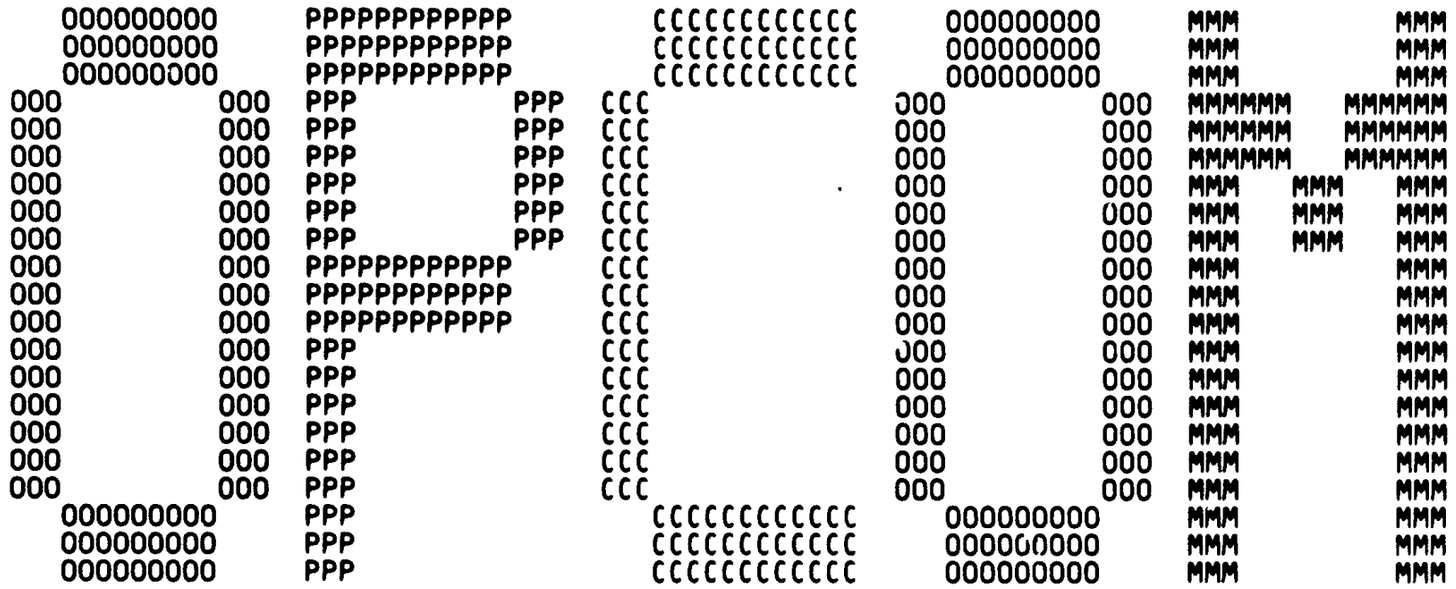

_$2

Sym

ALL

ASC

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

CLU
CLU
CLU
CLU
CLU
CLU
CLU

CLU
CLU
CLU
CLU
CLU
CLU
CLU
CLU

CLU
CLU

FILEID**REPLYBRD

OP
VO

```
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
```

```
;     1          0001  0 MODULE  opc$replybrd    (       %TITLE 'Broadcast command module' %SBTTL 'Copyright notice'
;     2          0002  0                          LANGUAGE (BLISS32),
;     3          0003  0                          IDENT = 'V04-000'
;     4          0004  0                         ) =
;     5          0005  0
;     6          0006  0 !*****************************************************************************
;     7          0007  0 !*                                                                           *
;     8          0008  0 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  0 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  0 !*  ALL RIGHTS RESERVED.                                                     *
;    11          0011  0 !*                                                                           *
;    12          0012  0 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  0 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  0 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  0 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  0 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  0 !*  TRANSFERRED.                                                             *
;    18          0018  0 !*                                                                           *
;    19          0019  0 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  0 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  0 !*  CORPORATION.                                                             *
;    22          0022  0 !*                                                                           *
;    23          0023  0 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  0 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25          0025  0 !*                                                                           *
;    26          0026  0 !*                                                                           *
;    27          0027  0 !*****************************************************************************
;    28          0028  0
;    29          0029  0 !++
;    30          0030  0 ! FACILITY:
;    31          0031  0 !
;    32          0032  0 !       OPCOM - OPCOM process and REPLY command
;    33          0033  0 !
;    34          0034  0 ! ABSTRACT:
;    35          0035  0 !
;    36          0036  0 !       This module formats and outputs user broadcast requests.
;    37          0037  0 !
;    38          0038  0 ! Environment:
;    39          0039  0 !
;    40          0040  0 !       VAX/VMS operating system.
;    41          0041  0 !
;    42          0042  0 ! Author:
;    43          0043  0 !
;    44          0044  0 !       CW Hobbs
;    45          0045  0 !
;    46          0046  0 ! Creation date:
;    47          0047  0 !
;    48          0048  0 !       3-Aug-1983
;    49          0049  0 !
;    50          0050  0 ! Revision history:
;    51          0051  0 !
;    52          0052  0 !       V03-003 CWH3169         CW Hobbs                        5-May-1984
;    53          0053  0 !               Second pass for cluster-wide OPCOM:
;    54          0054  0 !                - Liberal rewrite to use ast-driven, queued brkthru writes.
;    55          0055  0 !
;    56          0056  0 !       V03-002 CWH3002         CW Hobbs                      16-Sep-1983
;    57          0057  0 !               Improve timeout calculation and error handling.
```

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

```
;    60          0059  1 BEGIN                                       %SBTTL 'Start of replybrd'
;    61          0060  1
;    62          0061  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    63          0062  1 LIBRARY 'LIB$:OPCOMLIB';
;    64          0063  1
;    65          0064  1 FORWARD ROUTINE
;    66          0065  1     replybrd_brkthru_ast        : NOVALUE,      ! AST routine for brkthru completion
;    67          0066  1     replybrd_brkthru_queue      : NOVALUE,      ! Add an entry to the brkthru work queue
;    68          0067  1     replybrd_format,                            ! Format the message
;    69          0068  1     replybrd_io,                                ! Do the actual io to the devices
;    70          0069  1     replybrd_io_list            : NOVALUE,      ! Do the actual io to a list of targets
;    71          0070  1     replybrd_notify_all         : NOVALUE,      ! Format message for /ALL or /USER
;    72          0071  1     replybrd_notify_dev         : NOVALUE,      ! Format message for /TERM=(...)
;    73          0072  1     replybrd_notify_io          : NOVALUE,      ! Do something to actually display the message
;    74          0073  1     replybrd_notify_use         : NOVALUE,      ! Format message for /USER=(...)
;    75          0074  1     replybrd_ok_to_queue_write;                 ! Is it ok to actually start the $brkthru?
;    76          0075  1
;    77          0076  1 EXTERNAL ROUTINE
;    78          0077  1     cluscomm_send : WEAK,                       ! The reply image doesn't need this
;    79          0078  1     share_full_devname;
;    80          0079  1
;    81          0080  1 EXTERNAL
;    82          0081  1     lcl_csid : WEAK;                            ! The reply image doesn't need this
;    83          0082  1
;    84          0083  1 GLOBAL
;    85          0084  1     bod_allocated,                              ! Count of BODs created
;    86          0085  1     bod_busy_count,                             ! Current count of i/os pending
;    87          0086  1     bod_busy_max        : INITIAL (32),         ! Maximum number of brkthrus pending
;    88          0087  1     bod_requests,                               ! Count of requests made
;    89          0088  1     bod_busy_queue      : VECTOR [2, LONG]      ! Queue of BODs pending for I/O
;    90          0089  1                           INITIAL (REP 2 OF (bod_busy_queue)),
;    91          0090  1     bod_free_queue      : VECTOR [2, LONG]      ! Queue of BODs available for use
;    92          0091  1                           INITIAL (REP 2 OF (bod_free_queue)),
;    93          0092  1     bod_wait_queue      : VECTOR [2, LONG]      ! Queue of BODs waiting for actual $BRKTHRU to be queued
;    94          0093  1                           INITIAL (REP 2 OF (bod_wait_queue)),
;    95          0094  1     bod_garbage_queue   : VECTOR [2, LONG]      ! Pointer to list of virtual memory to deallocate
;    96          0095  1                           INITIAL (REP 2 OF (bod_garbage_queue)),
;    97          0096  1     replybrd_status,                            ! Status for REPLY image
;    98          0097  1     reply_image         : INITIAL (0);          ! Zero if in OPCOM image, one if in REPLY image
;    99          0098  1
;   100          0099  1 !
;   101          0100  1 ! A macro to put virtual memory back on the queue of garbage to be deallocated
;   102          0101  1 !
;   103          0102  1 MACRO
;   104        M 0103  1     COLLECT_GARBAGE (INP_DESC) =
;   105        M 0104  1                     BEGIN
;   106        M 0105  1                     BIND
;   107        M 0106  1                         desc = (INP_DESC) : VECTOR [, LONG],
;   108        M 0107  1                         garbage = .desc [1] : VECTOR [, LONG];
;   109        M 0108  1                     garbage [2] = .desc [0];                ! Store length as second longword in block
;   110        M 0109  1                     $queue_insert_tail (garbage, bod_garbage_queue);
;   111          0110  1                     END %;
;   112          0111  1
;   113          0112  1 BUILTIN
;   114          0113  1     CALLG;
;   115          0114  1
;   116          0115  1 !
```

```
;  117          0116  1 ! Define the positions in the argument list for the calls to replybrd_brkthru_queue and replybrd_io_list
;  118          0117  1 !
;  119          0118  1 LITERAL
;  120          0119  1     count = 0,          ! ar
;  121          0120  1     text = 1,
;  122          0121  1     targ = 2,
;  123          0122  1     type = 3,
;  124          0123  1     cntl = 4,
;  125          0124  1     bflg = 5,
;  126          0125  1     rqid = 6,
;  127          0126  1     rtn = 7,
;  128          0127  1     node = 8,
;  129          0128  1     csid = 9,
;  130          0129  1     sflg = 10,
;  131          0130  1     term = 11,         ! last arg for replybrd_brkthru_queue
;  132          0131  1     queue_args = 11,
;  133          0132  1     llen = 12,
;  134          0133  1     lptr = 13;
```

```
;  136          0134  1 GLOBAL ROUTINE replybrd_brkthru_ast (parm) : NOVALUE =
;  137          0135  1
;  138          0136  1 !++
;  139          0137  1 ! Functional descripton:
;  140          0138  1 !
;  141          0139  1 !       This routine is the I/O completion for a $BRKTHRU write
;  142          0140  1 !
;  143          0141  1 ! Input:
;  144          0142  1 !       None.
;  145          0143  1 !
;  146          0144  1 ! Output:
;  147          0145  1 !       None.
;  148          0146  1 !
;  149          0147  1 ! Routine Value:
;  150          0148  1 !       Address of block
;  151          0149  1 !--
;  152          0150  1
;  153          0151  2 BEGIN                                           ! Start of REPLYBRD_BRKTHRU_AST
;  154          0152  2
;  155          0153  2 LOCAL
;  156          0154  2     bod : $ref_bblock,
;  157          0155  2     status;
;  158          0156  2
;  159          0157  2 !
;  160          0158  2 ! If the parameter is non-zero, release that block
;  161          0159  2 !
;  162          0160  2 bod = .parm;
;  163          0161  2 IF .bod NEQ 0
;  164          0162  2 THEN
;  165          0163  3     BEGIN
;  166          0164  3     !
;  167          0165  3     ! Check the I/O status, if not ok, save the worst status.  If ok,
;  168          0166  3     ! call the completion routine if there is one
;  169          0167  3     !
;  170          0168  3     IF NOT .bod [bod_w_iosb0]
;  171          0169  3     THEN
;  172          0170  4         BEGIN
;  173          0171  4         IF .(bod [bod_q_iosb])<0,3,0> GTR .replybrd_status<0,3,0>
;  174          0172  4         THEN
;  175          0173  4             replybrd_status = .bod [bod_w_iosb0];
;  176          0174  4         END
;  177          0175  3     ELSE
;  178          0176  3         IF .bod [bod_a_completion_routine] NEQ 0
;  179          0177  3         THEN
;  180          0178  3             (.bod [bod_a_completion_routine]) (.bod);
;  181          0179  3     !
;  182          0180  3     ! Free the message string buffer
;  183          0181  3     !
;  184          0182  3     collect_garbage (bod [bod_q_msgbuf]);
;  185          0183  3     !
;  186          0184  3     ! Place the BOD in the free queue, put it at the head so that any old bods that get
;  187          0185  3     ! out-faulted will stay out until we really need 'em.
;  188          0186  3     !
;  189          0187  3     $queue_remove (.bod);                       ! Remove it from the que (should be in the busy queue)
;  190          0188  3     $queue_insert_head (.bod, bod_free_queue);
;  191          0189  3     bod_busy_count = .bod_busy_count - 1;
;  192          0190  2     END;
```

```
;  193          0191  2 !
;  194          0192  2 ! If we can queue another brkthru, then do so
;  195          0193  2 !
;  196          0194  2 bod = .bod_wait_queue [0];
;  197          0195  2 WHILE .bod NEQ bod_wait_queue                    ! Loop until we see the end
;  198          0196  2     AND
;  199          0197  2      .bod_busy_count LSS .bod_busy_max           ! or until we have filled our quota
;  200          0198  2 DO
;  201          0199  3     BEGIN
;  202          0200  3     LOCAL
;  203          0201  3         next;
;  204          0202  3     next = .bod [bod_l_flink];                  ! Save the pointer to the next, since we might pull it out
;  205          0203  3     IF replybrd_ok_to_queue_write (.bod, bod_busy_queue)        ! Control number of $BRKTHRUs pending per ta
;  206          0204  3     THEN
;  207          0205  4         BEGIN
;  208          0206  4         $queue_remove (.bod);                   ! Remove it from the waiting queue
;  209        P 0207  4         status = $brkthru (efn = efn_k_asynch,
;  210        P 0208  4                            msgbuf = bod [bod_q_msgbuf],
;  211        P 0209  4                            sendto = bod [bod_q_sendto],
;  212        P 0210  4                            iosb   = bod [bod_q_iosb],
;  213        P 0211  4                            sndtyp = .bod [bod_l_sndtyp],
;  214        P 0212  4                            carcon = .bod [bod_l_carcon],
;  215        P 0213  4                            flags  = .bod [bod_l_flags],
;  216        P 0214  4                            reqid  = .bod [bod_l_reqid],
;  217        P 0215  4                            timout = (IF .bod [bod_v_short_timeout]        ! If the human is waiting for a repl
;  218        P 0216  4                                      THEN 20                              !  then timeout after 20 seconds
;  219        P 0217  4                                      ELSE 20*60),                         !  otherwise try for 20 minutes
;  220        P 0218  4                            astadr = replybrd_brkthru_ast,
;  221          0219  4                            astprm = .bod);
;  222          0220  4         IF NOT .status                                      ! If it didn't work, free the bod
;  223          0221  4         THEN
;  224          0222  5             BEGIN
;  225          0223  5             IF .status<0,3,0> GTR .replybrd_status<0,3,0>   ! Save the worst status
;  226          0224  5             THEN
;  227          0225  5                 replybrd_status = .status;
;  228          0226  5             collect_garbage (bod [bod_q_msgbuf]);           ! Free the message
;  229          0227  5             $queue_insert_head (.bod, bod_free_queue);      ! Free the bod
;  230          0228  5             END
;  231          0229  4         ELSE
;  232          0230  5             BEGIN
;  233          0231  5             bod_busy_count = .bod_busy_count + 1;
;  234          0232  5             $queue_insert_tail (.bod, bod_busy_queue);
;  235          0233  5             $gettim (timadr=bod [bod_q_quetime]);
;  236          0234  4             END;
;  237          0235  3         END;
;  238          0236  3     !
;  239          0237  3     ! Advance to the next one, using the saved next pointer
;  240          0238  3     !
;  241          0239  3     bod = .next;
;  242          0240  2     END;
;  243          0241  2
;  244          0242  2 !
;  245          0243  2 ! Set an event flag if we have emptied both of the queues
;  246          0244  2 !
;  247          0245  3 IF $queue_empty (bod_wait_queue) AND $queue_empty (bod_busy_queue)
;  248          0246  2 THEN
;  249          0247  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
```

```
;  254          0251  1 GLOBAL ROUTINE replybrd_brkthru_queue (msgbuf : $ref_bblock, sendto : $ref_bblock,
;  255          0252  1                                        sndtyp, carcon, flags, reqid,
;  256          0253  1                                        routin, node : $ref_bblock, csid,
;  257          0254  1                                        stflg, term : $ref_bblock) : NOVALUE =
;  258          0255  1
;  259          0256  1 !++
;  260          0257  1 ! Functional descripton:
;  261          0258  1 !
;  262          0259  1 !       This routine queues a breakthru write
;  263          0260  1 !
;  264          0261  1 ! Input:
;  265          0262  1 !       msgbuf  Text string to broadcast
;  266          0263  1 !       sendto  Target name for $BRKTHRU
;  267          0264  1 !       sndtyp  Type of target
;  268          0265  1 !       carcon  Carriage control
;  269          0266  1 !       flags   Misc flags
;  270          0267  1 !       reqid   Type of requestor
;  271          0268  1 !       routin  Address of completion routine (following arguments used by completion routine)
;  272          0269  1 !       node    Nodename
;  273          0270  1 !       csid    CSID of originating system
;  274          0271  1 !       stflg   Initial contents of STATUS flags
;  275          0272  1 !       term    Name of originating terminal
;  276          0273  1 !
;  277          0274  1 ! Output:
;  278          0275  1 !       None.
;  279          0276  1 !
;  280          0277  1 ! Routine Value:
;  281          0278  1 !       Address of block
;  282          0279  1 !--
;  283          0280  1
;  284          0281  2 BEGIN                                           ! Start of REPLYBRD_BRKTHRU_QUEUE
;  285          0282  2
;  286          0283  2 LOCAL
;  287          0284  2     bod         : $ref_bblock,                  ! BOD data structure
;  288          0285  2     garb        : REF VECTOR [, LONG],
;  289          0286  2     ptr,
;  290          0287  2     status;
;  291          0288  2
;  292          0289  2 !
;  293          0290  2 ! If any garbage nodes are in the hopper, send them away.  Garbage is reclaimed this
;  294          0291  2 ! way so that the user ast routines do not do free_vm calls on memory allocated
;  295          0292  2 ! from user non-ast mode.
;  296          0293  2 !
;  297          0294  2 $queue_remove_head (bod_garbage_queue, garb);
;  298          0295  2 WHILE .garb NEQ 0
;  299          0296  2 DO
;  300          0297  3     BEGIN
;  301          0298  4     IF NOT (status = opc$free_vm (garb [2], garb))
;  302          0299  3     THEN
;  303          0300  3         $signal_stop (.status);
;  304          0301  3     $queue_remove_head (bod_garbage_queue, garb);
;  305          0302  2     END;
;  306          0303  2 !
;  307          0304  2 ! Get a BOD, a Brkthru Output Descriptor, if none available on the queue then
;  308          0305  2 ! allocate and initialize one.
;  309          0306  2 !
;  310          0307  2 $queue_remove_head (bod_free_queue, bod);
```

```
;  311          0308  2 IF .bod EQL 0
;  312          0309  2 THEN
;  313          0310  3     BEGIN
;  314          0311  4     IF NOT (status = opc$get_vm (%ref (bod_k_size), ptr))
;  315          0312  3     THEN
;  316          0313  3         $signal_stop (.STATUS);
;  317          0314  3     bod_allocated = .bod_allocated + 1;
;  318          0315  3     bod = .ptr;
;  319          0316  3     CH$FILL (0, bod_k_size, .bod);
;  320          0317  3     bod [bod_w_size] = bod_k_size;
;  321          0318  3     bod [bod_b_type] = %x'01';
;  322          0319  3     bod [bod_a_senptr] = bod [bod_t_senbuf];
;  323          0320  3     bod [bod_a_nodptr] = bod [bod_t_nodbuf];
;  324          0321  3     bod [bod_a_trmptr] = bod [bod_t_trmbuf];
;  325          0322  2     END;
;  326          0323  2 !
;  327          0324  2 ! Init the block and copy the longword inputs
;  328          0325  2 !
;  329          0326  2 (bod [bod_q_quetime]) = 0;
;  330          0327  2 (bod [bod_q_quetime]+4) = 0;
;  331          0328  2 bod [bod_l_senlen] = 0;
;  332          0329  2 bod [bod_l_nodlen] = 0;
;  333          0330  2 bod [bod_l_trmlen] = 0;
;  334          0331  2 bod [bod_l_sndtyp] = .sndtyp;
;  335          0332  2 bod [bod_l_carcon] = .carcon;
;  336          0333  2 bod [bod_l_flags]  = .flags;
;  337          0334  2 bod [bod_l_reqid]  = .reqid;
;  338          0335  2 bod [bod_a_completion_routine] = .routin;
;  339          0336  2 bod [bod_l_csid] = .csid;
;  340          0337  2 bod [bod_l_status] = .stflg;
;  341          0338  2 !
;  342          0339  2 ! Copy the target string to the BOD, zero fill for a later test
;  343          0340  2 !
;  344          0341  2 IF .sendto NEQ 0
;  345          0342  2 THEN
;  346          0343  3     BEGIN
;  347          0344  3     IF .sndtyp EQL brk$c_device                 ! If a device breakthru, then see if we can see the device
;  348          0345  3     THEN
;  349          0346  4         BEGIN
;  350          0347  4         LOCAL
;  351          0348  4             terminal,
;  352          0349  4             dvi_items   : VECTOR [4, LONG];
;  353          0350  4         dvi_items [0] = (dvi$_trm^16 OR 4);
;  354          0351  4         dvi_items [1] = terminal;
;  355          0352  4         dvi_items [2] = dvi_items [3] = 0;
;  356          0353  4         status = $getdvi (devnam=.sendto, itmlst=dvi_items);    ! Can we see the device?
;  357          0354  4         IF .status                                              ! If we can see it, is it a terminal
;  358          0355  4         THEN
;  359          0356  4             IF NOT .terminal
;  360          0357  4             THEN
;  361          0358  4                 status = ss$_ivdevnam;                          ! Invalid device name seems ok
;  362          0359  4         IF NOT .status                                          ! Something went wrong
;  363          0360  4         THEN
;  364          0361  5             BEGIN
;  365          0362  5             replybrd_status = .status;                      ! Save the status
;  366          0363  5             $queue_insert_head (.bod, bod_free_queue);      ! Replace the new bod
;  367          0364  5             RETURN;                                         ! And return
```

```
;  368          0365  4              END;
;  369          0366  3          END;
;  370          0367  3      bod [bod_l_senlen] = .sendto [dsc$w_length];
;  371          0368  3      CH$COPY (.sendto [dsc$w_length], .sendto [dsc$a_pointer], 0, bod_s_senbuf, bod [bod_t_senbuf]);
;  372          0369  3      END
;  373          0370  2 ELSE
;  374          0371  2      CH$FILL (0, bod_s_senbuf, bod [bod_t_senbuf]);      ! For the test
;  375          0372  2 !
;  376          0373  2 ! Copy the output string to the BOD
;  377          0374  2 !
;  378          0375  2 bod [bod_l_msglen] = MAX (12, .msgbuf [dsc$w_length]);
;  379          0376  3 IF NOT (status = opc$get_vm (bod [bod_l_msglen], bod [bod_a_msgptr]))
;  380          0377  2 THEN
;  381          0378  2      $signal_stop (.status);
;  382          0379  2 CH$COPY (.msgbuf [dsc$w_length], .msgbuf [dsc$a_pointer], 0, .bod [bod_l_msglen], .bod [bod_a_msgptr]);
;  383          0380  2 !
;  384          0381  2 ! Copy the nodename to the BOD, if present
;  385          0382  2 !
;  386          0383  2 IF .node NEQ 0
;  387          0384  2 THEN
;  388          0385  3      BEGIN
;  389          0386  3      bod [bod_l_nodlen] = .node [dsc$w_length];
;  390          0387  3      CH$COPY (.node [dsc$w_length], .node [dsc$a_pointer], 0, bod_s_nodbuf, bod [bod_t_nodbuf]);
;  391          0388  2      END;
;  392          0389  2 !
;  393          0390  2 ! Copy the terminal to the BOD, if present
;  394          0391  2 !
;  395          0392  2 IF .term NEQ 0
;  396          0393  2 THEN
;  397          0394  3      BEGIN
;  398          0395  3      bod [bod_l_trmlen] = .term [dsc$w_length];
;  399          0396  3      CH$COPY (.term [dsc$w_length], .term [dsc$a_pointer], 0, bod_s_trmbuf, bod [bod_t_trmbuf]);
;  400          0397  2      END;
;  401          0398  2 !
;  402          0399  2 ! Place the BOD on the queue of outputs waiting
;  403          0400  2 !
;  404          0401  2 $queue_insert_tail (.bod, bod_wait_queue);
;  405          0402  2 !
;  406          0403  2 ! Declare an AST, the 0 parameter means try to start
;  407          0404  2 !
;  408          0405  2 bod_requests = .bod_requests + 1;
;  409          0406  3 IF NOT (status = $dclast (astadr=replybrd_brkthru_ast, astprm=0))
;  410          0407  2 THEN
;  411          0408  2      $signal_stop (.status);
;  412          0409  2
;  413          0410  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                                ; 0385
                                           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
```

```
;  415     0411  1 GLOBAL ROUTINE replybrd_format (msg : $ref_bblock, local_node : $ref_bblock) =  %SBTTL 'Replybrd_Format (msg
;  416     0412  1
;  417     0413  1 !++
;  418     0414  1 ! Functional description:
;  419     0415  1 !
;  420     0416  1 !       This routine makes a formatted message from a broadcast message block.  The formatted
;  421     0417  1 !       message is placed off the message block.
;  422     0418  1 !
;  423     0419  1 ! Input:
;  424     0420  1 !
;  425     0421  1 !       msg        - Pointer to RPYBRD message block
;  426     0422  1 !       local_node - Pointer to string descriptor with local node name
;  427     0423  1 !
;  428     0424  1 ! Implicit Input:
;  429     0425  1 !
;  430     0426  1 !       None.
;  431     0427  1 !
;  432     0428  1 ! Output:
;  433     0429  1 !
;  434     0430  1 !       None.
;  435     0431  1 !
;  436     0432  1 ! Implict output:
;  437     0433  1 !
;  438     0434  1 !       None.
;  439     0435  1 !
;  440     0436  1 ! Side effects:
;  441     0437  1 !
;  442     0438  1 !       None.
;  443     0439  1 !
;  444     0440  1 ! Routine value:
;  445     0441  1 !
;  446     0442  1 !       None.
;  447     0443  1 !--
;  448     0444  1
;  449     0445  2 BEGIN                                           ! Start of replybrd_format
;  450     0446  2
;  451     0447  2 LOCAL
;  452     0448  2         send_term_dsc   : VECTOR [2, LONG],
;  453     0449  2         send_user_dsc   : VECTOR [2, LONG],
;  454     0450  2         send_node_dsc   : VECTOR [2, LONG],
;  455     0451  2         message_dsc     : VECTOR [2, LONG],
;  456     0452  2         on_buf          : VECTOR [64, BYTE],
;  457     0453  2         on_dsc          : VECTOR [2, LONG] INITIAL (64, on_buf),
;  458     0454  2         tim_buf         : VECTOR [12, BYTE],
;  459     0455  2         tim_dsc         : VECTOR [2, LONG] INITIAL (11, tim_buf),
;  460     0456  2         fmt_buf         : VECTOR [512, BYTE],
;  461     0457  2         fmt_dsc         : VECTOR [2, LONG] INITIAL (512, fmt_buf),
;  462     0458  2         ctr_buf         : VECTOR [256, BYTE],
;  463     0459  2         ctr_dsc         : VECTOR [2, LONG] INITIAL (256, ctr_buf),
;  464     0460  2         bell_start,
;  465     0461  2         bell_mid,
;  466     0462  2         bell_end,
;  467     0463  2         msgid,
;  468     0464  2         ptr,
;  469     0465  2         status;
;  470     0466  2
;  471     0467  2 BIND
```

```
;  472          0468  2     bells  = UPLIT BYTE (7,7,7,7);
;  473          0469  2
;  474          0470  2 !
;  475          0471  2 ! If /BELLS, then set things up to make some noise
;  476          0472  2 !
;  477          0473  2 bell_start = bell_mid = bell_end = 0;
;  478          0474  2 IF .msg [rpybrd_v_bell]
;  479          0475  2 THEN
;  480          0476  3     BEGIN
;  481          0477  3     bell_start = 1;
;  482          0478  3     IF .msg [rpybrd_v_shutdown]
;  483          0479  3     THEN
;  484          0480  4         BEGIN
;  485          0481  4         bell_end = 2;
;  486          0482  4         bell_mid = 1;
;  487          0483  3         END;
;  488          0484  3     IF .msg [rpybrd_v_urgent]
;  489          0485  3     THEN
;  490          0486  3         bell_mid = 1;
;  491          0487  2     END;
;  492          0488  2 !
;  493          0489  2 ! Make descriptors for the four required items in the message (the optional items don't change the
;  494          0490  2 ! formatting)
;  495          0491  2 !
;  496          0492  2 ptr = msg [rpybrd_t_text];
;  497          0493  2 send_term_dsc [1] = .ptr;
;  498          0494  2 ptr = .ptr + (send_term_dsc [0] = .msg [rpybrd_w_send_term_len]);
;  499          0495  2 send_user_dsc [1] = .ptr;
;  500          0496  2 ptr = .ptr + (send_user_dsc [0] = .msg [rpybrd_w_send_user_len]);
;  501          0497  2 send_node_dsc [1] = .ptr;
;  502          0498  2 ptr = .ptr + (send_node_dsc [0] = .msg [rpybrd_w_send_node_len]);
;  503          0499  2 message_dsc [1] = .ptr;
;  504          0500  2 ptr = .ptr + (message_dsc [0] = .msg [rpybrd_w_message_len]);
;  505          0501  2 !
;  506          0502  2 ! Set the message id.  This will be a general broadcast unless /URGENT or /SHUTDOWN is requested.
;  507          0503  2 !
;  508          0504  3 msgid =     (SELECTONE true OF
;  509          0505  3              SET
;  510          0506  3                  [.msg [rpybrd_v_urgent]] :      opc$_reply_urgent;
;  511          0507  3                  [.msg [rpybrd_v_shutdown]] :    opc$_reply_shutdown;
;  512          0508  3                  [OTHERWISE] :                   opc$_reply_general;
;  513          0509  2              TES);
;  514          0510  2 !
;  515          0511  2 ! Get the $FAO control string text associated with the message id via $GETMSG.
;  516          0512  2 !
;  517          0513  3 IF NOT (status = $GETMSG (MSGID=.msgid, MSGLEN=ctr_dsc, BUFADR=ctr_dsc, FLAGS=1))
;  518          0514  2 THEN
;  519          0515  2     RETURN .status;
;  520          0516  2 !
;  521          0517  2 ! If we have a net or cluster, then set the preposition to precede the node name
;  522          0518  2 !
;  523          0519  2 IF .local_node [dsc$w_length] NEQ 0
;  524          0520  2 THEN
;  525          0521  3     BEGIN
;  526          0522  4     IF NOT (status = $GETMSG (MSGID=opc$_on_node, MSGLEN=on_dsc, BUFADR=on_dsc, FLAGS=1))
;  527          0523  3     THEN
;  528          0524  3         RETURN .status;
```

```
;  529          0525  3     END
;  530          0526  2 ELSE
;  531          0527  2     on_dsc [0] = 0;                                  ! Null prepositions
;  532          0528  2 !
;  533          0529  2 ! Format the time, we can make it look a little cleaner if we strip the hundredths
;  534          0530  2 !
;  535          0531  2 $ASCTIM (timbuf=tim_dsc, cvtflg=1);
;  536          0532  2 tim_dsc [0] = 8;
;  537          0533  2 !
;  538          0534  2 ! Format the message, the general form of the control string is:
;  539          0535  2 !
;  540          0536  2 !       "!/!ADReply received!AS!AF from user !AF at !AS   !AS!AD!/!AS!AD!/"
;  541          0537  2 !
;  542        P 0538  3 IF NOT (status = $FAO ( ctr_dsc,                         ! Control string
;  543        P 0539  3                         fmt_dsc, fmt_dsc,               ! Return length, output buffer
;  544        P 0540  3                         .bell_start, bells,             ! First bell
;  545        P 0541  3                         on_dsc,                         ! Preposition (like in "on " for "on ATHENS")
;  546        P 0542  3                         .local_node [dsc$w_length],     ! Local node name
;  547        P 0543  3                         .local_node [dsc$a_pointer],    ! Local node name
;  548        P 0544  3                         .send_user_dsc [0],             ! Sender's username
;  549        P 0545  3                         .send_user_dsc [1],             ! Sender's username
;  550        P 0546  3                         send_term_dsc,                  ! Name of sender's terminal
;  551        P 0547  3                         tim_dsc,                        ! Use the reformatted time
;  552        P 0548  3                         .bell_mid, bells,               ! Second bell
;  553        P 0549  3                         message_dsc,                    ! Actual message text
;  554          0550  3                         .bell_end, bells))              ! Third bell
;  555          0551  2 THEN
;  556          0552  2     RETURN .status;
;  557          0553  2 !
;  558          0554  2 ! Get a chunk of memory to hang the formatted message from the block.  We make a descriptor in the block.
;  559          0555  2 !
;  560          0556  2 msg [rpybrd_l_format_len] = .fmt_dsc [0];
;  561          0557  3 IF NOT (status = OPC$GET_VM (msg [rpybrd_l_format_len], msg [rpybrd_a_format_ptr]))
;  562          0558  2 THEN
;  563          0559  2     RETURN .status;
;  564          0560  2 !
;  565          0561  2 ! Move the formatted message to the block.  As we copy the message, change
;  566          0562  2 ! any escape characters to dollar signs.
;  567          0563  2 !
;  568          0564  3 BEGIN
;  569          0565  3 REGISTER
;  570          0566  3     char : BYTE,
;  571          0567  3     iptr,
;  572          0568  3     optr;
;  573          0569  3 iptr = .fmt_dsc [1];
;  574          0570  3 optr = .msg [rpybrd_a_format_ptr];
;  575          0571  3 INCR i FROM 1 TO .fmt_dsc [0]
;  576          0572  3 DO
;  577          0573  4     BEGIN
;  578          0574  4     char = CH$RCHAR_A (iptr);
;  579          0575  4     IF .char EQL 27 THEN char = %C '$';
;  580          0576  4     CH$WCHAR_A (.char, optr);
;  581          0577  3     END;
;  582          0578  2 END;
;  583          0579  2
;  584          0580  2 RETURN true;
;  585          0581  1 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$:     MOVAB   48(R2), PTR                           ; 0492
                          FC    AD              50  D0 00066        MOVL    PTR, SEND_TERM_DSC+4                  ; 0493
                                51        14    A2  3C 0006A        MOVZWL  20(R2), R1                            ; 0494
                          F8    AD              51  D0 0006E        MOVL    R1, SEND_TERM_DSC
                                50              51  C0 00072        ADDL2   R1, PTR
                          F4    AD              50  D0 00075        MOVL    PTR, SEND_USER_DSC+4                  ; 0495
                                51        16    A2  3C 00079        MOVZWL  22(R2), R1                            ; 0496
                          F0    AD              51  D0 0007D        MOVL    R1, SEND_USER_DSC
                                50              51  C0 00081        ADDL2   R1, PTR
                          EC    AD              50  D0 00084        MOVL    PTR, SEND_NODE_DSC+4                  ; 0497
                                51        18    A2  3C 00088        MOVZWL  24(R2), R1                            ; 0498
                          E8    AD              51  D0 0008C        MOVL    R1, SEND_NODE_DSC
                                50              51  C0 00090        ADDL2   R1, PTR
                          E4    AD              50  D0 00093        MOVL    PTR, MESSAGE_DSC+4                    ; 0499
                                51        1A    A2  3C 00097        MOVZWL  26(R2), R1                            ; 0500
                          E0    AD              51  D0 0009B        MOVL    R1, MESSAGE_DSC
                                50              51  C0 0009F        ADDL2   R1, PTR
                    09          63              06  E1 000A2        BBC     #6, (R3), 3$                          ; 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        PUSHA3  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
```

```
;  587          0582  1 GLOBAL ROUTINE replybrd_io (msg : $ref_bblock, local_node : $ref_bblock) =      %SBTTL 'Replybrd_IO (msg, lo
;  588          0583  1
;  589          0584  1 !++
;  590          0585  1 ! Functional description:
;  591          0586  1 !
;  592          0587  1 !       This routine sends the formatted message from a broadcast message block to the terminals,
;  593          0588  1 !       using the $BRKTHRU system service.
;  594          0589  1 !
;  595          0590  1 ! Input:
;  596          0591  1 !
;  597          0592  1 !       msg         - Pointer to RPYBRD message block
;  598          0593  1 !       local_node - Pointer to string descriptor with local node name
;  599          0594  1 !
;  600          0595  1 ! Implicit Input:
;  601          0596  1 !
;  602          0597  1 !       None.
;  603          0598  1 !
;  604          0599  1 ! Output:
;  605          0600  1 !
;  606          0601  1 !       None.
;  607          0602  1 !
;  608          0603  1 ! Implict output:
;  609          0604  1 !
;  610          0605  1 !       Messages might be sent to terminals.
;  611          0606  1 !
;  612          0607  1 ! Side effects:
;  613          0608  1 !
;  614          0609  1 !       None.
;  615          0610  1 !
;  616          0611  1 ! Routine value:
;  617          0612  1 !
;  618          0613  1 !       None.
;  619          0614  1 !--
;  620          0615  1
;  621          0616  2 BEGIN                                           ! Start of replybrd_io
;  622          0617  2
;  623          0618  2 LOCAL
;  624          0619  2         flags,                                  ! Initial flags for BOD_L_STATUS
;  625          0620  2         sndtyp,                                 ! Type of brkthru
;  626          0621  2         reqid,                                  ! ID of brkthru sender
;  627          0622  2         arglist         : VECTOR [16, LONG],    ! Argument list for callg
;  628          0623  2         trmdsc          : VECTOR [2, LONG],
;  629          0624  2         ptr,
;  630          0625  2         timout,
;  631          0626  2         status;
;  632          0627  2
;  633          0628  2 !
;  634          0629  2 ! Initialize the global status to zero.  Determine the miscellaneous flags.
;  635          0630  2 !
;  636          0631  2 replybrd_status = 0;
;  637          0632  2 flags = bod_m_short_timeout;
;  638          0633  2 IF .msg [rpybrd_v_wait]
;  639          0634  2 THEN
;  640          0635  2     flags = .flags + bod_m_wait;
;  641          0636  2 IF .msg [rpybrd_v_local_node]
;  642          0637  2 THEN
;  643          0638  2     flags = .flags + bod_m_local_node;
```

```
;  644          0639  2
;  645          0640  2 !
;  646          0641  2 ! Set the send type.  If REPLY /ALL, then send to all terms.  If /USER, send to all logged in users,
;  647          0642  2 ! If /USER=(...), send to those users.  If /TERM=(...), send to those terminals.
;  648          0643  2 !
;  649          0644  3 sndtyp =    (SELECTONE true OF
;  650          0645  3                 SET
;  651          0646  3                     [.msg [rpybrd_v_all]] :         brk$c_allterms;
;  652          0647  3                     [.msg [rpybrd_v_username]] :    IF .msg [rpybrd_w_targ_user_len] EQL 0
;  653          0648  3                                                     THEN brk$c_allusers
;  654          0649  3                                                     ELSE brk$c_username;
;  655          0650  3                     [.msg [rpybrd_v_terminal]] :    brk$c_device;
;  656          0651  3                     [OTHERWISE] :                   RETURN ss$_badparam;
;  657          0652  2                 TES);
;  658          0653  2 !
;  659          0654  2 ! Set the request id.  This will be a general broadcast unless /URGENT or /SHUTDOWN is requested.
;  660          0655  2 !
;  661          0656  3 reqid =     (SELECTONE true OF
;  662          0657  3                 SET
;  663          0658  3                     [.msg [rpybrd_v_urgent]] :      brk$c_urgent;
;  664          0659  3                     [.msg [rpybrd_v_shutdown]] :    brk$c_shutdown;
;  665          0660  3                     [OTHERWISE] :                   brk$c_general;
;  666          0661  2                 TES);
;  667          0662  2 !
;  668          0663  2 ! Build the argument list for the callg.  Since most of the arguments are the same, this both saves
;  669          0664  2 ! execution time and makes it easier to update.  Also note that the list io routine will modify
;  670          0665  2 ! this arglist
;  671          0666  2 !
;  672          0667  2 arglist [count] = queue_args;                        ! assume args for an ALL reply
;  673          0668  2 arglist [text]  = msg [rpybrd_q_format_desc];        ! text to send
;  674          0669  2 arglist [targ]  = 0;                                ! target of send
;  675          0670  2 arglist [type]  = .sndtyp;                          ! type of target
;  676          0671  2 arglist [cntl]  = 32;                               ! carriage control
;  677          0672  2 arglist [bflg]  = 0;                                ! brkthru flags                                  ;
;  678          0673  2 arglist [rqid]  = .reqid;                           ! requestor id
;  679          0674  2 arglist [rtn]   = 0;                                ! assume no completion routine
;  680          0675  2 arglist [node]  = .local_node;                      ! local node name
;  681          0676  2 arglist [csid]  = .msg [rpybrd_l_send_csid];        ! remote csid name
;  682          0677  2 arglist [sflg]  = .flags;                           ! miscellaneous flags for BOD_L_STATUS
;  683          0678  2 arglist [term]  = trmdsc;                           ! pass the address of and
;  684          0679  2 trmdsc [0]      = .msg [rpybrd_w_send_term_len];    !  create a descriptor
;  685          0680  2 trmdsc [1]      = msg [rpybrd_t_text];              !  for the sending terminal
;  686          0681  2
;  687          0682  2 !
;  688          0683  2 ! If we are supposed to wait, then disable ASTs while we are queueing the brkthrus
;  689          0684  2 !
;  690          0685  2 IF .msg [rpybrd_v_wait]
;  691          0686  2 THEN
;  692          0687  2     $setast (enbflg=0);
;  693          0688  2
;  694          0689  2 !
;  695          0690  2 ! Now send it on to the world.  If it is /ALL or /USER, then do a brkthru and
;  696          0691  2 ! return.  If /USER=(...) or /TERM=(...), set up pointers so that we can do one
;  697          0692  2 ! brkthru per specified user or terminal and call the list io routine.
;  698          0693  2 !
;  699          0694  2 CASE .sndtyp FROM 1 TO brk$c_maxsendtype OF
;  700          0695  2 SET
```

```
;  701     0696  2      [brk$c_allterms, brk$c_allusers] :
;  702     0697  3                  BEGIN
;  703     0698  3                  IF .msg [rpybrd_v_notify]                          ! if notify then pass
;  704     0699  3                  THEN arglist [rtn] = replybrd_notify_all;          !  address of notification routine
;  705     0700  3                  CALLG (arglist, replybrd_brkthru_queue);
;  706     0701  2                  END;
;  707     0702  2
;  708     0703  2      [brk$c_device] :
;  709     0704  3                  BEGIN
;  710     0705  3                  IF .msg [rpybrd_v_notify]                          ! if notify then pass
;  711     0706  3                  THEN arglist [rtn] = replybrd_notify_dev;          !  address of notification routine
;  712     0707  3                  arglist [lptr] = .msg + .msg [rpybrd_w_optional_off];  ! send list address
;  713     0708  3                  arglist [llen] = .msg [rpybrd_w_targ_term_len];    ! send list length
;  714     0709  3                  arglist [count] = queue_args+2;
;  715     0710  3                  CALLG (arglist, replybrd_io_list);
;  716     0711  2                  END;
;  717     0712  2
;  718     0713  2      [brk$c_username] :
;  719     0714  3                  BEGIN
;  720     0715  3                  IF .msg [rpybrd_v_notify]                          ! if notify then pass
;  721     0716  3                  THEN arglist [rtn] = replybrd_notify_use;          !  address of notification routine
;  722     0717  3                  arglist [lptr] = .msg + .msg [rpybrd_w_optional_off] +  ! send list address
;  723     0718  3                                  .msg [rpybrd_w_targ_term_len];
;  724     0719  3                  arglist [llen] = .msg [rpybrd_w_targ_user_len];    ! send list length
;  725     0720  3                  arglist [count] = queue_args+2;
;  726     0721  3                  CALLG (arglist, replybrd_io_list);
;  727     0722  2                  END;
;  728     0723  2
;  729     0724  2      [INRANGE, OUTRANGE] :
;  730     0725  2                  RETURN ss$_badparam;
;  731     0726  2  TES;
;  732     0727  2
;  733     0728  2  !
;  734     0729  2  ! If we are supposed to wait, then fire off an AST to start the ball rolling and wait for our flag
;  735     0730  2  !
;  736     0731  2  IF .msg [rpybrd_v_wait]
;  737     0732  2    AND
;  738     0733  2      .replybrd_status EQL 0                                   ! Don't wait if an error occurred
;  739     0734  2  THEN
;  740     0735  3      BEGIN
;  741     0736  3      $clref (efn=efn_k_brkthru);                              ! Clear our flag
;  742     0737  3      $setast (enbflg=1);                                      ! Enable pending ASTs
;  743     0738  3      $waitfr (efn=efn_k_brkthru);                             ! Wait for the queues to empty
;  744     0739  2      END;
;  745     0740  2
;  746     0741  2  !
;  747     0742  2  ! If nobody has modified the status, then return success
;  748     0743  2  !
;  749     0744  2  IF (status = .replybrd_status) EQL 0
;  750     0745  2  THEN
;  751     0746  2      status = ss$_normal;
;  752     0747  2
;  753     0748  2  RETURN .status;
;  754     0749  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
```

```
;  756          0750  1 GLOBAL ROUTINE replybrd_io_list : NOVALUE =     %SBTTL 'Replybrd_IO_list (callg with AP)'
;  757          0751  1
;  758          0752  1 !++
;  759          0753  1 ! Functional description:
;  760          0754  1 !
;  761          0755  1 !       This routine sends the formatted message from a broadcast message block to the terminals,
;  762          0756  1 !       using the $BRKTHRU system service.
;  763          0757  1 !
;  764          0758  1 ! Input:
;  765          0759  1 !
;  766          0760  1 !       msg         - Pointer to RPYBRD message block
;  767          0761  1 !       local_node - Pointer to string descriptor with local node name
;  768          0762  1 !
;  769          0763  1 ! Implicit Input:
;  770          0764  1 !
;  771          0765  1 !       None.
;  772          0766  1 !
;  773          0767  1 ! Output:
;  774          0768  1 !
;  775          0769  1 !       None.
;  776          0770  1 !
;  777          0771  1 ! Implict output:
;  778          0772  1 !
;  779          0773  1 !       Messages might be sent to terminals.
;  780          0774  1 !
;  781          0775  1 ! Side effects:
;  782          0776  1 !
;  783          0777  1 !       None.
;  784          0778  1 !
;  785          0779  1 ! Routine value:
;  786          0780  1 !
;  787          0781  1 !       None.
;  788          0782  1 !--
;  789          0783  1
;  790          0784  2 BEGIN                                           ! Start of replybrd_io
;  791          0785  2
;  792          0786  2 BUILTIN
;  793          0787  2         AP;
;  794          0788  2
;  795          0789  2 LOCAL
;  796          0790  2         mlen,                                   ! Length of message field
;  797          0791  2         mptr            : $ref_bvector,         ! Pointer to message field
;  798          0792  2         sendto          : VECTOR [2, LONG];     ! User or terminal name
;  799          0793  2
;  800          0794  2 BIND
;  801          0795  2         arglist = .AP : VECTOR [, LONG];
;  802          0796  2
;  803          0797  2 mlen = .arglist [llen];                         ! Get length and pointer to list
;  804          0798  2 mptr = .arglist [lptr];
;  805          0799  2 arglist [targ] = sendto;                        ! Point the arglist at the new desc
;  806          0800  2 arglist [count] = queue_args;                   ! Set correct argument count for queue routine
;  807          0801  2 !
;  808          0802  2 ! We are doing it to a list of terminals or usernames, mptr and mlen are set up
;  809          0803  2 !
;  810          0804  2 WHILE .mlen GTR 0
;  811          0805  2 DO
;  812          0806  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
```

```
;  834     0827  1 GLOBAL ROUTINE replybrd_notify_all (bod : $ref_bblock) : NOVALUE = %SBTTL 'Replybrd_notify_all (bod)'
;  835     0828  1 !++
;  836     0829  1 ! Functional description:
;  837     0830  1 !
;  838     0831  1 !       Format a message for REPLY /ALL /NOTIFY or REPLY /USER /NOTIFY, then send the message.
;  839     0832  1 !
;  840     0833  1 ! Input:
;  841     0834  1 !
;  842     0835  1 !       bod         - Pointer to brkthru output descriptor
;  843     0836  1 !
;  844     0837  1 ! Implicit Input:
;  845     0838  1 !
;  846     0839  1 !       None.
;  847     0840  1 !
;  848     0841  1 ! Output:
;  849     0842  1 !
;  850     0843  1 !       None.
;  851     0844  1 !
;  852     0845  1 ! Implict output:
;  853     0846  1 !
;  854     0847  1 !       None.
;  855     0848  1 !
;  856     0849  1 ! Side effects:
;  857     0850  1 !
;  858     0851  1 !       None.
;  859     0852  1 !
;  860     0853  1 ! Routine value:
;  861     0854  1 !
;  862     0855  1 !       None.
;  863     0856  1 !--
;  864     0857  1
;  865     0858  2 BEGIN                                           ! Start of replybrd_notify_all
;  866     0859  2
;  867     0860  2 LOCAL
;  868     0861  2         on_buf          : VECTOR [64, BYTE],
;  869     0862  2         on_dsc          : VECTOR [2, LONG] INITIAL (64, on_buf),
;  870     0863  2         fmt_buf         : VECTOR [512, BYTE],
;  871     0864  2         fmt_dsc         : VECTOR [2, LONG] INITIAL (512, fmt_buf),
;  872     0865  2         ctr_buf         : VECTOR [256, BYTE],
;  873     0866  2         ctr_dsc         : VECTOR [2, LONG] INITIAL (256, ctr_buf),
;  874     0867  2         id,
;  875     0868  2         status;
;  876     0869  2
;  877     0870  2 !
;  878     0871  2 ! Get the $FAO control string text associated with the message id via $GETMSG.
;  879     0872  2 !
;  880     0873  2 id = (IF .bod [bod_w_iosb1] EQL 1 THEN opc$_reply_notall ELSE opc$_reply_notalln);
;  881     0874  3 IF NOT (status = $GETMSG (MSGID=.id, MSGLEN=ctr_dsc, BUFADR=ctr_dsc, FLAGS=1))
;  882     0875  2 THEN
;  883     0876  2     RETURN .status;
;  884     0877  2 !
;  885     0878  2 ! If we have a net or cluster, then set the preposition to precede the node name
;  886     0879  2 !
;  887     0880  2 IF .bod [bod_l_nodlen] NEQ 0
;  888     0881  2 THEN
;  889     0882  3     BEGIN
;  890     0883  4     IF NOT (status = $GETMSG (MSGID=opc$_on_node, MSGLEN=on_dsc, BUFADR=on_dsc, FLAGS=1))
```

```
;  891          0884  3     THEN
;  892          0885  3         RETURN .status;
;  893          0886  3     END
;  894          0887  2 ELSE
;  895          0888  2     on_dsc [0] = 0;                         ! Null prepositions
;  896          0889  2 !
;  897          0890  2 ! Format the message, the general form of the control string is:
;  898          0891  2 !
;  899        P 0892  3 IF NOT (status = $FAO ( ctr_dsc,                        ! Control string
;  900        P 0893  3                         fmt_dsc, fmt_dsc,               ! Return length, output buffer
;  901        P 0894  3                         .bod [bod_w_iosb1],             ! Number of terminals notified
;  902        P 0895  3                         on_dsc,                         ! Preposition (like in "on " for "on ATHENS")
;  903        P 0896  3                         .bod [bod_l_nodlen],            ! Local node name
;  904          0897  3                         .bod [bod_a_nodptr]))           ! Local node name
;  905          0898  2 THEN
;  906          0899  2     RETURN .status;
;  907          0900  2 !
;  908          0901  2 ! Call the routine to queue the io
;  909          0902  2 !
;  910          0903  2 replybrd_notify_io (.bod, fmt_dsc);
;  911          0904  2
;  912          0905  2 RETURN;
;  913          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
```

```
;  915          0907  1 GLOBAL ROUTINE replybrd_notify_dev (bod : $ref_bblock) : NOVALUE = %SBTTL 'Replybrd_notify_dev (bod)'
;  916          0908  1
;  917          0909  1 !++
;  918          0910  1 ! Functional description:
;  919          0911  1 !
;  920          0912  1 !       Format a message for REPLY /NOTIFY /TERMINAL, then send the message.
;  921          0913  1 !
;  922          0914  1 ! Input:
;  923          0915  1 !
;  924          0916  1 !       bod         - Pointer to brkthru output descriptor
;  925          0917  1 !
;  926          0918  1 ! Implicit Input:
;  927          0919  1 !
;  928          0920  1 !       None.
;  929          0921  1 !
;  930          0922  1 ! Output:
;  931          0923  1 !
;  932          0924  1 !       None.
;  933          0925  1 !
;  934          0926  1 ! Implict output:
;  935          0927  1 !
;  936          0928  1 !       None.
;  937          0929  1 !
;  938          0930  1 ! Side effects:
;  939          0931  1 !
;  940          0932  1 !       None.
;  941          0933  1 !
;  942          0934  1 ! Routine value:
;  943          0935  1 !
;  944          0936  1 !       None.
;  945          0937  1 !--
;  946          0938  1
;  947          0939  2 BEGIN                                           ! Start of replybrd_notify_dev
;  948          0940  2
;  949          0941  2 LOCAL
;  950          0942  2         fmt_buf         : VECTOR [512, BYTE],
;  951          0943  2         fmt_dsc         : VECTOR [2, LONG] INITIAL (512, fmt_buf),
;  952          0944  2         ctr_buf         : VECTOR [256, BYTE],
;  953          0945  2         ctr_dsc         : VECTOR [2, LONG] INITIAL (256, ctr_buf),
;  954          0946  2         full_desc       : $ref_bblock,
;  955          0947  2         status;
;  956          0948  2
;  957          0949  2 !
;  958          0950  2 ! Get the $FAO control string text associated with the message id via $GETMSG.
;  959          0951  2 !
;  960          0952  3 IF NOT (status = $GETMSG (MSGID=opc$_reply_notdev, MSGLEN=ctr_dsc, BUFADR=ctr_dsc, FLAGS=1))
;  961          0953  2 THEN
;  962          0954  2     RETURN .status;
;  963          0955  2 !
;  964          0956  2 ! Get the full device name
;  965          0957  2 !
;  966          0958  2 full_desc = share_full_devname (bod [bod_q_sendto], dvi$_fulldevnam);
;  967          0959  2 !
;  968          0960  2 ! Format the message
;  969          0961  2 !
;  970        P 0962  3 IF NOT (status = $FAO ( ctr_dsc,                      ! Control string
;  971        P 0963  3                         fmt_dsc, fmt_dsc,             ! Return length, output buffer
```

```
;  972     0964  3                          .full_desc))                    ! Device name
;  973     0965  2 THEN
;  974     0966  2     RETURN .status;
;  975     0967  2 !
;  976     0968  2 ! Call the routine to dispose of it
;  977     0969  2 !
;  978     0970  2 replybrd_notify_io (.bod, fmt_dsc);
;  979     0971  2
;  980     0972  2 RETURN;
;  981     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    010C   8F   3C 00015  MOVZWL  #268, 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        50   DD 00049  PUSHL   FULL_DESC                                ; 0964
                                                0110  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
```

```
1 GLOBAL ROUTINE replybrd_notify_io (bod : $ref_bblock, text : $ref_bblock) : NOVALUE =   %SBTTL 'Replybrd_not
1
1 !++
1 ! Functional description:
1 !
1 !       This routine sends the formatted message from a reply notification back to the sending
1 !       terminal.
1 !
1 ! Input:
1 !
1 !       bod        - Pointer to brkthru output descriptor
1 !       text       - Pointer to string descriptor with reply notification message
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_notify_io
2
2 LOCAL
2       rpynot  : $bblock [512],
2       ptr     : $ref_bvector,
2       len,
2       status;
2
2 !
2 ! If /WAIT, then we can simply write it to SYS$OUTPUT
2 !
2 IF .bod [bod_v_wait]
2 THEN
2     RETURN lib$put_output (.text);
2
2 !
2 ! Following code executes on OPCOM process only, cannot be executed from within REPLY image
2 !
2
2 !
2 ! If on the local node, queue up the broadcast from here and return
2 !
2 IF .bod [bod_v_local_node]
```

```
;  1040          1031  2 THEN
;  1041          1032  3     BEGIN
;  1042          1033  3     replybrd_brkthru_queue (
;  1043          1034  3                             .text,                  ! text to send
;  1044          1035  3                             bod [bod_q_trmdsc],     ! target
;  1045          1036  3                             brk$c_device,           ! send type
;  1046          1037  3                             32,                     ! carriage control
;  1047          1038  3                             0,                      ! brkthru flags
;  1048          1039  3                             brk$c_general,          ! request id
;  1049          1040  3                             0,0,0,0,0);             ! no completion routine or arguments
;  1050          1041  3     RETURN;
;  1051          1042  2     END;
;  1052          1043  2 !
;  1053          1044  2 ! Not wait or local, send it back to the OPCOM on the host node.  First init the message header.
;  1054          1045  2 !
;  1055          1046  2 CH$FILL (0, rpynot_k_min_size, rpynot);
;  1056          1047  2 rpynot [clm_b_rqstcode]         = opc$_x_clusmsg;
;  1057          1048  2 rpynot [clm_b_clm_code]         = clm__rpynot;
;  1058          1049  2 rpynot [clm_b_ds_version]       = rpynot_k_ds_version;
;  1059          1050  2 rpynot [clm_b_sw_version]       = opc$k_sw_version;
;  1060          1051  2 rpynot [clm_l_csid]             = .lcl_csid;
;  1061          1052  2 !
;  1062          1053  2 ! Copy the originating terminal to the message
;  1063          1054  2 !
;  1064          1055  2 ptr = rpynot [rpynot_t_text];
;  1065          1056  2 len = .bod [bod_l_trmlen];
;  1066          1057  2 rpynot [rpynot_w_term_len] = .len;
;  1067          1058  2 CH$MOVE (.len, .bod [bod_a_trmptr], .ptr);
;  1068          1059  2 ptr = .ptr + .len;
;  1069          1060  2 !
;  1070          1061  2 ! Copy the text to the message
;  1071          1062  2 !
;  1072          1063  2 len = .text [dsc$w_length];
;  1073          1064  2 rpynot [rpynot_w_message_len] = .len;
;  1074          1065  2 CH$MOVE (.len, .text [dsc$a_pointer], .ptr);
;  1075          1066  2 ptr = .ptr + .len;
;  1076          1067  2 !
;  1077          1068  2 ! Store the final length and send it back from whence it came
;  1078          1069  2 !
;  1079          1070  2 len = .ptr - rpynot;
;  1080          1071  2 rpynot [clm_w_length] = .len;
;  1081          1072  2 cluscomm_send (.bod [bod_l_csid], .len, rpynot);
;  1082          1073  2
;  1083          1074  2 RETURN;
;  1084          1075  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
```

```
;  1086          1076  1 GLOBAL ROUTINE replybrd_notify_use (bod : $ref_bblock) : NOVALUE = %SBTTL 'Replybrd_notify_use (bod)'
;  1087          1077  1 !++
;  1088          1078  1 ! Functional description:
;  1089          1079  1 !
;  1090          1080  1 !       Format a message for REPLY /USER=(...) /NOTIFY, then send the message.
;  1091          1081  1 !
;  1092          1082  1 ! Input:
;  1093          1083  1 !
;  1094          1084  1 !       bod         - Pointer to brkthru output descriptor
;  1095          1085  1 !
;  1096          1086  1 ! Implicit Input:
;  1097          1087  1 !
;  1098          1088  1 !       None.
;  1099          1089  1 !
;  1100          1090  1 ! Output:
;  1101          1091  1 !
;  1102          1092  1 !       None.
;  1103          1093  1 !
;  1104          1094  1 ! Implict output:
;  1105          1095  1 !
;  1106          1096  1 !       None.
;  1107          1097  1 !
;  1108          1098  1 ! Side effects:
;  1109          1099  1 !
;  1110          1100  1 !       None.
;  1111          1101  1 !
;  1112          1102  1 ! Routine value:
;  1113          1103  1 !
;  1114          1104  1 !       None.
;  1115          1105  1 !--
;  1116          1106  1
;  1117          1107  2 BEGIN                                           ! Start of replybrd_notify_use
;  1118          1108  2
;  1119          1109  2 LOCAL
;  1120          1110  2         on_buf          : VECTOR [64, BYTE],
;  1121          1111  2         on_dsc          : VECTOR [2, LONG] INITIAL (64, on_buf),
;  1122          1112  2         fmt_buf         : VECTOR [512, BYTE],
;  1123          1113  2         fmt_dsc         : VECTOR [2, LONG] INITIAL (512, fmt_buf),
;  1124          1114  2         ctr_buf         : VECTOR [256, BYTE],
;  1125          1115  2         ctr_dsc         : VECTOR [2, LONG] INITIAL (256, ctr_buf),
;  1126          1116  2         id,
;  1127          1117  2         status;
;  1128          1118  2
;  1129          1119  2 !
;  1130          1120  2 ! Get the $FAO control string text associated with the message id via $GETMSG.
;  1131          1121  2 !
;  1132          1122  2 id = (IF .bod [bod_w_iosb1] EQL 1 THEN opc$_reply_notuse ELSE opc$_reply_notusen);
;  1133          1123  3 IF NOT (status = $GETMSG (MSGID=.id, MSGLEN=ctr_dsc, BUFADR=ctr_dsc, FLAGS=1))
;  1134          1124  2 THEN
;  1135          1125  2     RETURN .status;
;  1136          1126  2 !
;  1137          1127  2 ! If we have a net or cluster, then set the preposition to precede the node name
;  1138          1128  2 !
;  1139          1129  2 IF .bod [bod_l_nodlen] NEQ 0
;  1140          1130  2 THEN
;  1141          1131  3     BEGIN
;  1142          1132  4     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                                  ;
                                      FEDO  CF  02 FB 000A2       CALLS   #2, REPLYBRD_NOTIFY_IO              ;
                                                04 000A7 5$:      RET                                         ; 1156

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

```
;1168     1157  1 GLOBAL ROUTINE replybrd_ok_to_queue_write (bod : $ref_bblock, queue : $ref_bblock) =
;1169     1158  1
;1170     1159  1 !++
;1171     1160  1 ! Functional descripton:
;1172     1161  1 !
;1173     1162  1 !       Check to see if the SENDTO field in the BOD is in any of the bods in the queue.
;1174     1163  1 !       We assume that we are operating at AST level so that we do not have to worry
;1175     1164  1 !       about interlocking the queue.  After counting the number of times it appears
;1176     1165  1 !       in the queue, make a decision as to whether another write can be started.
;1177     1166  1 !
;1178     1167  1 ! Input:
;1179     1168  1 !       bod     pointer to a bod
;1180     1169  1 !       queue   head of a queue of bods
;1181     1170  1 !
;1182     1171  1 ! Output:
;1183     1172  1 !       None.
;1184     1173  1 !
;1185     1174  1 ! Routine Value:
;1186     1175  1 !       true if OK to send another write, false otherwise
;1187     1176  1 !--
;1188     1177  1
;1189     1178  2 BEGIN                                           ! Start of REPLYBRD_OK_TO_QUEUE_WRITE
;1190     1179  2
;1191     1180  2 LOCAL
;1192     1181  2     count,
;1193     1182  2     ok,
;1194     1183  2     head : $ref_bblock,
;1195     1184  2     cur : $ref_bblock;
;1196     1185  2
;1197     1186  2 !
;1198     1187  2 ! Scan the queue, counting the number of times the target appears
;1199     1188  2 !
;1200     1189  2 count = 0;
;1201     1190  2 head = .queue;
;1202     1191  2 cur = .head [bod_l_flink];
;1203     1192  2 WHILE .cur NEQ .head                            ! Loop until we see the end
;1204     1193  2 DO
;1205     1194  3     BEGIN
;1206     1195  3     IF CH$EQL (bod_s_senbuf, bod [bod_t_senbuf], bod_s_senbuf, cur [bod_t_senbuf])
;1207     1196  3     THEN
;1208     1197  3         count = .count + 1;
;1209     1198  3     cur = .cur [bod_l_flink];                  ! Get the next bod
;1210     1199  2     END;
;1211     1200  2 !
;1212     1201  2 ! Now make a decision based on the type of breakthru
;1213     1202  2 !
;1214     1203  2 ok = false;
;1215     1204  2 CASE .bod [bod_l_sndtyp]
;1216     1205  2 FROM MIN (brk$c_device, brk$c_username, brk$c_allusers, brk$c_allterms)
;1217     1206  2 TO   MAX (brk$c_device, brk$c_username, brk$c_allusers, brk$c_allterms) OF
;1218     1207  2 SET
;1219     1208  2     [brk$c_username, brk$c_device] :
;1220     1209  3                         BEGIN
;1221     1210  3                         IF .count EQL 0
;1222     1211  3                         THEN ok = true                  ! Only one to devices or users
;1223     1212  3                         ELSE
;1224     1213  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
```

0291 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

REPLYBRD
LIS

SECURITY
LIS

OPRENABLE
LIS

REPLYMAIN
LIS

RQSTMAIN
LIS