L 5

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  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
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
```

```
_$2

Sym
---
SS1
SS1

SS1
SS1
SS1
SS1
SS1
SS1
SYS
SYS
SYS
TRY
UNL
WRI
```

C 12

```
**FILE**ID**CSPCALL
```

```
  CCCCCCCC    SSSSSSSS  PPPPPPPP      CCCCCCCC    AAAAAA    LL          LL
  CCCCCCCC    SSSSSSSS  PPPPPPPP      CCCCCCCC    AAAAAA    LL          LL
CC          SS          PP      PP  CC          AA      AA  LL          LL
CC          SS          PP      PP  CC          AA      AA  LL          LL
CC          SS          PP      PP  CC          AA      AA  LL          LL
CC          SS          PP      PP  CC          AA      AA  LL          LL
CC            SSSSSS    PPPPPPPP    CC          AA      AA  LL          LL
CC            SSSSSS    PPPPPPPP    CC          AA      AA  LL          LL
CC                  SS  PP          CC          AAAAAAAAAA  LL          LL
CC                  SS  PP          CC          AAAAAAAAAA  LL          LL
CC                  SS  PP          CC          AA      AA  LL          LL              ....
CC                  SS  PP          CC          AA      AA  LL          LL              ....
  CCCCCCCC  SSSSSSSS    PP            CCCCCCCC  AA      AA  LLLLLLLLLL  LLLLLLLLLL      ....
  CCCCCCCC  SSSSSSSS    PP            CCCCCCCC  AA      AA  LLLLLLLLLL  LLLLLLLLLL      ....

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

Table of contents

```
0000            .TITLE   CSPCALL          - Loadable Exec support for CSP
0000            .IDENT   'V04-000'
0000
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;++
0000    ;
0000    ; FACILITY:     VMS
0000    ;
0000    ; ABSTRACT:     Routine to call the Cluster Server Process on another node.
0000    ;
0000    ; AUTHOR:       Paul R. Beck
0000    ;
0000    ; DATE:         21-MAR-1983
0000    ;
0000    ; REVISION HISTORY:
0000    ;
0000    ;       V03-016 ADE0010         Alan D. Eldridge        18-Jul-1984
0000    ;               Consmetic (comments only) cleanup.
0000    ;
0000    ;       V03-015 ADE0008         Alan D. Eldridge        24-May-1984
0000    ;               Add bug-checks to avoid pool corruption when deallocating
0000    ;               packets.  This has proven to be a problem area.
0000    ;
0000    ;       V03-014 ADE0008         Alan D. Eldridge        22-May-1984
0000    ;               Bias ACB$W_WAIT_CNT in EXE$CSP_BRDCST while the routine is
0000    ;               referencing the master ACB copy.  This is needed since the code
0000    ;               is a referencer -- race conditions could otherwise cause the
0000    ;               ACB$V_STS_WAIT flag to be cleared prematurely by DEALL_CSD.
0000    ;
0000    ;       V03-013 ADE0007         Alan D. Eldridge        18-May-1984
0000    ;               Clear parent pointer in offspring ACB when deallocating
0000    ;               offspring.  It was being deallocated in the parent ACB.
0000    ;
0000    ;       V03-011 ADE0006         Alan D. Eldridge        26-Apr-1984
0000    ;               Erase ACB$V_WAIT at end of EXE$CSP_BRDCST if ACB$W_WAIT_CNT
```

```
0000   ;               is zero.
0000   ;
0000   ;       V03-010 ADE0005         Alan D. Eldridge        12-Apr-1984
0000   ;               Make default retry count 4 -- it was 30.
0000   ;
0000   ;       V03-010 ADE0004         Alan D. Eldridge        22-Mar-1984
0000   ;               Fix EXE$CSP_COMMAND handling of CSP$_LOCAL request.
0000   ;
0000   ;       V03-009 DWT0193         David W. Thiel          15-MAR-1984
0000   ;               Change interface to ACKMSG block transfer.
0000   ;
0000   ;       V03-008 ADE0003         Alan D. Eldridge        28-Feb-1984
0000   ;               Add support for CSP$_LOCAL call in EXE$CSP_COMMAND.
0000   ;
0000   ;       V03-007 ADE0002         Alan D. Eldridge        6-Feb-1984
0000   ;               Move CSD address to R2 in EXE$CSP_BRDCST before call to WAIT.
0000   ;               Call scheduler at IPL$_SYNCH.  Check ACB$W_WAIT_CNT before
0000   ;               clear ACB$V_STS_WAIT.
0000   ;
0000   ;       V03-006 ADE0001         Alan D. Eldridge        9-Dec-1983
0000   ;               Rewrite to use the ACKMSG of the Connection Manager rather
0000   ;               than DECnet.  Merge module CSPALLOC into this one in order
0000   ;               keep all special buffering details local to one module.
0000   ;               Add state table, etc.
0000   ;
0000   ;       V03-005 JLV0309         Jake VanNoy             5-OCT-1983
0000   ;               Check status after call to EXE$ALLOC_CSD.
0000   ;
0000   ;       V03-004 JLV0305         Jake VanNoy             29-AUG-1983
0000   ;               Add error checking to EXE$CSP_CALL call in EXE$CSP_BRDCST.
0000   ;               Call EXE$DEANONPGDSIZ instead of EXE$DEANONPAGED.
0000   ;
0000   ;       V03-003 PRB0231         Paul R. Beck    13-JUL-1983 21:33
0000   ;               Fix bugs in broadcast.
0000   ;               Change ''empty slot'' test in main routine.
0000   ;
0000   ;       V03-002 PRB0203         Paul R. Beck    7-JUN-1983 22:53
0000   ;               Fix non-PIC definition of NET0:
0000   ;               Add broadcast capability.
0000   ;
0000   ;       V03-001 PRB0164         Paul R. Beck    22-APR-1983 14:28:31
0000   ;               Add PSECT.
0000   ;
0000   ;--
```

```
0000  ;+
0000  ;
0000  ;     Future enhancements:
0000  ;
0000  ;     1.  Create a better bug-check code.  INCONSTATE is temporary.
0000  ;
0000  ;     2.  Do a better job about image rundown.
0000  ;
0000  ;     3.  What happens if a user tries to ^Y-Stop in various places (especially
0000  ;         after depleting the JIB quota and while in a a wait state allocating
0000  ;         memory).
0000  ;-
0000
0000  ;
0000  ;   Definitions
0000  ;
0000          $ACBDEF
0000          $CSBDEF
0000          $CSDDEF
0000          $CSPDEF
0000          $CDRPDEF
0000          $CLSMSGDEF
0000          $CLUBDEF
0000          $CLUBTXDEF
0000          $DYNDEF
0000          $FKBDEF
0000          $IPLDEF
0000          $JIBDEF
0000          $PCBDEF
0000          $PHDDEF
0000          $PRIDEF
0000          $RSNDEF
0000          $SBODEF
0000          $SSDEF
0000          $VADEF
0000
0000
0000
```

```
          0000   ;
          0000   ;    Macro to setup up a routine dispatch table
          0000   ;
          0000   .MACRO  $DSP_TABLE  list                          ; Setup dispatch table
          0000
          0000           .MACRO  $dspent _$dspinx, _$dspact
          0000                   .IIF GT,  <_$dspinx-_$maxinx>,  _$maxinx = _$dspinx
          0000                   .          = _$tmp + <4 * _$dspinx>
          0000                   .long   _$dspact - _$tmp
          0000           .ENDM   $dspent
          0000
          0000           _$tmp    = .
          0000           _$maxinx = 0
          0000           .IRP    a,<LIST>
          0000                   $dspent a
          0000           .ENDR
          0000
          0000           . = _$tmp + <4 * _$maxinx> + 4
          0000   .ENDM   $DSP_TABLE
          0000
          0000   ;
          0000   ;    Macro to create and fill the event state table.
          0000   ;
00000006  0000   CEV$K_STATES  =  6                                ; Number of columns in the table
FFFFFFFF  0000   CEV$_MAX_EVT  = -1                                ; Init the number of rows
00000000  0000   CEV$_EXIT     =  0                                ; Define termination event
          0000
          0000   .MACRO  $CEV  event, i,f,x,k,a,s                  ; Create state table entries
          0000                                                     ; for the specified event
          0000           CEV$_MAX_EVT = CEV$_MAX_EVT + 1           ; Bump max event value
          0000           CEV$_'event' = CEV$_MAX_EVT               ; Define circuit event symbol
          0000
          0000                   $ENT     i,_i                     ; Create table entry
          0000                   $ENT     f,_f
          0000                   $ENT     x,_x
          0000                   $ENT     k,_k
          0000                   $ENT     a,_a
          0000                   $ENT     s,_s
          0000   .ENDM   $CEV
          0000
          0000   .MACRO  $ENT     entry,def_sta                    ; Create state table entry
          0000
          0000                   _$ent = %LENGTH(entry)-1
          0000                   CEV$K_sta_. = CEV$K_sta'def_sta'; Define default next state
          0000
          0000           .IF IDN,entry,?                           ; ? => bug
          0000               .BYTE  CEV$K_sta_.                    ; Use current state
          0000               .BYTE  2                              ; Action is bug-check
          0000           .IFF
          0000               .BYTE  CEV$K_sta_%EXTRACT(0,1,entry); Setup next state
          0000               .BYTE  %EXTRACT(1,_$ent,entry)        ; Setup action routine index
          0000           .ENDC
          0000   .ENDM   $ENT
          0000
          0000
```

```
0000
0000    .MACRO  $RSP_CEV_TAB, LIST                          ; CSPMSG$K_RSP to CEV$_ mapping
0000
0000            .MACRO  $make_entry, rsp, cev
0000                    . = _$START + cspmsg$k_rsp_'rsp'
0000                    .byte cev$_'cev'
0000            .ENDM   $make_entry
0000
0000            _$start = .
0000                    .byte   0 [cspmsg$k_rsp_max+1]          ; Init table
0000            _$end   = .
0000
0000            .IRP    member,<list>                           ; Fill table
0000                    $make_entry member
0000            .ENDR
0000            .=_$end
0000
0000    .ENDM   $RSP_CEV_TAB
0000
```

```
                0000  ;
                0000  ;  Define CLSMSG format
                0000  ;
                0000  $DEFINI CSPMSG
                0000  $EQULST CSPMSG$K_RSP_,,0,1,-              ; Define response codes
                0000  <-
                0000            <NOP>,-                         ; Should never be used
                0000            <ILL>,-                         ; Illegal CSPMSG$K_RSP_xx code specified
                0000            <BUSY>,-                        ; Remote CSP is busy, try later
                0000            <NOCSP>,-                       ; No CSP process
                0000            <RO>,-                          ; Read/only completion
                0000            <RW>,-                          ; Read/write completion
                0000            <BADCSD>,-                      ; Illegal CSD detected
                0000            <ASYNERR>,-                     ; Asynchronous block transfer failure
                0000            <SYNERR>,-                      ; Synchronous block transfer failure
                0000            <MAX>,-                         ; Not a legal response code -- used
                0000  >                                         ; to mark end of list
                0000
      00000018  0000  .= CLMHDR$K_BT_LENGTH                     ; Skip over ACKMSG header
                0018
                0018  $DEF      CSPMSG$B_RSP       .BLKB 1      ; Response code
                0019  $DEF      CSPMSG$B_SPARE     .BLKB 1      ; Reserved -- used here for alignment
                001A  $DEF      CSPMSG$W_CLIENT    .BLKW 1      ; Client i.d.
                001C  $DEF      CSPMSG$L_CSD_SIZE  .BLKL 1      ; Size of CSD
      00000020  0020            CSPMSG$K_LENGTH = .             ;
                0020  $DEFEND CSPMSG                            ;
                0000
                0000  $DEFINI ACB                               ; Define our own ACB extensions
                0000                                            ;
      00000020  0000  .= <ACB$K_LENGTH + 15>  & ^C<15>          ; Goto end of normal ACB honoring normal
                0020                                            ; pool granularty
                0020            ;
                0020            ;  A copy of the AST and PID are needed in the ACB to prevent a block
                0020            ;  tranfer or a client from corrupting the ones in the CSD.
                0020            ;
                0020  $DEF      ACB$L_USER_AST   .BLKL    1     ; User's AST address
                0024  $DEF      ACB$L_USER_PID   .BLKL    1     ; User's PID
                0028  $DEF      ACB$W_WAIT_CNT   .BLKW    1     ; Used if ACB$V_STS_BCST is set
                002A                                            ; -- # of outstanding broadcasts
                002A  $DEF      ACB$W_LAST_INX   .BLKW    1     ; -  Last CSB index used
                002C  $DEF      ACB$L_PARENT     .BLKL    1     ; Used if ACB$V_STS_BCST is clear
                0030                                            ; -- 0 means no parent
                0030  $DEF      ACB$B_STA        .BLKB    1     ; CEV$K_STA_xxx code used by state table
                0031  $DEF      ACB$B_STS        .BLKB    1     ; The following:
                0032                                            ;
                0032            $VIELD  ACB,0,-                 ;
                0032                             <<STS_ASY,,M>  -; Used to determine if return was async
                0032                             ,<STS_QUE,,M>  -; Set if ACB queue header is in use
                0032                             ,<STS_WAIT,,M> -; While set, don't return to user
                0032                             ,<STS_BCST,,M> -; Set if part of broadcast
                0032                             ,<STS_PCNT,,M> -; Set if part of parent's WAIT_CNT
                0032                             >              ;
                0032  $DEF      ACB$W_RETRY      .BLKW    1     ; Retries allowed (signed value)
      00000004  0034            ACB$K_RETRY      = 4            ; Max number of retries allowed
      00000034  0034            ACB$K_CSPLNG     = .            ; Length of ACB we use
                0034  $DEFEND ACB                               ;
```

```
    0000
00000000                .PSECT  $$$200,NOPIC,EXE,QUAD,RD,WRT
    0000
    0000  CSP$BEGIN::                                       ; Starting address for reading
    0000                                                    ; map while debugging
    0000  ;
    0000  ; OWN STORAGE:
    0000  ;
    0000
    0000  ;
    0000  ;  ACB states
    0000  ;
    0000  $EQULST CEV$K_STA_,,0,1,-
    0000  <-
    0000            <I>      -; Initial:    Initial state upon being allocated.
    0000                     -;             On the 'idle CSD' queue.
    0000                     -;
    0000            <F>      -; Forking:    Waiting 1 sec. before requesting a ''warm'' CDRP.
    0000                     -;             On either some system fork or wait queue.
    0000                     -;
    0000            <X>      -; Transfer:   Undergoing block transfer.
    0000                     -;             On the 'active transfer' queue.
    0000                     -;
    0000            <K>      -; KAST:       In use as a 'special kernel' AST block.
    0000                     -;             On the PCB AST queue.
    0000                     -;
    0000            <A>      -; AST:        In use as a normal AST block.
    0000                     -;             On the PCB AST queue.
    0000                     -;
    0000            <S>      -; System:     The ACB is being processed by system CSP code.
    0000                     -;             Not on any queue.
    0000  >
    0000
    0000  CEV$AL_ACTTAB:
    0000    $DSP_TABLE -
    0000       <-
    0000            < 0, ACT_NOP>           -; Nop action routine
    0000            < 2, ACT_BUG>           -; Bugcheck
    0000            < 4, ACT_NYI>           -; Not yet implemented
    0000            <10, ACT_INSQUE>        -; Queue ACB to 'idle' queue, resignal the event
    0000            <12, ACT_REMQUE>        -; Remove ACB from current queue, resignal event
    0000            <14, ACT_REQ_ILL_BT>    -; User requested block transfer on via a CSD
    0000                                    -; that is in the wrong state
    0000            <16, ACT_REQ_DEAL>      -; User requested CSD deallocation before AST
    0000                                    -; was delivered
    0000            <18, ACT_GET_CDRP>      -; Allocate warm CDRP
    0000            <20, ACT_FORK_WAIT>     -; Put ACB on FORK and WAIT queue
    0000            <22, ACT_BLOCK_XFER>    -; Request ACKMSG block transfer
    0000            <24, ACT_SYN_ERROR>     -; Process synchronous block transfer error
    0000            <26, ACT_QUE_KAST>      -; Request Special Kernel AST
    0000            <28, ACT_QUE_AST>       -; Request Normal  Kernel AST
    0000            <32, ACT_DEALL>         -; Deallocate CSD
    0000            <34, ACT_GIVE_UP>       -; Retry count exceeded
    0000            <36, ACT_NO_AST>        -; No client AST to deliver
    0000       >
    0094
```

```
                        0094
                        0094   CEV$AW_STA_TAB:
                        0094   ;
                        0094   ;                   I    F    X    K    A    S
                        0094   ;       -----------+----+----+----+----+----+----+
                        0094   $CEV   EXIT         ?    ?    ?    ?    ?    ?    ; Exit state table processing
                        00A0   $CEV   BUG          ?    ?    ?    ?    ?    ?    ; Bug detected
                        00AC                                                    ;
                        00AC   $CEV   REQ_BT       S12  .14  .14  .14  .14  .18  ; User block-transfer request
                        00B8   $CEV   REQ_DEALL    S12  .16  .16  .16  .16  .32  ; User's deallocate CSD request
                        00C4                                                    ;
                        00C4   $CEV   NO_CDRP      ?    ?    ?    ?    ?    F20  ; No CDRP's available
                        00D0   $CEV   FORK_DONE    ?    S18  ?    ?    ?    ?    ; Back from FORK_WAIT
                        00DC   $CEV   GOT_CDRP     ?    ?    ?    ?    ?    X22  ; CDRP was allocated
                        00E8   $CEV   BT_DONE      ?    ?    K26  ?    ?    ?    ; Block-transfer done
                        00F4   $CEV   BT_SYNERR    .24  ?    I10  ?    ?    ?    ; Synchronous transfer error
                        0100                                                    ;
                        0100   $CEV   CSP_BUSY     ?    ?    F20  ?    ?    ?    ; Remote CSP is busy
                        010C   $CEV   NO_CSP       ?    ?    F20  ?    ?    ?    ; No CSP on remote node
                        0118   $CEV   GIVE_UP      ?    K34  ?    ?    ?    ?    ; Retry count exceeded
                        0124                                                    ;
                        0124   $CEV   KAST_DEL     ?    ?    ?    A28  ?    ?    ; Special Kernel AST delivered
                        0130   $CEV   AST_DEL      .    ?    ?    ?    I10  ?    ; Normal Kernel AST delivered
                        013C   $CEV   NO_AST       .36  ?    ?    I10  S32  ?    ; No user AST to deliver
                        0148   $CEV   INV_PID      S12  ?    ?    ?    ?    .32  ; Event is "invalid PID"
                        0154
                        0154
                        0154   ;
                        0154   ;  Table to map CSPMSG$K_RSP codes to CEV$_ events
                        0154   ;
                        0154   CEV$AB_RSP_CEV:
                        0154       $RSP_CEV_TAB -
                        0154          <-
                        0154             <NOP,       BUG>                  -; Not supposed to be used
                        0154             <BUSY,      CSP_BUSY>             -; Remote CSP is busy, try later
                        0154             <NOCSP,     NO_CSP>               -; No CSP process
                        0154             <RO,        BT_DONE>              -; Read/only completion
                        0154             <RW,        BT_DONE>              -; Read/write completion
                        0154             <BADCSD,    BUG>                  -; Illegal CSD detected
                        0154             <ASYNERR,   BT_DONE>              -; Asynchronous block transfer failure
                        0154             <SYNERR,    BT_SYNERR>            -; Synchronous block transfer failure
                        0154             <MAX,       BUG>                  -; Not supposed to be used
                        0154          >
                        015E
                        015E   ;
                        015E   ;  Queue headers
                        015E   ;
                        015E                    .ALIGN QUAD
                        0160
   00000000 00000000    0160   CSP$Q_ACB_IDLE:  .QUAD  0          ; ACB/CSD's allocated to some process but
                        0168                                      ; which are otherwise idle
   00000000 00000000    0168   CSP$Q_ACB_XFER:  .QUAD  0          ; ACB/CSD's with block transfer in progress
                        0170                                      ;
                  00    0170   CSP$B_RCVCSDCNT: .BYTE  0          ; Number of rcv'd CSD's being processed
                        0171                                      ; currently.
                  00    0171   CSP$B_INITED:    .BYTE  0          ; Zero only if queue's not inited
                        0172
```

```
                 0172  ;
                 0172  ;  Define CSP specific receive CDRP fields and extensions
                 0172  ;
      00000060   0172  CDRP$L_CSP_CSD =  0+CDRP$K_CM_LENGTH     ; Pointer to allocated CSD
      00000064   0172  CDRP$L_CSP_SP1 =  4+CDRP$L_CSP_CSD       ; Spare
                 0172
                 0172  $VIELD  CDRP,0,-                         ; Define CDRP$B_CLTSTS flags
                 0172  <-
                 0172          <CSP_ERROR,,M>,-                 ; ACKMSG error experienced
                 0172          <CSP_QUEUED,,M>,-                ; CSD is queued to CSP process
                 0172          <CSP_FLWCTL,,M>,-                ; CSD accounted against flow control
                 0172  >
                 0172
                 0172
             00000172          .PSECT  $$$200,EXE               ; Go to code .PSECT
                 0172
```

```
                            0172  .SBTTL  'CSP$INIT        - Init CSP data structures upon load'
                            0172  ;++
                            0172  ;
                            0172  ;  This code is called once when the CLUSTRLOA is loaded.  It init's the
                            0172  ;  queue headers.
                            0172  ;
                            0172  ;  INPUTS:         NONE
                            0172  ;
                            0172  ;  OUTPUTS:        R0       SS$_NORMAL
                            0172  ;
                            0172  ;--
                            0172  CSP$INIT::                                              ; Init data structures
          25 FC AF     E8   0172          BLBS    CSP$B_INITED,100$                       ; If LBS, we've been here
                            0176
                            0176          ASSUME  CSP$Q_ACB_XFER  EQ  8+CSP$Q_ACB_IDLE
                            0176
       50    E7 AF     9E   0176          MOVAB   CSP$Q_ACB_IDLE,R0                       ; Get queue header address
          80    60     9E   017A          MOVAB   (R0),(R0)+                              ; Setup ACB_IDLE queue header
       80    FC A0     9E   017D          MOVAB   -4(R0),(R0)+                            ;
          80    60     9E   0181          MOVAB   (R0),(R0)+                              ; Setup ACB_XFER queue header
       80    FC A0     9E   0184          MOVAB   -4(R0),(R0)+                            ;
                            0188                                                  ;
       00000088 8F     C1   0188          ADDL3   #CLUB$L_CSPFL,-                         ; Get queue header address
  50   00000000'GF          018E                  G^CLU$GL_CLUB,R0                        ;
          60    60     9E   0194          MOVAB   (R0),(R0)                               ; Setup forward link
       04 A0    60     9E   0197          MOVAB   (R0),4(R0)                              ; Setup backward link
                            019B                                                  ;
       D2 AF    01     90   019B  100$:   MOVB    #1,CSP$B_INITED                         ; Say "initialized"
          50    01     D0   019F          MOVL    #SS$_NORMAL,R0                          ; Always successful
                       05   01A2          RSB                                             ; Done
                            01A3
```

```
                           01A3  .SBTTL  'CLEAN_UP        - ACKMSG Rcv cleanup routine'
                           01A3  ;++
                           01A3  ;
                           01A3  ;   This routine is called by ACKMSG when a fatal virtual circuit error is
                           01A3  ;   encountered.  ACKMSG is going to drop this thread on the floor and will
                           01A3  ;   deallocate the CLUBTX structure.  It is up to us to eventually deallocate
                           01A3  ;   the CDRP and the CSD.
                           01A3  ;
                           01A3  ;   INPUTS:          R5       CDRP Pointer
                           01A3  ;                    R4       N/A
                           01A3  ;                    R3       CSB (or zero)
                           01A3  ;                    R2       Pointer to message stored in CLUBTX
                           01A3  ;                    R1       Pointer to extension space at end of CLUBTX (0 if none)
                           01A3  ;                    R0       Scratch
                           01A3  ;
                           01A3  ;   OUTPUTS:         ??
                           01A3  ;
                           01A3  ;--
                           01A3                 .ENABL  LSB
                           01A3  CLEAN_UP:                                                           ; Cleanup upon error
       4B A5    01    88   01A3          BISB   #CDRP$M_CSP_ERROR,CDRP$B_CLTSTS(R5)                  ; Remember error
                           01A7  CLEAN_UP1:                                                          ; Internal cleanup
    20 4B A5    01    E0   01A7          BBS    #CDRP$V_CSP_QUEUED,CDRP$B_CLTSTS(R5),100$            ; If BS, CSD is
                           01AC                                                                      ; queued to CSP
    03 4B A5    02    E5   01AC          BBCC   #CDRP$V_CSP_FLWCTL,CDRP$B_CLTSTS(R5),50$             ; If BS, accounted
                           01B1                                                                      ; against flow control
          BC AF       97   01B1          DECB   CSP$B_RCVCSDCNT                                      ; Return flow credit
    50    60 A5       D0   01B4  50$:    MOVL   CDRP$L_CSP_CSD(R5),R0                                ; Get CSD
             09       13   01B8          BEQL   70$                                                  ; If EQL, none
          60 A5       D4   01BA          CLRL   CDRP$L_CSP_CSD(R5)                                   ; Clear ptr
   00000000'GF        16   01BD          JSB    G^EXE$DEANONPAGED                                    ; Deallocate CSD
      50    55        D0   01C3  70$:    MOVL   R5,R0                                                ; Get CDRP
   00000000'GF        16   01C6          JSB    G^EXE$DEANONPAGED                                    ; Deallocate CDRP
              05           01CC  100$:   RSB                                                         ; Done
                           01CD
                           01CD                 .DSABL  LSB
                           01CD
```

```
                              01CD      .SBTTL  'CSP$DISPATCH   - Dispatch on received ACKMSG message'
                              01CD      ;++
                              01CD      ;
                              01CD      ;  INPUTS:          R5      Unitialized CDRP
                              01CD      ;                   R4      PDT address
                              01CD      ;                   R3      CSB address
                              01CD      ;                   R2      Message address
                              01CD      ;                   R1-R0   Scratch
                              01CD      ;
                              01CD      ;  OUTPUTS:         R5-R0   Garbage
                              01CD      ;
                              01CD      ;--
                              01CD                      .ENABL  LSB
                              01CD      CSP$DISPATCH::                                           ; CSP ACMKSG dispatcher
                              01CD              ;
                              01CD              ;
                              01CD              ;       Call CNX$PARTNER_INIT_CSB to allocate new BTX (R2) and to init CDRP
                              01CD              ;
                              01CD              ;
                   51    D4   01CD              CLRL    R1                                      ; No BTX extension space needed
          54    D1 AF    9E   01CF              MOVAB   CLEAN_UP,R4                             ; Address of cleanup routine
               FE2A'     30   01D3              BSBW    CNX$PARTNER_INIT_CSB                    ; Prepare for block transfer
                              01D6                                                              ; - may return to our caller
                              01D6                                                              ; - may never return if
                              01D6                                                              ;   connection breaks
                4B A5    94   01D6              CLRB    CDRP$B_CLTSTS(R5)                       ; Init client (us) status
                60 A5    D4   01D9              CLRL    CDRP$L_CSP_CSD(R5)                      ; Init CSD pointer
                64 A5    D4   01DC              CLRL    CDRP$L_CSP_SP1(R5)                      ; Init spare longword
             51    02    D0   01DF              MOVL    #CSP$_ABORT,R1                          ; Say "no CSP process"
   50   00000000'GF      D0   01E2              MOVL    G^CLU$GL_CLUB,R0                        ; Get CLUB
                   10    13   01E9              BEQL    20$                                     ; If EQL, none
              0090 C0    D5   01EB              TSTL    CLUB$L_CSPIPID(R0)                      ; CSP there ?
                   0A    13   01EF              BEQL    20$                                     ; If EQL, no
        FF7A CF    08    91   01F1              CMPB    #CSP$K_MAX_FLWCTL,CSP$B_RCVCSDCNT         ; Within limit?
                   09    1A   01F6              BGTRU   30$                                     ; If GTRU yes, okay to continue
             51    06    D0   01F8      10$:    MOVL    #CSP$_REJECT,R1                         ; "reject due to flow control"
                 010A    30   01FB      20$:    BSBW    CSP_COMMAND                             ; Issue command
                 008C    31   01FE              BRW     100$                                    ; Done
                              0201      30$:    ;
                              0201              ;
                              0201              ;       Flow control allows us to continue.  Allocate a CSD to receive the
                              0201              ;       remote request.
                              0201              ;
                              0201              ;
      3C A5    1C A2     D0   0201              MOVL    CSPMSG$L_CSD_SIZE(R2),CDRP$L_XCT_LEN(R5); Save CSD size
   51    3C A5    0C     C1   0206              ADDL3   #12,CDRP$L_XCT_LEN(R5),R1               ; Get total CSD size
        00000000'GF      16   020B              JSB     G^EXE$ALONONPAGED                       ; Allocate CSD
                E4 50    E9   0211              BLBC    R0,10$                                  ; If LBC no, treat as
                              0214                                                              ; flow control problem
              FF58 CF    96   0214              INCB    CSP$B_RCVCSDCNT                         ; Consume flow control
          4B A5    04    88   0218              BISB    #CDRP$M_CSP_FLWCTL,CDRP$B_CLTSTS(R5)    ; And mark the fact
                              021C              ;
                              021C              ;
                              021C              ;       Setup the CDRP for the block transfer, and read the remote command
                              021C              ;       into the allocated buffer.
                              021C              ;
                              021C              ;       The call to CNX$BLOCK_READ returns to our caller immediately, and
```

```
                          021C            ;       returns in-line only after the transfer completes.  If an error is
                          021C            ;       encountered and our error routine (CLEAN_UP) is called, then there
                          021C            ;       is no return in-line.
                          021C            ;
                          021C            ;
        60 A5    52   D0  021C                    MOVL    R2,CDRP$L_CSP_CSD(R5)                   ; Setup pointer
        08 A2    51   3C  0220                    MOVZWL  R1,8(R2)                                ; Setup size
              62    55   D0  0224                 MOVL    R5,(R2)                                 ; Setup CDRP pointer
              52    0C   C0  0227                 ADDL    #12,R2                                  ; Go to CSD area
  51   52    15    09   EF  022A                  EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R1               ; Get page number
   50   00000000'GF     D0  022F                  MOVL    G^MMG$GL_SPTBASE,R0                     ; Get base of SPT
       40 A5    6041    DE  0236                  MOVAL   (R0)[R1],CDRP$L_CNXSVAPTE(R5)           ; Setup SVAPTE
 44 A5   52   FE00 8F   AB  023B                  BICW3   #^C<VA$M_BYTE>,R2,CDRP$W_CNXBOFF(R5)    ; Setup BOFF
      46 A5      3C A5  D0  0242                  MOVL    CDRP$L_XCT_LEN(R5),CDRP$L_CNXBCNT(R5)   ; Setup BCNT
                 4A A5  94  0247                  CLRB    CDRP$B_CNXRMOD(R5)                      ; Setup for kernel mode
                 38 A5  D4  024A                  CLRL    CDRP$L_RBOFF(R5)                        ; Start at begining of
                 30 A5  D4  024D                  CLRL    CDRP$L_LBOFF(R5)                        ;  buffer on both sides
                 FDAD'  30  0250                  BSBW    CNX$BLOCK_READ                          ; Read remote request
                          0253            ;
                          0253            ;
                          0253            ;       We only get here if the READ completed successfully.  Pickup the
                          0253            ;       CSD, queue it, and wake the CSP process to come and get it.
                          0253            ;
                          0253            ;       If the CSP is no longer there (SCH$WAKE fails), empty the CSD queue
                          0253            ;       and send an approriate response.
                          0253            ;
                          0253            ;
   52   60 A5    0C   C1  0253                    ADDL3   #12,CDRP$L_CSP_CSD(R5),R2               ; Get the CSD
        4B A5    02   88  0258                    BISB    #CDRP$M_CSP_QUEUED,CDRP$B_CLTSTS(R5)    ; Say "queued to CSP"
                          025C                                                                    ;
                          025C      INSQUE_CLUB:                                                  ; Queue CSD to CLUB
                          025C            ;
                          025C            ;
                          025C            ;       Inputs:         R0      Scratch
                          025C            ;                       R1      Scratch
                          025C            ;                       R2      CSD pointer
                          025C            ;                       R3      Scratch
                          025C            ;                       R4      Scratch
                          025C            ;                       R5      CDRP pointer, if any
                          025C            ;
                          025C            ;
   50   00000000'GF     D0  025C                  MOVL    G^CLU$GL_CLUB,R0                        ; Get CLUB
      008C D0    62   0E  0263                    INSQUE  (R2),@CLUB$L_CSPBL(R0)                  ; Queue the CSD
      51    0090 C0   D0  0268                    MOVL    CLUB$L_CSPIPID(R0),R1                   ; Get CSP's IPID
                   09   13  026D                  BEQL    80$                                     ; If EQL, no CSP
        00000000'GF     16  026F                  JSB     G^SCH$WAKE                              ; Wake CSP
                15 50   E8  0275                  BLBS    R0,100$                                 ; If LBS, okay
   54   00000000'GF     D0  0278    80$:          MOVL    G^CLU$GL_CLUB,R4                        ; Get the CLUB
      52    0088 D4   0F  027F      90$:          REMQUE  @CLUB$L_CSPFL(R4),R2                    ; Get the CSD
                   07   1D  0284                  BVS     100$                                    ; If VS, none left
              51   02   D0  0286                  MOVL    #CSP$_ABORT,R1                          ; Setup function code
                   03   10  0289                  BSBB    EXE$CSP_COMMAND                         ; Process CSD
                   F2   11  028B                  BRB     90$                                     ; Loop
                        05  028D    100$:         RSB                                             ; Done
                          028E                            .DSABL  LSB
                          028E
                        028E  .SBTTL  'EXE$CSP_COMMAND  Receive commnad from CSP process'
                        028E  ;++
                        028E  ;
                        028E  ;  The CSP process calls this routine when it is done processing a CSD.  The
                        028E  ;  action is to conditionally send the CSD back to the requestor (if it contains
                        028E  ;  new data) and to terminate the block transfer sequence with a response
                        028E  ;  message.
                        028E  ;
                        028E  ;  This routine is also used to process the CSP$_LOCAL command.  This command
                        028E  ;  is used to pass locally generated requests to the CSP process.
                        028E  ;
                        028E  ;  INPUTS:        R4      client code       (CSP$_LOCAL only)
                        028E  ;                 R3      0                 (CSP$_LOCAL only)
                        028E  ;                         Will someday be used for message build call back
                        028E  ;                 R2      Address of CSD
                        028E  ;                 R1      Function code:
                        028E  ;
                        028E  ;                                 CSP$_ABORT  - Abort the request
                        028E  ;                                 CSP$_BADCSD - Illegal CSD structure detected
                        028E  ;                                 CSP$_DONE   - Terminate the exchange
                        028E  ;                                 CSP$_REJECT - Reject request due to flow control
                        028E  ;                                 CSP$_REPLY  - Send CSD back to requestor
                        028E  ;                                 CSP$_LOCAL  - Send local CSD to CSP
                        028E  ;
                        028E  ;                 R0      Scratch
                        028E  ;
                        028E  ;  OUTPUTS:       R2-R0   Garbage
                        028E  ;
                        028E  ;--
                        028E  EXE$CSP_COMMAND::                                        ; Command from CSP
              38   BB   028E          PUSHR   #^M<R3,R4,R5>                            ; Save regs
                        0290          DSBINT  #IPL$_SYNCH                              ; Go to proper IPL
                        0296                                                           ;
              06   10   0296          BSBB    50$                                      ; Process the command
                        0298                                                           ;
                        0298          ENBINT                                           ; Restore IPL
              38   BA   029B          POPR    #^M<R3,R4,R5>                            ; Restore regs
                   05   029D          RSB                                              ; Done
                        029E
        07    51   D1   029E  50$:    CMPL    R1,#CSP$_LOCAL                           ; "Local" request ?
              4D   12   02A1          BNEQ    CSP_COMMAND_1                            ; If NEQ, no
                        02A3          ;
                        02A3          ;
                        02A3          ;       This is a "local" request
                        02A3          ;
                        02A3          ;
  FEC8 CF     08   91   02A3          CMPB    #CSP$K_MAX_FLWCTL,CSP$B_RCVCSDCNT   ; Within limit?
              07   1A   02A8          BGTRU   70$                                      ; If GTRU, okay
  50   0294   8F   3C   02AA  60$:    MOVZWL  #SS$_REJECT,R0                           ; Tell caller we failed
              3E   11   02AF          BRB     100$                                     ; Take common exit
  51   005E   8F   3C   02B1  70$:    MOVZWL  #12+CSD$K_LENGTH,R1                      ; Setup block size
   00000000'GF     16   02B6          JSB     G^EXE$ALONONPAGED                        ; Allocate the block
          EB  50   E9   02BC          BLBC    R0,60$                                   ; If LBC, failed
                        02BF
              3F   BB   02BF          PUSHR   #^M<R0,R1,R2,R3,R4,R5>                   ; Save regs
62  51  00  6E  00  2C  02C1          MOVC5   #0,(SP),#0,R1,(R2)                       ; Zero the block
              3F   BA   02C7          POPR    #^M<R0,R1,R2,R3,R4,R5>                   ; Restore regs
```

```
                        02C9                      ;
      08 A2    51   3C  02C9          MOVZWL  R1,8(R2)                   ; Setup size, zero type
         52    0C   C0  02CD          ADDL    #12,R2                     ; Goto CSD area
         51    0C   C2  02D0          SUBL    #12,R1                     ; Reduce size
      08 A2    51   B0  02D3          MOVW    R1,8(R2)                   ; Setup size
   0A A2    65 8F   90  02D7          MOVB    #DYN$C_CLU,CSD$B_TYPE(R2)  ; Setup type
   0B A2    64 8F   90  02DC          MOVB    #DYN$C_CSD,CSD$B_SUBTYPE(R2) ; Setup subtype
      0C A2    54   B0  02E1          MOVW    R4,CSD$W_CODE(R2)          ; Enter client code
       FE87 CF      96  02E5          INCB    CSP$B_RCVCSDCNT            ; Consume flow control
          FF70      30  02E9          BSBW    INSQUE_CLUB                ; Queue the CSD
                        02EC          ;
                        02EC          ;
                        02EC          ;       *** NOTE ***
                        02EC          ;
                        02EC          ;       For a variety of reasons (CSP not there yet, CSP was there when
                        02EC          ;       CSD was queued but exitted shortly thereafter), a return with
                        02EC          ;       the low bit set does not mean that the request actually made
                        02EC          ;       it.  A return with the low bit clear does mean that it didn't.
                        02EC          ;
                        02EC          ;       A more sophisticated mechanism for status reporting will need
                        02EC          ;       to be invented if this is not adequate for future users of
                        02EC          ;       this interface (currenly only the Quorum disk thread uses this).
                        02EC          ;
         50    01   D0  02EC          MOVL    #1,R0                      ; Assume success (error at
                        02EF                                             ; this point is untrustworthy)
               05       02EF  100$:   RSB                                ; Return status to caller
                        02F0
                        02F0
                        02F0  CSP_COMMAND_1:                             ; Process CSP command
                        02F0          ;
                        02F0          ;
                        02F0          ;        If the CDRP pointer is zero, then this is a "local" CSD being
                        02F0          ;        returned -- simply restore the flow control taken and deallocate
                        02F0          ;        the CSD.  Otherwise,
                        02F0          ;
                        02F0          ;
      55    F4 A2   D0  02F0          MOVL    -12(R2),R5                 ; Get CDRP
               0E   12  02F4          BNEQ    5$                         ; If NEQ, not local CSD
       FE76 CF      97  02F6          DECB    CSP$B_RCVCSDCNT            ; Restore flow control
      50    F4 A2   9E  02FA          MOVAB   -12(R2),R0                 ; Get block address
    00000000'GF     17  02FE          JMP     G^EXE$DEANONPAGED          ; Deallocate the block
               02   8A  0304  5$:     BICB    #CDRP$M_CSP_QUEUED,-       ; CSP is done with CSD
            4B A5       0306            CDRP$B_CLTSTS(R5)                ;
                        0308  CSP_COMMAND:                               ; Process CSP command
                        0308                                             ;
  38 4B A5    00    E0  0308          BBS     #CDRP$V_CSP_ERROR,CDRP$B_CLTSTS(R5),900$ ; If BS, ACKMSG error
                        030D                                             ; occurred
                        030D          DISPATCH  R1,-
                        030D          <-
                        030D            <CSP$_DONE,    100$>,-           ; Terminate the exchange
                        030D            <CSP$_BADCSD,  300$>,-           ; Illegal CSD structure
                        030D            <CSP$_ABORT,   310$>,-           ; CSP is not there or is going
                        030D            <CSP$_REJECT,  320$>,-           ; Reject due to no flow control
                        030D            <CSP$_REPLY,   800$>,-           ; Send CSD back to requestor
                        030D          >
                        031B
                        031B                 BUG_CHECK  INCONSTATE,FATAL                   ; Unknown command
                        031F
                        031F  100$:          ;
                        031F                 ;
                        031F                 ;       Send CSD back to requestor before finishing up the block transfer
                        031F                 ;
                        031F                 ;
           FCDE'   30   031F                 BSBW       CNX$BLOCK_WRITE                    ; Send CSD back to requestor
     51      05    D0   0322                 MOVL       #CSPMSG$K_RSP_RW,R1                ; Setup response code
             12    11   0325                 BRB        810$                               ; Finish up block transfer
                        0327
                        0327                 ;
                        0327                 ;
                        0327                 ;       Miscellaneous failures
                        0327                 ;
                        0327                 ;
     51      06    D0   0327  300$:          MOVL       #CSPMSG$K_RSP_BADCSD,R1            ; Inidicate 'bad csd'
             0D    11   032A                 BRB        810$                               ; Finish up block transfer
     51      03    D0   032C  310$:          MOVL       #CSPMSG$K_RSP_NOCSP,R1             ; Indicate 'no CSP process'
             08    11   032F                 BRB        810$                               ; Finish up block transfer
     51      02    D0   0331  320$:          MOVL       #CSPMSG$K_RSP_BUSY,R1              ; Indicate 'no flow credits'
             03    11   0334                 BRB        810$                               ; Finish up block transfer
                        0336
                        0336  800$:          ;
                        0336                 ;
                        0336                 ;       Finish up the block transfer and deallocate the CDRP and CSD
                        0336                 ;       Store the response code in low byte of CDRP$L_VAL2.
                        0336                 ;
     51      04    D0   0336                 MOVL       #CSPMSG$K_RSP_RO,R1                ; Setup response code
  30 A5      51    90   0339  810$:          MOVB       R1,CDRP$L_VAL2(R5)                 ; Enter response code
4C A5     49'AF    9E   033D                 MOVAB      B^RSP_MSGBLD,CDRP$L_MSGBLD(R5)     ; Setup message build routine
           FCBB'   30   0342                 BSBW       CNX$PARTNER_RESPOND                ; Finish up block transfer
            FE5F   30   0345  900$:          BSBW       CLEAN_UP1                          ; Cleanup CDRP, CSD, etc
                   05   0348                 RSB                                           ; Done
                        0349
                        0349  RSP_MSGBLD:
                        0349                 ;
                        0349                 ;     ACKMSG calls us here to build the response message.
                        0349                 ;
                        0349                 ;     INPUTS:         R5      CDRP ptr
                        0349                 ;                     R4      PDT ptr
                        0349                 ;                     R3      CSB ptr
                        0349                 ;                     R2      Message pointer
                        0349                 ;                     R0      Scratch
                        0349                 ;
                        0349                 ;
                        0349                 ;                                                        ;
18 A2     30 A5    90   0349                 MOVB       CDRP$L_VAL2+0(R5),CSPMSG$B_RSP(R2)            ; Copy CSP response
08 A2     86 8F    90   034E                 MOVB       #<CLSMSG$K_FAC_CSP ! CLSMSG$M_RESPMSG>, -     ; Copy code/flag
                        0353                            CLSMSG$B_FACILITY(R2)
        09 A2      94   0353                 CLRB       CLSMSG$B_FUNC(R2)                             ; Copy our fct
                   05   0356                 RSB                                                      ; Done
                        0357
```

```
0357  .SBTTL  'EXE$CSP_BRDCST - Send CSP request to all nodes'
0357  ;++
0357  ;
0357  ;  Send specified message to all other nodes in the cluster.  A list is made of
0357  ;  all nodes currently in the cluster, and the message is sent to the CSP in
0357  ;  each.  A new list is then made and compared with the first; if any new nodes
0357  ;  have appeared, the message is sent to them.  This repeats until the no new
0357  ;  nodes appear.  Note that the local node is excluded from the list of
0357  ;  recipients.
0357  ;
0357  ;
0357  ;  Allocation and Deallocation of CSD's
0357  ;  ------------------------------------
0357  ;
0357  ;  EXE$ALLOC_CSD   should be used to allocate all CSD's.
0357  ;  EXE$DEALLOC_CSD should be used to deallocate all CSD's.
0357  ;
0357  ;  Because some fields in the CSD need reinitializing, and since the call to
0357  ;  EXE$DEALLOC_CSD is merely a request (the actual deallocation can only happen
0357  ;  when the CSD ''runs down''), CSD's should not be recycled by the clients, but
0357  ;  rather a fresh one should be allocated for each use.
0357  ;
0357  ;  The template CSD is allocated by the caller and this routine allocates the
0357  ;  rest.  However, the AST routine is responsible for deallocating each CSD;
0357  ;  this is true of every CSD the AST routine is called with, including the
0357  ;  template CSD.  If there is no AST routine specified, then EXE$CSP_BRDCST will
0357  ;  cause the CSD's used in the node dialogues to be automatically deallocated.
0357  ;  Note that the AST routine need not deallocate a CSD immediately -- it may
0357  ;  queue for later deallocation at normal process level.
0357  ;
0357  ;  The caller is always responsible for deallocating the template CSD as listed
0357  ;  in the table below.  Basically, if the call to this routine returns an error,
0357  ;  or if no AST is specified, then the caller should deallocate the CSD upon
0357  ;  return.  Otherwise, the AST routine should cause the CSD (in this case
0357  ;  CSD$L_CSID = -1) to be deallocated.
0357  ;
0357  ;
0357  ;
0357  ;  The CSD$L_USER_AST field
0357  ;  ------------------------
0357  ;
0357  ;  If this field is zero, then no AST's will be delivered and control is not
0357  ;  returned to the caller until the completion of the dialoque with the final
0357  ;  node.
0357  ;
0357  ;  If this field is non-zero, then control is returned to the user as soon as
0357  ;  possible.  An AST will be delivered after the completion of a dialogue with
0357  ;  each node.  The CSD address is the AST parameter.  The AST routine should
0357  ;  check the CSD$L_CSID field to determine the remote node, and CSD$Q_INT_IOSB
0357  ;  to determine the status.  Also, it may read the response data described by
0357  ;  CSD$L_RECVLEN and CSD$L_RECVOFF.
0357  ;
0357  ;  If EXE$CSP_BRDCST returns with the low bit set in R0, then an AST will be
0357  ;  delivered using the template CSD as a parameter (i.e. CSD$L_CSID=-1) after
0357  ;  completion of the dialogue with the final node.  This allows the caller to
0357  ;  know when the all of the EXE$CSP_BRDCST operations are done.
0357  ;
```

```
0357  ;     If EXE$CSP_BRDCST returns with the low bit clear in R0, then no further AST
0357  ;     will be queued to the process (those already in the queue will be delivered
0357  ;     when process state allows).  This means that the AST routine will not be
0357  ;     called with the template CSD.
0357  ;
0357  ;
0357  ;     Danger of Disabling AST's
0357  ;     -------------------------
0357  ;
0357  ;     Since the allocation of CSD's is charged against the user's BYTCNT quota,
0357  ;     and if the caller has specified an AST routine, then calling EXE$CSP_BRDCST
0357  ;     could hang the process.  This is because the quota is only returned when a
0357  ;     CSD is deallocated, and that does not happen until the AST causes to happen.
0357  ;     This also implies that the CSD should be deallocated as soon as possible
0357  ;     after the AST is delivered.
0357  ;
0357  ;     AST's may be disabled if no AST routine is specified since in that case
0357  ;     an AST does not have to be delivered before the quota is returned since the
0357  ;     CSD is deallocated in the 'Special Kernel' AST routine that is delivered
0357  ;     when the block transfer completes or fails.  Note that 'Special Kernel' AST's
0357  ;     are not disabled by the $SETAST service.
0357  ;
0357  ;
0357  ;     Waiting for Pool or Process Quota
0357  ;     ---------------------------------
0357  ;
0357  ;     When system resources or process quotas are not available, EXE$CSP_BRDCST
0357  ;     will optionally wait, depending on the setting of PCB$V_SSRWAIT, in the
0357  ;     current mode (kernel) at IPL 0.  This will allow the process to be deleted
0357  ;     (cleanup any allocated pool is eventually done when the timer ticks or some
0357  ;     block transfer completes), but will not allow the user to ''^Y, STOP'' the
0357  ;     current running image.  The later problem should be solved someday, but it
0357  ;     it is non-trivial since our caller is not the ''user'' but is some internal
0357  ;     system service code which may have resources to clean up.
0357  ;
0357  ;          NOTE:    Caller's of this routine are therefore cautioned from making
0357  ;                   this eventual solution overly difficult by calling
0357  ;                   EXE$CSP_BRDCST from awkward places.
0357  ;
0357  ;
0357  ;
0357  ;
0357  ;     In summary
0357  ;     ----------
0357  ;
0357  ;     R0's      AST           When to EXE$DEALLOC_CSD       When EXE$CSP_BRDCST
0357  ;     low bit   specified     the template CSD              returns to caller
0357  ;     -------   ---------     -------------------------     -------------------
0357  ;       LBC        no         Upon return - no further      When the error is
0357  ;                             AST's are delivered.          encountered.
0357  ;       LBC        yes        Upon return - no further      When the error is
0357  ;                             AST's are delivered.          encountered.
0357  ;       LBS        no         Upon return                   When all dialogues
0357  ;                                                           have completed.
0357  ;       LBS        yes        By the AST routine or by      As soon as possible.
0357  ;                             some action it schedules.
0357  ;
0357  ;
0357  ;
```

```
                        0357  ;
                        0357  ;
                        0357  ;       CALLING SEQUENCE:     JSB     EXE$CSP$BRDCST  at IPL 0
                        0357  ;
                        0357  ;
                        0357  ;       INPUTS:         R2      Address of template CSD which is completely filled in
                        0357  ;                               (including user data) with the exception CSD$L_CSID.
                        0357  ;
                        0357  ;       OUTPUTS:        R0      Status
                        0357  ;
                        0357  ;                       All other registers are preserved.
                        0357  ;
                        0357  ;--
                        0357  EXE$CSP_BRDCST::
           03FE 8F  BB  0357          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save volatile reg's
                        035B                                                  ;
              0216  30  035B          BSBW    COMMON_SETUP                    ; Check IPL, get ACB, etc
             7D 50  E9  035E          BLBC    R0,100$                         ; If LBC, error
          56    52  D0  0361          MOVL    R2,R6                           ; Save ptr to the template CSD
          59    54  D0  0364          MOVL    R4,R9                           ; Save ACB pointer
             28 A9  B6  0367          INCW    ACB$W_WAIT_CNT(R9)              ; Bias the wait count while
                        036A                                                  ; this routine is using the ACB
     50    028C 8F  3C  036A          MOVZWL  #SS$_NOSUCHNODE,R0              ; set up other escape code
                        036F                                                  ;
       0E A6    01  CE  036F          MNEGL   #1,CSD$L_CSID(R6)               ; Mark CSD as "template"
2A A9  00000000'GF  B0  0373          MOVW    G^CLU$GW_MAXINDEX,ACB$W_LAST_INX(R9) ; Init final CSB index
                        037B                                                  ;
                        037B  10$:    ;
                        037B          ;
                        037B          ;       Get the next CSB.  If there is one, allocate a CSD and copy the
                        037B          ;       the template to it.
                        037B          ;
                        037B          ;
                66  10  037B          BSBB    GET_NEXT_CSB                    ; Get next CSB, if any
             48 50  E9  037D          BLBC    R0,70$                          ; If LBC, we're done
       51    08 A6  3C  0380          MOVZWL  CSD$W_SIZE(R6),R1               ; Get the allocation size
    00000435'GF     16  0384          JSB     G^EXE$ALLOC_CSD                 ; Get a new CSD for this node
             3B 50  E9  038A          BLBC    R0,70$                          ; Error if LBC (no recovery)
                        038D                                                  ;
                52  DD  038D          PUSHL   R2                              ; Save its address
     62    66   51  28  038F          MOVC3   R1,(R6),(R2)                    ; Fill it in from the template
                52 8ED0 0393          POPL    R2                              ; Retrieve the CSD
                        0396                                                  ;
                        0396          ;
                        0396          ;
                        0396          ;       Make the CSP call to tranfer the CSD.
                        0396          ;
                        0396          ;
       54    CC A2  9E  0396          MOVAB   -ACB$K_CSPLNG(R2),R4            ; Get ACB
       2C A4    59  D0  039A          MOVL    R9,ACB$L_PARENT(R4)             ; Remember parent
             28 A9  B6  039E          INCW    ACB$W_WAIT_CNT(R9)              ; Account for this broadcast
       31 A4    18  88  03A1          BISB    #ACB$M_STS_BCST!-               ; Mark it as part of broadcast
                        03A5                  ACB$M_STS_PCNT,ACB$B_STS(R4)    ; and part of broadcast count
       0E A2    58  D0  03A5          MOVL    R8,CSD$L_CSID(R2)               ; Fill in CSID
    0000051E'GF     16  03A9          JSB     G^EXE$CSP_CALL                  ; Send it to its fate
             C9 50  E8  03AF          BLBS    R0,10$                          ; Loop if ok
    06 31 A4    04  E5  03B2          BBCC    #ACB$V_STS_PCNT,ACB$B_STS(R4),60$; If BC, no longer part of count
```

```
          2C A4   D4  03B7               CLRL    ACB$L_PARENT(R4)                    ; Erase pointer
          28 A9   B7  03BA               DECW    ACB$W_WAIT_CNT(R9)                  ; Account for this broadcast
       50    52   D0  03BD      60$:     MOVL    R2,R0                               ; Set for deallocation
   0000050C'GF    16  03C0               JSB     G^EXE$DEALLOC_CSD                   ; Deallocate
             B3   11  03C6               BRB     10$                                 ; Loop
                      03C8      70$:     ;
                      03C8               ;
                      03C8               ;       We're done.
                      03C8               ;
                      03C8               ;
       50    01   D0  03C8               MOVL    #SS$_NORMAL,R0                      ; Indicate success
                      03CB                                                           ;
          10 50   E9  03CB      80$:     BLBC    R0,100$                             ; If LBC, return immediately
          28 A9   B7  03CE               DECW    ACB$W_WAIT_CNT(R9)                  ; Take back this routine's
                      03D1                                                           ; reference
             05   12  03D1               BNEQ    90$                                 ; If NEQ, may need to wait
    00 31 A9 02   E5  03D3               BBCC    #ACB$V_STS_WAIT,ACB$B_STS(R9),90$ ; Else our waiting is done
          52 56   D0  03D8      90$:     MOVL    R6,R2                               ; Setup original CSD address
           016F   30  03DB               BSBW    WAIT                                ; Wait if necessary
                      03DE                                                           ;
        03FE 8F   BA  03DE      100$:    POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9>     ; Restore registers
                  05  03E2               RSB                                         ; Done
                      03E3
                      03E3
                      03E3      GET_NEXT_CSB:                                        ;
                      03E3               DSBINT  #IPL$_SCS                           ; Lock cluster database
                      03E9                                                           ;
             50   D4  03E9               CLRL    R0                                  ; Assume no new CSB's
       51 2A A9   3C  03EB               MOVZWL  ACB$W_LAST_INX(R9),R1               ; Get next index to use
             3C   13  03EF               BEQL    60$                                 ; If EQL, done
57 00000000'GF    D0  03F1               MOVL    G^CLU$GL_CLUSVEC,R7                 ; Address the cluster vector
             33   13  03F8               BEQL    60$                                 ; If EQL, none
54 00000000'GF    D0  03FA               MOVL    G^CLU$GL_CLUB,R4                    ; Get Cluster Block
             2A   13  0401               BEQL    60$                                 ; If EQL, not in cluster (?)
53 00000000'GF    3C  0403               MOVZWL  G^CLU$GW_MAXINDEX,R3                ; Get vector length counter
             21   13  040A               BEQL    60$                                 ; If EQL, none
          51 53   B1  040C               CMPW    R3,R1                               ; Compare against last index
             03   1E  040F               BGEQU   30$                                 ; If LSSU, it shrunk
          51 53   D0  0411               MOVL    R3,R1                               ; Update current index
      52   6741   D0  0414      30$:     MOVL    (R7)[R1],R2                         ; Get CSB
             10   18  0418               BGEQ    50$                                 ; If GEQ, this slot is empty
       58 4C A2   D0  041A               MOVL    CSB$L_CSID(R2),R8                   ; Get the CSID
       52 10 A4   D1  041E               CMPL    CLUB$L_LOCAL_CSB(R4),R2             ; Is this the local node?
             06   13  0422               BEQL    50$                                 ; If EQL yes, don't use it
             50   D6  0424               INCL    R0                                  ; Else, say "CSB found"
             51   B7  0426               DECW    R1                                  ; Update index for next time
             03   11  0428               BRB     60$                                 ; Exit loop
          E7 51   F5  042A      50$:     SOBGTR  R1,30$                              ; Still in the vector? Continue.
                      042D
    2A A9    51   B0  042D      60$:     MOVW    R1,ACB$W_LAST_INX(R9)               ; Update index for next time
                      0431                                                           ;
                      0431               ENBINT                                      ; Done with the vector
                  05  0434               RSB                                         ; Return
                      0435
```

```
                          0435  .SBTTL  'EXE$ALLOC_CSD  - Allocate and initialize a CSD block'
                          0435  ;++
                          0435  ;
                          0435  ;  Allocate and initialize fixed portions of CSD structure and an ACB to be
                          0435  ;  used as an internal work block.
                          0435  ;
                          0435  ;  EXE$ALLOC_CSD   should be used to allocate all CSD's.
                          0435  ;  EXE$DEALLOC_CSD should be used to deallocate all CSD's.
                          0435  ;
                          0435  ;  Because some fields in the CSD need reinitializing, and since the call to
                          0435  ;  EXE$DEALLOC_CSD is merely a request (the actual deallocation can only happen
                          0435  ;  when the CSD "runs down"), CSD's should not be recycled by the clients, but
                          0435  ;  rather a fresh one should be allocated for each use.
                          0435  ;
                          0435  ;
                          0435  ;  CALLING SEQUENCE:      JSB     EXE$ALLOC_CSD  at IPL 0
                          0435  ;
                          0435  ;  INPUTS:        R2      Scratch
                          0435  ;                 R1      Size of structure to allocate (minimum CSD$AB_DATA)
                          0435  ;                 R0      Scratch
                          0435  ;
                          0435  ;  OUTPUTS:       R2      Address of allocated structure
                          0435  ;                 R1      Size allocated
                          0435  ;                 R0      Completion status:
                          0435  ;                                 SS$_NORMAL      => normal success
                          0435  ;                                 Low bit clear   => no buffer allocated
                          0435  ;
                          0435  ;--
                          0435  EXE$ALLOC_CSD::
          50    14    D0  0435          MOVL    S^#SS$_BADPARAM,R0                      ; Assume error
                          0438  5$:     SAVIPL                                          ; Push IPL
                8E    D5  043B          TSTL    (SP)+                                   ; Was is 0 ?
                01    13  043D          BEQL    10$                                     ; If EQL, okay
                      05  043F          RSB                                             ; Else illegal IPL
                          0440                                                          ;
                38    BB  0440  10$:    PUSHR   #^M<R3,R4,R5>                           ; Save critical regs
                          0442          ;
                          0442          ;
                          0442          ;       Check BYTCNT quota, wait if necessary.  The ACB is allocated along
                          0442          ;       with the CSD block for simplicity.  BYTCNT quota is decremented for
                          0442          ;       the ACB in order to prevent a process from gobbling up too much
                          0442          ;       pool in case the CSD is small.
                          0442          ;
                          0442          ;
 00000052 8F    51    D1  0442          CMPL    R1,#CSD$AB_DATA                         ; Is the request large enough ?
                4A    1F  0449          BLSSU   60$                                     ; If LSSU, no
          51    34    C0  044B          ADDL    #ACB$K_CSPLNG,R1                        ; Add in ACB size
 54   00000000'GF     D0  044E          MOVL    G^CTL$GL_PCB,R4                         ; Get address of PCB
      00000000'GF     16  0455          JSB     G^EXE$BUFQUOPRC                         ; Wait for adequate BYTCNT quota
             2E 50    E9  045B          BLBC    R0,50$                                  ; If LBC, not enough
                          045E          ;
                          045E          ;
                          045E          ;       EXE$BUFQUOPRC put us at IPL$_ASTDEL to prevent AST's from consuming
                          045E          ;       any quota from the JIB.  Take the quota and restore IPL to 0 to
                          045E          ;       allow the call to EXE$ALLOCBUF to wait if needed without blocking
                          045E          ;       AST delivery (AST's may cause memory to be returned to pool) and
                          045E          ;       hence avoiding a deadlock.  There is no need to stay at IPL$_ASTDEL
                          045E           ;       to avoid process deallocation since we have not yet allocated any
                          045E           ;       system wide resources (such as pool).
                          045E           ;
                          045E           ;
 50   0080 C4     D0      045E                   MOVL    PCB$L_JIB(R4),R0                ; Get JIB
   20 A0   51     C2      0463                   SUBL    R1,JIB$L_BYTCNT(R0)             ; Take the quota
                          0467
           12     BB      0467           30$:    PUSHR   #^M<R1,R4>                      ; Save quota taken, PCB
  00000000'GF     16      0469                   JSB     G^EXE$ALLOCBUF                  ; Allocate the buffer
                          046F                                                           ; ...return at IPL$_ASTDEL (2)
           12     BA      046F                   POPR    #^M<R1,R4>                      ; Restore requested size, PCB
                          0471
        25 50     E8      0471                   BLBS    R0,80$                          ; If LBS, successful allocation
 13 24 A4  0A     E0      0474                   BBS     #PCB$V_SSRWAIT,PCB$L_STS(R4),50$; If BS, wait mode DISABLED
        50 03     D0      0479                   MOVL    #RSN$_NPDYNMEM,R0               ; Resource to wait for
                          047C                                                           ;
                          047C                   SETIPL  #IPL$_SYNCH                     ; SCH$RWAIT requires this
           7E     DC      047F                   MOVPSL  -(SP)                           ; PSL onto stack for SCH$RWAIT
  00000000'GF     16      0481                   JSB     G^SCH$RWAIT                     ; Wait for resource
                          0487                   SETIPL  #0                              ; Restore IPL
                          048A                                                           ;
           DB     11      048A                   BRB     30$                             ; Loop
                          048C           50$:
                          048C           ;
                          048C           ;       Error return
                          048C           ;
                          048C           ;
 52   0080 C4     D0      048C                   MOVL    PCB$L_JIB(R4),R2                ; Get JIB
   20 A2   51     C0      0491                   ADDL    R1,JIB$L_BYTCNT(R2)             ; Restore the quota taken
           52     D4      0495           60$:    CLRL    R2                              ; Invalidate buffer pointer
           6D     11      0497           70$:    BRB     100$                            ; Take common exit
                          0499           80$:
                          0499           ;
                          0499           ;       Got a buffer. Initialize the fixed portions.
                          0499           ;
                          0499           ;
           3E     BB      0499                   PUSHR   #^M<R1,R2,R3,R4,R5>             ; Protect volatile registers
 00   6E   00     2C      049B                   MOVC5   #0,(SP),#0,-                    ; Clear the front end
 62   0086 8F             049F                           #ACB$K_CSPLNG+CSD$AB_DATA,(R2)  ;
           3E     BA      04A3                   POPR    #^M<R1,R2,R3,R4,R5>             ; Restore
                          04A5           ;
                          04A5           ;
                          04A5           ;       Fill in the ACB fields as appropriate.  ACB$B_RMOD and ACB$L_PID
                          04A5           ;       must be filled in just prior to queuing the AST since it may be
                          04A5           ;       used as a fork block until then.
                          04A5           ;
                          04A5           ;       The PID must be saved in the ACB since it may not be trusted if
                          04A5           ;       saved only in the CSD, especially if the CSD is to recieve a block
                          04A5           ;       transfer.
                          04A5           ;
                          04A5           ;
                          04A5                   ASSUME  FKB$B_FIPL   EQ  ACB$B_RMOD
                          04A5                   ASSUME  FKB$L_FPC    EQ  ACB$L_PID
                          04A5
   08 A2   51     B0      04A5                   MOVW    R1,              ACB$W_SIZE(R2)     ; Setup total size
   0A A2   02     90      04A9                   MOVB    #DYN$C_ACB,      ACB$B_TYPE(R2)     ; Setup block type
   32 A2   04     B0      04AD                   MOVW    #ACB$K_RETRY,    ACB$W_RETRY(R2)    ; Setup retry count
```

```
     30 A2       00   90  04B1  1097          MOVB    #CEV$K_STA_I, ACB$B_STA(R2)          ; Initialize ACB state
  18 A2     05C4'CF   9E  04B5  1098          MOVAB   W^KAST,       ACB$L_KAST(R2)         ; Setup special-kernel AST ptr
  10 A2     05CE'CF   9E  04BB  1099          MOVAB   W^AST,        ACB$L_AST(R2)          ; Setup normal kernel AST ptr
    24 A2     60 A4   D0  04C1  1100          MOVL    PCB$L_PID(R4),ACB$L_USER_PID(R2);    Copy internal PID
              20 A2   D4  04C6  1101          CLRL                  ACB$L_USER_AST(R2);    Zero user's AST address
    14 A2     34 A2   9E  04C9  1102          MOVAB   ACB$K_CSPLNG(R2),ACB$L_ASTPRM(R2)   ; CSD address is AST parameter
                          04CE  1103                                                       ;
         52      34   C0  04CE  1104          ADDL    #ACB$K_CSPLNG,R2                     ; Advance to the CSD structure
         51      34   C2  04D1  1105          SUBL    #ACB$K_CSPLNG,R1                     ; Reduce size appropriately
                          04D4  1106
                          04D4  1107          ASSUME  CSD$B_SUBTYPE  EQ  1+CSD$B_TYPE
                          04D4  1108
  0A A2     6465 8F   B0  04D4  1109          MOVW    #<DYN$C_CSD@8>!-                     ; Fill in type/subtype
                          04DA  1110                    DYN$C_CLU,CSD$B_TYPE(R2)           ;
       08 A2     51   B0  04DA  1111          MOVW    R1,CSD$W_SIZE(R2)                    ; Save allocation size
                          04DE  1112
  42 A2     0084 C4   7D  04DE  1113          MOVQ    PCB$Q_PRIV(R4),CSD$Q_PROCPRIV(R2)    ; Copy privileges
  4A A2     00BC C4   D0  04E4  1114          MOVL    PCB$L_UIC(R4), CSD$L_PROCUIC(R2)     ; Copy UIC
    36 A2     60 A4   D0  04EA  1115          MOVL    PCB$L_PID(R4), CSD$L_IPID(R2)        ; Copy internal PID
50    00000000'GF     D0  04EF  1116          MOVL    G^CTL$GL_PHD,R0                      ; Get address of header
  4E A2     00F4 C0   D0  04F6  1117          MOVL    PHD$L_IMGCNT(R0),CSD$L_IMGCNT(R2)    ; Copy image activation count
                          04FC  1118
       54     CC A2   9E  04FC  1119          MOVAB   -ACB$K_CSPLNG(R2),R4                 ; Get ACB address
                0174  30  0500  1120          BSBW    ACT_INSQUE                           ; Queue ACB to 'idle' queue
         50      01   D0  0503  1121          MOVL    #SS$_NORMAL,R0                       ; Success
                          0506  1122          ;
                          0506  1123          ;       That's it.
                          0506  1124          ;
                          0506  1125  100$:   SETIPL  #0                                   ; Restore IPL
                 38   BA  0509  1126          POPR    #^M<R3,R4,R5>                        ; Restore regs
                      05  050B  1127          RSB                                          ; Done
                          050C  1128
                          050C  1129
```

```
                        050C  .SBTTL  'EXE$DEALLOC_CSD  Deallocate CSD or mark it for deletion'
                        050C  ;++
                        050C  ;
                        050C  ;  Deallocate CSD structure.   The deallocation is done via the PROC_EVENT
                        050C  ;  mechanism to protect against deallocating the CSD if it active on some
                        050C  ;  queue or there is a transfer in progress (there is no cancel request as
                        050C  ;  part of the ACKMSG services).  Depending upon the current state, the CSD
                        050C  ;  is either deallocated immediately or marked for delete when the CSD becomes
                        050C  ;  free.
                        050C  ;
                        050C  ;  EXE$ALLOC_CSD   should be used to allocate all CSD's.
                        050C  ;  EXE$DEALLOC_CSD should be used to deallocate all CSD's.
                        050C  ;
                        050C  ;  Because some fields in the CSD need reinitializing, and since the call to
                        050C  ;  EXE$DEALLOC_CSD is merely a request (the actual deallocation can only happen
                        050C  ;  when the CSD ''runs down''), CSD's should not be recycled by the clients, but
                        050C  ;  rather a fresh one should be allocated for each use.
                        050C  ;
                        050C  ;
                        050C  ;  CALLING SEQUENCE:     JSB      EXE$DEALLOC_CSD at IPL 0 or 2.
                        050C  ;
                        050C  ;  INPUTS:        R0      Address of CSD to deallocate
                        050C  ;                        CSD$W_SIZE(R0) = size of CSD
                        050C  ;
                        050C  ;  OUTPUTS:       R0-R3   Clobbered
                        050C  ;
                        050C  ;
                        050C  ;--
                        050C  EXE$DEALLOC_CSD::
              30    BB  050C          PUSHR   #^M<R4,R5>                      ; Save regs
                        050E                                                  ;
   54    CC A0      9E  050E          MOVAB   -ACB$K_CSPLNG(R0),R4            ; Get ACB block
         51    03   9A  0512          MOVZBL  #CEV$_REQ_DEALL,R1              ; Setup event code
            0110    30  0515          BSBW    PROC_EVENT                      ; Process the event
                        0518                                                  ;
              30    BA  0518          POPR    #^M<R4,R5>                      ; Restore regs
         50    01   D0  051A          MOVL    S^#SS$_NORMAL,R0                ; Setup return status
                    05  051D          RSB                                     ; Done
                        051E
                    051E  .SBTTL  'EXE$CSP_CALL   - Send a request message to local or remote CSP'
                    051E  ;++
                    051E  ;
                    051E  ;  Call the Cluster Server Process on another node.
                    051E  ;
                    051E  ;  A block of data (the CSD) is sent to the CSP on the target node, and
                    051E  ;  optionally recieve a response message into the same CSD.
                    051E  ;
                    051E  ;  If CSD$L_USER_AST is 0, then this routine does not return until the block
                    051E  ;  transfer has completed, or has failed.
                    051E  ;
                    051E  ;  If CSD$L_USER_AST is non-zero, then this routine returns immediately.  If
                    051E  ;  the return is with the low bit clear, then the AST will not be delivered and
                    051E  ;  the CSD should be deallocated upon return.  If the return is with the low
                    051E  ;  bit set in R0, then the AST routine should deallocate teh CSD.
                    051E  ;
                    051E  ;
                    051E  ;
                    051E  ;  R0's     AST        When to EXE$DEALLOC_CSD         When EXE$CSP_CALL
                    051E  ;  low bit  specified  the CSD                         returns to caller
                    051E  ;  -------  ---------  ------------------------        -------------------
                    051E  ;    LBC      no       Upon return - no further        When the error is
                    051E  ;                      AST's are delivered.            encountered.
                    051E  ;    LBC      yes      Upon return - no further        When the error is
                    051E  ;                      AST's are delivered.            encountered.
                    051E  ;    LBS      no       Upon return                     When all dialogues
                    051E  ;                                                      have completed.
                    051E  ;    LBS      yes      By the AST routine or by        As soon as possible.
                    051E  ;                      some action it schedules.
                    051E  ;
                    051E  ;
                    051E  ;
                    051E  ;  EXE$ALLOC_CSD   should be used to allocate all CSD's.
                    051E  ;  EXE$DEALLOC_CSD should be used to deallocate all CSD's.
                    051E  ;
                    051E  ;  Because some fields in the CSD need reinitializing, and since the call to
                    051E  ;  EXE$DEALLOC_CSD is merely a request (the actual deallocation can only happen
                    051E  ;  when the CSD "runs down"), CSD's should not be recycled by the clients, but
                    051E  ;  rather a fresh one should be allocated for each use.
                    051E  ;
                    051E  ;
                    051E  ;  CALLING SEQUENCE:    JSB     EXE$CSP_CALL at IPL 0
                    051E  ;
                    051E  ;  INPUTS:        R2      Address of CSD structure
                    051E  ;                 R0      Scratch
                    051E  ;
                    051E  ;  OUTPUTS:       R0      SS$_... status code.
                    051E  ;
                    051E  ;                 All other registers are preserved.
                    051E  ;
                    051E  ;--
                    051E  EXE$CSP_CALL::                                          ; Send request to CSP
   007E 8F    BB    051E          PUSHR   #^M<R1,R2,R3,R4,R5,R6>                  ; Save volatile registers
                    0522                                                          ;
      004F    30    0522          BSBW    COMMON_SETUP                            ; Check IPL, get ACB, etc
     20 50    E9    0525          BLBC    R0,200$                                 ; If LBC, error
                    0528          SETIPL  #IPL$_ASTDEL                            ; Go to IPL 2 to prevent AST's
```

```
                        052B            ;
                        052B            ;
                        052B            ;       Request the start of the block transfer sequence
                        052B            ;
                        052B            ;
         51      02  D0 052B            MOVL    #CEV$_REQ_BT,R1                         ; Event is 'request block xfer
         56      01  3C 052E            MOVZWL  #SS$_NORMAL,R6                          ; Initialize status register
               00F4  30 0531            BSBW    PROC_EVENT                              ; Process it
         50      56  D0 0534            MOVL    R6,R0                                   ; Pickup status
                        0537                                                            ;
                        0537   100$:    SETIPL  #0                                      ; Return to IPL 0
   09 31 A4      03  E0 053A            BBS     #ACB$V_STS_BCST,ACB$B_STS(R4),200$ ; If BS, part of "broadcast"
         06      50  E9 053F            BLBC    R0,200$                                 ; If error, then return now
      52   34    A4  9E 0542            MOVAB   ACB$K_CSPLNG(R4),R2                     ; Pickup CSD
                 05  10 0546            BSBB    WAIT                                    ; Wait if necessary
                        0548                                                            ;
          007E   8F  BA 0548   200$:    POPR    #^M<R1,R2,R3,R4,R5,R6>                  ; Restore volatile registers
                     05 054C   300$:    RSB                                             ; Return to caller
                        054D
                        054D
                        054D   WAIT:    ;
                        054D            ;
                        054D            ;       We are waiting here for the block transfer to complete so that
                        054D            ;       we can return to the user.  This is done whenever CSD$L_USER_AST
                        054D            ;       is 0.  It allows a synchronous return.
                        054D            ;
                        054D            ;
                        054D            ;       Inputs:         R4      Scratch
                        054D            ;                       R2      CSD address
                        054D            ;                       R0      SS$_NORMAL
                        054D            ;
                        054D            ;       Outputs:        R4      Garbage
                        054D            ;
                        054D            ;                       All other registers are preserved
                        054D            ;
         50      01  3C 054D            MOVZWL  #SS$_NORMAL,R0                          ; Setup return status
      54    CC   A2  9E 0550            MOVAB   -ACB$K_CSPLNG(R2),R4                    ; Get ACB
   1A 31 A4      02  E1 0554            BBC     #ACB$V_STS_WAIT,ACB$B_STS(R4),100$ ; If BC, not suspended
   54     00000000'GF  D0 0559          MOVL    G^CTL$GL_PCB,R4                         ; Get PCB
         50      01  D0 0560            MOVL    #RSN$_ASTWAIT,R0                        ; Setup wait condition
                        0563                                                            ;
                        0563            SETIPL  #IPL$_SYNCH                             ; SCH$RWAIT requires this
                 7E  DC 0566            MOVPSL  -(SP)                                   ; Put PSL on the stack
        00000000'GF  16 0568            JSB     G^SCH$RWAIT                             ; Wait for resource
                        056E            SETIPL  #0                                      ; Restore IPL
                        0571                                                            ;
              DA     11 0571            BRB     WAIT                                    ; Loop
                     05 0573   100$:    RSB                                             ; Done
                        0574
                        0574
```

```
                         0574  COMMON_SETUP:
          50   14    D0  0574          MOVL    #SS$_BADPARAM,R0                     ; Assume error
                         0577          SAVIPL                                       ; Push IPL
               8E    D5  057A          TSTL    (SP)+                                ; Was is 0 ?
               45    12  057C          BNEQ    100$                                 ; If NEQ, illegal IPL
                         057E          ;
                         057E          ;
                         057E          ;       Sanity check various fields in the CSD
                         057E          ;
                         057E          ;
                         057E          ASSUME  CSD$B_SUBTYPE  EQ  1+CSD$B_TYPE
                         057E
 0A A2    6465 8F    B1  057E          CMPW    #<DYN$C_CSD@8>!DYN$C_CLU,CSD$B_TYPE(R2) ; Right structure?
               3D    12  0584          BNEQ    100$                                 ; If NEQ, return error
     54     CC A2    9E  0586          MOVAB   -ACB$K_CSPLNG(R2),R4                 ; Pickup ACB address
            3A A2    7C  058A          CLRQ    CSD$Q_INT_IOSB(R2)                   ; Zero initial status
     53     08 A4    3C  058D          MOVZWL  ACB$W_SIZE(R4),R3                    ; Get ACB total size
          53   54    C0  0591          ADDL    R4,R3                                ; Calculate end
 51  52     16 A2    C1  0594          ADDL3   CSD$L_SENDOFF(R2),R2,R1              ; Get begining of region
     51     12 A2    C0  0599          ADDL    CSD$L_SENDLEN(R2),R1                 ; Calc end of region
          53   51    D1  059D          CMPL    R1,R3                                ; Within bounds ?
               21    1A  05A0          BGTRU   100$                                 ; If GTRU, out of bounds
 51  52     1E A2    C1  05A2          ADDL3   CSD$L_RECVOFF(R2),R2,R1              ; Get begining of region
     51     1A A2    C0  05A7          ADDL    CSD$L_RECVLEN(R2),R1                 ; Calc end of region
          53   51    D1  05AB          CMPL    R1,R3                                ; Within bounds ?
               13    1A  05AE          BGTRU   100$                                 ; If GTRU, out of bounds
                         05B0          ;
                         05B0          ;
                         05B0          ;       If the user want's an AST, let him have it without decrementing
                         05B0          ;       its AST quota.  The ACB is needed anyway as a work block, and
                         05B0          ;       the user has been charged for it via JIB$L_BYTCNT.
                         05B0          ;
                         05B0          ;
  20 A4     22 A2    D0  05B0          MOVL    CSD$A_ASTADR(R2),ACB$L_USER_AST(R4)  ; Save user AST address
               09    12  05B5          BNEQ    70$                                  ; If NEQ, continue
  04 31 A4     03    E0  05B7          BBS     #ACB$V_STS_BCST,ACB$B_STS(R4),70$    ; If BCST, never wait
     31 A4     04    88  05BC          BISB    #ACB$M_STS_WAIT,ACB$B_STS(R4)        ; Else, wait until done
          50   01    D0  05C0  70$:    MOVL    #SS$_NORMAL,R0                       ; Say "success"
                     05  05C3  100$:   RSB                                          ; Done
                         05C4
```

```
                          05C4  .SBTTL  'KAST             - Special Kernel AST entry point'
                          05C4  .SBTTL  'AST              - Normal  Kernel AST entry point'
                          05C4  ;++
                          05C4  ;
                          05C4  ;  The proper event is determined and the event processor is called.
                          05C4  ;
                          05C4  ;--
                          05C4  KAST:                                                  ; Special Kernel AST
                          05C4          ;
                          05C4          ;
                          05C4          ;       The ACB is in R5.  IPL is IPL$_ASTDEL (2).
                          05C4          ;
                          05C4          ;       R0 thru R5 may be clobbered upon return to caller
                          05C4          ;
                          05C4          ;
        52    14 A5   D0  05C4          MOVL    ACB$L_ASTPRM(R5),R2                    ; Get CSD
           51    0C   D0  05C8          MOVL    #CEV$_KAST_DEL,R1                      ; Setup event code
                 1A   10  05CB          BSBB    ASTEVT                                 ; Process event
                      05  05CD          RSB                                            ; Done
                          05CE
                          05CE  AST:                                                   ; Normal Kernel AST
                          05CE          ;
                          05CE          ;
                          05CE          ;       The ACB is the AST parameter.  IPL is 0.
                          05CE          ;
                          05CE          ;       All regs but R0,R1 must be saved/restored.
                          05CE          ;
                          05CE          ;
                    003C  05CE          .WORD   ^M<R2,R3,R4,R5>                        ; Entry mask
        52    04 AC   D0  05D0          MOVL    4(AP),R2                               ; Get CSD address
           51    0D   D0  05D4          MOVL    #CEV$_AST_DEL,R1                       ; Setup event code
                 0E   10  05D7          BSBB    ASTEVT                                 ; Do AST common processing
                 54   D5  05D9          TSTL    R4                                     ; Still have an ACB ?
                 09   13  05DB          BEQL    30$                                    ; If EQL, no
        50    20 A4   D0  05DD          MOVL    ACB$L_USER_AST(R4),R0                  ; Get AST address
                 03   13  05E1          BEQL    30$                                    ; If EQL, none
           60    6C   FA  05E3          CALLG   (AP),(R0)                              ; Call the user AST routine
                      04  05E6  30$:    RET                                            ; Done
                          05E7
        54    CC A2   9E  05E7  ASTEVT: MOVAB   -ACB$K_CSPLNG(R2),R4                   ; Get ACB address
  1B 31 A4       01   E5  05EB          BBCC    #ACB$V_STS_QUE,ACB$B_STS(R4),90$;     ACB no longer queued to PCB
           20 A4      D5  05F0          TSTL    ACB$L_USER_AST(R4)                     ; Does user want AST delivered?
                 0F   13  05F3          BEQL    50$                                    ; If EQL, no
  50   00000000'GF    D0  05F5          MOVL    G^CTL$GL_PHD,R0                        ; Get current PHD
  4E A2   00F4 C0     D1  05FC          CMPL    PHD$L_IMGCNT(R0),CSD$L_IMGCNT(R2)      ; Compare image deactivations
                 03   13  0602          BEQL    70$                                    ; If EQL, same image is running
           51    0E   D0  0604  50$:    MOVL    #CEV$_NO_AST,R1                        ; No user AST to deliver
              001E    30  0607  70$:    BSBW    PROC_EVENT                             ; Process the event
                      05  060A          RSB                                            ; Done
                          060B
                          060B  90$:    BUG_CHECK  INCONSTATE,FATAL                    ; Queued state is inconsistent
                          060F
                          060F
```

```
                         060F  .SBTTL  'PROC_EVENT_ASY - Process CSD event if process is still around'
                         060F  .SBTTL  'PROC_EVENT     - Process CSD event'
                         060F  ;+
                         060F  ;
                         060F  ;  This routine processes all CSD events and is state table driven.  Action
                         060F  ;  routines are called until the null event is detected.  Each action routine
                         060F  ;  generates a new event, which it returns in R1, and returns with the low bit
                         060F  ;  set in R0 only if the indicated state change is to be performed.
                         060F  ;
                         060F  ;
                         060F  ;  CALLING SEQUENCE:     JSB PROC_EVENT at IPL$_SYNCH or lower
                         060F  ;
                         060F  ;  INPUTS:        R5       Scratch
                         060F  ;                R4       ACB ptr
                         060F  ;                R3       Scratch
                         060F  ;                R2       Optional event parameter
                         060F  ;                R1       Standard event longword
                         060F  ;                R0       Scratch
                         060F  ;
                         060F  ;                All other registers are scratch.
                         060F  ;
                         060F  ;  OUTPUTS:       R4       Unchanged, or zero if deallocated
                         060F  ;
                         060F  ;                All other registers between R0 and R5 are clobbered
                         060F  ;
                         060F  ;-
                         060F  PROC_EVENT_ASY:                                        ; Process asynch event
        50    24 A4   3C 060F          MOVZWL  ACB$L_USER_PID(R4),R0                  ; Get process index
   52  00000000'GF    D0 0613          MOVL    G^SCH$GL_PCBVEC,R2                     ; Get address of PCB vector
           52   6240  D0 061A          MOVL    (R2)[R0],R2                            ; Get PCB itself
     60 A2    24 A4   D1 061E          CMPL    ACB$L_USER_PID(R4),PCB$L_PID(R2); Is this process still here?
                 03   13 0623          BEQL    PROC_EVENT                             ; If EQL, yes
               021C   31 0625          BRW     DEALL_CSD                              ; Else, deallocate CSD/ACB
                         0628                                                         ;
                         0628  PROC_EVENT:                                            ; Process all CSD events
                         0628          ASSUME   IPL$_SYNCH  EQ  IPL$_SCS
                         0628          DSBINT  #IPL$_SYNCH                            ; Synchronize
                         062E  10$:    ;
                         062E          ;
                         062E          ;       Find appropriate state table entry
                         062E          ;
                         062E          ;
        51        0F  D1 062E          CMPL    S^#CEV$_MAX_EVT,R1             ; Is event within range ?
                  40  1F 0631          BLSSU   200$                           ; If LSSU then bug exists
     50    51     06  C5 0633          MULL3   S^#CEV$K_STATES,R1,R0          ; Bias for current event
        53    30 A4   9A 0637          MOVZBL  ACB$B_STA(R4),R3               ; Get ACB state
           50     53  C0 063B          ADDL    R3,R0                          ; Add current state offset
   53   FA51 CF40     3E 063E          MOVAW   W^CEV$AW_STA_TAB[R0],R3 ; Address state table entry
                         0644          ;
                         0644          ;
                         0644          ;
                         0644          ;       Dispatch to the action routine with the following:
                         0644          ;
                         0644          ;       INPUTS:         R5       Scratch
                         0644          ;                       R4       ACB pointer
                         0644          ;                       R3       CSID of target system
                         0644          ;                       R2       CSD pointer
```

```
                             0644  1431             ;                    R1      Event code
                             0644  1432             ;                    R0      Scratch
                             0644  1433             ;
                             0644  1434             ;       ON RETURN:   R5      Garbage
                             0644  1435             ;                    R4      ACB pointer
                             0644  1436             ;                    R3      Garbage
                             0644  1437             ;                    R2      Garbage
                             0644  1438             ;                    R1      Next event code to chain to
                             0644  1439             ;                    R0      Low bit set to request state change
                             0644  1440             ;                            Low bit clear to inhibit state change
                             0644  1441             ;
                             0644  1442             ;
               83     9F     0644  1443             PUSHAB  (R3)+                         ; Save table address
          53   63     9A     0646  1444             MOVZBL  (R3),R3                       ; Get action routine index
   50    F9B3  CF     9E     0649  1445             MOVAB   W^CEV$AL_ACTTAB,R0            ; Get action routine table
     50    6043       C0     064E  1446             ADDL    (R0)[R3],R0                   ; Get action routine address
   52    14  A4       D0     0652  1447             MOVL    ACB$L_ASTPRM(R4),R2           ; Get the CSD
   53    0E  A2       D0     0656  1448             MOVL    CSD$L_CSID(R2),R3             ; Get the CSID
               60     16     065A  1449             JSB     (R0)                          ; Dispatch
               53   8ED0     065C  1450             POPL    R3                            ; Get next state, cleanup stack
               54     D5     065F  1451             TSTL    R4                            ; Is ACB still there ?
               0C     13     0661  1452             BEQL    100$                          ; If EQL, its been deallocated
          04   50     E9     0663  1453             BLBC    R0,50$                        ; Avoid state change if LBC
     30  A4    63     90     0666  1454             MOVB    (R3),ACB$B_STA(R4)            ; Change state
          51   00     D1     066A  1455  50$:       CMPL    S^#CEV$_EXIT,R1               ; Are we done ?
               BF     12     066D  1456             BNEQ    10$                           ; If NEQ then process next event
                             066F  1457                                                   ;
                             066F  1458  100$:      ENBINT                                ; Restore IPL
                      05     0672  1459             RSB                                   ; Done
                             0673  1460
                             0673  1461  200$:      BUG_CHECK  INCONSTATE,FATAL           ; Signal the bug
                             0677  1462
                             0677  1463
```

```
CS
Sy

$$
$$
$$
$$
$$
$$
$$
$$
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AC
AS
AS
BI
BU
```

```
                         0677  .SBTTL  'ACT_INSQUE      - Queue  ACB to CSP$Q_ACB_IDLE'
                         0677  .SBTTL  'ACT_REMQUE      - Remove ACB from current (internal) queue'
                         0677  ;+
                         0677  ;
                         0677  ; The ACB queue operation is performed.  Upon return, the event code passed
                         0677  ; in R1 is unchanged and the low bit of R0 is set.  This will force the same
                         0677  ; event to be reprocessed after the state change.
                         0677  ;
                         0677  ;
                         0677  ; INPUTS:          R4        ACB pointer
                         0677  ;                  R1        Event to be processed
                         0677  ;                  R0        Scratch
                         0677  ;
                         0677  ; OUTPUTS:         R4        Unchanged
                         0677  ;                  R1        Unchanged
                         0677  ;                  R0        Low bit set to force state change
                         0677  ;
                         0677  ;-
                         0677  ACT_INSQUE:                                             ; Put ACB on 'idle' queue
  09 31 A4    01    E2   0677          BBSS     #ACB$V_STS_QUE,ACB$B_STS(R4),10$; Mark ACB as 'queued'
     FADF CF  64    0E   067C          INSQUE   (R4),CSP$Q_ACB_IDLE               ; Remove from current queue
          50  01    D0   0681          MOVL     #1,R0                             ; Request state change
                    05   0684          RSB                                        ; Return to reprocess same event
                         0685
                         0685  10$:    BUG_CHECK  INCONSTATE,FATAL                ; Queued state is inconsistent
                         0689
                         0689  ACT_REMQUE:                                        ; Dequeue ACB and deallocate it
  07 31 A4    01    E5   0689          BBCC     #ACB$V_STS_QUE,ACB$B_STS(R4),10$; Mark ACB as 'not queued'
          54  64    0F   068E          REMQUE   (R4),R4                           ; Remove from current queue
          50  01    D0   0691          MOVL     #1,R0                             ; Request state change
                    05   0694          RSB                                        ; Return to reprocess same event
                         0695
                         0695  10$:    BUG_CHECK  INCONSTATE,FATAL                ; Queued state is inconsistent
                         0699
```

```
                                 0699  .SBTTL  'ACT_GET_CDRP   - Allocate a warm CDRP for block transfer'
                                 0699  ;+
                                 0699  ;
                                 0699  ;  INPUTS:         R5        Scratch
                                 0699  ;                  R4        ACB pointer
                                 0699  ;                  R3        CSID of target system
                                 0699  ;                  R2        CSD pointer
                                 0699  ;                  R1        Scratch
                                 0699  ;                  R0        Scratch
                                 0699  ;
                                 0699  ;  OUTPUTS:        R5        CDRP pointer if allocation was a success
                                 0699  ;                  R4        ACB pointer
                                 0699  ;                  R3        Garbage
                                 0699  ;                  R2        Garbage
                                 0699  ;                  R1        CEV$_NO_CDRP     if no CDRP was available
                                 0699  ;                            CEV$_GOT_CDRP    if CDRP allocation was successful
                                 0699  ;                  R0        Low bit set to request state change
                                 0699  ;
                                 0699  ;-
                                 0699  ACT_GET_CDRP:                                             ; Allocate warm CDRP
                                 0699          ;
                                 0699          ;       Allocate a warm CDRP and fill it in as appropriate
                                 0699          ;
                                 0699          ;
          00000000'GF     16     0699          JSB     G^CNX$ALLOC_WARMCDRP                      ; Get the CDRP
                51   04   9A     069F          MOVZBL  #CEV$_NO_CDRP,R1                          ; Assume allocation failure
                  31 50   E9     06A2          BLBC    R0,100$                                   ; If LBC, allocation failed
             52  14 A4    D0     06A5          MOVL    ACB$L_ASTPRM(R4),R2                       ; Get the CSD again
            3C A5    54   D0     06A9          MOVL    R4,CDRP$L_VAL5(R5)                        ; Save ACB address
       4C A5   06DA'CF    9E     06AD          MOVAB   W^REQ_MSGBLD,CDRP$L_MSGBLD(R5)            ; Setup message build routine
                  4A A5   94     06B3          CLRB    CDRP$B_CNXRMOD(R5)                        ; Kernel mode
                                 06B6
       50   00000000'GF   D0     06B6          MOVL    G^MMG$GL_SPTBASE,R0                       ; Get SPT base address
      51   52   15   09   EF     06BD          EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R1                 ; Get page number
          40 A5    6041   DE     06C2          MOVAL   (R0)[R1],CDRP$L_CNXSVAPTE(R5)                ; Store SVAPTE
         46 A5   08 A2    3C     06C7          MOVZWL  CSD$W_SIZE(R2),CDRP$L_CNXBCNT(R5)                ; Store BCNT
  44 A5   52    FE00 8F   AB     06CC          BICW3   #^C<VA$M_BYTE>,R2,CDRP$W_CNXBOFF(R5)             ; Store BOFF
                                 06D3          ;
                                 06D3          ;
                                 06D3          ;       Exit with proper new event code
                                 06D3          ;
                                 06D3          ;
                51   06   9A     06D3          MOVZBL  #CEV$_GOT_CDRP,R1                         ; Setup new event
                50   01   D0     06D6  100$:   MOVL    #1,R0                                     ; Request state change
                          05     06D9          RSB                                               ; Done
                                 06DA
                                 06DA  REQ_MSGBLD:
                                 06DA          ;
                                 06DA          ;       ACKMSG calls us here to build the request message.
                                 06DA          ;
                                 06DA          ;       INPUTS:         R5      CDRP ptr
                                 06DA          ;                       R4      PDT ptr
                                 06DA          ;                       R3      CSB ptr
                                 06DA          ;                       R2      Message pointer
                                 06DA          ;
                                 06DA          ;
          50    3C A5   D0  06DA  1557        MOVL    CDRP$L_VAL5(R5),R0                      ; Get ACB address
          50    14 A0   D0  06DE  1558        MOVL    ACB$L_ASTPRM(R0),R0                     ; Get the CSD again
       1C A2    08 A0   3C  06E2  1559        MOVZWL  CSD$W_SIZE(R0),CSPMSG$L_CSD_SIZE(R2)    ; Setup size
       1A A2    0C A0   B0  06E7  1560        MOVW    CSD$W_CODE(R0),CSPMSG$W_CLIENT(R2)      ; Setup client code
          08 A2    06   90  06EC  1561        MOVB    #CLSMSG$K_FAC_CSP,CLSMSG$B_FACILITY(R2) ; Tell ACKMSG it's us
             09 A2      94  06F0  1562        CLRB      CLSMSG$B_FUNC(R2)                     ; Our func code
                            06F3  1563                                                        ; - not used yet
                        05  06F3  1564        RSB                                             ; Done
                            06F4  1565
```

```
                          06F4  .SBTTL  'ACT_FORK_WAIT  - Fork and wait for up to 1 second'
                          06F4  ;+
                          06F4  ;
                          06F4  ;  INPUTS:         R5       Scratch
                          06F4  ;                  R4       ACB pointer
                          06F4  ;                  R3       CSID of target system
                          06F4  ;                  R2       CSD pointer
                          06F4  ;                  R1       Scratch
                          06F4  ;                  R0       Scratch
                          06F4  ;
                          06F4  ;  OUTPUTS:        R5       CDRP pointer if allocation was a success
                          06F4  ;                  R4       ACB pointer
                          06F4  ;                  R3       Garbage
                          06F4  ;                  R2       Garbage
                          06F4  ;                  R1       CEV$_EXIT   if okay to retry
                          06F4  ;                           CEV$_GIVEUP if retry count exceeded
                          06F4  ;                  R0       Low bit set to request state change
                          06F4  ;
                          06F4  ;  SIDE EFFECTS:            When the fork returns, PROC_EVENT is called with the
                          06F4  ;                           event CEV$_FORK_DONE
                          06F4  ;
                          06F4  ;-
                          06F4  ACT_FORK_WAIT:                                          ; Fork and wait for up to 1 sec.
         51    0B   D0    06F4          MOVL    #CEV$_GIVE_UP,R1                        ; Assume retry count exceeded
            32 A4   B7    06F7          DECW    ACB$W_RETRY(R4)                         ; Account for retry
               13   15    06FA          BLEQ    30$                                     ; If LEQ, count exceeded
                          06FC
                          06FC          ASSUME  FKB$B_FIPL  EQ  ACB$B_RMOD
                          06FC          ASSUME  FKB$L_FPC   EQ  ACB$L_PID
                          06FC          ASSUME  FKB$L_FR3   EQ  ACB$L_AST
                          06FC          ASSUME  FKB$L_FR4   EQ  ACB$L_ASTPRM
                          06FC
         55    54   D0    06FC          MOVL    R4,R5                                   ; Setup fork block address
      53    10 A5   7D    06FF          MOVQ    FKB$L_FR3(R5),R3                        ; Get ACB fields to be saved
      0B A5    08   90    0703          MOVB    #IPL$_SCS,FKB$B_FIPL(R5)                ; Setup fork IPL
               0A   10    0707          BSBB    50$                                     ; Create fork thread
         54    55   D0    0709          MOVL    R5,R4                                   ; Re-establish ACB pointer
         51    00   9A    070C          MOVZBL  #CEV$_EXIT,R1                           ; Setup next event code
         50    01   D0    070F  30$:    MOVL    #1,R0                                   ; Request state change
                    05    0712          RSB                                             ; Done
                          0713
15 31 A5       01   E2    0713  50$:    BBSS    #ACB$V_STS_QUE,ACB$B_STS(R5),90$;  Mark ACB as 'queued'
                          0718          FORK_WAIT                                       ; Fork and wait for a second
0A 31 A5       01   E5    071E          BBCC    #ACB$V_STS_QUE,ACB$B_STS(R5),90$;  Mark ACB as 'not queued'
         54    55   D0    0723          MOVL    R5,R4                                   ; Re-establish ACB pointer
         51    05   D0    0726          MOVL    #CEV$_FORK_DONE,R1                      ; Setup event
            FEE3    30    0729          BSBW    PROC_EVENT_ASY                          ; Process event if process is
                          072C                                                          ; still here, else deallocate
                          072C                                                          ; the ACB/CSD
                    05    072C          RSB                                             ; Done
                          072D
                          072D  90$:    BUG_CHECK  INCONSTATE,FATAL                     ; Queued state is inconsistent
                          0731
```

```
                         0731  .SBTTL  'ACT_REQ_ILL_BT - Request illegal block-transfer'
                         0731  .SBTTL  'ACT_BLOCK_XFER - Request ACKMSG Block Transfer'
                         0731  ;+
                         0731  ;
                         0731  ;  INPUTS:        R5      CDRP pointer
                         0731  ;                 R4      ACB pointer
                         0731  ;                 R3      CSID of target system
                         0731  ;                 R2      CSD pointer
                         0731  ;                 R1      Scratch
                         0731  ;                 R0      Scratch
                         0731  ;
                         0731  ;  OUTPUTS:       R5      Garbage
                         0731  ;                 R4      ACB pointer
                         0731  ;                 R3      Garbage
                         0731  ;                 R2      Garbage
                         0731  ;                 R1      CEV$_EXIT
                         0731  ;                         CEV$_BT_DONE
                         0731  ;                         CEV$_CSP_BUSY
                         0731  ;                 R0      Low bit set to request state change
                         0731  ;
                         0731  ;  SIDE EFFECTS:          When the fork returns, PROC_EVENT is called with the
                         0731  ;                         event CEV$_FORK_DONE
                         0731  ;
                         0731  ;-
                         0731  ACT_REQ_ILL_BT:                                          ; User requested block transfer
                         0731                                                           ; with CSD in the wrong state
  56  000002C4  8F  D0   0731          MOVL    #SS$_DEVACTIVE,R6                        ; Say 'CSD in wrong state'
            51  00  D0   0738          MOVL    S^#CEV$_EXIT,R1                          ; No further events
            50  01  90   073B          MOVB    #1,R0                                    ; Allow state transition
                    05   073E          RSB
                         073F
                         073F  ACT_BLOCK_XFER:                                          ; Request ACKMSG block transfer
                         073F          ;
                         073F          ;
                         073F          ;       CNX$BLOCK_XFER usually  returns asynchronously.  Therefore, we
                         073F          ;       must call a routine to call CNX$BLOCK_XFER so that we can return
                         073F          ;       to our caller with the correct values in the registers.
                         073F          ;
                         073F          ;
         31 A4  01  88   073F          BISB    #ACB$M_STS_ASY,ACB$B_STS(R4)             ; Mark ACB for asynch access
                54  DD   0743          PUSHL   R4                                       ; Save ACB pointer
                13  10   0745          BSBB    30$                                      ; Make request and return
                54 8ED0  0747          POPL    R4 .                                     ; Restore ACB pointer
      03 31 A4  00  E5   074A          BBCC    #ACB$V_STS_ASY,ACB$B_STS(R4),10$; If BC, CNX$BLOCK_XFER returned
                         074F                                                           ; synchronously.
            51  00  9A   074F          MOVZBL  #CEV$_EXIT,R1                            ; No further events for now
            50  01  D0   0752  10$:    MOVL    #1,R0                                    ; Request state cnange
                    05   0755          RSB                                              ; Done
                         0756
                         0756  20$:    BUG_CHECK  INCONSTATE,FATAL                      ; Queued state is inconsistent
                         075A
                         075A  30$:    ;
                         075A          ;       Request block transfer.
                         075A          ;
                         075A          ;       We are resumed after the call to BLOCK_XFER when block transfer
                         075A          ;       sequence has completed with the following registers setup:
                         075A          ;
```

```
                          075A             ;       R5      Address of CDRP
                          075A             ;       R4      Address of PDT
                          075A             ;       R3      CSB address
                          075A             ;       R2      Address of response message buffer      (if R0 has LBS)
                          075A             ;       R1      Scratch
                          075A             ;       R0      Status
                          075A             ;
                          075A             ;
F3 31 A4     01     E2    075A                 BBSS    #ACB$V_STS_QUE,ACB$B_STS(R4),10$; Mark ACB as 'queued'
  FA04 CF    64     0E    075F                 INSQUE  (R4),CSP$Q_ACB_XFER             ; Queue to 'active xfer' queue
        F899'       30    0764                 BSBW    CNX$BLOCK_XFER                  ; Do block transfer seqeuence
   54     3C A5     D0    0767                 MOVL    CDRP$L_VAL5(R5),R4              ; Get ACB pointer
E6 31 A4     01     E5    076B                 BBCC    #ACB$V_STS_QUE,ACB$B_STS(R4),20$; Mark ACB as 'not queued'
   54        64     0F    0770                 REMQUE  (R4),R4                         ; Remove from 'active xfer' list
                          0773
        0D   50     E8    0773                 BLBS    R0,50$                          ; If LBS, then no error
      51     08     D0    0776                 MOVL    #CSPMSG$K_RSP_SYNERR,R1         ; Assume synchronous error
09 31 A4     00     E0    0779                 BBS     #ACB$V_STS_ASY,ACB$B_STS(R4),60$; If BS, return was synchronous
      51     07     D0    077E                 MOVL    #CSPMSG$K_RSP_ASYNERR,R1        ; Asynchronous error
             04     11    0781                 BRB     60$                             ; Continue
   51     18 A2     9A    0783      50$:       MOVZBL  CSPMSG$B_RSP(R2),R1             ; Get the response code
   6E A4     50     7D    0787      60$:       MOVQ    R0,ACB$K_CSPLNG+CSD$Q_INT_IOSB(R4) ; Save status info
             1B     10    078B                 BSBB    DUMP_CDRP                       ; Dump CDRP using R0 status
   50     6E A4     7D    078D                 MOVQ    ACB$K_CSPLNG+CSD$Q_INT_IOSB(R4),R0 ; Recover status info
                          0791             ;
                          0791             ;
                          0791             ;       If ACB$V_STS_ASY is still set then the return is synchronous and
                          0791             ;       all we have to do, after clearing the flag, is to return and let
                          0791             ;       our caller chain to the next event since we are still in the event
                          0791             ;       processing loop.
                          0791             ;
                          0791             ;       Otherwise, we must call PROC_EVENT_ASY to check to see if the
                          0791             ;       process is still there, and if so, to process the new event.
                          0791             ;
                          0791             ;
        09   51     D1    0791                 CMPL    R1,#CSPMSG$K_RSP_MAX            ; Within range ?
             03     1B    0794                 BLEQU   70$                             ; If LEQU, okay
      51     01     90    0796                 MOVB    #CSPMSG$K_RSP_ILL,R1            ; Override with our own code
51   F9B6 CF41      9A    0799      70$:       MOVZBL  CEV$AB_RSP_CEV[R1],R1           ; Convert response to an event
03 31 A4     00     E4    079F                 BBSC    #ACB$V_STS_ASY,ACB$B_STS(R4),90$; If BS, return was synchronous
           FE68     30    07A4                 BSBW    PROC_EVENT_ASY                  ; Process event
                    05    07A7      90$:       RSB                                     ; Return
                          07A8
                          07A8
                          07A8      DUMP_CDRP:                                         ; Dump CDRP according to status
        03 50       E9    07A8                 BLBC    R0,10$                          ; If LBC, special cleanup
          F852'     31    07AB                 BRW     CNX$DEALL_WARMCDRP_CSB          ; Deallocate ACKMSG resources
                          07AE      10$:       ;
                          07AE             ;
                          07AE             ;       The following code assumes that the CDRP is "cold", that is,
                          07AE             ;       contains no associated buffer or RSPID.
                          07AE             ;
                          07AE             ;
             0C     BB    07AE                 PUSHR   #^M<R2,R3>                      ; Save regs
      50     55     D0    07B0                 MOVL    R5,R0                           ; Get address for deallocation
             55     D4    07B3                 CLRL    R5                              ; CDRP is now gone
   00000000'GF      16    07B5                 JSB     G^EXE$DEANONPAGED               ; Deallocate it
```

```
                  07BB  1735                                              ;
      0C    BA    07BB  1736          POPR     #^M<R2,R3>                 ; Restore regs
            05    07BD  1737          RSB                                 ; Done
                  07BE  1738
```

```
                    07BE  .SBTTL  'ACT_NO_AST      - No AST to deliver - deallocate CSD if broadcast'
                    07BE  .SBTTL  'ACT_GIVE_UP     - Retry count has be exhausted, give up'
                    07BE  .SBTTL  'ACT_QUE_KAST    - Queue Special Kernel AST to process'
                    07BE  .SBTTL  'ACT_QUE_AST     - Queue Normal  Kernel AST to process'
                    07BE  ;+
                    07BE  ;
                    07BE  ;  Come here when the Block transfer has completed or failed.
                    07BE  ;
                    07BE  ;
                    07BE  ;  INPUTS:        R5          Scratch
                    07BE  ;                 R4          ACB pointer
                    07BE  ;                 R3          CSID of target system
                    07BE  ;                 R2          CSD pointer
                    07BE  ;                 R1          Scratch
                    07BE  ;                 R0          Scratch
                    07BE  ;
                    07BE  ;  OUTPUTS:       R5          Garbage
                    07BE  ;                 R4          ACB pointer
                    07BE  ;                 R3          Garbage
                    07BE  ;                 R2          Garbage
                    07BE  ;                 R1          CEV$_EXIT
                    07BE  ;                 R0          Low bit set to request state change
                    07BE  ;
                    07BE  ;-
                    07BE                  .ENABL  LSB
                    07BE  ACT_NO_AST:                                            ; No AST to deliver
   31 A4    04   8A 07BE          BICB    #ACB$M_STS_WAIT,ACB$B_STS(R4)          ; No need to wait any longer
34 31 A4    03   E1 07C2          BBC     #ACB$V_STS_BCST,ACB$B_STS(R4),30$      ; If BC, not part of broadcast
      51    03   D0 07C7          MOVL    #CEV$_REQ_DEALL,R1                     ; Else, request deallocation
            32   11 07CA          BRB     40$                                    ; Continue
                    07CC                                                         ;
                    07CC  ACT_GIVE_UP:                                           ; Retry count exceeded
    022C 8F      3C 07CC          MOVZWL  #SS$_TIMEOUT,-                         ; Setup status
      6E A4         07D0           ACB$R_CSPLNG+CSD$Q_INT_IOSB(R4)               ;
2B 31 A4    01   E4 07D2          BBSC    #ACB$V_STS_QUE,ACB$B_STS(R4),50$; Make sure ACB is not queued
                    07D7                                                         ;
                    07D7  ACT_QUE_KAST:                                          ; Queue Special Kernel AST
0C A4    24 A4   D0 07D7          MOVL    ACB$L_USER_PID(R4),ACB$L_PID(R4); Copy internal PID
0B A4    A0 8F   90 07DC          MOVB    #ACB$M_KAST!-                          ; Mark as 'special kernel'
                    07E1           ACB$M_NODELETE,ACB$B_RMOD(R4)                 ; and don't delete ACB     ,
      52    01   D0 07E1          MOVL    #PRI$_IOCOM,R2                         ; Setup priority increment class
            02   11 07E4          BRB     10$                                    ; Continue
                    07E6                                                         ;
                    07E6  ACT_QUE_AST:                                           ; Queue Normal Kernel AST
            52   D4 07E6          CLRL    R2                                     ; Use null priority inc. class
15 31 A4    01   E2 07E8  10$:    BBSS    #ACB$V_STS_QUE,ACB$B_STS(R4),50$; ACB will be queued to the PCB
      55    54   D0 07ED          MOVL    R4,R5                                  ; Setup ACB pointer
            54   DD 07F0          PUSHL   R4                                     ; Save ACB address
  00000000'GF    16 07F2          JSB     G^SCH$QAST                             ; Queue the AST
            54 8ED0 07F8          POPL    R4                                     ; Restore ACB address
      51    00   D0 07FB  30$:    MOVL    #CEV$_EXIT,R1                          ; No new events
      50    01   D0 07FE  40$:    MOVL    #1,R0                                  ; Request state change
                 05 0801          RSB                                            ; Done
                    0802
                    0802  50$:    BUG_CHECK  INCONSTATE,FATAL                    ; Queued state is inconsistent
                    0806
                    0806                  .DSABL  LSB
```

```
                      0806  .SBTTL  'ACT_SYN_ERROR  - Synchronous block transfer error'
                      0806  ;+
                      0806  ;
                      0806  ;   INPUTS:         R5        Scratch
                      0806  ;                   R4        ACB pointer
                      0806  ;                   R3        CSID of target system
                      0806  ;                   R2        CSD pointer
                      0806  ;                   R1        Scratch
                      0806  ;                   R0        Scratch
                      0806  ;
                      0806  ;   OUTPUTS:        R5        Garbage
                      0806  ;                   R4        ACB pointer
                      0806  ;                   R3        Garbage
                      0806  ;                   R2        Garbage
                      0806  ;                   R1        CEV$_EXIT
                      0806  ;                   R0        Low bit set to request state change
                      0806  ;
                      0806  ;-
                      0806  ACT_SYN_ERROR:                                          ; Synchronous block transfer err
         20 A4    D4  0806          CLRL    ACB$L_USER_AST(R4)                      ; No AST delivery if synchronous
                      0809                                                          ; error return
    56    3A A2    3C  0809          MOVZWL  CSD$W_IOSB_STAT(R2),R6                  ; Setup status to be returned
                      080D                                                          ; to EXE$CALL_CSP
       51    00    9A  080D          MOVZBL  #CEV$_EXIT,R1                           ; No further events
       50    01    D0  0810          MOVL    #1,R0                                   ; Request state change
                 05  0813          RSB                                             ; Done
                      0814
```

```
                         0814  1826 .SBTTL  'ACT_REQ_DEAL   - Illegal user deallocation request'
                         0814  1827 ;+
                         0814  1828 ;
                         0814  1829 ;  INPUTS:        R5           Scratch
                         0814  1830 ;                 R4           ACB pointer
                         0814  1831 ;                 R3           CSID of target system
                         0814  1832 ;                 R2           CSD pointer
                         0814  1833 ;                 R1           Scratch
                         0814  1834 ;                 R0           Scratch
                         0814  1835 ;
                         0814  1836 ;  OUTPUTS:       R5           Garbage
                         0814  1837 ;                 R4           0 to indicate CSD has been deallocated
                         0814  1838 ;                 R3           Garbage
                         0814  1839 ;                 R2           Garbage
                         0814  1840 ;                 R1           CEV$_EXIT
                         0814  1841 ;                 R0           Low bit clear to avoid state change
                         0814  1842 ;
                         0814  1843 ;-
                         0814  1844 ACT_REQ_DEAL:                                      ; Illegal user dealloc. request
                         0814  1845           ;
                         0814  1846           ;
                         0814  1847           ;     The user has requested that the CSD be deallocated while the CSD
                         0814  1848           ;     is in the wrong state (e.g., a block transfer is in progress).
                         0814  1849           ;     Since this is a user error just prevent user AST notification and
                         0814  1850           ;     let the transfer run its course.  When the transfer completes and
                         0814  1851           ;     the 'special kernel' AST is delivered, return quotas and deallocate
                         0814  1852           ;     the CSD.
                         0814  1853           ;
                         0814  1854           ;     Note:
                         0814  1855           ;        This action routine could be rewritten to bug-check, but since
                         0814  1856           ;        since not all users have been updated yet to request AST
                         0814  1857           ;        notification, and since there is no adequate mechanism yet in
                         0814  1858           ;        place to detect image run-down (an interactive user may have
                         0814  1859           ;        Control-Y'd and issued a STOP) we do the next best thing:  stop
                         0814  1860           ;        the user AST delivery and return quota's when the operation
                         0814  1861           ;        actual completes.  The choice of when to return quota's is not
                         0814  1862           ;        perfect, but the choice was made since it may save the system
                         0814  1863           ;        from running out of pool at the expense of the process possibly
                         0814  1864           ;        running out of quota.
                         0814  1865           ;
                         0814  1866           ;        Eventually, each client must be updated to request AST
                         0814  1867           ;        notification even if it is not receiving any response.  Also, an
                         0814  1868           ;        image run-down hook is needed and a hook in ACKMSG to abort a
                         0814  1869           ;        transfer in progress.
                         0814  1870           ;
                         0814  1871           ;
      22 A2      D4      0814  1872           CLRL    CSD$A_ASTADR(R2)              ; Prevent AST notification
      20 A4      D4      0817  1873           CLRL    ACB$L_USER_AST(R4)            ; Here too
   51    00      9A      081A  1874           MOVZBL  #CEV$_EXIT,R1                 ; No further events
   50    01      D0      081D  1875           MOVL    #1,R0                         ; Allow state change
                 05      0820  1876           RSB                                   ; Done
                         0821  1877
```

```
                             0821  .SBTTL  'ACT_DEALL       - Deallocate CSD, return quotas'
                             0821  ;+
                             0821  ;
                             0821  ;   INPUTS:         R5      Scratch
                             0821  ;                   R4      ACB pointer
                             0821  ;                   R3      CSID of target system
                             0821  ;                   R2      CSD pointer
                             0821  ;                   R1      Scratch
                             0821  ;                   R0      Scratch
                             0821  ;
                             0821  ;   OUTPUTS:        R5      Garbage
                             0821  ;                   R4      0 to indicate CSD has been deallocated
                             0821  ;                   R3      Garbage
                             0821  ;                   R2      Garbage
                             0821  ;                   R1      CEV$_EXIT
                             0821  ;                   R0      Low bit clear to avoid state change
                             0821  ;
                             0821  ;
                             0821  ;-
                             0821  ACT_DEALL:                                                 ; Deallocate CSD, return quota
        50    24 A4     3C   0821          MOVZWL  ACB$L_USER_PID(R4),R0                      ; Get process index
51    00000000'GF       D0   0825          MOVL    G^SCH$GL_PCBVEC,R1                         ; Get address of PCB vector
        50    6140      D0   082C          MOVL    (R1)[R0],R0                                ; Get PCB itself
   60 A0    24 A4       D1   0830          CMPL    ACB$L_USER_PID(R4),PCB$L_PID(R0)           ; Is this process still here?
                0D      12   0835          BNEQ    DEALL_CSD                                  ; If NEQ, no
                             0837                                                             ;
        51    08 A4     3C   0837          MOVZWL  ACB$W_SIZE(R4),R1                          ; Get quota taken
      50    0080 C0     D0   083B          MOVL    PCB$L_JIB(R0),R0                           ; Get JIB
     20 A0      51      C0   0840          ADDL    R1,JIB$L_BYTCNT(R0)                        ; Return quota
                             0844                                                             ;
                             0844  DEALL_CSD:                                                 ; Deallocate CSD/ACB
   17 31 A4     04      E5   0844          BBCC    #ACB$V_STS_PCNT,ACB$B_STS(R4),30$          ; If BC, not part of Bcst count
        50    2C A4     D0   0849          MOVL    ACB$L_PARENT(R4),R0                        ; Get parent ACB, if any
              2C A4     D4   084D          CLRL    ACB$L_PARENT(R4)                           ; Erase pointer
        02    0A A0     91   0850          CMPB    ACB$B_TYPE(R0),#DYN$C_ACB                  ; Check packet type
                27      12   0854          BNEQ    200$                                       ; If NEQ, pool corruption
              28 A0     B7   0856          DECW    ACB$W_WAIT_CNT(R0)                         ; Decrement the wait count
                05      12   0859          BNEQ    30$                                        ; If NEQ, not done yet
   00 31 A0     02      E5   085B          BBCC    #ACB$V_STS_WAIT,ACB$B_STS(R0),30$          ; If BC, not waiting
        50      54      D0   0860  30$:    MOVL    R4,R0                                      ; Get address for deallocation
                54      D4   0863          CLRL    R4                                         ; Erase offical pointer
        02    0A A0     91   0865          CMPB    ACB$B_TYPE(R0),#DYN$C_ACB                  ; Check packet type
                12      12   0869          BNEQ    200$                                       ; If NEQ, pool corruption
              28 A0     B5   086B          TSTW    ACB$W_WAIT_CNT(R0)                         ; Any lingering references?
                11      12   086E          BNEQ    210$                                       ; If NEQ yes, bug
      00000000'GF       16   0870          JSB     G^EXE$DEANONPAGED                          ; Deallocate the block
                             0876                                                             ;
        50      01      D0   0876          MOVL    S^#SS$_NORMAL,R0                           ; Why not
        51      00      9A   0879          MOVZBL  #CEV$_EXIT,R1                              ; No further events
                        05   087C          RSB                                                ; Done
                             087D
                             087D  200$:   BUG_CHECK  INCONSTATE,FATAL                        ; ACB$B_TYPE is wrong
                             0881  210$:   BUG_CHECK  INCONSTATE,FATAL                        ; WAIT_CNT non-zero
                             0885
```

```
                  0885  .SBTTL  'ACT_BUG         - Bugcheck failure'
                  0885  .SBTTL  'ACT_NYI         - Not-yet-implemented error'
                  0885  .SBTTL  'ACT_NOP         - No-operation'
                  0885  ;+
                  0885  ;
                  0885  ;
                  0885  ;    INPUTS:     R5      ACB ptr or zero
                  0885  ;
                  0885  ;    OUTPUTS:    R5      Unchanged
                  0885  ;
                  0885  ;
                  0885  ;-
                  0885  ACT_BUG:
                  0885          BUG_CHECK  INCONSTATE,FATAL      ; Signal the bug
                  0889  ACT_NYI:
                  0889          BUG_CHECK  INCONSTATE,FATAL      ; Signal the bug
                  088D
                  088D  ACT_NOP:                                 ; Nop action routine
51    00    D0    088D          MOVL    S^#CEV$_EXIT,R1          ; No further events
50    01    90    0890          MOVB    #1,R0                    ; Allow state transition
            05    0893          RSB
                  0894
                  0894
                  0894  .END
```

```
$$BASE                         = 00000002                CDRP$B_CLTSTS                  = 0000004B
$$DISPL                        = 00000007                CDRP$B_CNXRMOD                 = 0000004A
$$GENSW                        = 00000001                CDRP$K_CM_LENGTH               = 00000060
$$HIGH                         = 00000006                CDRP$L_CNXBCNT                 = 00000046
$$LIMIT                        = 00000004                CDRP$L_CNXSVAPTE               = 00000040
$$LOW                          = 00000002                CDRP$L_CSP_CSD                 = 00000060
$$MNSW                         = 00000001                CDRP$L_CSP_SP1                 = 00000064
$$MXSW                         = 00000001                CDRP$L_LBOFF                   = 00000030
ACB$B_RMOD                     = 0000000B                CDRF$L_MSGBLD                  = 0000004C
ACB$B_STA                        00000030                CDRP$L_RBOFF                   = 00000038
ACB$B_STS                        00000031                CDRP$L_VAL2                    = 00000030
ACB$B_TYPE                     = 0000000A                CDRP$L_VAL5                    = 0000003C
ACB$K_CSPLNG                   = 00000034                CDRP$L_XCT_LEN                 = 0000003C
ACB$K_LENGTH                   = 0000001C                CDRP$M_CSP_ERROR               = 00000001
ACB$K_RETRY                    = 00000004                CDRP$M_CSP_FLWCTL              = 00000004
ACB$L_AST                      = 00000010                CDRP$M_CSP_QUEUED              = 00000002
ACB$L_ASTPRM                   = 00000014                CDRP$V_CSP_ERROR               = 00000000
ACB$L_KAST                     = 00000018                CDRP$V_CSP_FLWCTL              = 00000002
ACB$L_PARENT                     0000002C                CDRP$V_CSP_QUEUED              = 00000001
ACB$L_PID                      = 0000000C                CDRP$W_CNXBOFF                 = 00000044
ACB$L_USER_AST                   00000020                CEV$AB_RSP_CEV                   00000154 R     02
ACB$L_USER_PID                   00000024                CEV$AL_ACTTAB                    00000000 R     02
ACB$M_KAST                     = 00000080                CEV$AW_STA_TAB                   00000094 R     02
ACB$M_NODELETE                 = 00000020                CEV$K_STATES                   = 00000006
ACB$M_STS_ASY                  = 00000001                CEV$K_STA_.                    = 00000005
ACB$M_STS_BCST                 = 00000008                CEV$K_STA_A                    = 00000004
ACB$M_STS_PCNT                 = 00000010                CEV$K_STA_F                    = 00000001
ACB$M_STS_WAIT                 = 00000004                CEV$K_STA_I                    = 00000000
ACB$V_STS_ASY                  = 00000000                CEV$K_STA_K                    = 00000003
ACB$V_STS_BCST                 = 00000003                CEV$K_STA_S                    = 00000005
ACB$V_STS_PCNT                 = 00000004                CEV$K_STA_X                    = 00000002
ACB$V_STS_QUE                  = 00000001                CEV$_AST_DEL                   = 0000000D
ACB$V_STS_WAIT                 = 00000002                CEV$_BT_DONE                   = 00000007
ACB$W_LAST_INX                   0000002A                CEV$_BT_SYNERR                 = 00000008
ACB$W_RETRY                      00000032                CEV$_BUG                       = 00000001
ACB$W_SIZE                     = 00000008                CEV$_CSP_BUSY                  = 00000009
ACB$W_WAIT_CNT                   00000028                CEV$_EXIT                      = 00000000
ACT_BLOCK_XFER                   0000073F R     02       CEV$_FORK_DONE                 = 00000005
ACT_BUG                          00000885 R     02       CEV$_GIVE_UP                   = 0000000B
ACT_DEALL                        00000821 R     02       CEV$_GOT_CDRP                  = 00000006
ACT_FORK_WAIT                    000006F4 R     02       CEV$_INV_PID                   = 0000000F
ACT_GET_CDRP                     00000699 R     02       CEV$_KAST_DEL                  = 0000000C
ACT_GIVE_UP                      000007CC R     02       CEV$_MAX_EVT                   = 0000000F
ACT_INSQUE                       00000677 R     02       CEV$_NO_AST                    = 0000000E
ACT_NOP                          0000088D R     02       CEV$_NO_CDRP                   = 00000004
ACT_NO_AST                       000007BE R     02       CEV$_NO_CSP                    = 0000000A
ACT_NYI                          00000889 R     02       CEV$_REQ_BT                    = 00000002
ACT_QUE_AST                      000007E6 R     02       CEV$_REQ_DEALL                 = 00000003
ACT_QUE_KAST                     000007D7 R     02       CLEAN_UP                         000001A3 R     02
ACT_REMQUE                       00000689 R     02       CLEAN_UP1                        000001A7 R     02
ACT_REQ_DEAL                     00000814 R     02       CLMHDR$K_BT_LENGTH             = 00000018
ACT_REQ_ILL_BT                   00000731 R     02       CLSMSG$B_FACILITY              = 00000008
ACT_SYN_ERROR                    00000806 R     02       CLSMSG$B_FUNC                  = 00000009
AST                              000005CE R     02       CLSMSG$K_FAC_CSP               = 00000006
ASTEVT                           000005E7 R     02       CLSMSG$M_RESPMSG               = 00000080
BIT...                         = 00000003                CLU$GL_CLUB                      ********   X 02
BUG$_INCONSTATE                  ********   X   02       CLU$GL_CLUSVEC                   ********   X 02
```

```
CLU$GW_MAXINDEX                   ********   X   02     CSPMSG$K_RSP_SYNERR              = 00000008
CLUB$L_CSPBL                    = 0000008C              CSPMSG$L_CSD_SIZE                  0000001C
CLUB$L_CSPFL                    = 00000088              CSPMSG$W_CLIENT                    0000001A
CLUB$L_CSPIPID                  = 00000090              CSP_COMMAND                        00000308 R      02
CLUB$L_LOCAL_CSB                = 00000010              CSP_COMMAND_1                      000002F0 R      02
CNX$ALLOC_WARMCDRP                ********   X   02     CTL$GL_PCB                         ********   X    02
CNX$BLOCK_READ                    ********   X   02     CTL$GL_PHD                         ********   X    02
CNX$BLOCK_WRITE                   ********   X   02     DEALL_CSD                          00000844 R      02
CNX$BLOCK_XFER                    ********   X   02     DUMP_CDRP                          000007A8 R      02
CNX$DEALL_WARMCDRP_CSB            ********   X   02     DYN$C_ACB                        = 00000002
CNX$PARTNER_INIT_CSB              ********   X   02     DYN$C_CLU                        = 00000065
CNX$PARTNER_RESPOND               ********   X   02     DYN$C_CSD                        = 00000064
COMMON_SETUP                      00000574 R     02     EXE$ALLOCBUF                       ********   X    02
CSB$L_CSID                      = 0000004C              EXE$ALLOC_CSD                      00000435 RG     02
CSD$AB_DATA                     = 00000052              EXE$ALONONPAGED                    ********   X    02
CSD$A_ASTADR                    = 00000022              EXE$BUFQUOPRC                      ********   X    02
CSD$B_SUBTYPE                   = 0000000B              EXE$CSP_BRDCST                     00000357 RG     02
CSD$B_TYPE                      = 0000000A              EXE$CSP_CALL                       0000051E RG     02
CSD$K_LENGTH                    = 00000052              EXE$CSP_COMMAND                    0000028E RG     02
CSD$L_CSID                      = 0000000E              EXE$DEALLOC_CSD                    0000050C RG     02
CSD$L_IMGCNT                    = 0000004E              EXE$DEANONPAGED                    ********   X    02
CSD$L_IPID                      = 00000036              EXE$FORK_WAIT                      ********   X    02
CSD$L_PROCUIC                   = 0000004A              FKB$B_FIPL                       = 0000000B
CSD$L_RECVLEN                   = 0000001A              FKB$L_FPC                        = 0000000C
CSD$L_RECVOFF                   = 0000001E              FKB$L_FR3                        = 00000010
CSD$L_SENDLEN                   = 00000012              FKB$L_FR4                        = 00000014
CSD$L_SENDOFF                   = 00000016              GET_NEXT_CSB                       000003E3 R      02
CSD$Q_INT_IOSB                  = 0000003A              INSQUE_CLUB                        0000025C R      02
CSD$Q_PROCPRIV                  = 00000042              IPL$_ASTDEL                      = 00000002
CSD$W_CODE                      = 0000000C              IPL$_SCS                         = 00000008
CSD$W_IOSB_STAT                 = 0000003A              IPL$_SYNCH                       = 00000008
CSD$W_SIZE                      = 00000008              JIB$L_BYTCNT                     = 00000020
CSP$BEGIN                         00000000 RG    02     KAST                               000005C4 R      02
CSP$B_INITED                      00000171 R     02     MMG$GL_SPTBASE                     ********   X    02
CSP$B_RCVCSDCNT                   00000170 R     02     PCB$L_JIB                        = 00000080
CSP$DISPATCH                      000001CD RG    02     PCB$L_PID                        = 00000060
CSP$INIT                          00000172 RG    02     PCB$L_STS                        = 00000024
CSP$K_MAX_FLWCTL                = 00000008              PCB$L_UIC                        = 000000BC
CSP$Q_ACB_IDLE                    00000160 R     02     PCB$Q_PRIV                       = 00000084
CSP$Q_ACB_XFER                    00000168 R     02     PCB$V_SSRWAIT                    = 0000000A
CSP$_ABORT                      = 00000002              PHD$L_IMGCNT                     = 000000F4
CSP$_BADCSD                     = 00000003              PR$_IPL                            ********   X    02
CSP$_DONE                       = 00000004              PRI$_IOCOM                       = 00000001
CSP$_LOCAL                      = 00000007              PROC_EVENT                         00000628 R      02
CSP$_REJECT                     = 00000006              PROC_EVENT_ASY                     0000060F R      02
CSP$_REPLY                      = 00000005              REQ_MSGBLD                         000006DA R      02
CSPMSG$B_RSP                      00000018              RSN$_ASTWAIT                     = 00000001
CSPMSG$B_SPARE                    00000019              RSN$_NPDYNMEM                    = 00000003
CSPMSG$K_RSP_ASYNERR            = 00000007              RSP_MSGBLD                         00000349 R      02
CSPMSG$K_RSP_BADCSD             = 00000006              SCH$GL_PCBVEC                      ********   X    02
CSPMSG$K_RSP_BUSY               = 00000002              SCH$QAST                           ********   X    02
CSPMSG$K_RSP_ILL                = 00000001              SCH$RWAIT                          ********   X    02
CSPMSG$K_RSP_MAX                = 00000009              SCH$WAKE                           ********   X    02
CSPMSG$K_RSP_NOCSP              = 00000003              SIZ...                           = 00000001
CSPMSG$K_RSP_NOP                = 00000000              SS$_BADPARAM                     = 00000014
CSPMSG$K_RSP_RO                 = 00000004              SS$_DEVACTIVE                    = 000002C4
CSPMSG$K_RSP_RW                 = 00000005              SS$_NORMAL                       = 00000001
```

```
SS$_NOSUCHNODE                 = 0000028C
SS$_REJECT                     = 00000294
SS$_TIMEOUT                    = 0000022C
VA$M_BYTE                      = 000001FF
VA$S_VPN                       = 00000015
VA$V_VPN                       = 00000009
WAIT                             0000054D R     02
_$END                          = 0000015E R     02
_$ENT                          = 00000002
_$MAXINX                       = 00000024
_$START                        = 00000154 R     02
_$TMP                          = 00000000 R     02

                                                +-----------------+
                                                ! Psect synopsis !
                                                +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (      0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000034  (     52.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$200                          00000894  (   2196.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC QUAD

                                             +------------------------+
                                             ! Performance indicators !
                                             +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    36    00:00:00.05    00:00:01.28
Command processing               137    00:00:00.48    00:00:04.21
Pass 1                           556    00:00:16.55    00:00:54.85
Symbol table sort                  0    00:00:02.15    00:00:08.44
Pass 2                           338    00:00:04.20    00:00:12.97
Symbol table output               29    00:00:00.13    00:00:00.98
Psect synopsis output              1    00:00:00.02    00:00:00.02
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals            1099    00:00:23.58    00:01:22.75

The working set limit was 2400 pages.
141751 bytes (277 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 1983 non-local and 75 local symbols.
1956 source lines were read in Pass 1, producing 21 object records in Pass 2.
48 pages of virtual memory were used to define 46 macros.

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

Macro library name                           Macros defined
------------------                           --------------
_$255$DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1                     4
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                           21
_$255$DUA28:[SYSLIB]STARLET.MLB;2                         7
TOTALS (all libraries)                                   32

2066 GETS were required to define 32 macros.
```

```
There were no errors, warnings or information messages.

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

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CSPCALL
LIS

CSPCJFMAS
LIS

CSP
LIS

CSPBRKTHR
LIS

CONUTIL
LIS

CSPCALLACT
LIS

CONSUBS
LIS