
1 ! Functional descripton:
1 !
1 !       This routine allocates a COD for a cluster write
1 !
1 ! Input:
1 !       None.
1 !
1 ! Output:
1 !       None.
1 !
1 ! Routine Value:
1 !       Address of block allocated
1 !--
1
2 BEGIN                                         ! Start of cluscomm_cod_allocate
2
2 LOCAL
2     cod       : $ref_bblock,                  ! cod data structure
2     garb      : REF VECTOR [, LONG],
2     ptr,
2     status;
2
2 !
2 ! If any garbage nodes are in the hopper, send them away.  Garbage is reclaimed this
2 ! way so that the kernel ast routines do not do free_vm calls on memory allocated
2 ! from user mode.
2 !
2 $queue_remove_head (cod_garbage_queue, garb);
2 WHILE .garb NEQ 0
2 DO
3     BEGIN
4     IF NOT (status = opc$free_vm (garb [2], garb))
3     THEN
3         $signal_stop (.status);
3     $queue_remove_head (cod_garbage_queue, garb);
2     END;
2 !
2 ! Get a cod, a Cluster Output Descriptor, if none available on the queue then
2 ! allocate and initialize one.
2 !
2 $queue_remove_head (cod_free_queue, cod);
2 IF .cod EQL 0
2 THEN
3     BEGIN
4     IF NOT (status = opc$get_vm (%ref (cod_k_size), ptr))
3     THEN
3         $signal_stop (.STATUS);
3     cod_allocated = .cod_allocated + 1;
3     cod = .ptr;
3     CH$FILL (0, cod_k_size, .cod);
3     cod [cod_w_size] = cod_k_size;
3     cod [cod_b_type] = %x'77';
2     END;
2 !
```

```
;  180        0371  2 ! Init the block
;  181        0372  2 !
;  182        0373  2 (cod [cod_q_quetime]) = 0;
;  183        0374  2 (cod [cod_q_quetime]+4) = 0;
;  184        0375  2 cod [cod_a_csd] = 0;
;  185        0376  2 cod [cod_l_msglen] = 0;
;  186        0377  2
;  187        0378  2 RETURN .cod;
;  188        0379  1 END;                                                ! End of cluscomm_cod_allocate


                                                                          .TITLE  OPC$CLUSCOMM
                                                                          .IDENT  \V04-000\

                                                                          .PSECT  $GLOBAL$,NOEXE,2

                                            00000 COD_ALLOCATED::
                                                                          .BLKB   4
                                            00004 COD_BUSY_COUNT::
                                                                          .BLKB   4
                                  00000002  00008 COD_BUSY_MAX::
                                                                          .LONG   2                                          ;
                                  00000001  0000C COD_BUSY_NODE::
                                                                          .LONG   1                                          ;
                                            00010 COD_ERRORS::
                                                                          .BLKB   4
                                            00014 COD_FLUSHED::
                                                                          .BLKB   4
                                            00018 COD_REQUESTS::
                                                                          .BLKB   4
                                            0001C COD_QUEUED::
                                                                          .BLKB   4
                                  00000000' 00020 COD_BUSY_QUEUE::
                                                                          .ADDRESS COD_BUSY_QUEUE                            ;
                                  00000000' 00024                         .ADDRESS COD_BUSY_QUEUE                            ;
                                  00000000' 00028 COD_FREE_QUEUE::
                                                                          .ADDRESS COD_FREE_QUEUE                            ;
                                  00000000' 0002C                         .ADDRESS COD_FREE_QUEUE                            ;
                                  00000000' 00030 COD_WAIT_QUEUE::
                                                                          .ADDRESS COD_WAIT_QUEUE                            ;
                                  00000000' 00034                         .ADDRESS COD_WAIT_QUEUE                            ;
                                  00000000' 00038 COD_GARBAGE_QUEUE::
                                                                          .ADDRESS COD_GARBAGE_QUEUE                         ;
                                  00000000' 0003C                         .ADDRESS COD_GARBAGE_QUEUE                         ;

                                                  _QH_=                            COD_GARBAGE_QUEUE
                                                  _QH_=                            COD_GARBAGE_QUEUE
                                                  _QH_=                            COD_FREE_QUEUE
                                                                          .EXTRN  CLUSUTIL_FIND_NOD_BY_CSID
                                                                          .EXTRN  CLUSUTIL_NODE_MESSAGE
                                                                          .EXTRN  DUMP_LOG_FILE, SHARE_FAO_BUFFER
                                                                          .EXTRN  WRITE_LOG_FILE, OPC$FREE_VM
                                                                          .EXTRN  LIB$STOP, OPC$GET_VM

                                                                          .PSECT  $CODE$,NOWRT,2

                                       007C 00000                         .ENTRY  CLUSCOMM_COD_ALLOCATE, Save R2,R3,R4,R5,R6  ; 0314
```

```
                          5E              0C  C2 00002       SUBL2   #12, SP                          ;
                   04     AE      0000'   DF  0F 00005 1$:   REMQUE  @_QH_, _T_                       ; 0344
                                          03  1C 0000B       BVC     2$                               ;
                                  04      AE  D4 0000D       CLRL    _T_                              ;
                                  04      AE  D5 00010 2$:   TSTL    GARB                             ; 0345
                                          15  13 00013       BEQL    3$                               ;
                                  04      AE  9F 00015       PUSHAB  GARB                             ; 0348
            7E     08     AE              08  C1 00018       ADDL3   #8, GARB, -(SP)                  ;
                0000G     CF              02  FB 0001D       CALLS   #2, OPC$FREE_VM                  ;
                          51              50  D0 00022       MOVL    R0, STATUS                       ;
                          DD              51  E8 00025       BLBS    STATUS, 1$                       ;
                                          22  11 00028       BRB     5$                               ; 0350
                          56      0000'   DF  0F 0002A 3$:   REMQUE  @_QH_, _T_                       ; 0357
                                          02  1C 0002F       BVC     4$                               ;
                                          56  D4 00031       CLRL    _T_                              ;
                                          56  D5 00033 4$:   TSTL    COD                              ; 0358
                                          36  12 00035       BNEQ    7$                               ;
                                  08      AE  9F 00037       PUSHAB  PTR                              ; 0361
                   04     AE              30  D0 0003A       MOVL    #48, 4(SP)                       ;
                                  04      AE  9F 0003E       PUSHAB  4(SP)                            ;
                0000G     CF              02  FB 00041       CALLS   #2, OPC$GET_VM                   ;
                          51              50  D0 00046       MOVL    R0, STATUS                       ;
                          0A              51  E8 00049       BLBS    STATUS, 6$                       ;
                                          51  DD 0004C 5$:   PUSHL   STATUS                           ; 0363
            00000000G     00              01  FB 0004E       CALLS   #1, LIB$STOP                     ;
                                              04 00055       RET                                      ;
                                  0000'   CF  D6 00056 6$:   INCL    COD_ALLOCATED                    ; 0364
                          56      08      AE  D0 0005A       MOVL    PTR, COD                         ; 0365
      30           00     6E              00  2C 0005E       MOVC5   #0, (SP), #0, #48, (COD)         ; 0366
                                          66     00063                                                ;
                   08     A6              30  B0 00064       MOVW    #48, 8(COD)                      ; 0367
                   0A     A6      77      8F  90 00068       MOVB    #119, 10(COD)                    ; 0368
                                  28      A6  7C 0006D 7$:   CLRQ    40(COD)                          ; 0373
                                  14      A6  D4 00070       CLRL    20(COD)                          ; 0375
                                  20      A6  D4 00073       CLRL    32(COD)                          ; 0376
                          50              56  D0 00076       MOVL    COD, R0                          ; 0378
                                              04 00079       RET                                      ; 0379

; Routine Size:  122 bytes,    Routine Base:  $CODE$ + 0000
;   190          0380  1 GLOBAL ROUTINE cluscomm_declare_kernel_ast =
;   191          0381  1
;   192          0382  1 !++
;   193          0383  1 ! Functional descripton:
;   194          0384  1 !
;   195          0385  1 !       This routine declares an ast to start the I/O, both it and the AST operate in kernel mode
;   196          0386  1 !
;   197          0387  1 ! Input:
;   198          0388  1 !       None.
;   199          0389  1 !
;   200          0390  1 ! Output:
;   201          0391  1 !       None.
;   202          0392  1 !
;   203          0393  1 ! Routine Value:
;   204          0394  1 !       Value from DCLAST
;   205          0395  1 !--
;   206          0396  1
;   207          0397  2 BEGIN                                             ! Start of cluscomm_declare_kernel_ast
;   208          0398  2
;   209          0399  2 RETURN $DCLAST (ASTADR=cluscomm_output_kernel_ast, ASTPRM=0);   ! 0 means start
;   210          0400  2
;   211          0401  1 END;                                              ! End of cluscomm_declare_kernel_ast


                                                        .EXTRN  SYS$DCLAST

                                     0000 00000         .ENTRY  CLUSCOMM_DECLARE_KERNEL_AST, Save nothing     ; 0380
                                  7E   7C 00002         CLRQ    -(SP)                                         ; 0399
                         0000V    CF   9F 00004         PUSHAB  CLUSCOMM_OUTPUT_KERNEL_AST                    ;
              00000000G  00       03   FB 00008         CALLS   #3, SYS$DCLAST                                ;
                                       04 0000F         RET                                                   ; 0401

; Routine Size:  16 bytes,    Routine Base:  $CODE$ + 007A
```

```
1 GLOBAL ROUTINE cluscomm_output_kernel_ast (csd : $ref_bblock) : NOVALUE =
1
1 !++
1 ! Functional descripton:
1 !
1 !       This routine is the I/O completion for a EXE$CSP_CALL write, executes in kernel mode
1 !
1 ! Input:
1 !       csd     address of CSD for the transfer
1 !
1 ! Output:
1 !       None.
1 !
1 ! Routine Value:
1 !       None.
1 !--
1
2 BEGIN                                                 ! Start of cluscomm_output_kernel_ast
2
2 LOCAL
2       cod : $ref_bblock;
2
2 !
2 ! If the parameter is non-zero, release that block
2 !
2 IF .csd NEQ 0
2 THEN
3     BEGIN
3     cod = .(csd [csd$ab_data]);                       ! COD address is first longword of data field
3     !
3     ! Free the CSD and put the message buffer on the list of virtual memory blocks to be deallocated
3     !
3     EXE$DEALLOC_CSD (.csd);
3     collect_garbage (cod [cod_q_msgbuf]);
3     !
3     ! Place the cod in the free queue
3     !
3     $queue_remove (.cod);                             ! Remove it from the queue (should be in the busy queue)
3     $queue_insert_tail (.cod, cod_free_queue);
3     cod_busy_count = .cod_busy_count - 1;
2     END;
2 !
2 ! If we can queue another EXE$CSP_CALL, then do so
2 !
2 cod = .cod_wait_queue [0];
2 WHILE .cod NEQ cod_wait_queue                         ! Loop until we see the end
2     AND
2      .cod_busy_count LSS .cod_busy_max                ! or until we have filled our quota
2 DO
3     BEGIN
3     LOCAL
3         next,
3         nod : $ref_bblock;
3     next = .cod [cod_l_flink];                        ! Save the pointer to the next, since we might pull it out
3     nod = .cod [cod_a_nod];                           ! Pointer to the nod block for the system
3     !
3     ! Make sure that the target is still there, this means that the csid stored in the node must be
```

```
;  270        0459  3       ! valid and that the node must not be in the departed state.
;  271        0460  3       !
;  272        0461  3       IF .nod [nod_l_node_csid] NEQ .cod [cod_l_csid]      ! Node has rebooted with a new csid
;  273        0462  3         OR
;  274        0463  3          .nod [nod_b_state] EQL nod_k_state_departed      ! Node is gone, but not forgotten
;  275        0464  3       THEN
;  276        0465  4           BEGIN
;  277        0466  4           $queue_remove (.cod);                  ! Remove it from the waiting queue
;  278        0467  4           cod [cod_l_errstat] = ss$_nodeleave;   ! Give it a reasonable error status
;  279        0468  4           cod_flushed = .cod_flushed + 1;        ! Count flushes individually
;  280        0469  4           cluscomm_cod_error (.cod);             ! Signal and clean it up
;  281        0470  4           END
;  282        0471  3       ELSE IF cluscomm_target_in_queue (.cod, cod_busy_queue) LSS .cod_busy_node
;  283        0472  3       THEN
;  284        0473  4           BEGIN
;  285        0474  4           LOCAL
;  286        0475  4               status;
;  287        0476  4           !
;  288        0477  4           ! Remove the cod from the waiting queue
;  289        0478  4           !
;  290        0479  4           $queue_remove (.cod);
;  291        0480  4           !
;  292        0481  4           ! Allocate a CSD block for the transfer.  Common fields in the CSD are initialized by
;  293        0482  4           ! the allocate routine.
;  294        0483  4           !
;  295        0484  5           IF NOT (cod [cod_l_errstat] = EXE$ALLOC_CSD (csd$k_length + 4 + .cod [cod_l_msglen]; csd))
;  296        0485  4           THEN
;  297        0486  5               BEGIN
;  298        0487  5               cluscomm_cod_error (.cod);                          ! Signal error and clean up
;  299        0488  5               RETURN;                                             ! More serious error, exit the routine
;  300        0489  4               END;
;  301        0490  4           cod [cod_a_csd] = .csd;                                 ! Point the cod at the csd
;  302        0491  4           (csd [csd$ab_data]) = .cod;                             ! Store cod address as first longword in csd
;  303        0492  4           !
;  304        0493  4           ! Set the other message dependent fields in the CSD
;  305        0494  4           !
;  306        0495  4           csd [csd$w_code] = csd$k_opcom;                         ! Set the OPCOM client code
;  307        0496  4           csd [csd$l_sendoff] = (4 + (csd [csd$ab_data])) - .csd; ! Store offset to the actual message
;  308        0497  4           csd [csd$l_sendlen] = .cod [cod_l_msglen];              ! Store size of message
;  309        0498  4           CH$MOVE (.cod [cod_l_msglen], .cod [cod_a_msgptr],      ! Move the message into the CSD
;  310        0499  4                           (4 + (csd [csd$ab_data])));             !  right after the cod address
;  311        0500  4           csd [csd$l_recvoff] = csd [csd$l_recvlen] = 0;          ! We do not want a reply
;  312        0501  4           csd [csd$a_astadr] = cluscomm_output_kernel_ast;        ! Store address of completion AST routine
;  313        0502  4           csd [csd$l_csid] = .cod [cod_l_csid];                   ! Store the target node CSID
;  314        0503  5           IF NOT (cod [cod_l_errstat] = EXE$CSP_CALL (.cod [cod_a_csd]))
;  315        0504  4           THEN
;  316        0505  4               cluscomm_cod_error (.cod)                           ! Signal error and clean up
;  317        0506  4           ELSE
;  318        0507  5               BEGIN
;  319        0508  5               cod_busy_count = .cod_busy_count + 1;               ! Bump the busy count
;  320        0509  5               cod_queued = .cod_queued + 1;                       ! Bump the count of those queued
;  321        0510  5               $queue_insert_tail (.cod, cod_busy_queue);          ! Put it at the end of the busy queue
;  322        0511  5               $gettim (timadr=cod [cod_q_quetime]);               ! Store the current time in the cod
;  323        0512  4               END;
;  324        0513  3           END;
;  325        0514  3       !
;  326        0515  3       ! Advance to the next one, using the saved next pointer
;  327          0516  3         !
;  328          0517  3         cod = .next;
;  329          0518  2         END;
;  330          0519  2     !
;  331          0520  2     ! Check the validity of the queues, crash the system if anything is wrong
;  332          0521  2     !
;  333        L 0522  2     %IF %VARIANT EQL 7
;  334        U 0523  2     %THEN
;  335        U 0524  2     BEGIN
;  336        U 0525  2     EXTERNAL ROUTINE monitor_queue : NOVALUE;
;  337        U 0526  2     monitor_queue (cod_busy_queue, 0);
;  338        U 0527  2     monitor_queue (cod_free_queue, 1);
;  339        U 0528  2     monitor_queue (cod_wait_queue, 2);
;  340        U 0529  2     END;
;  341          0530  2     %FI
;  342          0531  2
;  343          0532  2     RETURN;
;  344          0533  1     END;                                      ! End of cluscomm_output_kernel_ast


                                                    _QH_=                   COD_GARBAGE_QUEUE
                                                    _QH_=                   COD_FREE_QUEUE
                                                    _QH_=                   COD_BUSY_QUEUE
                                                              .EXTRN   EXE$DEALLOC_CSD
                                                              .EXTRN   EXE$ALLOC_CSD, EXE$CSP_CALL
                                                              .EXTRN   SYS$GETTIM

                                          03FC 00000          .ENTRY   CLUSCOMM_OUTPUT_KERNEL_AST, Save R2,R3,R4,-  ; 0402
                                                                       R5,R6,R7,R8,R9                               ;
                   59       0000'   CF   9E 00002             MOVAB    COD_BUSY_COUNT, R9                           ;
                   50          04   AC   D0 00007             MOVL     CSD, R0                                      ; 0427
                                    23   13 0000B             BEQL     1$                                           ;
                   57          52   A0   D0 0000D             MOVL     82(R0), COD                                  ; 0430
                        00000000G   00   16 00011             JSB      EXE$DEALLOC_CSD                              ; 0434
                   51          20   A7   9E 00017             MOVAB    32(COD), R1                                  ; 0435
                   50          04   A1   D0 0001B             MOVL     4(R1), R0                                    ;
              08   A0               61   D0 0001F             MOVL     (R1), 8(R0)                                  ;
              38   B9               60   0E 00023             INSQUE   (R0), @_QH_+4                                ;
                   50               67   0F 00027             REMQUE   (COD), _T_                                   ; 0439
              28   B9               67   0E 0002A             INSQUE   (COD), @_QH_+4                               ; 0440
                                    69   D7 0002E             DECL     COD_BUSY_COUNT                               ; 0441
                   57          2C   A9   D0 00030 1$:         MOVL     COD_WAIT_QUEUE, COD                          ; 0446
                   50          2C   A9   9E 00034 2$:         MOVAB    COD_WAIT_QUEUE, R0                           ; 0447
                   50               57   D1 00038             CMPL     COD, R0                                      ;
                                    01   12 0003B             BNEQ     3$                                           ;
                                         04 0003D             RET                                                   ;
              04   A9               69   D1 0003E 3$:         CMPL     COD_BUSY_COUNT, COD_BUSY_MAX                 ; 0449
                                    01   19 00042             BLSS     4$                                           ;
                                         04 00044             RET                                                   ;
                   58               67   D0 00045 4$:         MOVL     (COD), NEXT                                  ; 0455
                   50          18   A7   D0 00048             MOVL     24(COD), NOD                                 ; 0456
              10   A7          2C   A0   D1 0004C             CMPL     44(NOD), 16(COD)                             ; 0461
                                    06   12 00051             BNEQ     5$                                           ;
                   04          22   A0   91 00053             CMPB     34(NOD), #4                                  ; 0463
                                    0E   12 00057             BNEQ     6$                                           ;
                   50               67   0F 00059 5$:         REMQUE   (COD), _T_                                   ; 0466
```

```
                    1C   A7      223C   8F   3C 0005C         MOVZWL  #8764, 28(COD)                              ; 0467
                                   10   A9   D6 00062         INCL    COD_FLUSHED                                 ; 0468
                                        7C   11 00065         BRB     8$                                          ; 0469
                                   1C   A9   9F 00067 6$:     PUSHAB  COD_BUSY_QUEUE                              ; 0471
                                        57   DD 0006A         PUSHL   COD                                         ;
                 0000V   CF             02   FB 0006C         CALLS   #2, CLUSCOMM_TARGET_IN_QUEUE                ;
                    08   A9             50   D1 00071         CMPL    R0, COD_BUSY_NODE                           ;
                                        73   18 00075         BGEQ    9$                                          ;
                         50             67   0F 00077         REMQUE  (COD), T                                    ; 0479
           51       20   A7  00000056   8F   C1 0007A         ADDL3   #86, 32(COD), R1                            ; 0484
                             00000000G  00   16 00083         JSB     EXE$ALLOC_CSD                               ;
                    04   AC             52   D0 00089         MOVL    R2, CSD                                     ;
                    1C   A7             50   D0 0008D         MOVL    R0, 28(COD)                                 ;
                         08             50   E8 00091         BLBS    R0, 7$                                      ;
                                        57   DD 00094         PUSHL   COD                                         ; 0487
                 0000V   CF             01   FB 00096         CALLS   #1, CLUSCOMM_COD_ERROR                      ;
                                             04 0009B         RET                                                 ; 0486
                         56        04   AC   D0 0009C 7$:     MOVL    CSD, R6                                     ; 0490
                    14   A7             56   D0 000A0         MOVL    R6, 20(COD)                                 ;
                    52   A6             57   D0 000A4         MOVL    COD, 82(R6)                                 ; 0491
                    0C   A6             05   B0 000A8         MOVW    #5, 12(R6)                                  ; 0495
           50            56             56   C3 000AC         SUBL3   R6, R6, R0                                  ; 0496
                    16   A6        56   A0   9E 000B0         MOVAB   86(R0), 22(R6)                              ;
                    12   A6        20   A7   D0 000B5         MOVL    32(COD), 18(R6)                             ; 0497
                         50             56   D0 000BA         MOVL    R6, R0                                      ; 0499
     56    A0       24   B7        20   A7   28 000BD         MOVC3   32(COD), @36(COD), 86(R0)                   ;
                                   1A   A6   7C 000C4         CLRQ    26(R6)                                      ; 0500
                    22   A6      FF35   CF   9E 000C7         MOVAB   CLUSCOMM_OUTPUT_KERNEL_AST, 34(R6)          ; 0501
                    0E   A6        10   A7   D0 000CD         MOVL    16(COD), 14(R6)                             ; 0502
                         52        14   A7   D0 000D2         MOVL    20(COD), R2                                 ; 0503
                             00000000G  00   16 000D6         JSB     EXE$CSP_CALL                                ;
                    1C   A7             50   D0 000DC         MOVL    R0, 28(COD)                                 ;
                         09             50   E8 000E0         BLBS    R0, 10$                                     ;
                                        57   DD 000E3 8$:     PUSHL   COD                                         ; 0505
                 0000V   CF             01   FB 000E5         CALLS   #1, CLUSCOMM_COD_ERROR                      ;
                                        13   11 000EA 9$:     BRB     11$                                         ;
                                        69   D6 000EC 10$:    INCL    COD_BUSY_COUNT                              ; 0508
                                   18   A9   D6 000EE         INCL    COD_QUEUED                                  ; 0509
                    20   B9             67   0E 000F1         INSQUE  (COD), @_QH_+4                              ; 0510
                                   28   A7   9F 000F5         PUSHAB  40(COD)                                     ; 0511
             00000000G   00             01   FB 000F8         CALLS   #1, SYS$GETTIM                              ;
                         57             58   D0 000FF 11$:    MOVL    NEXT, COD                                   ; 0517
                                      FF2F   31 00102         BRW     2$                                          ; 0447
                                             04 00105         RET                                                 ; 0533

; Routine Size:  262 bytes,    Routine Base:  $CODE$ + 008A
;  346          0534   1 GLOBAL ROUTINE cluscomm_cod_error (cod : $ref_bblock) : NOVALUE =
;  347          0535   1
;  348          0536   1 !++
;  349          0537   1 ! Functional descripton:
;  350          0538   1 !
;  351          0539   1 !       This routine handles an error in CSP communications, executes in kernel mode.
;  352          0540   1 !       The error is given to a user-mode AST to actually handle
;  353          0541   1 !
;  354          0542   1 ! Input:
;  355          0543   1 !       cod     address of COD for the transfer
;  356          0544   1 !
;  357          0545   1 ! Output:
;  358          0546   1 !       None.
;  359          0547   1 !
;  360          0548   1 ! Routine Value:
;  361          0549   1 !       None.
;  362          0550   1 !--
;  363          0551   1
;  364          0552   2 BEGIN                                          ! Start of cluscomm_cod_error
;  365          0553   2
;  366          0554   2 LOCAL
;  367          0555   2     csd : $ref_bblock;
;  368          0556   2
;  369          0557   2 cod_errors = .cod_errors + 1;
;  370          0558   2 !
;  371          0559   2 ! Deallocate the CSD if present
;  372          0560   2 !
;  373          0561   2 IF (csd = .cod [cod_a_csd]) NEQ 0
;  374          0562   2 THEN
;  375          0563   2     EXE$DEALLOC_CSD (.csd);
;  376          0564   2 !
;  377          0565   2 ! Return any virtual memory to the free list
;  378          0566   2 !
;  379          0567   2 IF .cod [cod_l_msglen] NEQ 0
;  380          0568   2 THEN
;  381          0569   2     collect_garbage (cod [cod_q_msgbuf]);
;  382          0570   2 !
;  383          0571   2 ! Declare an AST in user mode, so that we can use RMS/etc
;  384          0572   2 !
;  385          0573   2 $DCLAST (astadr=cluscomm_cod_error_ast, astprm=.cod, acmode=psl$c_user);
;  386          0574   2
;  387          0575   2 RETURN;
;  388          0576   1 END;                                           ! End of cluscomm_cod_error


                                       _QH_=              COD_GARBAGE_QUEUE


                           001C 00000          .ENTRY  CLUSCOMM_COD_ERROR, Save R2,R3,R4          ; 0534
                 0000'  CF  D6 00002           INCL    COD_ERRORS                                 ; 0557
           54       04  AC  D0 00006           MOVL    COD, R4                                    ; 0561
           50       14  A4  D0 0000A           MOVL    20(R4), CSD                                ;
                        06  13 0000E           BEQL    1$                                         ;
            00000000G   00  16 00010           JSB     EXE$DEALLOC_CSD                            ; 0563
                    20  A4  D5 00016 1$:       TSTL    32(R4)                                     ; 0567
                        11  13 00019           BEQL    2$                                         ;
```

```
                   51      20  A4  9E 0001B          MOVAB   32(R4), R1                          ; 0569
                   50      04  A1  D0 0001F          MOVL    4(R1), R0                           ;
             08    A0          61  D0 00023          MOVL    (R1), 8(R0)                         ;
          0000'    DF          60  0E 00027          INSQUE  (R0), @_QH_+4                       ;
                               03  DD 0002C  2$:     PUSHL   #3                                  ; 0573
                           04  AC  DD 0002E          PUSHL   COD                                 ;
                        0000V  CF  9F 00031          PUSHAB  CLUSCOMM_COD_ERROR_AST              ;
        00000000G  00          03  FB 00035          CALLS   #3, SYS$DCLAST                      ;
                                   04 0003C          RET                                         ; 0576

; Routine Size:  61 bytes,    Routine Base:  $CODE$ + 0190
```

```
  1 GLOBAL ROUTINE cluscomm_cod_error_ast (cod : $ref_bblock) : NOVALUE =
  1
  1 !++
  1 ! Functional descripton:
  1 !
  1 !       This routine signals an error in CSP communications, executes in user mode.
  1 !
  1 ! Input:
  1 !       cod     address of COD for the transfer
  1 !
  1 ! Output:
  1 !       None.
  1 !
  1 ! Routine Value:
  1 !       None.
  1 !--
  1
  2 BEGIN                                           ! Start of cluscomm_cod_error
  2
  2 LOCAL
  2     leaving,
  2     dsc : VECTOR [2, LONG],
  2     nod : $ref_bblock;
  2
  2 nod = .cod [cod_a_nod];
  2 leaving = (.cod [cod_l_errstat] EQL ss$_nodeleave);
  3 IF (NOT .leaving)                               ! If any other error
  2   OR
  3    (NOT .nod [nod_v_node_leaving])              ! or if the first node_leaving error
  2 THEN
  3     BEGIN
  3     !
  3     ! Put a message in the logfile
  3     !
  3     clusutil_node_message (.nod, opc$_cluscomm, false);
  3     !
  3     ! If any error besides leaving, then put a message in the logfile about the exact reason
  3     !
  3     IF .leaving                                 ! Mark the first message so that we can skip the others
  3     THEN
  3         nod [nod_v_node_leaving] = true
  3     ELSE
  4         BEGIN
  4         write_log_file (
L 4             share_fao_buffer (%ASCID %STRING ('Unable to communicate with !AS (!XL), system status code !XL!
L 4                                             '  Current statistics for cluster message activity:!/',
L 4                                             '       Msg desc allocated!8UL     Errors       !8UL!/',
L 4                                             '       Msg requests       !8UL     Msgs flushed !8UL!/',
  4                                             '       Msgs queued        !8UL     Current busy !8UL'),
  4                                             nod [nod_q_name_desc], .cod [cod_l_csid], .cod [cod_l_errstat],
  4                                             .cod_allocated, .cod_errors-.cod_flushed,
  4                                             .cod_requests, .cod_flushed,
  4                                             .cod_queued, .cod_busy_count));
  4         !
  4         ! Write some more arcane, but useful messages if we are debugging
  4         !
L 4         %IF %VARIANT NEQ 0
```

```
;  447          U 0634  4           %THEN
;  448          U 0635  4           dsc [0] = cod_k_size;  dsc [1] = .cod;
;  449          U 0636  4           dump_log_file (dsc, %ASCID 'Dump of COD used in transfer');
;  450          U 0637  4           dump_log_file (cod [cod_q_msgbuf], %ASCID 'Dump of COD text field');
;  451            0638  4           %FI
;  452            0639  3           END;
;  453            0640  2       END;
;  454            0641  2
;  455            0642  2 $queue_insert_tail (.cod, cod_free_queue);       ! All done, put it back in the queue
;  456            0643  2
;  457            0644  2 RETURN;
;  458            0645  1 END;                                             ! End of cluscomm_cod_error_ast



                                                                         .PSECT  $PLIT$,NOWRT,NOEXE,2

75 6D 6D 6F 63 20 6F 74 20 65 6C 62 61 6E 55 00000 P.AAB:   .ASCII  \Unable to communicate with !AS (!XL), sy\   ;
53 41 21 20 68 74 69 77 20 65 74 61 63 69 6E 0000F                                                           :
               79 73 20 2C 29 4C 58 21 28 20 0001E                                                           :
64 6F 63 20 73 75 74 61 74 73 20 6D 65 74 73 00028          .ASCII  \stem status code !XL!/  Current statisti\   :
6E 65 72 72 75 43 20 20 2F 21 4C 58 21 20 65 00037                                                           :
               69 74 73 69 74 61 74 73 20 74 00046                                                           :
20 72 65 74 73 75 6C 63 20 72 6F 66 20 73 63 00050          .ASCII  \cs for cluster message activity:!/\<9>   :
74 69 76 69 74 63 61 20 65 67 61 73 73 65 6D 0005F                                                           :
                              09 2F 21 3A 79 0006E                                                           :
61 63 6F 6C 6C 61 20 63 73 65 64 20 67 73 4D 00073          .ASCII  \Msg desc allocated!8UL      Errors      \   :
72 72 45 20 20 20 20 20 4C 55 38 21 64 65 74 00082                                                           :
               20 20 20 20 20 20 20 73 72 6F 00091                                                           :
75 71 65 72 20 67 73 4D 09 2F 21 4C 55 38 21 0009B          .ASCII  \!8UL!/\<9>\Msg requests       !8UL      \   :
20 4C 55 38 21 20 20 20 20 20 20 73 74 73 65 000AA                                                           :
                                 20 20 20 20 000B9                                                           :
38 21 20 64 65 68 73 75 6C 66 20 73 67 73 4D 000BD          .ASCII  \Msgs flushed !8UL!/\<9>\Msgs queued     \   :
65 75 65 75 71 20 73 67 73 4D 09 2F 21 4C 55 000CC                                                           :
                                 20 20 20 64 000DB                                                           :
75 43 20 20 20 20 20 4C 55 38 21 20 20 20 20 000DF          .ASCII  \     !8UL      Current busy !8UL\<0><0>   :
4C 55 38 21 20 79 73 75 62 20 74 6E 65 72 72 000EE                                                           :
                                       00 00 000FD                                                           :
                                          00 000FF          .ASCII  <0>                                      :
                                   010E00FD  00100 P.AAA:   .LONG   17694973                                 :
                                   00000000' 00104          .ADDRESS P.AAB                                   :

                                                   _QH_=                     COD_FREE_QUEUE


                                                            .PSECT  $CODE$,NOWRT,2

                                       003C 00000           .ENTRY  CLUSCOMM_COD_ERROR_AST, Save R2,R3,R4,R5  ; 0577
                   55      0000'    CF    9E 00002          MOVAB   COD_FLUSHED, R5                          :
                   5E         08    C2    00007             SUBL2   #8, SP                                   :
                   53         04    AC    D0 0000A          MOVL    COD, R3                                  : 0601
                   52         18    A3    D0 0000E          MOVL    24(R3), NOD                              :
                              50    D4    00012             CLRL    R0                                       : 0602
         0000223C  8F         1C    A3    D1 00014          CMPL    28(R3), #8764                            :
                              02    12    0001C             BNEQ    1$                                       :
                              50    D6    0001E             INCL    R0                                       :
                   54         50    D0    00020 1$:         MOVL    R0, LEAVING                              :
                      05          54  E9 00023        BLBC    LEAVING, 2$                                   ; 0603
   44        2A       A2          03  E0 00026        BBS     #3, 42(NOD), 4$                               ; 0605
                                  7E  D4 0002B 2$:    CLRL    -(SP)                                         ; 0611
                      00058253    8F  DD 0002D        PUSHL   #361043                                       :
                                  52  DD 00033        PUSHL   NOD                                           :
             0000G    CF          03  FB 00035        CALLS   #3, CLUSUTIL_NODE_MESSAGE                     :
                      06          54  E9 0003A        BLBC    LEAVING, 3$                                   ; 0615
             2A       A2          08  88 0003D        BISB2   #8, 42(NOD)                                   ; 0617
                                  2C  11 00041        BRB     4$                                            :
                             F0   A5  DD 00043 3$:    PUSHL   COD_BUSY_COUNT                                ; 0629
                             08   A5  DD 00046        PUSHL   COD_QUEUED                                    :
                                  65  DD 00049        PUSHL   COD_FLUSHED                                   ; 0628
                             04   A5  DD 0004B        PUSHL   COD_REQUESTS                                  :
   7E        FC       A5          65  C3 0004E        SUBL3   COD_FLUSHED, COD_ERRORS, -(SP)                ; 0627
                             EC   A5  DD 00053        PUSHL   COD_ALLOCATED                                 :
                             1C   A3  DD 00056        PUSHL   28(R3)                                        ; 0626
                             10   A3  DD 00059        PUSHL   16(R3)                                        :
                             30   A2  9F 0005C        PUSHAB  48(NOD)                                       :
                          0000'   CF  9F 0005F        PUSHAB  P.AAA                                         ; 0625
             0000G    CF          0A  FB 00063        CALLS   #10, SHARE_FAO_BUFFER                         ; 0626
                                  50  DD 00068        PUSHL   R0                                            :
             0000G    CF          01  FB 0006A        CALLS   #1, WRITE_LOG_FILE                            :
                18    B5          63  0E 0006F 4$:    INQUE   (R3), @_QH_+4                                 ; 0642
                                      04 00073        RET                                                   ; 0645

; Routine Size:  116 bytes,    Routine Base:  $CODE$ + 01CD
;  460          0646  1 GLOBAL ROUTINE CLUSCOMM_SEND (CSID, MSG_LEN, MSG_PTR) =          %SBTTL 'CLUSCOMM_SEND (CSID, MSG_LEN, MSG_PT
;  461          0647  1
;  462          0648  1 !++
;  463          0649  1 ! Functional description:
;  464          0650  1 !
;  465          0651  1 !       Jacket routine to send a message to remote node(s), and wait for completion.
;  466          0652  1 !
;  467          0653  1 ! Input:
;  468          0654  1 !
;  469          0655  1 !       CSID    - Id of target node, -1 for broadcast to all nodes except local
;  470          0656  1 !       MSG_LEN - Length of message
;  471          0657  1 !       MSG_PTR - Address of message
;  472          0658  1 !
;  473          0659  1 ! Implicit Input:
;  474          0660  1 !
;  475          0661  1 !       None.
;  476          0662  1 !
;  477          0663  1 ! Output:
;  478          0664  1 !
;  479          0665  1 !       None.
;  480          0666  1 !
;  481          0667  1 ! Implict output:
;  482          0668  1 !
;  483          0669  1 !       None.
;  484          0670  1 !
;  485          0671  1 ! Side effects:
;  486          0672  1 !
;  487          0673  1 !       Messages will be sent to remote nodes.
;  488          0674  1 !
;  489          0675  1 ! Routine value:
;  490          0676  1 !
;  491          0677  1 !       Status from comm primitive.
;  492          0678  1 !--
;  493          0679  1
;  494          0680  2 BEGIN                                               ! Start of CLUSCOMM_SEND
;  495          0681  2
;  496          0682  2 EXTERNAL
;  497          0683  2         GLOBAL_STATUS   : BITVECTOR [32],
;  498          0684  2         LCL_CSID        : LONG,
;  499          0685  2         NOD_HEAD        : VECTOR [2, LONG];
;  500          0686  2
;  501          0687  2 LOCAL
;  502          0688  2         FINAL_STAT      : LONG,
;  503          0689  2         NOD             : $ref_bblock,
;  504          0690  2         STATUS          : LONG;
```

```
2 !
2 ! Assume success for final status
2 !
2 FINAL_STAT = SS$_NORMAL;
2 !
2 ! If not in a cluster we are done, return with success
2 !
2 IF NOT .GLOBAL_STATUS [GBLSTS_K_IN_VAXcluster]
2 THEN
2     RETURN .FINAL_STAT;
2 !
2 ! If CSID is -1, send it to everyone
2 !
2 IF .CSID EQL -1
2 THEN
3     BEGIN
3     NOD = .NOD_HEAD [0];
3     WHILE .NOD NEQ NOD_HEAD [0]
3     DO
4         BEGIN
4         LOCAL
4             TARGET;
4         !
4         ! Send to all nodes but local
4         !
4         TARGET = .NOD [NOD_L_NODE_CSID];
4         IF .TARGET NEQ .LCL_CSID
4         THEN
5             BEGIN
5             STATUS = CLUSCOMM_SEND_ONE (.TARGET, .NOD, .MSG_LEN, .MSG_PTR);
5             IF NOT .STATUS
5             THEN
5                 FINAL_STAT = .STATUS;
4             END;
4         !
4         ! Move to the next node
4         !
4         NOD = .NOD [NOD_L_FLINK];
3         END;
3     END
3 !
3 ! CSID is real, send it to a single node
3 !
2 ELSE
3     BEGIN
3     NOD = CLUSUTIL_FIND_NOD_BY_CSID (.CSID);
4     FINAL_STAT = (IF .NOD EQL 0
4                   THEN SS$_NOSUCHNODE
3                   ELSE CLUSCOMM_SEND_ONE (.CSID, .NOD, .MSG_LEN, .MSG_PTR));
2     END;
2
2 RETURN .FINAL_STAT;
1 END;                                              ! End of CLUSCOMM_SEND


                                                    .EXTRN  GLOBAL_STATUS, LCL_CSID
```

```
                                                         .EXTRN  NOD_HEAD

                                      000C 00000         .ENTRY  CLUSCOMM_SEND, Save R2,R3                    ; 0646
                 53             01  D0 00002             MOVL    #1, FINAL_STAT                                ; 0694
                 5F      0000G  CF  E9 00005             BLBC    GLOBAL_STATUS+1, 5$                           ; 0698
      FFFFFFFF   8F         04  AC  D1 0000A             CMPL    CSID, #-1                                     ; 0704
                                30  12 00012             BNEQ    3$                                            ;
                 52      0000G  CF  D0 00014             MOVL    NOD_HEAD, NOD                                 ; 0707
                 51      0000G  CF  9E 00019 1$:         MOVAB   NOD_HEAD, R1                                  ; 0708
                 51             52  D1 0001E             CMPL    NOD, R1                                       ;
                                46  13 00021             BEQL    5$                                            ;
                 51         2C  A2  D0 00023             MOVL    44(NOD), TARGET                               ; 0716
          0000G  CF             51  D1 00027             CMPL    TARGET, LCL_CSID                              ; 0717
                                11  13 0002C             BEQL    2$                                            ;
                 7E         08  AC  7D 0002E             MOVQ    MSG_LEN, -(SP)                                ; 0720
                                06  BB 00032             PUSHR   #^M<R1,R2>                                    ;
          0000V  CF             04  FB 00034             CALLS   #4, CLUSCOMM_SEND_ONE                         ;
                 03             50  E8 00039             BLBS    STATUS, 2$                                    ; 0721
                 53             50  D0 0003C             MOVL    STATUS, FINAL_STAT                            ; 0723
                 52             62  D0 0003F 2$:         MOVL    (NOD), NOD                                    ; 0728
                                D5  11 00042             BRB     1$                                            ; 0708
                            04  AC  DD 00044 3$:         PUSHL   CSID                                          ; 0736
          0000G  CF             01  FB 00047             CALLS   #1, CLUSUTIL_FIND_NOD_BY_CSID                 ;
                 52             50  D0 0004C             MOVL    R0, NOD                                       ;
                                07  12 0004F             BNEQ    4$                                            ; 0737
                 53       028C  8F  3C 00051             MOVZWL  #652, FINAL_STAT                              ;
                                11  11 00056             BRB     5$                                            ;
                 7E         08  AC  7D 00058 4$:         MOVQ    MSG_LEN, -(SP)                                ; 0739
                                52  DD 0005C             PUSHL   NOD                                           ;
                            04  AC  DD 0005E             PUSHL   CSID                                          ;
          0000V  CF             04  FB 00061             CALLS   #4, CLUSCOMM_SEND_ONE                         ;
                 53             50  D0 00066             MOVL    R0, FINAL_STAT                                ;
                 50             53  D0 00069 5$:         MOVL    FINAL_STAT, R0                                ; 0742
                                    04 0006C             RET                                                   ; 0743

; Routine Size:  109 bytes,    Routine Base:  $CODE$ + 0241
;  560        0744  1 GLOBAL ROUTINE CLUSCOMM_SEND_ONE (CSID, NOD, MSG_LEN, MSG_PTR) =          %SBTTL 'CLUSCOMM_SEND_ONE'
;  561        0745  1
;  562        0746  1 !++
;  563        0747  1 ! Functional description:
;  564        0748  1 !
;  565        0749  1 !       Send a message to a remote node, and wait for completion.
;  566        0750  1 !
;  567        0751  1 ! Input:
;  568        0752  1 !
;  569        0753  1 !       CSID    - Id of target node
;  570        0754  1 !       NOD     - Address of NOD block for target node
;  571        0755  1 !       MSG_LEN - Length of message
;  572        0756  1 !       MSG_PTR - Address of message
;  573        0757  1 !
;  574        0758  1 ! Implicit Input:
;  575        0759  1 !
;  576        0760  1 !       None.
;  577        0761  1 !
;  578        0762  1 ! Output:
;  579        0763  1 !
;  580        0764  1 !       None.
;  581        0765  1 !
;  582        0766  1 ! Implict output:
;  583        0767  1 !
;  584        0768  1 !       None.
;  585        0769  1 !
;  586        0770  1 ! Side effects:
;  587        0771  1 !
;  588        0772  1 !       Messages will be sent to remote nodes.
;  589        0773  1 !
;  590        0774  1 ! Routine value:
;  591        0775  1 !
;  592        0776  1 !       Status from comm primitive.
;  593        0777  1 !--
;  594        0778  1
;  595        0779  2 BEGIN                                               ! Start of CLUSCOMM_SEND_ONE
;  596        0780  2
;  597        0781  2 EXTERNAL
;  598        0782  2         GLOBAL_STATUS   : BITVECTOR [32];
;  599        0783  2
;  600        0784  2 LOCAL
;  601        0785  2         ARGLIST         : VECTOR [2, LONG],
;  602        0786  2         COD             : $ref_bblock,
;  603        0787  2         STATUS          : LONG;
;  604        0788  2
;  605        0789  2 !
;  606        0790  2 ! If not in a cluster we are done, return with error.
;  607        0791  2 !
;  608        0792  2 IF NOT .GLOBAL_STATUS [GBLSTS_K_IN_VAXcluster]
;  609        0793  2 THEN
;  610        0794  2     RETURN SS$_NOSUCHNODE;
;  611        0795  2 !
;  612        0796  2 ! Allocate and fill in the COD
;  613        0797  2 !
;  614        0798  2 COD = CLUSCOMM_COD_ALLOCATE ();                      ! Get a new COD
;  615        0799  2 COD [COD_L_CSID] = .CSID;                           ! Keep a copy of the CSID in the COD
;  616        0800  2 COD [COD_A_NOD] = .NOD;                             ! Keep the NOD address too
;  617          0801  2 COD [COD_L_MSGLEN] = MAXU (12, .MSG_LEN);       ! Store the length of the message, make sure garbage header
;  618          0802  3 IF NOT (STATUS = OPC$GET_VM (COD [COD_L_MSGLEN], COD [COD_A_MSGPTR]))
;  619          0803  2 THEN
;  620          0804  2     $signal_stop (.STATUS);
;  621          0805  2 CH$MOVE (.MSG_LEN, .MSG_PTR, .COD [COD_A_MSGPTR]);
;  622          0806  2 !
;  623          0807  2 ! Place the cod on the queue of outputs waiting
;  624          0808  2 !
;  625          0809  2 $QUEUE_INSERT_TAIL (.COD, COD_WAIT_QUEUE);
;  626          0810  2 COD_REQUESTS = .COD_REQUESTS + 1;
;  627          0811  2 !
;  628          0812  2 ! Change to kernel mode to start the transfer, call the ast routine with a zero
;  629          0813  2 ! parameter (arglst not relevant)
;  630          0814  2 !
;  631          0815  2 STATUS = $CMKRNL (ROUTIN = CLUSCOMM_DECLARE_KERNEL_AST, ARGLST = COD);
;  632          0816  2 !
;  633          0817  2 ! Signal errors.  If ast quota exceeded, then write a message, since it is almost certain that
;  634          0818  2 ! kernel ASTs are already active.  If not ast quota error, stop the process.
;  635          0819  2 !
;  636          0820  2 IF NOT .STATUS
;  637          0821  2 THEN
;  638          0822  3     BEGIN
;  639          0823  3     IF .STATUS NEQ SS$_EXQUOTA
;  640          0824  3     THEN
;  641          0825  4         $signal_stop (.STATUS)
;  642          0826  3     ELSE
;  643          0827  3         WRITE_LOG_FILE (%ASCID 'AST quota error in cluster communication');
;  644          0828  2     END;
;  645          0829  2
;  646          0830  2 RETURN .STATUS;
;  647          0831  1 END;                                          ! End of CLUSCOMM_SEND_ONE


                                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

72 6F 72 72 65 20 61 74 6F 75 71 20 54 53 41  00108 P.AAD:  .ASCII  \AST quota error in cluster communication\  ;
6D 6F 63 20 72 65 74 73 75 6C 63 20 6E 69 20  00117                                                            ;
               6E 6F 69 74 61 63 69 6E 75 6D  00126                                                            ;
                                   010E0028  00130 P.AAC:  .LONG   17694760                                   ;
                                   00000000' 00134         .ADDRESS P.AAD                                     ;

                                                   _QH_=           COD_WAIT_QUEUE
                                                           .EXTRN  SYS$CMKRNL

                                                           .PSECT  $CODE$,NOWRT,2

                                       00FC 00000          .ENTRY  CLUSCOMM_SEND_ONE, Save R2,R3,R4,R5,R6,R7    ; 0744
                      5E          0C   C2 00002            SUBL2   #12, SP                                      ;
                      06   0000G  CF   E8 00005            BLBS    GLOBAL_STATUS+1, 1$                          ; 0792
                      50   028C   8F   3C 0000A            MOVZWL  #652, R0                                     ; 0794
                                       04 0000F            RET                                                  ;
               FD3D   CF          00   FB 00010 1$:        CALLS   #0, CLUSCOMM_COD_ALLOCATE                    ; 0798
                      6E          50   D0 00015            MOVL    R0, COD                                      ;
                      56          6E   D0 00018            MOVL    COD, R6                                      ; 0799
                10    A6    04    AC   D0 0001B            MOVL    CSID, 16(R6)                                 ;
                18    A6    08    AC   D0 00020            MOVL    NOD, 24(R6)                                  ; 0800
```

```
                            50      0C   AC  D0 00025          MOVL     MSG_LEN, R0                                  ; 0801
                            0C           50  D1 00029          CMPL     R0, #12                                      ;
                                         03  1E 0002C          BGEQU    2$                                           ;
                            50           0C  D0 0002E          MOVL     #12, R0                                      ;
                      20    A6           50  D0 00031 2$:      MOVL     R0, 32(R6)                                   ;
                                   24    A6  9F 00035          PUSHAB   36(R6)                                       ; 0802
                                   20    A6  9F 00038          PUSHAB   32(R6)                                       ;
                   0000G    CF           02  FB 0003B          CALLS    #2, OPC$GET_VM                               ;
                            57           50  D0 00040          MOVL     R0, STATUS                                   ;
                            28           57  E9 00043          BLBC     STATUS, 3$                                   ;
        24    B6      10    BC      0C   AC  28 00046          MOVC3    MSG_LEN, @MSG_PTR, @36(R6)                   ; 0805
                   0000'    DF           66  0E 0004D          INSQUE   (R6), @ QH +4                                ; 0809
                                 0000'   CF  D6 00052          INCL     COD_REQUESTS                                 ; 0810
                                         5E  DD 00056          PUSHL    SP                                           ; 0815
                                  FD70   CF  9F 00058          PUSHAB   CLUSCOMM_DECLARE_KERNEL_AST                  ;
               00000000G    00           02  FB 0005C          CALLS    #2, SYS$CMKRNL                               ;
                            57           50  D0 00063          MOVL     R0, STATUS                                   ;
                            18           57  E8 00066          BLBS     STATUS, 5$                                   ; 0820
                            1C           57  D1 00069          CMPL     STATUS, #28                                  ; 0823
                                         0A  13 0006C          BEQL     4$                                           ;
                                         57  DD 0006E 3$:      PUSHL    STATUS                                       ; 0825
               00000000G    00           01  FB 00070          CALLS    #1, LIB$STOP                                 ;
                                             04 00077          RET                                                   ;
                                 0000'   CF  9F 00078 4$:      PUSHAB   P.AAC                                        ; 0827
                   0000G    CF           01  FB 0007C          CALLS    #1, WRITE_LOG_FILE                           ;
                            50           57  D0 00081 5$:      MOVL     STATUS, R0                                   ; 0830
                                             04 00084          RET                                                   ; 0831

; Routine Size:  133 bytes,    Routine Base:  $CODE$ + 02AE
;  649       0832  1 GLOBAL ROUTINE cluscomm_target_in_queue (cod : $ref_bblock, queue : $ref_bblock) =
;  650       0833  1
;  651       0834  1 !++
;  652       0835  1 ! Functional descripton:
;  653       0836  1 !
;  654       0837  1 !       Check to see if the CSID field in the cod is in any of the CODs in the queue.
;  655       0838  1 !       We assume that we are operating at AST level so that we do not have to worry
;  656       0839  1 !       about interlocking the queue.
;  657       0840  1 !
;  658       0841  1 ! Input:
;  659       0842  1 !       cod     pointer to a cod
;  660       0843  1 !       queue   head of a queue of CODs
;  661       0844  1 !
;  662       0845  1 ! Output:
;  663       0846  1 !       None.
;  664       0847  1 !
;  665       0848  1 ! Routine Value:
;  666       0849  1 !       number of matches in the queue
;  667       0850  1 !--
;  668       0851  1
;  669       0852  2 BEGIN                                               ! Start of cluscomm_TARGET_IN_QUEUE
;  670       0853  2
;  671       0854  2 LOCAL
;  672       0855  2     count,
;  673       0856  2     csid,
;  674       0857  2     head : $ref_bblock,
;  675       0858  2     cur : $ref_bblock;
;  676       0859  2
;  677       0860  2 !
;  678       0861  2 ! Scan the queue, counting the number of times the target appears
;  679       0862  2 !
;  680       0863  2 count = 0;
;  681       0864  2 csid = .cod [cod_l_csid];
;  682       0865  2 head = .queue;
;  683       0866  2 cur = .head [cod_l_flink];
;  684       0867  2 WHILE .cur NEQ .head                                ! Loop until we see the end
;  685       0868  2 DO
;  686       0869  3     BEGIN
;  687       0870  3     IF .csid EQL .cur [cod_l_csid]
;  688       0871  3     THEN
;  689       0872  3         count = .count + 1;
;  690       0873  3     cur = .cur [cod_l_flink];                       ! Get the next cod
;  691       0874  2     END;
;  692       0875  2
;  693       0876  2 RETURN .count;
;  694       0877  1 END;                                                ! End of cluscomm_TARGET_IN_QUEUE



                                         000C 00000          .ENTRY  CLUSCOMM_TARGET_IN_QUEUE, Save R2,R3              ; 0832
                                      52  D4 00002          CLRL    COUNT                                             ; 0863
                    50          04    AC  7D 00004          MOVQ    COD, R0                                           ; 0864
                    53          10    A0  D0 00008          MOVL    16(R0), CSID                                      ;
                    50                61  D0 0000C          MOVL    (HEAD), CUR                                       ; 0866
                    51                50  D1 0000F 1$:      CMPL    CUR, HEAD                                         ; 0867
```

```
                                        0D  13 00012          BEQL     3$                                        ;
                      10    A0          53  D1 00014          CMPL     CSID, 16(CUR)                             ; 0870
                                        02  12 00018          BNEQ     2$                                        ;
                                        52  D6 0001A          INCL     COUNT                                     ; 0872
                            50          60  D0 0001C 2$:      MOVL     (CUR), CUR                                ; 0873
                                        EE  11 0001F          BRB      1$                                        ; 0867
                            50          52  D0 00021 3$:      MOVL     COUNT, R0                                 ; 0876
                                            04 00024          RET                                                ; 0877

; Routine Size:  37 bytes,    Routine Base:  $CODE$ + 0333
;  696             0878  1 END                                    ! End of CLUSCOMM
;  697             0879  0 ELUDOM





;                                     PSECT SUMMARY
;
;       Name                        Bytes                       Attributes
;
;   $GLOBAL$                                64  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                                 856  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $PLIT$                                 312  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                             Library Statistics
;
;                                                 -------- Symbols --------        Pages        Processing
;       File                                      Total    Loaded   Percent       Mapped       Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1                18619        12         0         1000          00:01.8
;   _$255$DUA28:[OPCOM.OBJ]OPCOMLIB.L32;1          633        34         5           43          00:00.9





;                                      COMMAND QUALIFIERS

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

; Size:           856 code + 376 data bytes
; Run Time:          00:22.2
; Elapsed Time:      01:18.8
; Lines/CPU Min:       2375
; Lexemes/CPU-Min:  22248
; Memory Used:   161 pages
; Compilation Complete
```

0289 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CLUMBX
LIS

DEVICE
LIS

LOGEVENT
LIS

CLUSREPLY
LIS

CLUSUTIL
LIS

DEBUG
LIS

CLUSCOMM
LIS

CLUSMSG
LIS