```
  000000000   PPPPPPPPPPPP     CCCCCCCCCCCC   000000000   MMM         MMM
  000000000   PPPPPPPPPPPP     CCCCCCCCCCCC   000000000   MMM         MMM
  000000000   PPPPPPPPPPPP     CCCCCCCCCCCC   000000000   MMM         MMM
000       000 PPP       PPP CCC            000       000 MMMMMM   MMMMMM
000       000 PPP       PPP CCC            000       000 MMMMMM   MMMMMM
000       000 PPP       PPP CCC            000       000 MMMMMM   MMMMMM
000       000 PPP       PPP CCC            000       000 MMM   MMM   MMM
000       000 PPP       PPP CCC            000       000 MMM   MMM   MMM
000       000 PPP       PPP CCC            000       000 MMM   MMM   MMM
000       000 PPPPPPPPPPPP  CCC            000       000 MMM         MMM
000       000 PPPPPPPPPPPP  CCC            000       000 MMM         MMM
000       000 PPPPPPPPPPPP  CCC            000       000 MMM         MMM
000       000 PPP           CCC            000       000 MMM         MMM
000       000 PPP           CCC            000       000 MMM         MMM
000       000 PPP           CCC            000       000 MMM         MMM
000       000 PPP           CCC            000       000 MMM         MMM
000       000 PPP           CCC            000       000 MMM         MMM
000       000 PPP           CCC            000       000 MMM         MMM
  000000000   PPP             CCCCCCCCCCCC   000000000   MMM         MMM
  000000000   PPP             CCCCCCCCCCCC   000000000   MMM         MMM
  000000000   PPP             CCCCCCCCCCCC   000000000   MMM         MMM
```

```
**FILE**ID**REPLYBRD                                              D 5
```

```
RRRRRRRR    EEEEEEEEEE  PPPPPPPP    LL          YY      YY  BBBBBBBB    RRRRRRRR    DDDDDDDD
RRRRRRRR    EEEEEEEEEE  PPPPPPPP    LL          YY      YY  BBBBBBBB    RRRRRRRR    DDDDDDDD
RR      RR  EE          PP      PP  LL          YY      YY  BB      BB  RR      RR  DD      DD
RR      RR  EE          PP      PP  LL          YY      YY  BB      BB  RR      RR  DD      DD
RR      RR  EE          PP      PP  LL            YY  YY    BB      BB  RR      RR  DD      DD
RR      RR  EE          PP      PP  LL            YY  YY    BB      BB  RR      RR  DD      DD
RRRRRRRR    EEEEEEEE    PPPPPPPP    LL              YY      BBBBBBBB    RRRRRRRR    DD      DD
RRRRRRRR    EEEEEEEE    PPPPPPPP    LL              YY      BBBBBBBB    RRRRRRRR    DD      DD
RR  RR      EE          PP          LL              YY      BB      BB  RR  RR      DD      DD
RR  RR      EE          PP          LL              YY      BB      BB  RR  RR      DD      DD
RR    RR    EE          PP          LL              YY      BB      BB  RR    RR    DD      DD      ....
RR    RR    EE          PP          LL              YY      BB      BB  RR    RR    DD      DD      ....
RR      RR  EEEEEEEEEE  PP          LLLLLLLLLL      YY      BBBBBBBB    RR      RR  DDDDDDDD        ....
RR      RR  EEEEEEEEEE  PP          LLLLLLLLLL      YY      BBBBBBBB    RR      RR  DDDDDDDD        ....

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

```
0 MODULE  opc$replybrd    (        %TITLE 'Broadcast command module' %SBTTL 'Copyright notice'
0                           LANGUAGE (BLISS32),
0                           IDENT = 'V04-000'
0                           ) =
0
0 !****************************************************************************
0 !*                                                                          *
0 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0 !*  ALL RIGHTS RESERVED.                                                    *
0 !*                                                                          *
0 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0 !*  TRANSFERRED.                                                            *
0 !*                                                                          *
0 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0 !*  CORPORATION.                                                            *
0 !*                                                                          *
0 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0 !*                                                                          *
0 !*                                                                          *
0 !****************************************************************************
0
0 !++
0 ! FACILITY:
0 !
0 !       OPCOM - OPCOM process and REPLY command
0 !
0 ! ABSTRACT:
0 !
0 !       This module formats and outputs user broadcast requests.
0 !
0 ! Environment:
0 !
0 !       VAX/VMS operating system.
0 !
0 ! Author:
0 !
0 !       CW Hobbs
0 !
0 ! Creation date:
0 !
0 !       3-Aug-1983
0 !
0 ! Revision history:
0 !
0 !       V03-003 CWH3169         CW Hobbs                        5-May-1984
0 !               Second pass for cluster-wide OPCOM:
0 !                - Liberal rewrite to use ast-driven, queued brkthru writes.
0 !
0 !       V03-002 CWH3002         CW Hobbs                16-Sep-1983
0 !               Improve timeout calculation and error handling.
```

```
;   58      0058  0 !--
```

```
1 BEGIN                                                        %SBTTL 'Start of replybrd'
1
1 LIBRARY 'SYS$LIBRARY:LIB.L32';
1 LIBRARY 'LIB$:OPCOMLIB';
1
1 FORWARD ROUTINE
1     replybrd_brkthru_ast          : NOVALUE,          ! AST routine for brkthru completion
1     replybrd_brkthru_queue        : NOVALUE,          ! Add an entry to the brkthru work queue
1     replybrd_format,                                  ! Format the message
1     replybrd_io,                                      ! Do the actual io to the devices
1     replybrd_io_list              : NOVALUE,          ! Do the actual io to a list of targets
1     replybrd_notify_all           : NOVALUE,          ! Format message for /ALL or /USER
1     replybrd_notify_dev           : NOVALUE,          ! Format message for /TERM=(...)
1     replybrd_notify_io            : NOVALUE,          ! Do something to actually display the message
1     replybrd_notify_use           : NOVALUE,          ! Format message for /USER=(...)
1     replybrd_ok_to_queue_write;                       ! Is it ok to actually start the $brkthru?
1
1 EXTERNAL ROUTINE
1     cluscomm_send : WEAK,                             ! The reply image doesn't need this
1     share_full_devname;
1
1 EXTERNAL
1     lcl_csid : WEAK;                                  ! The reply image doesn't need this
1
1 GLOBAL
1     bod_allocated,                                    ! Count of BODs created
1     bod_busy_count,                                   ! Current count of i/os pending
1     bod_busy_max          : INITIAL (32),             ! Maximum number of brkthrus pending
1     bod_requests,                                     ! Count of requests made
1     bod_busy_queue        : VECTOR [2, LONG]          ! Queue of BODs pending for I/O
1                             INITIAL (REP 2 OF (bod_busy_queue)),
1     bod_free_queue        : VECTOR [2, LONG]          ! Queue of BODs available for use
1                             INITIAL (REP 2 OF (bod_free_queue)),
1     bod_wait_queue        : VECTOR [2, LONG]          ! Queue of BODs waiting for actual $BRKTHRU to be queued
1                             INITIAL (REP 2 OF (bod_wait_queue)),
1     bod_garbage_queue     : VECTOR [2, LONG]          ! Pointer to list of virtual memory to deallocate
1                             INITIAL (REP 2 OF (bod_garbage_queue)),
1     replybrd_status,                                  ! Status for REPLY image
1     reply_image           : INITIAL (0);              ! Zero if in OPCOM image, one if in REPLY image
1
1 !
1 ! A macro to put virtual memory back on the queue of garbage to be deallocated
1 !
1 MACRO
M 1     COLLECT_GARBAGE (INP_DESC) =
M 1                     BEGIN
M 1                     BIND
M 1                         desc = (INP_DESC) : VECTOR [, LONG],
M 1                         garbage = .desc [1] : VECTOR [, LONG];
M 1                     garbage [2] = .desc [0];              ! Store length as second longword in block
M 1                     $queue_insert_tail (garbage, bod_garbage_queue);
1                       END %;
1
1 BUILTIN
1     CALLG;
1
1 !
```

```
1 ! Define the positions in the argument list for the calls to replybrd_brkthru_queue and replybrd_io_list
1 !
1 LITERAL
1       count = 0,          ! ar
1       text = 1,
1       targ = 2,
1       type = 3,
1       cntl = 4,
1       bflg = 5,
1       rqid = 6,
1       rtn = 7,
1       node = 8,
1       csid = 9,
1       sflg = 10,
1       term = 11,          ! last arg for replybrd_brkthru_queue
1       queue_args = 11,
1       llen = 12,
1       lptr = 13;
```

```
GLOBAL ROUTINE replybrd_brkthru_ast (parm) : NOVALUE =

!++
! Functional descripton:
!
!       This routine is the I/O completion for a $BRKTHRU write
!
! Input:
!       None.
!
! Output:
!       None.
!
! Routine Value:
!       Address of block
!--

BEGIN                                               ! Start of REPLYBRD_BRKTHRU_AST

LOCAL
    bod : $ref_bblock,
    status;

!
! If the parameter is non-zero, release that block
!
bod = .parm;
IF .bod NEQ 0
THEN
    BEGIN
    !
    ! Check the I/O status, if not ok, save the worst status.  If ok,
    ! call the completion routine if there is one
    !
    IF NOT .bod [bod_w_iosb0]
    THEN
        BEGIN
        IF .(bod [bod_q_iosb])<0,3,0> GTR .replybrd_status<0,3,0>
        THEN
            replybrd_status = .bod [bod_w_iosb0];
        END
    ELSE
        IF .bod [bod_a_completion_routine] NEQ 0
        THEN
            (.bod [bod_a_completion_routine]) (.bod);
    !
    ! Free the message string buffer
    !
    collect_garbage (bod [bod_q_msgbuf]);
    !
    ! Place the BOD in the free queue, put it at the head so that any old bods that get
    ! out-faulted will stay out until we really need 'em.
    !
    $queue_remove (.bod);                           ! Remove it from the que (should be in the busy queue)
    $queue_insert_head (.bod, bod_free_queue);
    bod_busy_count = .bod_busy_count - 1;
    END;
```

```
  2 !
  2 ! If we can queue another brkthru, then do so
  2 !
  2 bod = .bod_wait_queue [0];
  2 WHILE .bod NEQ bod_wait_queue                   ! Loop until we see the end
  2      AND
  2       .bod_busy_count LSS .bod_busy_max         ! or until we have filled our quota
  2 DO
  3      BEGIN
  3      LOCAL
  3          next;
  3      next = .bod [bod_l_flink];                 ! Save the pointer to the next, since we might pull it out
  3      IF replybrd_ok_to_queue_write (.bod, bod_busy_queue)       ! Control number of $BRKTHRUs pending per ta
  3      THEN
  4          BEGIN
  4          $queue_remove (.bod);                  ! Remove it from the waiting queue
P 4          status = $brkthru (efn = efn_k_asynch,
P 4                             msgbuf = bod [bod_q_msgbuf],
P 4                             sendto = bod [bod_q_sendto],
P 4                             iosb   = bod [bod_q_iosb],
P 4                             sndtyp = .bod [bod_l_sndtyp],
P 4                             carcon = .bod [bod_l_carcon],
P 4                             flags  = .bod [bod_l_flags],
P 4                             reqid  = .bod [bod_l_reqid],
P 4                             timout = (IF .bod [bod_v_short_timeout]      ! If the human is waiting for a repl
P 4                                       THEN 20                            !   then timeout after 20 seconds
P 4                                       ELSE 20*60),                       !   otherwise try for 20 minutes
P 4                             astadr = replybrd_brkthru_ast,
  4                             astprm = .bod);
  4          IF NOT .status                                          ! If it didn't work, free the bod
  4          THEN
  5              BEGIN
  5              IF .status<0,3,0> GTR .replybrd_status<0,3,0>       ! Save the worst status
  5              THEN
  5                  replybrd_status = .status;
  5              collect_garbage (bod [bod_q_msgbuf]);               ! Free the message
  5              $queue_insert_head (.bod, bod_free_queue);          ! Free the bod
  5              END
  4          ELSE
  5              BEGIN
  5              bod_busy_count = .bod_busy_count + 1;
  5              $queue_insert_tail (.bod, bod_busy_queue);
  5              $gettim (timadr=bod [bod_q_quetime]);
  4              END;
  3          END;
  3      !
  3      ! Advance to the next one, using the saved next pointer
  3      !
  3      bod = .next;
  2      END;
  2
  2 !
  2 ! Set an event flag if we have emptied both of the queues
  2 !
  3 IF $queue_empty (bod_wait_queue) AND $queue_empty (bod_busy_queue)
  2 THEN
  2      $setef (efn=efn_k_brkthru);
```

```
;  250      0248  2
;  251      0249  2 RETURN;
;  252      0250  1 END;                                      ! End of REPLYBRD_BRKTHRU_AST

                                                .TITLE  OPC$REPLYBRD Broadcast command module
                                                .IDENT  \V04-000\

                                                .PSECT  $GLOBAL$,NOEXE,2

                                00000 BOD_ALLOCATED::
                                                .BLKB   4
                                00004 BOD_BUSY_COUNT::
                                                .BLKB   4
                      00000020  00008 BOD_BUSY_MAX::
                                                .LONG   32                                    ;
                                0000C BOD_REQUESTS::
                                                .BLKB   4
                      00000000' 00010 BOD_BUSY_QUEUE::
                                                .ADDRESS BOD_BUSY_QUEUE                       ;
                      00000000' 00014           .ADDRESS BOD_BUSY_QUEUE                       ;
                      00000000' 00018 BOD_FREE_QUEUE::
                                                .ADDRESS BOD_FREE_QUEUE                       ;
                      00000000' 0001C           .ADDRESS BOD_FREE_QUEUE                       ;
                      00000000' 00020 BOD_WAIT_QUEUE::
                                                .ADDRESS BOD_WAIT_QUEUE                       ;
                      00000000' 00024           .ADDRESS BOD_WAIT_QUEUE                       ;
                      00000000' 00028 BOD_GARBAGE_QUEUE::
                                                .ADDRESS BOD_GARBAGE_QUEUE                    ;
                      00000000' 0002C           .ADDRESS BOD_GARBAGE_QUEUE                    ;
                                00030 REPLYBRD_STATUS::
                                                .BLKB   4
                      00000000  00034 REPLY_IMAGE::
                                                .LONG   0                                     ;

                                      _QH_=              BOD_GARBAGE_QUEUE
                                      _QH_=              BOD_FREE_QUEUE
                                      _QH_=              BOD_GARBAGE_QUEUE
                                      _QH_=              BOD_FREE_QUEUE
                                      _QH_=              BOD_BUSY_QUEUE
                                      _QH_=              BOD_WAIT_QUEUE
                                      _QH_=              BOD_BUSY_QUEUE
                                                .EXTRN  SHARE_FULL_DEVNAME
                                                .EXTRN  SYS$BRKTHRU, SYS$GETTIM
                                                .EXTRN  SYS$SETEF
                                                .WEAK   CLUSCOMM_SEND, LCL_CSID

                                                .PSECT  $CODE$,NOWRT,2

                                003C 00000      .ENTRY  REPLYBRD_BRKTHRU_AST, Save R2,R3,R4,R5      ; 0134
                    55    0000'  CF  9E 00002   MOVAB   REPLYBRD_STATUS, R5                         ;
                    52       04  AC  D0 00007   MOVL    PARM, BOD                                   ; 0160
                                 3C  13 0000B   BEQL    3$                                          ; 0161
                    13       3C  A2  E8 0000D   BLBS    60(BOD), 1$                                 ; 0168
 50          65     03           00  EF 00011   EXTZV   #0, #3, REPLYBRD_STATUS, R0                 ; 0171
 50     3C   A2     03           00  ED 00016   CMPZV   #0, #3, 60(BOD), R0                         ;
                                 11  15 0001C   BLEQ    2$                                          ;
```

```
                       65        3C  A2  3C 0001E       MOVZWL  60(BOD), REPLYBRD_STATUS                 ; 0173
                                     0B  11 00022       BRB     2$                                       ; 0168
                                 10  A2  D5 00024 1$:   TSTL    16(BOD)                                  ; 0176
                                     06  13 00027       BEQL    2$                                       ;
                                     52  DD 00029       PUSHL   BOD                                      ; 0178
                    10  B2           01  FB 0002B       CALLS   #1, @16(BOD)                             ;
                        51       28  A2  9E 0002F 2$:   MOVAB   40(BOD), R1                              ; 0182
                        50       04  A1  D0 00033       MOVL    4(R1), R0                                ;
                    08  A0           61  D0 00037       MOVL    (R1), 8(R0)                              ;
                    FC  B5           60  0E 0003B       INSQUE  (R0), @_QH_+4                            ;
                        50           62  0F 0003F       REMQUE  (BOD), _T_                               ; 0187
                    E8  A5           62  0E 00042       INSQUE  (BOD), _QH_                              ; 0188
                                 D4  A5  D7 00046       DECL    BOD_BUSY_COUNT                           ; 0189
                        52       F0  A5  D0 00049 3$:   MOVL    BOD_WAIT_QUEUE, BOD                      ; 0194
                        50       F0  A5  9E 0004D 4$:   MOVAB   BOD_WAIT_QUEUE, R0                       ; 0195
                        50           52  D1 00051       CMPL    BOD, R0                                  ;
                                     03  12 00054       BNEQ    6$                                       ;
                                   008B  31 00056 5$:   BRW     12$                                      ;
                    D8  A5       D4  A5  D1 00059 6$:   CMPL    BOD_BUSY_COUNT, BOD_BUSY_MAX             ; 0197
                                     F6  18 0005E       BGEQ    5$                                       ;
                        54           62  D0 00060       MOVL    (BOD), NEXT                              ; 0202
                                 E0  A5  9F 00063       PUSHAB  BOD_BUSY_QUEUE                           ; 0203
                                     52  DD 00066       PUSHL   BOD                                      ;
                 0000V  CF           02  FB 00068       CALLS   #2, REPLYBRD_OK_TO_QUEUE_WRITE           ;
                        6E           50  E9 0006D       BLBC    R0, 11$                                  ;
                        50           62  0F 00070       REMQUE  (BOD), _T_                               ; 0206
                                     52  DD 00073       PUSHL   BOD                                      ; 0219
                                 88  AF  9F 00075       PUSHAB  REPLYBRD_BRKTHRU_AST                     ;
           04       0C  A2           03  E1 00078       BBC     #3, 12(BOD), 7$                          ;
                                     14  DD 0007D       PUSHL   #20                                      ;
                                     05  11 0007F       BRB     8$                                       ;
                        7E     04B0  8F  3C 00081 7$:   MOVZWL  #1200, -(SP)                             ;
                        7E       48  A2  7D 00086 8$:   MOVQ    72(BOD), -(SP)                           ;
                                 44  A2  DD 0008A       PUSHL   68(BOD)                                  ;
                                 3C  A2  9F 0008D       PUSHAB  60(BOD)                                  ;
                                 38  A2  DD 00090       PUSHL   56(BOD)                                  ;
                                 30  A2  9F 00093       PUSHAB  48(BOD)                                  ;
                                 28  A2  9F 00096       PUSHAB  40(BOD)                                  ;
                                     01  DD 00099       PUSHL   #1                                       ;
             00000000G  00           0B  FB 0009B       CALLS   #11, SYS$BRKTHRU                         ;
                        53           50  D0 000A2       MOVL    R0, STATUS                               ;
                        25           53  E8 000A5       BLBS    STATUS, 10$                              ; 0220
  50       65           03           00  EF 000A8       EXTZV   #0, #3, REPLYBRD_STATUS, R0              ; 0223
  50       53           03           00  ED 000AD       CMPZV   #0, #3, STATUS, R0                       ;
                                     03  15 000B2       BLEQ    9$                                       ;
                        65           53  D0 000B4       MOVL    STATUS, REPLYBRD_STATUS                  ; 0225
                        51       28  A2  9E 000B7 9$:   MOVAB   40(BOD), R1                              ; 0226
                        50       04  A1  D0 000BB       MOVL    4(R1), R0                                ;
                    08  A0           61  D0 000BF       MOVL    (R1), 8(R0)                              ;
                    FC  B5           60  0E 000C3       INSQUE  (R0), @_QH_+4                            ;
                    E8  A5           62  0E 000C7       INSQUE  (BOD), _QH_                              ; 0227
                                     11  11 000CB       BRB     11$                                      ; 0220
                                 D4  A5  D6 000CD 10$:  INCL    BOD_BUSY_COUNT                           ; 0231
                    E4  B5           62  0E 000D0       INSQUE  (BOD), @_QH_+4                           ; 0232
                                 50  A2  9F 000D4       PUSHAB  80(BOD)                                  ; 0233
             00000000G  00           01  FB 000D7       CALLS   #1, SYS$GETTIM                           ;
                        52           54  D0 000DE 11$:  MOVL    NEXT, BOD                                ; 0239
```

```
                              FF69  31 000E1       BRW    4$                          ; 0195
              50        F0      A5  9E 000E4 12$:  MOVAB  _QH_, R0                    ; 0245
              50        F0      A5  D1 000E8       CMPL   _QH_, R0                    ;
                                13  12 000EC       BNEQ   13$                         ;
              50        E0      A5  9E 000EE       MOVAB  _QH_, R0                    ;
              50        E0      A5  D1 000F2       CMPL   _QH_, R0                    ;
                                09  12 000F6       BNEQ   13$                         ;
                                02  DD 000F8       PUSHL  #2                          ; 0247
       00000000G  00            01  FB 000FA       CALLS  #1, SYS$SETEF               ;
                                    04 00101 13$:  RET                                ; 0250

; Routine Size:  258 bytes,    Routine Base:  $CODE$ + 0000
```

```
GLOBAL ROUTINE replybrd_brkthru_queue (msgbuf : $ref_bblock, sendto : $ref_bblock,
                                       sndtyp, carcon, flags, reqid,
                                       routin, node : $ref_bblock, csid,
                                       stflg, term : $ref_bblock) : NOVALUE =

!++
! Functional descripton:
!
!       This routine queues a breakthru write
!
! Input:
!       msgbuf  Text string to broadcast
!       sendto  Target name for $BRKTHRU
!       sndtyp  Type of target
!       carcon  Carriage control
!       flags   Misc flags
!       reqid   Type of requestor
!       routin  Address of completion routine (following arguments used by completion routine)
!       node    Nodename
!       csid    CSID of originating system
!       stflg   Initial contents of STATUS flags
!       term    Name of originating terminal
!
! Output:
!       None.
!
! Routine Value:
!       Address of block
!--

BEGIN                                           ! Start of REPLYBRD_BRKTHRU_QUEUE

LOCAL
     bod          : $ref_bblock,                ! BOD data structure
     garb         : REF VECTOR [, LONG],
     ptr,
     status;

!
! If any garbage nodes are in the hopper, send them away.  Garbage is reclaimed this
! way so that the user ast routines do not do free_vm calls on memory allocated
! from user non-ast mode.
!
$queue_remove_head (bod_garbage_queue, garb);
WHILE .garb NEQ 0
DO
    BEGIN
    IF NOT (status = opc$free_vm (garb [2], garb))
    THEN
        $signal_stop (.status);
    $queue_remove_head (bod_garbage_queue, garb);
    END;
!
! Get a BOD, a Brkthru Output Descriptor, if none available on the queue then
! allocate and initialize one.
!
$queue_remove_head (bod_free_queue, bod);
```

```
2 IF .bod EQL 0
2 THEN
3     BEGIN
4     IF NOT (status = opc$get_vm (%ref (bod_k_size), ptr))
3     THEN
3         $signal_stop (.STATUS);
3     bod_allocated = .bod_allocated + 1;
3     bod = .ptr;
3     CH$FILL (0, bod_k_size, .bod);
3     bod [bod_w_size] = bod_k_size;
3     bod [bod_b_type] = %x'01';
3     bod [bod_a_senptr] = bod [bod_t_senbuf];
3     bod [bod_a_nodptr] = bod [bod_t_nodbuf];
3     bod [bod_a_trmptr] = bod [bod_t_trmbuf];
2     END;
2 !
2 ! Init the block and copy the longword inputs
2 !
2 (bod [bod_q_quetime]) = 0;
2 (bod [bod_q_quetime]+4) = 0;
2 bod [bod_l_senlen] = 0;
2 bod [bod_l_nodlen] = 0;
2 bod [bod_l_trmlen] = 0;
2 bod [bod_l_sndtyp] = .sndtyp;
2 bod [bod_l_carcon] = .carcon;
2 bod [bod_l_flags]  = .flags;
2 bod [bod_l_reqid]  = .reqid;
2 bod [bod_a_completion_routine] = .routin;
2 bod [bod_l_csid] = .csid;
2 bod [bod_l_status] = .stflg;
2 !
2 ! Copy the target string to the BOD, zero fill for a later test
2 !
2 IF .sendto NEQ 0
2 THEN
3     BEGIN
3     IF .sndtyp EQL brk$c_device                        ! If a device breakthru, then see if we can see the device
3     THEN
4         BEGIN
4         LOCAL
4             terminal,
4             dvi_items   : VECTOR [4, LONG];
4         dvi_items [0] = (dvi$_trm^16 OR 4);
4         dvi_items [1] = terminal;
4         dvi_items [2] = dvi_items [3] = 0;
4         status = $getdvi (devnam=.sendto, itmlst=dvi_items);    ! Can we see the device?
4         IF .status                                              ! If we can see it, is it a terminal
4         THEN
4             IF NOT .terminal
4             THEN
4                 status = ss$_ivdevnam;                          ! Invalid device name seems ok
4         IF NOT .status                                          ! Something went wrong
4         THEN
5             BEGIN
5             replybrd_status = .status;                          ! Save the status
5             $queue_insert_head (.bod, bod_free_queue);          ! Replace the new bod
5             RETURN;                                             ! And return
```

```
4               END;
3           END;
3       bod [bod_l_senlen] = .sendto [dsc$w_length];
3       CH$COPY (.sendto [dsc$w_length], .sendto [dsc$a_pointer], 0, bod_s_senbuf, bod [bod_t_senbuf]);
3       END
2   ELSE
2       CH$FILL (0, bod_s_senbuf, bod [bod_t_senbuf]);      ! For the test
2   !
2   ! Copy the output string to the BOD
2   !
2   bod [bod_l_msglen] = MAX (12, .msgbuf [dsc$w_length]);
3   IF NOT (status = opc$get_vm (bod [bod_l_msglen], bod [bod_a_msgptr]))
2   THEN
2       $signal_stop (.status);
2   CH$COPY (.msgbuf [dsc$w_length], .msgbuf [dsc$a_pointer], 0, .bod [bod_l_msglen], .bod [bod_a_msgptr]);
2   !
2   ! Copy the nodename to the BOD, if present
2   !
2   IF .node NEQ 0
2   THEN
3       BEGIN
3       bod [bod_l_nodlen] = .node [dsc$w_length];
3       CH$COPY (.node [dsc$w_length], .node [dsc$a_pointer], 0, bod_s_nodbuf, bod [bod_t_nodbuf]);
2       END;
2   !
2   ! Copy the terminal to the BOD, if present
2   !
2   IF .term NEQ 0
2   THEN
3       BEGIN
3       bod [bod_l_trmlen] = .term [dsc$w_length];
3       CH$COPY (.term [dsc$w_length], .term [dsc$a_pointer], 0, bod_s_trmbuf, bod [bod_t_trmbuf]);
2       END;
2   !
2   ! Place the BOD on the queue of outputs waiting
2   !
2   $queue_insert_tail (.bod, bod_wait_queue);
2   !
2   ! Declare an AST, the 0 parameter means try to start
2   !
2   bod_requests = .bod_requests + 1;
3   IF NOT (status = $dclast (astadr=replybrd_brkthru_ast, astprm=0))
2   THEN
2       $signal_stop (.status);
2
1   END;                                        ! End of REPLYBRD_BRKTHRU_QUEUE
; INFO#250          L1:0356
; Referenced LOCAL symbol TERMINAL is probably not initialized


                    _QH_=               BOD_GARBAGE_QUEUE
                    _QH_=               BOD_GARBAGE_QUEUE
                    _QH_=               BOD_FREE_QUEUE
                    _QH_=               BOD_FREE_QUEUE
                    _QH_=               BOD_WAIT_QUEUE
                            .EXTRN  OPC$FREE_VM, LIB$STOP
```

```
                                                    .EXTRN  OPC$GET_VM, SYS$GETDVI
                                                    .EXTRN  SYS$DCLAST

                              01FC 00000            .ENTRY  REPLYBRD_BRKTHRU_QUEUE, Save R2,R3,R4,R5,-   ; 0251
                                                            R6,R7,R8                                     ;
              58       0000'  CF 9E 00002           MOVAB   _QH_, R8                                     ;
              5E              20 C2 00007           SUBL2   #32, SP                                      ;
         04   AE         10   B8 0F 0000A 1$:       REMQUE  @_QH_, _T_                                   ; 0294
                              03 1C 0000F           BVC     2$                                           ;
                         04   AE D4 00011           CLRL    _T_                                          ;
                         04   AE D5 00014 2$:       TSTL    GARB                                         ; 0295
                              16 13 00017           BEQL    4$                                           ;
                         04   AE 9F 00019           PUSHAB  GARB                                         ; 0298
    7E        08   AE         08 C1 0001C           ADDL3   #8, GARB, -(SP)                              ;
           0000G   CF         02 FB 00021           CALLS   #2, OPC$FREE_VM                              ;
              57              50 D0 00026           MOVL    R0, STATUS                                   ;
              DE              57 E8 00029           BLBS    STATUS, 1$                                   ;
                          0147 31 0002C 3$:         BRW     15$                                          ; 0300
              56         00   B8 0F 0002F 4$:       REMQUE  @_QH_, _T_                                   ; 0307
                              02 1C 00033           BVC     5$                                           ;
                              56 D4 00035           CLRL    _T_                                          ;
                              56 D5 00037 5$:       TSTL    BOD                                          ; 0308
                              3F 12 00039           BNEQ    6$                                           ;
                         08   AE 9F 0003B           PUSHAB  PTR                                          ; 0311
         04   AE         BC   8F 9A 0003E           MOVZBL  #188, 4(SP)                                  ;
                         04   AE 9F 00043           PUSHAB  4(SP)                                        ;
           0000G   CF         02 FB 00046           CALLS   #2, OPC$GET_VM                               ;
              57              50 D0 0004B           MOVL    R0, STATUS                                   ;
              DB              57 E9 0004E           BLBC    STATUS, 3$                                   ;
                         E8   A8 D6 00051           INCL    BOD_ALLOCATED                                ; 0314
              56         08   AE D0 00054           MOVL    PTR, BOD                                     ; 0315
 00BC   8F       00   6E      00 2C 00058           MOVC5   #0, (SP), #0, #188, (BOD)                    ; 0316
                              66    0005F                                                                ;
         08   A6         BC   8F 9B 00060           MOVZBW  #188, 8(BOD)                                 ; 0317
         0A   A6              01 90 00065           MOVB    #1, 10(BOD)                                  ; 0318
         34   A6         58   A6 9E 00069           MOVAB   88(R6), 52(BOD)                              ; 0319
         1C   A6       0098   C6 9E 0006E           MOVAB   152(R6), 28(BOD)                             ; 0320
         24   A6       00A8   C6 9E 00074           MOVAB   168(R6), 36(BOD)                             ; 0321
                         50   A6 7C 0007A 6$:       CLRQ    80(BOD)                                      ; 0326
                         30   A6 D4 0007D           CLRL    48(BOD)                                      ; 0328
                         18   A6 D4 00080           CLRL    24(BOD)                                      ; 0329
                         20   A6 D4 00083           CLRL    32(BOD)                                      ; 0330
         38   A6         0C   AC D0 00086           MOVL    SNDTYP, 56(BOD)                              ; 0331
         44   A6         10   AC 7D 0008B           MOVQ    CARCON, 68(BOD)                              ; 0332
         4C   A6         18   AC D0 00090           MOVL    REQID, 76(BOD)                               ; 0334
         10   A6         1C   AC D0 00095           MOVL    ROUTIN, 16(BOD)                              ; 0335
         14   A6         24   AC D0 0009A           MOVL    CSID, 20(BOD)                                ; 0336
         0C   A6         28   AC D0 0009F           MOVL    STFLG, 12(BOD)                               ; 0337
                         08   AC D5 000A4           TSTL    SENDTO                                       ; 0341
                              57 13 000A7           BEQL    10$                                          ;
              01         0C   AC D1 000A9           CMPL    SNDTYP, #1                                   ; 0344
                              3D 12 000AD           BNEQ    9$                                           ;
         10   AE   004A0004   8F D0 000AF           MOVL    #4849668, DVI_ITEMS                          ; 0350
         14   AE         0C   AE 9E 000B7           MOVAB   TERMINAL, DVI_ITEMS+4                        ; 0351
                         18   AE 7C 000BC           CLRQ    DVI_ITEMS+8                                  ; 0352
                              7E 7C 000BF           CLRQ    -(SP)                                        ; 0353
                              7E 7C 000C1           CLRQ    -(SP)                                        ;
```

```
                                          20   AE   9F 000C3          PUSHAB  DVI_ITEMS                                  ;
                                          08   AC   DD 000C6          PUSHL   SENDTO                                     ;
                                               7E   7C 000C9          CLRQ    -(SP)                                      ;
                    00000000G   00             08   FB 000CB          CALLS   #8, SYS$GETDVI                             ;
                                57             50   D0 000D2          MOVL    R0, STATUS                                 ;
                                0C             57   E9 000D5          BLBC    STATUS, 8$                                 ; 0354
                                05        0C   AE   E8 000D8          BLBS    TERMINAL, 7$                               ; 0356
                                57      0144   8F   3C 000DC          MOVZWL  #324, STATUS                               ; 0358
                                08             57   E8 000E1 7$:      BLBS    STATUS, 9$                                 ; 0359
                          18    A8             57   D0 000E4 8$:      MOVL    STATUS, REPLYBRD_STATUS                    ; 0362
                                68             66   0E 000E8          INSQUE  (BOD), _QH_                                ; 0363
                                                    04 000EB          RET                                                ; 0361
                                50        08   AC   D0 000EC 9$:      MOVL    SENDTO, R0                                 ; 0367
                                30   A6        60   3C 000F0          MOVZWL  (R0), 48(BOD)                              ;
      0040    8F          00    04   B0        60   2C 000F4          MOVC5   (R0), @4(R0), #0, #64, 88(BOD)             ; 0368
                                          58   A6      000FC                                                             ;
                                               09   11 000FE          BRB     11$                                        ; 0341
      0040    8F          00    6E             00   2C 00100 10$:     MOVC5   #0, (SP), #0, #64, 88(BOD)                 ; 0371
                                          58   A6      00107                                                             ;
                                52        04   AC   D0 00109 11$:     MOVL    MSGBUF, R2                                 ; 0375
                                50             62   3C 0010D          MOVZWL  (R2), R0                                   ;
                                0C             50   B1 00110          CMPW    R0, #12                                    ;
                                               03   1E 00113          BGEQU   12$                                        ;
                                50             0C   D0 00115          MOVL    #12, R0                                    ;
                          28    A6             50   D0 00118 12$:     MOVL    R0, 40(BOD)                                ;
                                          2C   A6   9F 0011C          PUSHAB  44(BOD)                                    ; 0376
                                          28   A6   9F 0011F          PUSHAB  40(BOD)                                    ;
                        0000G   CF             02   FB 00122          CALLS   #2, OPC$GET_VM                             ;
                                57             50   D0 00127          MOVL    R0, STATUS                                 ;
                                49             57   E9 0012A          BLBC    STATUS, 15$                                ;
        28    A6          00    04   B2        62   2C 0012D          MOVC5   (R2), @4(R2), #0, 40(BOD), @44(BOD)        ; 0379
                                          2C   B6      00134                                                             ;
                                50        20   AC   D0 00136          MOVL    NODE, R0                                   ; 0383
                                               0D   13 0013A          BEQL    13$                                        ;
                                18   A6        60   3C 0013C          MOVZWL  (R0), 24(BOD)                              ; 0386
              10          00    04   B0        60   2C 00140          MOVC5   (R0), @4(R0), #0, #16, 152(BOD)            ; 0387
                                        0098   C6      00146                                                             ;
                                50        2C   AC   D0 00149 13$:     MOVL    TERM, R0                                   ; 0392
                                               0D   13 0014D          BEQL    14$                                        ;
                                20   A6        60   3C 0014F          MOVZWL  (R0), 32(BOD)                              ; 0395
              14          00    04   B0        60   2C 00153          MOVC5   (R0), @4(R0), #0, #20, 168(BOD)            ; 0396
                                        00A8   C6      00159                                                             ;
                                0C   B8        66   0E 0015C 14$:     INSQUE  (BOD), @_QH_+4                             ; 0401
                                          F4   A8   D6 00160          INCL    BOD_REQUESTS                               ; 0405
                                               7E   7C 00163          CLRQ    -(SP)                                      ; 0406
                                        FD95   CF   9F 00165          PUSHAB  REPLYBRD_BRKTHRU_AST                       ;
                    00000000G   00             03   FB 00169          CALLS   #3, SYS$DCLAST                             ;
                                57             50   D0 00170          MOVL    R0, STATUS                                 ;
                                09             57   E8 00173          BLBS    STATUS, 16$                                ;
                                               57   DD 00176 15$:     PUSHL   STATUS                                     ; 0408
                    00000000G   00             01   FB 00178          CALLS   #1, LIB$STOP                               ;
                                                    04 0017F 16$:     RET                                                ; 0410

; Routine Size:  384 bytes,    Routine Base:  $CODE$ + 0102
```

```
1 GLOBAL ROUTINE replybrd_format (msg : $ref_bblock, local_node : $ref_bblock) =  %SBTTL 'Replybrd_Format (msg
1
1 !++
1 ! Functional description:
1 !
1 !       This routine makes a formatted message from a broadcast message block.  The formatted
1 !       message is placed off the message block.
1 !
1 ! Input:
1 !
1 !       msg        - Pointer to RPYBRD message block
1 !       local_node - Pointer to string descriptor with local node name
1 !
1 ! Implicit Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implict output:
1 !
1 !       None.
1 !
1 ! Side effects:
1 !
1 !       None.
1 !
1 ! Routine value:
1 !
1 !       None.
1 !--
1
2 BEGIN                                                 ! Start of replybrd_format
2
2 LOCAL
2       send_term_dsc   : VECTOR [2, LONG],
2       send_user_dsc   : VECTOR [2, LONG],
2       send_node_dsc   : VECTOR [2, LONG],
2       message_dsc     : VECTOR [2, LONG],
2       on_buf          : VECTOR [64, BYTE],
2       on_dsc          : VECTOR [2, LONG] INITIAL (64, on_buf),
2       tim_buf         : VECTOR [12, BYTE],
2       tim_dsc         : VECTOR [2, LONG] INITIAL (11, tim_buf),
2       fmt_buf         : VECTOR [512, BYTE],
2       fmt_dsc         : VECTOR [2, LONG] INITIAL (512, fmt_buf),
2       ctr_buf         : VECTOR [256, BYTE],
2       ctr_dsc         : VECTOR [2, LONG] INITIAL (256, ctr_buf),
2       bell_start,
2       bell_mid,
2       bell_end,
2       msgid,
2       ptr,
2       status;
2
2 BIND
```

```
2       bells  = UPLIT BYTE (7,7,7,7);
2
2   !
2   ! If /BELLS, then set things up to make some noise
2   !
2   bell_start = bell_mid = bell_end = 0;
2   IF .msg [rpybrd_v_bell]
2   THEN
3       BEGIN
3       bell_start = 1;
3       IF .msg [rpybrd_v_shutdown]
3       THEN
4           BEGIN
4           bell_end = 2;
4           bell_mid = 1;
3           END;
3       IF .msg [rpybrd_v_urgent]
3       THEN
3           bell_mid = 1;
2       END;
2   !
2   ! Make descriptors for the four required items in the message (the optional items don't change the
2   ! formatting)
2   !
2   ptr = msg [rpybrd_t_text];
2   send_term_dsc [1] = .ptr;
2   ptr = .ptr + (send_term_dsc [0] = .msg [rpybrd_w_send_term_len]);
2   send_user_dsc [1] = .ptr;
2   ptr = .ptr + (send_user_dsc [0] = .msg [rpybrd_w_send_user_len]);
2   send_node_dsc [1] = .ptr;
2   ptr = .ptr + (send_node_dsc [0] = .msg [rpybrd_w_send_node_len]);
2   message_dsc [1] = .ptr;
2   ptr = .ptr + (message_dsc [0] = .msg [rpybrd_w_message_len]);
2   !
2   ! Set the message id.  This will be a general broadcast unless /URGENT or /SHUTDOWN is requested.
2   !
3   msgid =     (SELECTONE true OF
3                SET
3                    [.msg [rpybrd_v_urgent]] :        opc$_reply_urgent;
3                    [.msg [rpybrd_v_shutdown]] :      opc$_reply_shutdown;
3                    [OTHERWISE] :                     opc$_reply_general;
3                TES);
2   !
2   ! Get the $FAO control string text associated with the message id via $GETMSG.
2   !
3   IF NOT (status = $GETMSG (MSGID=.msgid, MSGLEN=ctr_dsc, BUFADR=ctr_dsc, FLAGS=1))
2   THEN
2       RETURN .status;
2   !
2   ! If we have a net or cluster, then set the preposition to precede the node name
2   !
2   IF .local_node [dsc$w_length] NEQ 0
2   THEN
3       BEGIN
4       IF NOT (status = $GETMSG (MSGID=opc$_on_node, MSGLEN=on_dsc, BUFADR=on_dsc, FLAGS=1))
3       THEN
3           RETURN .status;
```

```
    END
ELSE
    on_dsc [0] = 0;                                 ! Null prepositions
!
! Format the time, we can make it look a little cleaner if we strip the hundredths
!
$ASCTIM (timbuf=tim_dsc, cvtflg=1);
tim_dsc [0] = 8;
!
! Format the message, the general form of the control string is:
!
!       "!/!ADReply received!AS!AF from user !AF at !AS   !AS!AD!/!AS!AD!/"
!
IF NOT (status = $FAO ( ctr_dsc,                    ! Control string
                        fmt_dsc, fmt_dsc,            ! Return length, output buffer
                        .bell_start, bells,          ! First bell
                        on_dsc,                      ! Preposition (like in "on " for "on ATHENS")
                        .local_node [dsc$w_length],  ! Local node name
                        .local_node [dsc$a_pointer], ! Local node name
                        .send_user_dsc [0],          ! Sender's username
                        .send_user_dsc [1],          ! Sender's username
                        send_term_dsc,               ! Name of sender's terminal
                        tim_dsc,                     ! Use the reformatted time
                        .bell_mid, bells,            ! Second bell
                        message_dsc,                 ! Actual message text
                        .bell_end, bells))           ! Third bell
THEN
    RETURN .status;
!
! Get a chunk of memory to hang the formatted message from the block.  We make a descriptor in the block.
!
msg [rpybrd_l_format_len] = .fmt_dsc [0];
IF NOT (status = OPC$GET_VM (msg [rpybrd_l_format_len], msg [rpybrd_a_format_ptr]))
THEN
    RETURN .status;
!
! Move the formatted message to the block.  As we copy the message, change
! any escape characters to dollar signs.
!
BEGIN
REGISTER
    char : BYTE,
    iptr,
    optr;
iptr = .fmt_dsc [1];
optr = .msg [rpybrd_a_format_ptr];
INCR i FROM 1 TO .fmt_dsc [0]
DO
    BEGIN
    char = CH$RCHAR_A (iptr);
    IF .char EQL 27 THEN char = %C '$';
    CH$WCHAR_A (.char, optr);
    END;
END;

RETURN true;
END;                                                ! End of replybrd_format
```

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

                    07  07  07  07  00000 P.AAA:  .BYTE   7, 7, 7, 7                                      ;

                                      BELLS=               P.AAA
                                              .EXTRN  SYS$GETMSG, SYS$ASCTIM
                                              .EXTRN  SYS$FAO

                                              .PSECT  $CODE$,NOWRT,2

                              03FC  00000            .ENTRY  REPLYBRD_FORMAT, Save R2,R3,R4,R5,R6,R7,R8,-; 0411
                                                             R9
              59      0000'   CF  9E  00002          MOVAB   BELLS, R9                                 ;
              58  00000000G   00  9E  00007          MOVAB   SYS$GETMSG, R8                            ;
              5E       FC78   CE  9E  0000E          MOVAB   -904(SP), SP                              ;
         98   AD         40   8F  9A  00013          MOVZBL  #64, ON_DSC                               ; 0445
         9C   AD         A0   AD  9E  00018          MOVAB   ON_BUF, ON_DSC+4                          ;
         84   AD              0B  D0  0001D          MOVL    #11, TIM_DSC                              ;
         88   AD         8C   AD  9E  00021          MOVAB   TIM_BUF, TIM_DSC+4                        ;
       0104   CE       0200   8F  3C  00026          MOVZWL  #512, FMT_DSC                             ;
       0108   CE       010C   CE  9E  0002D          MOVAB   FMT_BUF, FMT_DSC+4                        ;
         7E            0100   8F  3C  00034          MOVZWL  #256, CTR_DSC                             ;
         04   AE         08   AE  9E  00039          MOVAB   CTR_BUF, CTR_DSC+4                        ;
                              55  7C  0003E          CLRQ    BELL_MID                                  ; 0473
                              57  D4  00040          CLRL    BELL_START                                ;
              52         04   AC  D0  00042          MOVL    MSG, R2                                   ; 0474
              53         0C   A2  9E  00046          MOVAB   12(R2), R3                                ;
   14         63              01  E1  0004A          BBC     #1, (R3), 2$                              ;
              57              01  D0  0004E          MOVL    #1, BELL_START                            ; 0477
   06         63              04  E1  00051          BBC     #4, (R3), 1$                              ; 0478
              56              02  D0  00055          MOVL    #2, BELL_END                              ; 0481
              55              01  D0  00058          MOVL    #1, BELL_MID                              ; 0482
   03         63              06  E1  0005B  1$:     BBC     #6, (R3), 2$                              ; 0484
              55              01  D0  0005F          MOVL    #1, BELL_MID                              ; 0486
              50         30   A2  9E  00062  2$:     MOVAB   48(R2), PTR                               ; 0492
         FC   AD              50  D0  00066          MOVL    PTR, SEND_TERM_DSC+4                      ; 0493
              51         14   A2  3C  0006A          MOVZWL  20(R2), RT                                ; 0494
         F8   AD              51  D0  0006E          MOVL    R1, SEND_TERM_DSC                         ;
              50              51  C0  00072          ADDL2   R1, PTR                                   ;
         F4   AD              50  D0  00075          MOVL    PTR, SEND_USER_DSC+4                      ; 0495
              51         16   A2  3C  00079          MOVZWL  22(R2), RT                                ; 0496
         F0   AD              51  D0  0007D          MOVL    R1, SEND_USER_DSC                         ;
              50              51  C0  00081          ADDL2   R1, PTR                                   ;
         EC   AD              50  D0  00084          MOVL    PTR, SEND_NODE_DSC+4                      ; 0497
              51         18   A2  3C  00088          MOVZWL  24(R2), RT                                ; 0498
         E8   AD              51  D0  0008C          MOVL    R1, SEND_NODE_DSC                         ;
              50              51  C0  00090          ADDL2   R1, PTR                                   ;
         E4   AD              50  D0  00093          MOVL    PTR, MESSAGE_DSC+4                        ; 0499
              51         1A   A2  3C  00097          MOVZWL  26(R2), R1                                ; 0500
         E0   AD              51  D0  0009B          MOVL    R1, MESSAGE_DSC                           ;
              50              51  C0  0009F          ADDL2   R1, PTR                                   ;
   09         63              06  E1  000A2          BBC     #6, (R3), 3$                              ; 0506
              50  0005827B    8F  D0  000A6          MOVL    #361083, MSGID                            ;
                              14  11  000AD          BRB     5$                                        ;
```

```
          09            63            04  E1 000AF 3$:      BBC     #4, (R3), 4$                         ; 0507
                        50 00058273   8F  D0 000B3          MOVL    #361075, MSGID                       ;
                                      07  11 000BA          BRB     5$                                   ;
                        50 0005826B   8F  D0 000BC 4$:      MOVL    #361067, MSGID                       ; 0508
                        7E            01  7D 000C3 5$:      MOVQ    #1, -(SP)                            ; 0513
                                08    AE  9F 000C6          PUSHAB  CTR_DSC                              ;
                                0C    AE  9F 000C9          PUSHAB  CTR_DSC                              ;
                                      50  DD 000CC          PUSHL   MSGID                                ;
                        68            05  FB 000CE          CALLS   #5, SYS$GETMSG                       ;
                        53            50  D0 000D1          MOVL    R0, STATUS                           ;
                        70            53  E9 000D4          BLBC    STATUS, 8$                           ;
                        54      08    AC  D0 000D7          MOVL    LOCAL_NODE, R4                       ; 0519
                                      64  B5 000DB          TSTW    (R4)                                 ;
                                      1A  13 000DD          BEQL    6$                                   ;
                        7E            01  7D 000DF          MOVQ    #1, -(SP)                            ; 0522
                                98    AD  9F 000E2          PUSHAB  ON_DSC                               ;
                                98    AD  9F 000E5          PUSHAB  ON_DSC                               ;
                          000582AB    8F  DD 000E8          PUSHL   #361131                              ;
                        68            05  FB 000EE          CALLS   #5, SYS$GETMSG                       ;
                        53            50  D0 000F1          MOVL    R0, STATUS                           ;
                        05            53  E8 000F4          BLBS    STATUS, 7$                           ;
                                      68  11 000F7          BRB     9$                                   ; 0524
                                98    AD  D4 000F9 6$:      CLRL    ON_DSC                               ; 0527
                                      01  DD 000FC 7$:      PUSHL   #1                                   ; 0531
                                      7E  D4 000FE          CLRL    -(SP)                                ;
                                84    AD  9F 00100          PUSHAB  TIM_DSC                              ;
                                      7E  D4 00103          CLRL    -(SP)                                ;
      00000000G  00                   04  FB 00105          CALLS   #4, SYS$ASCTIM                       ;
             84  AD                   08  D0 0010C          MOVL    #8, TIM_DSC                          ; 0532
                              0240    8F  BB 00110          PUSHR   #^M<R6,R9>                           ; 0550
                                E0    AD  9F 00114          PUSHAB  MESSAGE_DSC                          ;
                              0220    8F  BB 00117          PUSHR   #^M<R5,R9>                           ;
                                84    AD  9F 0011B          PUSHAB  TIM_DSC                              ;
                                F8    AD  9F 0011E          PUSHAB  SEND_TERM_DSC                        ;
                        7E      F0    AD  7D 00121          MOVQ    SEND_USER_DSC, -(SP)                 ;
                                04    A4  DD 00125          PUSHL   4(R4)                                ;
                        7E            64  3C 00128          MOVZWL  (R4), -(SP)                          ;
                                98    AD  9F 0012B          PUSHAB  ON_DSC                               ;
                              0280    8F  BB 0012E          PUSHR   #^M<R7,R9>                           ;
                              0140    CE  9F 00132          PUSHAB  FMT_DSC                              ;
                              0144    CE  9F 00136          PUSHAB  FMT_DSC                              ;
                                40    AE  9F 0013A          PUSHAB  CTR_DSC                              ;
      00000000G  00                   11  FB 0013D          CALLS   #17, SYS$FAO                         ;
                 53                   50  D0 00144          MOVL    R0, STATUS                           ;
                 17                   53  E9 00147 8$:      BLBC    STATUS, 9$                           ;
             28  A2               0108    CE  D0 0014A          MOVL    FMT_DSC, 40(R2)                      ; 0556
                                2C    A2  9F 00150          PUSHAB  44(R2)                               ; 0557
                                28    A2  9F 00153          PUSHAB  40(R2)                               ;
          0000G  CF                   02  FB 00156          CALLS   #2, OPC$GET_VM                       ;
                 53                   50  D0 0015B          MOVL    R0, STATUS                           ;
                 04                   53  E8 0015E          BLBS    STATUS, 10$                          ;
                 50                   53  D0 00161 9$:      MOVL    STATUS, R0                           ; 0559
                                          04 00164          RET                                          ;
                 51           010C    CE  D0 00165 10$:     MOVL    FMT_DSC+4, IPTR                      ; 0569
                 52             2C    A2  D0 0016A          MOVL    44(R2), OPTR                         ; 0570
                                      53  D4 0016E          CLRL    I                                    ; 0571
                                      0E  11 00170          BRB     13$                                  ;
```

```
                    50          81  90 00172 11$:    MOVB    (IPTR)+, CHAR                ; 0574
                    1B          50  91 00175         CMPB    CHAR, #27                    ; 0575
                                03  12 00178         BNEQ    12$                          ;
                    50          24  90 0017A         MOVB    #36, CHAR                    ;
                    82          50  90 0017D 12$:    MOVB    CHAR, (OPTR)+                ; 0576
        EC          53    0108  CE  F3 00180 13$:    AOBLEQ  FMT_DSC, I, 11$              ; 0571
                    50          01  D0 00186         MOVL    #1, R0                       ; 0580
                                    04 00189         RET                                  ; 0581

; Routine Size:  394 bytes,    Routine Base:  $CODE$ + 0282
```

```
1 GLOBAL ROUTINE replybrd_io (msg : $ref_bblock, local_node : $ref_bblock) =      %SBTTL 'Replybrd_IO (msg, lo
1
1 !++
1 ! Functional description:
1 !
1 !       This routine sends the formatted message from a broadcast message block to the terminals,
1 !       using the $BRKTHRU system service.
1 !
1 ! Input:
1 !
1 !       msg        - Pointer to RPYBRD message block
1 !       local_node - Pointer to string descriptor with local node name
1 !
1 ! Implicit Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implict output:
1 !
1 !       Messages might be sent to terminals.
1 !
1 ! Side effects:
1 !
1 !       None.
1 !
1 ! Routine value:
1 !
1 !       None.
1 !--
1
2 BEGIN                                             ! Start of replybrd_io
2
2 LOCAL
2       flags,                                      ! Initial flags for BOD_L_STATUS
2       sndtyp,                                     ! Type of brkthru
2       reqid,                                      ! ID of brkthru sender
2       arglist         : VECTOR [16, LONG],        ! Argument list for callg
2       trmdsc          : VECTOR [2, LONG],
2       ptr,
2       timout,
2       status;
2
2 !
2 ! Initialize the global status to zero.  Determine the miscellaneous flags.
2 !
2 replybrd_status = 0;
2 flags = bod_m_short_timeout;
2 IF .msg [rpybrd_v_wait]
2 THEN
2     flags = .flags + bod_m_wait;
2 IF .msg [rpybrd_v_local_node]
2 THEN
2     flags = .flags + bod_m_local_node;
```

```
2
2 !
2 ! Set the send type.  If REPLY /ALL, then send to all terms.  If /USER, send to all logged in users,
2 ! If /USER=(...), send to those users.  If /TERM=(...), send to those terminals.
2 !
3 sndtyp =    (SELECTONE true OF
3              SET
3                  [.msg [rpybrd_v_all]] :            brk$c_allterms;
3                  [.msg [rpybrd_v_username]] :       IF .msg [rpybrd_w_targ_user_len] EQL 0
3                                                     THEN brk$c_allusers
3                                                     ELSE brk$c_username;
3                  [.msg [rpybrd_v_terminal]] :       brk$c_device;
3                  [OTHERWISE] :                      RETURN ss$_badparam;
3              TES);
2 !
2 ! Set the request id.  This will be a general broadcast unless /URGENT or /SHUTDOWN is requested.
2 !
3 reqid =     (SELECTONE true OF
3              SET
3                  [.msg [rpybrd_v_urgent]] :         brk$c_urgent;
3                  [.msg [rpybrd_v_shutdown]] :       brk$c_shutdown;
3                  [OTHERWISE] :                      brk$c_general;
2              TES);
2 !
2 ! Build the argument list for the callg.  Since most of the arguments are the same, this both saves
2 ! execution time and makes it easier to update.  Also note that the list io routine will modify
2 ! this arglist
2 !
2 arglist [count] = queue_args;                                 ! assume args for an ALL reply
2 arglist [text]  = msg [rpybrd_q_format_desc];                 ! text to send
2 arglist [targ]  = 0;                                          ! target of send
2 arglist [type]  = .sndtyp;                                    ! type of target
2 arglist [cntl]  = 32;                                         ! carriage control
2 arglist [bflg]  = 0;                                          ! brkthru flags
2 arglist [rqid]  = .reqid;                                     ! requestor id
2 arglist [rtn]   = 0;                                          ! assume no completion routine
2 arglist [node]  = .local_node;                                ! local node name
2 arglist [csid]  = .msg [rpybrd_l_send_csid];                  ! remote csid name
2 arglist [sflg]  = .flags;                                     ! miscellaneous flags for BOD_L_STATUS
2 arglist [term]  = trmdsc;                                     ! pass the address of and
2 trmdsc [0]      = .msg [rpybrd_w_send_term_len];              !  create a descriptor
2 trmdsc [1]      = msg [rpybrd_t_text];                        !  for the sending terminal
2
2 !
2 ! If we are supposed to wait, then disable ASTs while we are queueing the brkthrus
2 !
2 IF .msg [rpybrd_v_wait]
2 THEN
2     $setast (enbflg=0);
2
2 !
2 ! Now send it on to the world.  If it is /ALL or /USER, then do a brkthru and
2 ! return.  If /USER=(...) or /TERM=(...), set up pointers so that we can do one
2 ! brkthru per specified user or terminal and call the list io routine.
2 !
2 CASE .sndtyp FROM 1 TO brk$c_maxsendtype OF
2 SET
```

```
2        [brk$c_allterms, brk$c_allusers] :
3                     BEGIN
3                     IF .msg [rpybrd_v_notify]                              ! if notify then pass
3                     THEN arglist [rtn] = replybrd_notify_all;               !  address of notification routine
3                     CALLG (arglist, replybrd_brkthru_queue);
2                     END;
2
2        [brk$c_device] :
3                     BEGIN
3                     IF .msg [rpybrd_v_notify]                              ! if notify then pass
3                     THEN arglist [rtn] = replybrd_notify_dev;               !  address of notification routine
3                     arglist [lptr] = .msg + .msg [rpybrd_w_optional_off];   ! send list address
3                     arglist [llen] = .msg [rpybrd_w_targ_term_len];         ! send list length
3                     arglist [count] = queue_args+2;
3                     CALLG (arglist, replybrd_io_list);
2                     END;
2
2        [brk$c_username] :
3                     BEGIN
3                     IF .msg [rpybrd_v_notify]                              ! if notify then pass
3                     THEN arglist [rtn] = replybrd_notify_use;               !  address of notification routine
3                     arglist [lptr] = .msg + .msg [rpybrd_w_optional_off] +  ! send list address
3                                     .msg [rpybrd_w_targ_term_len];
3                     arglist [llen] = .msg [rpybrd_w_targ_user_len];         ! send list length
3                     arglist [count] = queue_args+2;
3                     CALLG (arglist, replybrd_io_list);
2                     END;
2
2        [INRANGE, OUTRANGE] :
2                     RETURN ss$_badparam;
2   TES;
2
2   !
2   ! If we are supposed to wait, then fire off an AST to start the ball rolling and wait for our flag
2   !
2   IF .msg [rpybrd_v_wait]
2     AND
2      .replybrd_status EQL 0                                 ! Don't wait if an error occurred
2   THEN
3       BEGIN
3       $clref (efn=efn_k_brkthru);                            ! Clear our flag
3       $setast (enbflg=1);                                    ! Enable pending ASTs
3       $waitfr (efn=efn_k_brkthru);                           ! Wait for the queues to empty
2       END;
2
2   !
2   ! If nobody has modified the status, then return success
2   !
2   IF (status = .replybrd_status) EQL 0
2   THEN
2       status = ss$_normal;
2
2   RETURN .status;
1   END;                                              ! End of replybrd_io
```

```
                                                        .EXTRN   SYS$SETAST, SYS$CLREF
                                                        .EXTRN   SYS$WAITFR

                                       007C 00000       .ENTRY   REPLYBRD_IO, Save R2,R3,R4,R5,R6        ; 0582
                    56       0000'  CF  9E 00002        MOVAB    REPLYBRD_STATUS, R6                     ;
                    55  00000000G   00  9E 00007        MOVAB    SYS$SETAST, R5                          ;
                    5E         B8   AE  9E 0000E        MOVAB    -72(SP), SP                             ;
                                    66  D4 00012        CLRL     REPLYBRD_STATUS                         ; 0631
                    51              08  D0 00014        MOVL     #8, FLAGS                               ; 0632
                    52         04   AC  D0 00017        MOVL     MSG, R2                                 ; 0633
                    54         0C   A2  9E 0001B        MOVAB    12(R2), R4                              ;
                    03         01   A4  E9 0001F        BLBC     1(R4), 1$                               ;
                    51              02  C0 00023        ADDL2    #2, FLAGS                               ; 0635
              03    64              0C  E1 00026 1$:    BBC      #12, (R4), 2$                           ; 0636
                    51              04  C0 0002A        ADDL2    #4, FLAGS                               ; 0638
                    05              64  E9 0002D 2$:    BLBC     (R4), 3$                                ; 0646
                    53              04  D0 00030        MOVL     #4, SNDTYP                              ;
                                    1A  11 00033        BRB      6$                                      ;
                                    64  95 00035 3$:    TSTB     (R4)                                    ; 0647
                                    0F  18 00037        BGEQ     5$                                      ;
                               20   A2  B5 00039        TSTW     32(R2)                                  ;
                                    05  12 0003C        BNEQ     4$                                      ;
                    53              03  D0 0003E        MOVL     #3, SNDTYP                              ;
                                    0C  11 00041        BRB      6$                                      ;
                    53              02  D0 00043 4$:    MOVL     #2, SNDTYP                              ;
                                    07  11 00046        BRB      6$                                      ;
              62    64              05  E1 00048 5$:    BBC      #5, (R4), 12$                           ; 0650
                    53              01  D0 0004C        MOVL     #1, SNDTYP                              ;
              05    64              06  E1 0004F 6$:    BBC      #6, (R4), 7$                            ; 0658
                    50              05  D0 00053        MOVL     #5, REQID                               ;
                                    0B  11 00056        BRB      9$                                      ;
              05    64              04  E1 00058 7$:    BBC      #4, (R4), 8$                            ; 0659
                    50              04  D0 0005C        MOVL     #4, REQID                               ;
                                    02  11 0005F        BRB      9$                                      ;
                                    50  D4 00061 8$:    CLRL     REQID                                   ; 0660
         08         AE              0B  D0 00063 9$:    MOVL     #11, ARGLIST                            ; 0667
         0C         AE         28   A2  9E 00067        MOVAB    40(R2), ARGLIST+4                       ; 0668
                               10   AE  D4 0006C        CLRL     ARGLIST+8                               ; 0669
         14         AE              53  D0 0006F        MOVL     SNDTYP, ARGLIST+12                      ; 0670
         18         AE              20  7D 00073        MOVQ     #32, ARGLIST+16                         ; 0671
         20         AE              50  D0 00077        MOVL     REQID, ARGLIST+24                       ; 0673
                               24   AE  D4 0007B        CLRL     ARGLIST+28                              ; 0674
         28         AE         08   AC  D0 0007E        MOVL     LOCAL_NODE, ARGLIST+32                  ; 0675
         2C         AE         10   A2  D0 00083        MOVL     16(R2), ARGLIST+36                      ; 0676
         30         AE              51  D0 00088        MOVL     FLAGS, ARGLIST+40                       ; 0677
         34         AE              6E  9E 0008C        MOVAB    TRMDSC, ARGLIST+44                      ; 0678
                    6E         14   A2  3C 00090        MOVZWL   20(R2), TRMDSC                          ; 0679
         04         AE         30   A2  9E 00094        MOVAB    48(R2), TRMDSC+4                        ; 0680
                    05         01   A4  E9 00099        BLBC     1(R4), 10$                              ; 0685
                                    7E  D4 0009D        CLRL     -(SP)                                   ; 0687
                    65              01  FB 0009F        CALLS    #1, SYS$SETAST                          ;
              03    01              53  CF 000A2 10$:   CASEL    SNDTYP, #1, #3                          ; 0694
   000C       000C       0038     001E     000A6 11$:   .WORD    15$-11$,-                               ;
                                                                 17$-11$,-                               ;
                                                                 13$-11$,-                               ;
                                                                 13$-11$                                 ;
                    50              14  D0 000AE 12$:   MOVL     #20, R0                                 ; 0725
```

```
                                       04 000B1       RET                                                  ;
          06          64          03   E1 000B2 13$:  BBC     #3, (R4), 14$                                  ; 0698
            24        AE   0000V  CF   9E 000B6       MOVAB   REPLYBRD_NOTIFY_ALL, ARGLIST+28              ; 0699
          FC34        CF      08  AE   FA 000BC 14$:  CALLG   ARGLIST, REPLYBRD_BRKTHRU_QUEUE              ; 0700
                                  43   11 000C2       BRB     20$                                          ; 0694
          06          64          03   E1 000C4 15$:  BBC     #3, (R4), 16$                                ; 0705
            24        AE   0000V  CF   9E 000C8       MOVAB   REPLYBRD_NOTIFY_DEV, ARGLIST+28              ; 0706
                      50      1C  A2   3C 000CE 16$:  MOVZWL  28(R2), R0                                   ; 0707
     3C   AE          52          50   C1 000D2       ADDL3   R0, R2, ARGLIST+52                           ;
          38          AE      1E  A2   3C 000D7       MOVZWL  30(R2), ARGLIST+48                           ; 0708
                                  1F   11 000DC       BRB     19$                                          ; 0709
          06          64          03   E1 000DE 17$:  BBC     #3, (R4), 18$                                ; 0715
            24        AE   0000V  CF   9E 000E2       MOVAB   REPLYBRD_NOTIFY_USE, ARGLIST+28              ; 0716
                      50      1C  A2   3C 000E8 18$:  MOVZWL  28(R2), R0                                   ; 0717
                      50          52   C0 000EC       ADDL2   R2, R0                                       ;
                      51      1E  A2   3C 000EF       MOVZWL  30(R2), R1                                   ; 0718
     3C   AE          50          51   C1 000F3       ADDL3   R1, R0, ARGLIST+52                           ;
          38          AE      20  A2   3C 000F8       MOVZWL  32(R2), ARGLIST+48                           ; 0719
          08          AE          0D   D0 000FD 19$:  MOVL    #13, ARGLIST                                 ; 0720
          0000V       CF      08  AE   FA 00101       CALLG   ARGLIST, REPLYBRD_IO_LIST                    ; 0721
                      50      04  AC   D0 00107 20$:  MOVL    MSG, R0                                      ; 0731
                      1B      0D  A0   E9 0010B       BLBC    13(R0), 21$                                  ;
                                  66   D5 0010F       TSTL    REPLYBRD_STATUS                              ; 0733
                                  17   12 00111       BNEQ    21$                                          ;
                                  02   DD 00113       PUSHL   #2                                           ; 0736
          00000000G   00          01   FB 00115       CALLS   #1, SYS$CLREF                                ;
                                  01   DD 0011C       PUSHL   #1                                           ; 0737
                      65          01   FB 0011E       CALLS   #1, SYS$SETAST                               ;
                                  02   DD 00121       PUSHL   #2                                           ; 0738
          00000000G   00          01   FB 00123       CALLS   #1, SYS$WAITFR                               ;
                      50          66   D0 0012A 21$:  MOVL    REPLYBRD_STATUS, STATUS                      ; 0744
                                  03   12 0012D       BNEQ    22$                                          ;
                      50          01   D0 0012F       MOVL    #1, STATUS                                   ; 0746
                                       04 00132 22$:  RET                                                  ; 0749

; Routine Size:  307 bytes,    Routine Base:  $CODE$ + 040C
```

```
1 GLOBAL ROUTINE replybrd_io_list : NOVALUE =      %SBTTL 'Replybrd_IO_list (callg with AP)'
1
1 !++
1 ! Functional description:
1 !
1 !       This routine sends the formatted message from a broadcast message block to the terminals,
1 !       using the $BRKTHRU system service.
1 !
1 ! Input:
1 !
1 !       msg        - Pointer to RPYBRD message block
1 !       local_node - Pointer to string descriptor with local node name
1 !
1 ! Implicit Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implict output:
1 !
1 !       Messages might be sent to terminals.
1 !
1 ! Side effects:
1 !
1 !       None.
1 !
1 ! Routine value:
1 !
1 !       None.
1 !--
1
2 BEGIN                                                  ! Start of replybrd_io
2
2 BUILTIN
2         AP;
2
2 LOCAL
2         mlen,                                          ! Length of message field
2         mptr            : $ref_bvector,                ! Pointer to message field
2         sendto          : VECTOR [2, LONG];            ! User or terminal name
2
2 BIND
2         arglist = .AP : VECTOR [, LONG];
2
2 mlen = .arglist [llen];                                ! Get length and pointer to list
2 mptr = .arglist [lptr];
2 arglist [targ] = sendto;                               ! Point the arglist at the new desc
2 arglist [count] = queue_args;                          ! Set correct argument count for queue routine
2 !
2 ! We are doing it to a list of terminals or usernames, mptr and mlen are set up
2 !
2 WHILE .mlen GTR 0
2 DO
3     BEGIN
```

```
0807  3        REGISTER
0808  3            len;
0809  3        !
0810  3        ! Build a descriptor to the next ASCIC item, and move the pointers along
0811  3        !
0812  3        len = .mptr [0];                          ! Get the length
0813  3        sendto [0] = .len;                        ! Put the length in the desc
0814  3        sendto [1] = mptr [1];                    ! Put the address in the desc
0815  3        len = .len + 1;                           ! Adjust the length for the count byte
0816  3        mlen = .mlen - .len;                      ! Drop the total length by this item
0817  3        mptr = .mptr + .len;                      ! Bump pointer to next item
0818  3        !
0819  3        ! Queue the call
0820  3        !
0821  3        CALLG (.AP, replybrd_brkthru_queue);
0822  3
0823  2        END;
0824  2
0825  2  RETURN;
0826  1  END;                                        ! End of replybrd_io_list


                                000C 00000          .ENTRY  REPLYBRD_IO_LIST, Save R2,R3          ; 0750
          5E                08   C2 00002          SUBL2   #8, SP                                ;
          53        30      AC   D0 00005          MOVL    48(AP), MLEN                          ; 0797
          52        34      AC   D0 00009          MOVL    52(AP), MPTR                          ; 0798
     08   AC                6E   9E 0000D          MOVAB   SENDTO, 8(AP)                         ; 0799
          6C                0B   D0 00011          MOVL    #11, (AP)                             ; 0800
                            53   D5 00014 1$:      TSTL    MLEN                                  ; 0804
                            18   15 00016          BLEQ    2$                                    ;
          50                62   9A 00018          MOVZBL  (MPTR), LEN                           ; 0812
          6E                80   9E 0001B          MOVAB   (LEN)+, SENDTO                        ; 0813
     04   AE        01      A2   9E 0001E          MOVAB   1(R2), SENDTO+4                       ; 0814
          53                50   C2 00023          SUBL2   LEN, MLEN                             ; 0816
          52                50   C0 00026          ADDL2   LEN, MPTR                             ; 0817
   FB95   CF                6C   FA 00029          CALLG   (AP), REPLYBRD_BRKTHRU_QUEUE          ; 0821
                            E4   11 0002E          BRB     1$                                    ; 0804
                                 04 00030 2$:      RET                                           ; 0826

; Routine Size:  49 bytes,    Routine Base:  $CODE$ + 053F
```

```
1 GLOBAL ROUTINE replybrd_notify_all (bod : $ref_bblock) : NOVALUE = %SBTTL 'Replybrd_notify_all (bod)'
1 !++
1 ! Functional description:
1 !
1 !       Format a message for REPLY /ALL /NOTIFY or REPLY /USER /NOTIFY, then send the message.
1 !
1 ! Input:
1 !
1 !       bod         - Pointer to brkthru output descriptor
1 !
1 ! Implicit Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implict output:
1 !
1 !       None.
1 !
1 ! Side effects:
1 !
1 !       None.
1 !
1 ! Routine value:
1 !
1 !       None.
1 !--
1
2 BEGIN                                              ! Start of replybrd_notify_all
2
2 LOCAL
2       on_buf          : VECTOR [64, BYTE],
2       on_dsc          : VECTOR [2, LONG] INITIAL (64, on_buf),
2       fmt_buf         : VECTOR [512, BYTE],
2       fmt_dsc         : VECTOR [2, LONG] INITIAL (512, fmt_buf),
2       ctr_buf         : VECTOR [256, BYTE],
2       ctr_dsc         : VECTOR [2, LONG] INITIAL (256, ctr_buf),
2       id,
2       status;
2
2 !
2 ! Get the $FAO control string text associated with the message id via $GETMSG.
2 !
2 id = (IF .bod [bod_w_iosb1] EQL 1 THEN opc$_reply_notall ELSE opc$_reply_notalln);
3 IF NOT (status = $GETMSG (MSGID=.id, MSGLEN=ctr_dsc, BUFADR=ctr_dsc, FLAGS=1))
2 THEN
2     RETURN .status;
2 !
2 ! If we have a net or cluster, then set the preposition to precede the node name
2 !
2 IF .bod [bod_l_nodlen] NEQ 0
2 THEN
3     BEGIN
4     IF NOT (status = $GETMSG (MSGID=opc$_on_node, MSGLEN=on_dsc, BUFADR=on_dsc, FLAGS=1))
```

```
  0884 3        THEN
  0885 3            RETURN .status;
  0886 3        END
  0887 2  ELSE
  0888 2      on_dsc [0] = 0;                                    ! Null prepositions
  0889 2  !
  0890 2  ! Format the message, the general form of the control string is:
  0891 2  !
P 0892 3  IF NOT (status = $FAO ( ctr_dsc,                         ! Control string
P 0893 3                      fmt_dsc, fmt_dsc,                  ! Return length, output buffer
P 0894 3                      .bod [bod_w_iosb1],                ! Number of terminals notified
P 0895 3                      on_dsc,                            ! Preposition (like in "on " for "on ATHENS")
P 0896 3                      .bod [bod_l_nodlen],               ! Local node name
  0897 3                      .bod [bod_a_nodptr]))              ! Local node name
  0898 2  THEN
  0899 2      RETURN .status;
  0900 2  !
  0901 2  ! Call the routine to queue the io
  0902 2  !
  0903 2  replybrd_notify_io (.bod, fmt_dsc);
  0904 2
  0905 2  RETURN;
  0906 1  END;                                                   ! End of replybrd_notify_all


                            000C 00000       .ENTRY  REPLYBRD_NOTIFY_ALL, Save R2,R3          ; 0827
          53 00000000G   00   9E 00002       MOVAB   SYS$GETMSG, R3                           ;
          5E       FCAC  CE   9E 00009       MOVAB   -852(SP), SP                             ;
     B8   AD         40  8F   9A 0000E       MOVZBL  #64, ON_DSC                              ; 0858
     BC   AD         C0  AD   9E 00013       MOVAB   ON_BUF, ON_DSC+4                         ;
   0104   CE       0200  8F   3C 00018       MOVZWL  #512, FMT_DSC                            ;
   0108   CE       010C  CE   9E 0001F       MOVAB   FMT_BUF, FMT_DSC+4                       ;
          7E       0100  8F   3C 00026       MOVZWL  #256, CTR_DSC                            ;
     04   AE         08  AE   9E 0002B       MOVAB   CTR_BUF, CTR_DSC+4                       ;
          52         04  AC   D0 00030       MOVL    BOD, R2                                  ; 0873
          01         3E  A2   B1 00034       CMPW    62(R2), #1                               ;
                         09   12 00038       BNEQ    1$                                       ;
          50 00058283    8F   D0 0003A       MOVL    #361091, ID                              ;
                         07   11 00041       BRB     2$                                       ;
          50 0005828B    8F   D0 00043 1$:   MOVL    #361099, ID                              ;
          7E             01   7D 0004A 2$:   MOVQ    #1, -(SP)                                ; 0874
                     08  AE   9F 0004D       PUSHAB  CTR_DSC                                  ;
                     0C  AE   9F 00050       PUSHAB  CTR_DSC                                  ;
                         50   DD 00053       PUSHL   ID                                       ;
          63             05   FB 00055       CALLS   #5, SYS$GETMSG                           ;
          49             50   E9 00058       BLBC    STATUS, 5$                               ;
                     18  A2   D5 0005B       TSTL    24(R2)                                   ; 0880
                         16   13 0005E       BEQL    3$                                       ;
          7E             01   7D 00060       MOVQ    #1, -(SP)                                ; 0883
                     B8  AD   9F 00063       PUSHAB  ON_DSC                                   ;
                     B8  AD   9F 00066       PUSHAB  ON_DSC                                   ;
             000582AB    8F   DD 00069       PUSHL   #361131                                  ;
          63             05   FB 0006F       CALLS   #5, SYS$GETMSG                           ;
          04             50   E8 00072       BLBS    STATUS, 4$                               ;
```

```
                                       04 00075        RET                                  ; 0885
                           B8    AD    D4 00076 3$:    CLRL     ON_DSC                      ; 0888
              7E           18    A2    7D 00079 4$:    MOVQ     24(R2), -(SP)               ; 0897
                           B8    AD    9F 0007D        PUSHAB   ON_DSC                      ;
              7E           3E    A2    3C 00080        MOVZWL   62(R2), -(SP)               ;
                         0118    CE    9F 00084        PUSHAB   FMT_DSC                     ;
                         011C    CE    9F 00088        PUSHAB   FMT_DSC                     ;
                           18    AE    9F 0008C        PUSHAB   CTR_DSC                     ;
       00000000G  00                   07 FB 0008F     CALLS    #7, SYS$FAO                 ;
                  0B                   50 E9 00096     BLBC     STATUS, 5$                  ;
                         0108    CE    9F 00099        PUSHAB   FMT_DSC                     ; 0903
                                       52 DD 0009D     PUSHL    R2                          ;
           0000V  CF                   02 FB 0009F     CALLS    #2, REPLYBRD_NOTIFY_IO      ;
                                       04 000A4 5$:    RET                                  ; 0906

; Routine Size:  165 bytes,    Routine Base:  $CODE$ + 0570
```

```
GLOBAL ROUTINE replybrd_notify_dev (bod : $ref_bblock) : NOVALUE = %SBTTL 'Replybrd_notify_dev (bod)'

!++
! Functional description:
!
!       Format a message for REPLY /NOTIFY /TERMINAL, then send the message.
!
! Input:
!
!       bod         - Pointer to brkthru output descriptor
!
! Implicit Input:
!
!       None.
!
! Output:
!
!       None.
!
! Implict output:
!
!       None.
!
! Side effects:
!
!       None.
!
! Routine value:
!
!       None.
!--

BEGIN                                               ! Start of replybrd_notify_dev

LOCAL
        fmt_buf         : VECTOR [512, BYTE],
        fmt_dsc         : VECTOR [2, LONG] INITIAL (512, fmt_buf),
        ctr_buf         : VECTOR [256, BYTE],
        ctr_dsc         : VECTOR [2, LONG] INITIAL (256, ctr_buf),
        full_desc       : $ref_bblock,
        status;

!
! Get the $FAO control string text associated with the message id via $GETMSG.
!
IF NOT (status = $GETMSG (MSGID=opc$_reply_notdev, MSGLEN=ctr_dsc, BUFADR=ctr_dsc, FLAGS=1))
THEN
    RETURN .status;
!
! Get the full device name
!
full_desc = share_full_devname (bod [bod_q_sendto], dvi$_fulldevnam);
!
! Format the message
!
IF NOT (status = $FAO ( ctr_dsc,                    ! Control string
                        fmt_dsc, fmt_dsc,           ! Return length, output buffer
```

```
0964 3                         .full_desc))                         ! Device name
0965 2 THEN
0966 2     RETURN .status;
0967 2 !
0968 2 ! Call the routine to dispose of it
0969 2 !
0970 2 replybrd_notify_io (.bod, fmt_dsc);
0971 2
0972 2 RETURN;
0973 1 END;                                                     ! End of replybrd_notify_dev



                                  0004 00000       .ENTRY  REPLYBRD_NOTIFY_DEV, Save R2                ; 0907
                 5E     FCF4  CE  9E 00002         MOVAB   -780(SP), SP
          0104   CE     0200  8F  3C 00007         MOVZWL  #512, FMT_DSC                               ; 0939
          0108   CE     010C  CE  9E 0000E         MOVAB   FMT_BUF, FMT_DSC+4                          ;
                 7E     0100  8F  3C 00015         MOVZWL  #256, CTR_DSC                               ;
            04   AE       08  AE  9E 0001A         MOVAB   CTR_BUF, CTR_DSC+4                          ;
                 7E           01  7D 0001F         MOVQ    #1, -(SP)                                   ; 0952
                          08  AE  9F 00022         PUSHAB  CTR_DSC                                     ;
                          0C  AE  9F 00025         PUSHAB  CTR_DSC                                     ;
                    00058293  8F  DD 00028         PUSHL   #361107                                     ;
     00000000G   00               05  FB 0002E     CALLS   #5, SYS$GETMSG                              ;
                 52           50  D0 00035         MOVL    R0, STATUS                                  ;
                 34           52  E9 00038         BLBC    STATUS, 1$                                  ;
                 7E       E8  8F  9A 0003B         MOVZBL  #232, -(SP)                                 ; 0958
   7E       04   AC           30  C1 0003F         ADDL3   #48, BOD, -(SP)                             ;
         0000G   CF           02  FB 00044         CALLS   #2, SHARE_FULL_DEVNAME                      ;
                              50  DD 00049         PUSHL   FULL_DESC                                   ; 0964
                        010C  CE  9F 0004B         PUSHAB  FMT_DSC                                     ;
                        0110  CE  9F 0004F         PUSHAB  FMT_DSC                                     ;
                          0C  AE  9F 00053         PUSHAB  CTR_DSC                                     ;
     00000000G   00               04  FB 00056     CALLS   #4, SYS$FAO                                 ;
                 52           50  D0 0005D         MOVL    R0, STATUS                                  ;
                 0C           52  E9 00060         BLBC    STATUS, 1$                                  ;
                        0108  CE  9F 00063         PUSHAB  FMT_DSC                                     ; 0970
                          04  AC  DD 00067         PUSHL   BOD                                         ;
         0000V   CF           02  FB 0006A         CALLS   #2, REPLYBRD_NOTIFY_IO                      ;
                              04 0006F  1$:        RET                                                 ; 0973

; Routine Size:  112 bytes,    Routine Base:  $CODE$ + 0615
```

```
GLOBAL ROUTINE replybrd_notify_io (bod : $ref_bblock, text : $ref_bblock) : NOVALUE =   %SBTTL 'Replybrd_not

!++
! Functional description:
!
!       This routine sends the formatted message from a reply notification back to the sending
!       terminal.
!
! Input:
!
!       bod        - Pointer to brkthru output descriptor
!       text       - Pointer to string descriptor with reply notification message
!
! Implicit Input:
!
!       None.
!
! Output:
!
!       None.
!
! Implict output:
!
!       Messages might be sent to terminals.
!
! Side effects:
!
!       None.
!
! Routine value:
!
!       None.
!--

BEGIN                                                   ! Start of replybrd_notify_io

LOCAL
        rpynot  : $bblock [512],
        ptr     : $ref_bvector,
        len,
        status;

!
! If /WAIT, then we can simply write it to SYS$OUTPUT
!
IF .bod [bod_v_wait]
THEN
    RETURN lib$put_output (.text);

!
! Following code executes on OPCOM process only, cannot be executed from within REPLY image
!

!
! If on the local node, queue up the broadcast from here and return
!
IF .bod [bod_v_local_node]
```

```
2 THEN
3       BEGIN
3       replybrd_brkthru_queue (
3                       .text,                          ! text to send
3                       bod [bod_g_trmdsc],             ! target
3                       brk$c_device,                   ! send type
3                       32,                             ! carriage control
3                       0,                              ! brkthru flags
3                       brk$c_general,                  ! request id
3                       0,0,0,0,0);                     ! no completion routine or arguments
3       RETURN;
2       END;
2 !
2 ! Not wait or local, send it back to the OPCOM on the host node.  First init the message header.
2 !
2 CH$FILL (0, rpynot_k_min_size, rpynot);
2 rpynot [clm_b_rqstcode]           = opc$_x_clusmsg;
2 rpynot [clm_b_clm_code]           = clm__rpynot;
2 rpynot [clm_b_ds_version]         = rpynot_k_ds_version;
2 rpynot [clm_b_sw_version]         = opc$k_sw_version;
2 rpynot [clm_l_csid]               = .lcl_csid;
2 !
2 ! Copy the originating terminal to the message
2 !
2 ptr = rpynot [rpynot_t_text];
2 len = .bod [bod_l_trmlen];
2 rpynot [rpynot_w_term_len] = .len;
2 CH$MOVE (.len, .bod [bod_a_trmptr], .ptr);
2 ptr = .ptr + .len;
2 !
2 ! Copy the text to the message
2 !
2 len = .text [dsc$w_length];
2 rpynot [rpynot_w_message_len] = .len;
2 CH$MOVE (.len, .text [dsc$a_pointer], .ptr);
2 ptr = .ptr + .len;
2 !
2 ! Store the final length and send it back from whence it came
2 !
2 len = .ptr - rpynot;
2 rpynot [clm_w_length] = .len;
2 cluscomm_send (.bod [bod_l_csid], .len, rpynot);
2
2 RETURN;
1 END;                                                  ! End of replybrd_notify_io


                                                .EXTRN  LIB$PUT_OUTPUT

                                01FC 00000      .ENTRY  REPLYBRD_NOTIFY_IO, Save R2,R3,R4,R5,R6,R7,-; 0974
                                                        R8                                          ;
              5E      FE00  CE  9E 00002        MOVAB   -512(SP), SP                                ;
              56        04  AC  D0 00007        MOVL    BOD, R6                                     ; 1019
     0B    0C A6            01  E1 0000B        BBC     #1, 12(R6), 1$                              ;
                        08  AC  DD 00010        PUSHL   TEXT                                        ; 1021
       00000000G  00        01  FB 00013        CALLS   #1, LIB$PUT_OUTPUT                          ;
```

```
                                              04 0001A        RET                                            ;
            17        0C    A6           02   E1 0001B 1$:    BBC      #2, 12(R6), 2$                        ; 1030
                                         7E   7C 00020        CLRQ     -(SP)                                 ; 1035
                                         7E   7C 00022        CLRQ     -(SP)                                 ;
                                         7E   7C 00024        CLRQ     -(SP)                                 ;
                            7E           20   7D 00026        MOVQ     #32, -(SP)                            ;
                                         01   DD 00029        PUSHL    #1                                    ;
                                   20    A6   9F 0002B        PUSHAB   32(R6)                                ;
                                   08    AC   DD 0002E        PUSHL    TEXT                                  ;
                    FA47    CF           0B   FB 00031        CALLS    #11, REPLYBRD_BRKTHRU_QUEUE           ;
                                              04 00036        RET                                            ; 1032
   10               00      6E           00   2C 00037 2$:    MOVC5    #0, (SP), #0, #16, RPYNOT             ; 1046
                                         6E      0003C                                                       ;
                            6E 09021113  8F   D0 0003D        MOVL     #151130387, RPYNOT                    ; 1047
                    08      AE     0000G CF   D0 00044        MOVL     LCL_CSID, RPYNOT+8                    ; 1051
                            58        10 AE   9E 0004A        MOVAB    RPYNOT+16, PTR                        ; 1055
                            57        20 A6   D0 0004E        MOVL     32(R6), LEN                           ; 1056
                    0C      AE           57   B0 00052        MOVW     LEN, RPYNOT+12                        ; 1057
            68      24      B6           57   28 00056        MOVC3    LEN, @36(R6), (PTR)                   ; 1058
                            58           57   C0 0005B        ADDL2    LEN, PTR                              ; 1059
                            50        08 AC   D0 0005E        MOVL     TEXT, R0                              ; 1063
                            57           60   3C 00062        MOVZWL   (R0), LEN                             ;
                    0E      AE           57   B0 00065        MOVW     LEN, RPYNOT+14                        ; 1064
            68      04      B0           57   28 00069        MOVC3    LEN, @4(R0), (PTR)                    ; 1065
                            58           57   C0 0006E        ADDL2    LEN, PTR                              ; 1066
                            50           6E   9E 00071        MOVAB    RPYNOT, R0                            ; 1070
            57              58           50   C3 00074        SUBL3    R0, PTR, LEN                          ;
                    04      AE           57   B0 00078        MOVW     LEN, RPYNOT+4                         ; 1071
                                4080     8F   BB 0007C        PUSHR    #^M<R7,SP>                            ; 1072
                                  14     A6   DD 00080        PUSHL    20(R6)                                ;
                    0000G   CF           03   FB 00083        CALLS    #3, CLUSCOMM_SEND                     ;
                                              04 00088        RET                                            ; 1075

; Routine Size:  137 bytes,    Routine Base:  $CODE$ + 0685
```

```
GLOBAL ROUTINE replybrd_notify_use (bod : $ref_bblock) : NOVALUE = %SBTTL 'Replybrd_notify_use (bod)'
!++
! Functional description:
!
!       Format a message for REPLY /USER=(...) /NOTIFY, then send the message.
!
! Input:
!
!       bod         - Pointer to brkthru output descriptor
!
! Implicit Input:
!
!       None.
!
! Output:
!
!       None.
!
! Implict output:
!
!       None.
!
! Side effects:
!
!       None.
!
! Routine value:
!
!       None.
!--

BEGIN                                           ! Start of replybrd_notify_use

LOCAL
        on_buf          : VECTOR [64, BYTE],
        on_dsc          : VECTOR [2, LONG] INITIAL (64, on_buf),
        fmt_buf         : VECTOR [512, BYTE],
        fmt_dsc         : VECTOR [2, LONG] INITIAL (512, fmt_buf),
        ctr_buf         : VECTOR [256, BYTE],
        ctr_dsc         : VECTOR [2, LONG] INITIAL (256, ctr_buf),
        id,
        status;

!
! Get the $FAO control string text associated with the message id via $GETMSG.
!
id = (IF .bod [bod_w_iosb1] EQL 1 THEN opc$_reply_notuse ELSE opc$_reply_notusen);
IF NOT (status = $GETMSG (MSGID=.id, MSGLEN=ctr_dsc, BUFADR=ctr_dsc, FLAGS=1))
THEN
    RETURN .status;
!
! If we have a net or cluster, then set the preposition to precede the node name
!
IF .bod [bod_l_nodlen] NEQ 0
THEN
    BEGIN
    IF NOT (status = $GETMSG (MSGID=opc$_on_node, MSGLEN=on_dsc, BUFADR=on_dsc, FLAGS=1))
```

```
; 1143        1133 3       THEN
; 1144        1134 3           RETURN .status;
; 1145        1135 3       END
; 1146        1136 2   ELSE
; 1147        1137 2       on_dsc [0] = 0;                                  ! Null prepositions
; 1148        1138 2   !
; 1149        1139 2   ! Format the message, the general form of the control string is:
; 1150        1140 2   !
; 1151      P 1141 3   IF NOT (status = $FAO ( ctr_dsc,                     ! Control string
; 1152      P 1142 3                       fmt_dsc, fmt_dsc,                ! Return length, output buffer
; 1153      P 1143 3                       bod [bod_q_sendto],              ! Username
; 1154      P 1144 3                       on_dsc,                          ! Preposition (like in "on " for "on ATHENS")
; 1155      P 1145 3                       .bod [bod_l_nodlen],             ! Local node name
; 1156      P 1146 3                       .bod [bod_a_nodptr],             ! Local node name
; 1157        1147 3                       .bod [bod_w_iosb1]))             ! Number of terminals notified
; 1158        1148 2   THEN
; 1159        1149 2       RETURN .status;
; 1160        1150 2   !
; 1161        1151 2   ! Call the routine to dispose of it
; 1162        1152 2   !
; 1163        1153 2   replybrd_notify_io (.bod, fmt_dsc);
; 1164        1154 2
; 1165        1155 2   RETURN;
; 1166        1156 1   END;                                                 ! End of replybrd_notify_all


                                000C 00000          .ENTRY  REPLYBRD_NOTIFY_USE, Save R2,R3          ; 1076
           53 00000000G   00     9E 00002          MOVAB   SYS$GETMSG, R3                            ;
           5E       FCAC  CE     9E 00009          MOVAB   -852(SP), SP                              ;
      B8   AD         40  8F     9A 0000E          MOVZBL  #64, ON_DSC                               ; 1107
      BC   AD         C0  AD     9E 00013          MOVAB   ON_BUF, ON_DSC+4                          ;
    0104   CE       0200  8F     3C 00018          MOVZWL  #512, FMT_DSC                             ;
    0108   CE       010C  CE     9E 0001F          MOVAB   FMT_BUF, FMT_DSC+4                        ;
           7E       0100  8F     3C 00026          MOVZWL  #256, CTR_DSC                             ;
      04   AE         08  AE     9E 0002B          MOVAB   CTR_BUF, CTR_DSC+4                        ;
           52         04  AC     D0 00030          MOVL    BOD, R2                                   ; 1122
           01         3E  A2     B1 00034          CMPW    62(R2), #1                                ;
                          09     12 00038          BNEQ    1$                                        ;
           50 0005829B    8F     D0 0003A          MOVL    #361115, ID                               ;
                          07     11 00041          BRB     2$                                        ;
           50 000582A3    8F     D0 00043 1$:      MOVL    #361123, ID                               ;
           7E             01     7D 0004A 2$:      MOVQ    #1, -(SP)                                 ; 1123
                      08  AE     9F 0004D          PUSHAB  CTR_DSC                                   ;
                      0C  AE     9F 00050          PUSHAB  CTR_DSC                                   ;
                          50     DD 00053          PUSHL   ID                                        ;
           63             05     FB 00055          CALLS   #5, SYS$GETMSG                            ;
           4C             50     E9 00058          BLBC    STATUS, 5$                                ;
                      18  A2     D5 0005B          TSTL    24(R2)                                    ; 1129
                          16     13 0005E          BEQL    3$                                        ;
           7E             01     7D 00060          MOVQ    #1, -(SP)                                 ; 1132
                      B8  AD     9F 00063          PUSHAB  ON_DSC                                    ;
                      B8  AD     9F 00066          PUSHAB  ON_DSC                                    ;
                000582AB  8F     DD 00069          PUSHL   #361131                                   ;
           63             05     FB 0006F          CALLS   #5, SYS$GETMSG                            ;
```

```
               04        50  E8 00072          BLBS    STATUS, 4$                   ;
                             04 00075          RET                                  ; 1134
                   B8    AD  D4 00076 3$:      CLRL    ON_DSC                       ; 1137
               7E  3E    A2  3C 00079 4$:      MOVZWL  62(R2), -(SP)                ; 1147
               7E  18    A2  7D 0007D          MOVQ    24(R2), -(SP)                ;
                   B8    AD  9F 00081          PUSHAB  ON_DSC                       ;
                   30    A2  9F 00084          PUSHAB  48(R2)                       ;
                 011C    CE  9F 00087          PUSHAB  FMT_DSC                      ;
                 0120    CE  9F 0008B          PUSHAB  FMT_DSC                      ;
                   1C    AE  9F 0008F          PUSHAB  CTR_DSC                      ;
      00000000G    00        08 FB 00092       CALLS   #8, SYS$FAO                  ;
               0B        50  E9 00099          BLBC    STATUS, 5$                   ;
                 0108    CE  9F 0009C          PUSHAB  FMT_DSC                      ; 1153
                             52 DD 000A0       PUSHL   R2                           ;
           FED0    CF        02 FB 000A2       CALLS   #2, REPLYBRD_NOTIFY_IO       ;
                             04 000A7 5$:      RET                                  ; 1156

; Routine Size:  168 bytes,    Routine Base:  $CODE$ + 070E
```

```
1 GLOBAL ROUTINE replybrd_ok_to_queue_write (bod : $ref_bblock, queue : $ref_bblock) =
1
1 !++
1 ! Functional descripton:
1 !
1 !       Check to see if the SENDTO field in the BOD is in any of the bods in the queue.
1 !       We assume that we are operating at AST level so that we do not have to worry
1 !       about interlocking the queue.  After counting the number of times it appears
1 !       in the queue, make a decision as to whether another write can be started.
1 !
1 ! Input:
1 !       bod      pointer to a bod
1 !       queue    head of a queue of bods
1 !
1 ! Output:
1 !       None.
1 !
1 ! Routine Value:
1 !       true if OK to send another write, false otherwise
1 !--
1
2 BEGIN                                          ! Start of REPLYBRD_OK_TO_QUEUE_WRITE
2
2 LOCAL
2     count,
2     ok,
2     head : $ref_bblock,
2     cur : $ref_bblock;
2
2 !
2 ! Scan the queue, counting the number of times the target appears
2 !
2 count = 0;
2 head = .queue;
2 cur = .head [bod_l_flink];
2 WHILE .cur NEQ .head                           ! Loop until we see the end
2 DO
3     BEGIN
3     IF CH$EQL (bod_s_senbuf, bod [bod_t_senbuf], bod_s_senbuf, cur [bod_t_senbuf])
3     THEN
3         count = .count + 1;
3     cur = .cur [bod_l_flink];                  ! Get the next bod
2     END;
2 !
2 ! Now make a decision based on the type of breakthru
2 !
2 ok = false;
2 CASE .bod [bod_l_sndtyp]
2 FROM MIN (brk$c_device, brk$c_username, brk$c_allusers, brk$c_allterms)
2 TO   MAX (brk$c_device, brk$c_username, brk$c_allusers, brk$c_allterms) OF
2 SET
2     [brk$c_username, brk$c_device] :
3                         BEGIN
3                         IF .count EQL 0
3                         THEN ok = true                     ! Only one to devices or users
3                         ELSE
4                             BEGIN
```

```
; 1225    1214 4                      IF CH$FIND_SUB (bod_s_senbuf, bod [bod_t_senbuf], 4, UPLIT BYTE ('OPA0')) NEQ 0
; 1226    1215 4                       AND
; 1227    1216 4                         .count LEQ 2                       ! Allow 3 queued to the console
; 1228    1217 4                      THEN
; 1229    1218 4                         ok = true;
; 1230    1219 3                      END;
; 1231    1220 2                  END;
; 1232    1221 2      [brk$c_allusers, brk$c_allterms] :
; 1233    1222 2                  IF .count LEQ 3 THEN ok = true;           ! Allow four if alldevices or allusers
; 1234    1223 2      [INRANGE, OUTRANGE] :
; 1235    1224 2                  $signal_stop (ss$_illiofunc);             ! If anything else, we did bad
; 1236    1225 2  TES;
; 1237    1226 2
; 1238    1227 2  RETURN .ok;
; 1239    1228 1  END;                                  ! End of REPLYBRD_OK_TO_QUEUE_WRITE


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                           30  41  50  4F  00004 P.AAB: .ASCII  \OPA0\                                           ;


                                                        .PSECT  $CODE$,NOWRT,2

                                   00FC  00000          .ENTRY  REPLYBRD_OK_TO_QUEUE_WRITE, Save R2,R3,R4,-  ; 1157
                                                                R5,R6,R7                                     ;
                                   57  D4  00002        CLRL    COUNT                                        ; 1189
                     56       08   AC  D0  00004        MOVL    QUEUE, HEAD                                  ; 1190
                     55            66  D0  00008        MOVL    (HEAD), CUR                                  ; 1191
                     54       04   AC  D0  0000B        MOVL    BOD, R4                                      ; 1195
                     56            55  D1  0000F 1$:    CMPL    CUR, HEAD                                    ; 1192
                                   11  13  00012        BEQL    3$                                           ;
         58  A5      58  A4  0040  8F  29  00014        CMPC3   #64, 88(R4), 88(CUR)                         ; 1195
                                   02  12  0001C        BNEQ    2$                                           ;
                                   57  D6  0001E        INCL    COUNT                                        ; 1197
                     55            65  D0  00020 2$:    MOVL    (CUR), CUR                                   ; 1198
                                   EA  11  00023        BRB     1$                                           ; 1192
                                   56  D4  00025 3$:    CLRL    OK                                           ; 1203
             03      01       38   A4  CF  00027        CASEL   56(R4), #1, #3                               ; 1204
   0031    0031    0014     0014       0002C 4$:        .WORD   5$-4$,-                                      ;
                                                                5$-4$,-                                      ;
                                                                7$-4$,-                                      ;
                                                                7$-4$                                        ;
                     7E       F4   8F  9A  00034        MOVZBL  #244, -(SP)                                  ; 1224
          00000000G  00            01  FB  00038        CALLS   #1, LIB$STOP                                 ;
                                   04      0003F        RET                                                  ;
                                   57  D5  00040 5$:    TSTL    COUNT                                        ; 1210
                                   1E  13  00042        BEQL    9$                                           ;
  58  A4  0040  8F  0000'  CF      04  39  00044        MATCHC  #4, P.AAB, #64, 88(R4)                       ; 1214
                                   03  13  0004E        BEQL    6$                                           ;
                     53            04  D0  00050        MOVL    #4, R3                                       ;
                     53            04  C2  00053 6$:    SUBL2   #4, R3                                       ;
                                   0D  13  00056        BEQL    10$                                          ;
                     02            57  D1  00058        CMPL    COUNT, #2                                    ; 1216
                                   03  11  0005B        BRB     8$                                           ;
```

```
            03          57  D1 0005D 7$:     CMPL    COUNT, #3                      ; 1222
                        03  14 00060 8$:     BGTR    10$                            ;
            56          01  D0 00062 9$:     MOVL    #1, OK                         ;
            50          56  D0 00065 10$:    MOVL    OK, R0                         ; 1227
                            04 00068         RET                                    ; 1228

; Routine Size:  105 bytes,    Routine Base:  $CODE$ + 07B6
```

```
; 1241      1229  1 END
; 1242      1230  0 ELUDOM                                        ! End of replybrd



;                                  PSECT SUMMARY
;
;       Name                    Bytes                     Attributes
;
;   $GLOBAL$                       56  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                       2079  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $PLIT$                          8  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                            Library Statistics
;
;                                          -------- Symbols --------      Pages      Processing
;       File                               Total   Loaded  Percent       Mapped     Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1          18619       39        0        1000        00:01.8
;   _$255$DUA28:[OPCOM.OBJ]OPCOMLIB.L32;1    633       89       14          43        00:00.9



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



;                                  COMMAND QUALIFIERS

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

; Size:          2079 code + 64 data bytes
; Run Time:         00:42.3
; Elapsed Time:     01:51.3
; Lines/CPU Min:      1745
; Lexemes/CPU-Min: 23186
; Memory Used:   207 pages
; Compilation Complete
```

REPLYBRD
LIS

SECURITY
LIS

OPRENABLE
LIS

REPLYMAIN
LIS

RQSTMAIN
LIS