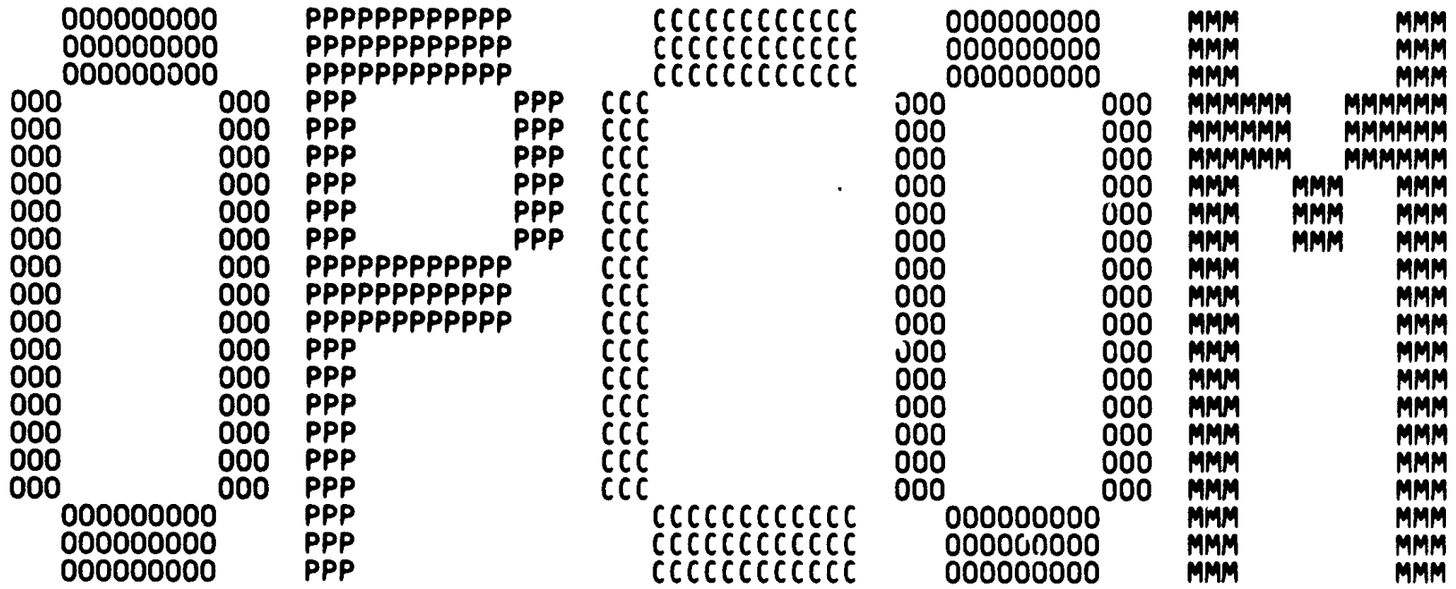


_$2

Sym

ALL

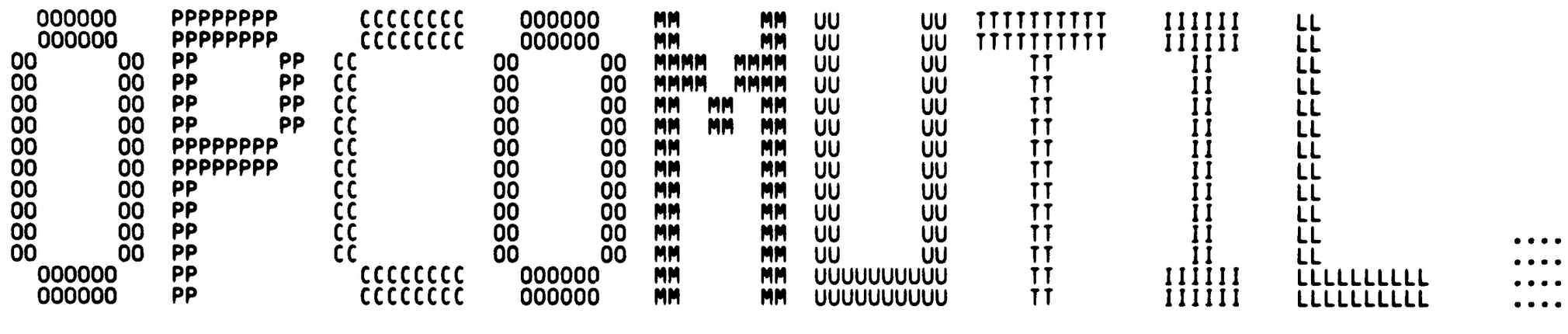
ASC

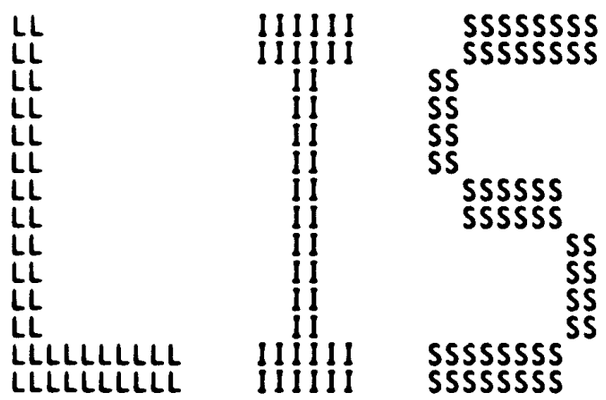
BOD
BOD
BOD
BOD
BOD
BOD
BOD
BOD
BUG
BYP
CAN
CAN
CAN
CHE
CHE

CLU
CLU
CLU
CLU
CLU
CLU
CLU

CLU
CLU
CLU
CLU
CLU
CLU
CLU
CLU

CLU
CLU





OP
VO

```
;     1          0001  0 MODULE  OPC$OPCOMUTIL   (
;     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
;    33          0033  0 !
;    34          0034  0 ! ABSTRACT:
;    35          0035  0 !
;    36          0036  0 !       This module contains all the various and sundry general
;    37          0037  0 !       purpose utility routines used by OPCOM's request handlers.
;    38          0038  0 !
;    39          0039  0 ! Environment:
;    40          0040  0 !
;    41          0041  0 !       VAX/VMS operating system.
;    42          0042  0 !
;    43          0043  0 ! Author:
;    44          0044  0 !
;    45          0045  0 !       Steven T. Jeffreys
;    46          0046  0 !
;    47          0047  0 ! Creation date:
;    48          0048  0 !
;    49          0049  0 !       March 10, 1981
;    50          0050  0 !
;    51          0051  0 ! Revision history:
;    52          0052  0 !
;    53          0053  0 !       V03-006 CWH3169         CW Hobbs                        5-May-1984
;    54          0054  0 !               Second pass for cluster-wide OPCOM:
;    55          0055  0 !                - Take BYPASS privilege to write to the user's mailbox,
;    56          0056  0 !                  don't let strange protections get in the way.
;    57          0057  0 !                - Add conditional assembly code to dump messages which
```

```
;    58          0058  0 !                   are going back to the operator.
;    59          0059  0 !
;    60          0060  0 !       V03-005 RSH0116         R. Scott Hanna                  14-Mar-1984
;    61          0061  0 !               FORMAT_MESSAGE / Replace the use of the local literal
;    62          0062  0 !               MESSAGE_LENGTH with the symbol OPC$K_MAXMESSAGE.
;    63          0063  0 !
;    64          0064  0 !       V03-004 CWH3004         CW Hobbs                        30-Oct-1983
;    65          0065  0 !               Remove linefeed at front of message, magtape ACP doesn't
;    66          0066  0 !               recognize the replies.
;    67          0067  0 !
;    68          0068  0 !       V03-003 CWH3003         CW Hobbs                        16-Sep-1983
;    69          0069  0 !               Fix accvio caused by appending the final line-feed as a
;    70          0070  0 !               longword.  Use jacket routines for VM calls.
;    71          0071  0 !
;    72          0072  0 !       V03-002 CWH3001         CW Hobbs                        30-Jul-1983
;    73          0073  0 !               Various and sundry things to make OPCOM distributed
;    74          0074  0 !               across the cluster.
;    75          0075  0 !
;    76          0076  0 ! V03-001       STJ51320        Steven T. Jeffreys,     01-Dec-1982
;    77          0077  0 !               Fix message length problem in SEND_REPLY.
;    78          0078  0 !
;    79          0079  0 ! V02-004       STJ0159         Steven T. Jeffreys,     08-Feb-1982
;    80          0080  0 !               Include a 'bell' character in front of each message
;    81          0081  0 !               that is formatted.  Care must be taken to insure that
;    82          0082  0 !               the 'bell' is not output to the log file.
;    83          0083  0 !
;    84          0084  0 ! V02-003       STJ0091         Steven T. Jeffreys,     13-Aug-1981
;    85          0085  0 !               Suppress facility code for INITAPE and BLANKTAPE.
;    86          0086  0 !
;    87          0087  0 ! V02-002       STJ0044         Steven T. Jeffreys,     23-May-1981
;    88          0088  0 !               Modify FORMAT_MESSAGE to suppress facility code for
;    89          0089  0 !               RQSTCMPLTE, RQSTABORT, and RQSTPEND messages.  This
;    90          0090  0 !               is necessary to enable existing code to parse the
;    91          0091  0 !               operator reply messages.
;    92          0092  0 !
;    93          0093  0 !--
;    94          0094  0
;    95          0095  1 BEGIN                                           ! Start of OPCOMUTIL
;    96          0096  1
;    97          0097  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    98          0098  1 LIBRARY 'LIB$:OPCOMLIB';
;    99          0099  1
;   100          0100  1 FORWARD ROUTINE
;   101          0101  1         CHECK_REQUEST,                  ! Common sanity checks
;   102          0102  1         FIND_OCD,                       ! Find an OCD with a given UIC or SCOPE
;   103          0103  1         FORMAT_MESSAGE,                 ! Format an OPCOM message for output
;   104          0104  1         INTERPRET_MASK,                 ! Interpret an attention mask
;   105          0105  1         SEND_REPLY,                     ! Send a reply to a requestor
;   106          0106  1         TRIM_LENGTH,                    ! Length of text of blank padded string
;   107          0107  1
;   108          0108  1 BUILTIN
;   109          0109  1
;   110          0110  1         INSQUE,                         ! Insert entry onto a queue
;   111          0111  1         REMQUE;                         ! Remove entry from a queue
;   112          0112  1
```

```
;  114     0113  1 GLOBAL ROUTINE CHECK_REQUEST (BUFFER_DESC, BLOCK) =
;  115     0114  1
;  116     0115  1 !++
;  117     0116  1 ! Functional description:
;  118     0117  1 !
;  119     0118  1 !       None.
;  120     0119  1 !
;  121     0120  1 ! Input:
;  122     0121  1 !
;  123     0122  1 !       BUFFER_DESC     : Address of a quadword buffer descriptor that
;  124     0123  1 !                         points to the buffer containing the request
;  125     0124  1 !                         read from the operator mailbox.
;  126     0125  1 !
;  127     0126  1 ! Implicit Input:
;  128     0127  1 !
;  129     0128  1 !       None.
;  130     0129  1 !
;  131     0130  1 ! Output:
;  132     0131  1 !
;  133     0132  1 !       RQCB            : Address of longword to receive the
;  134     0133  1 !                         address of an RQCB block.
;  135     0134  1 !
;  136     0135  1 ! Implict output:
;  137     0136  1 !
;  138     0137  1 !       None.
;  139     0138  1 !
;  140     0139  1 ! Side effects:
;  141     0140  1 !
;  142     0141  1 !       The RQCB block is partially filled in.
;  143     0142  1 !
;  14.     0143  1 ! Routine value:
;  145     0144  1 !
;  146     0145  1 !       TRUE            : The request passed the sanity checks
;  147     0146  1 !       <anything else> : The request failed a sanity check
;  148     0147  1 !--
;  149     0148  1
;  150     0149  2 BEGIN                                           ! Start of CHECK_REQUEST
;  151     0150  2
;  152     0151  2 MAP
;  153     0152  2         BUFFER_DESC     : $ref_bblock;          ! Buffer descriptor
;  154     0153  2
;  155     0154  2 EXTERNAL ROUTINE
;  156     0155  2         ALLOCATE_DS;                            ! Allocate a data structure
;  157     0156  2
;  158     0157  2 EXTERNAL LITERAL
;  159     0158  2         RQCB_K_TYPE,                            ! RQCB structure type
;  160     0159  2         MIN_SCOPE,                              ! Minimum scope value
;  161     0160  2         MAX_SCOPE;                              ! Maximum scope value
;  162     0161  2
;  163     0162  2 LOCAL
;  164     0163  2         RQCB            : $ref_bblock,          ! RQCB structure
;  165     0164  2         MSG             : $ref_bblock,          ! Pointer to request
;  166     0165  2         STATUS          : LONG;
;  167     0166  2
;  168     0167  2 .BLOCK = 0;                                     ! Assume the request will fail
;  169     0168  2 !
;  170     0169  2 ! Create a pointer to the request independent data within the message buffer.
```

```
;  171          0170  2 !
;  172          0171  2 MSG = .BUFFER_DESC [DSC$A_POINTER] + OPC$K_COMHDRSIZ;
;  173          0172  2 !
;  174          0173  2 ! If the SCOPE is bogus, then ignore the request.
;  175          0174  2 !
;  176          0175  3 IF (.MSG [OPC$B_SCOPE] LSS MIN_SCOPE) OR (.MSG [OPC$B_SCOPE] GTR MAX_SCOPE)
;  177          0176  2 THEN
;  178          0177  2     RETURN (SS$_BADPARAM);
;  179          0178  2 !
;  180          0179  2 ! Allocate an RQCB
;  181          0180  2 !
;  182          0181  3 IF NOT (STATUS = ALLOCATE_DS (RQCB_K_TYPE,RQCB))
;  183          0182  2 THEN
;  184          0183  2     RETURN (.STATUS);
;  185          0184  2 !
;  186          0185  2 ! Copy the request independent data to the RQCB.
;  187          0186  2 !
;  188          0187  2 CH$MOVE ((SBYTEOFFSET(RQCB_L_MCB) - SBYTEOFFSET(RQCB_W_MSGTYPE)),
;  189          0188  2          .BUFFER_DESC [DSC$A_POINTER],
;  190          0189  2          RQCB [RQCB_W_MSGTYPE]
;  191          0190  2         );
;  192          0191  2 !
;  193          0192  2 ! Set the lengths of the blank-padded strings
;  194          0193  2 !
;  195          0194  2 RQCB [RQCB_W_USERNAMELEN] = TRIM_LENGTH (RQCB_S_USERNAME, RQCB [RQCB_T_USERNAME]);
;  196          0195  2 RQCB [RQCB_W_ACCOUNTLEN] = TRIM_LENGTH (RQCB_S_ACCOUNT, RQCB [RQCB_T_ACCOUNT]);
;  197          0196  2 !
;  198          0197  2 ! Default the UIC to the sender's UIC.
;  199          0198  2 !
;  200          0199  2 IF .RQCB [RQCB_L_UIC] EQL 0
;  201          0200  2 THEN
;  202          0201  2     RQCB [RQCB_L_UIC] = .RQCB [RQCB_L_SENDERUIC];          ! Set UIC
;  203          0202  2
;  204          0203  2 .BLOCK = .RQCB;                                 ! Set RQCB address
;  205          0204  2 RETURN (TRUE);                                  ! If we get this far, it's ok.
;  206          0205  1 END;                                            ! End of CHECK_REQUEST


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

                                                                        .EXTRN  ALLOCATE_DS, RQCB_K_TYPE
                                                                        .EXTRN  MIN_SCOPE, MAX_SCOPE

                                                                        .PSECT  $CODE$,NOWRT,2

                                                 007C 00000             .ENTRY  CHECK_REQUEST, Save R2,R3,R4,R5,R6      ; 0113
                                              5E 04 C2 00002            SUBL2   #4, SP                                  ;
                                           08 BC D4 00005               CLRL    @BLOCK                                  ; 0167
                                              52 04 AC D0 00008         MOVL    BUFFER_DESC, R2                         ; 0171
                                        50 04 A2 26 C1 0000C            ADDL3   #38, 4(R2), MSG                         ;
00000000G 8F    01 A0          08       00 ED 00011                     CMPZV   #0, #8, 1(MSG), #MIN_SCOPE              ; 0175
                                         0C 19 0001B                    BLSS    1$                                      ;
00000000G 8F    01 A0          08       00 ED 0001D                     CMPZV   #0, #8, 1(MSG), #MAX_SCOPE              ;
                                         04 15 00027                    BLEQ    2$                                      ;
                                     50  14 D0 00029 1$:                MOVL    #20, R0                                 ; 0177
                                         04 0002C                       RET                                             ;
```

```
                                                5E  DD 0002D 2$:    PUSHL   SP                                      ; 0181
                                 00000000G      8F  DD 0002F        PUSHL   #RQCB_K_TYPE
                          0000G  CF             02  FB 00035        CALLS   #2, ALLOCATE_DS                         ;
                                 38             50  E9 0003A        BLBC    STATUS, 4$                              ;
                                 56             6E  D0 0003D        MOVL    RQCB, R6                                ; 0189
                    2C    A6       04   B2      0040  8F  28 00040  MOVC3   #64, @4(R2), 44(R6)                    ;
                                                 3C  A6  9F 00048  PUSHAB  60(R6)                                  ; 0194
                                                 0C  DD  0004B     PUSHL   #12                                     ;
                          0000V  CF              02  FB  0004D     CALLS   #2, TRIM_LENGTH                         ;
                            74   A6              50  B0  00052     MOVW    R0, 116(R6)                             ;
                                                 48  A6  9F 00056  PUSHAB  72(R6)                                  ; 0195
                                                 08  DD  00059     PUSHL   #8                                      ;
                          0000V  CF              02  FB  0005B     CALLS   #2, TRIM_LENGTH                         ;
                            76   A6              50  B0  00060     MOVW    R0, 118(R6)                             ;
                                                 68  A6  D5 00064  TSTL    104(R6)                                 ; 0199
                                                 05  12  00067     BNEQ    3$                                      ;
                            68   A6       38     A6  D0  00069     MOVL    56(R6), 104(R6)                         ; 0201
                            08   BC              56  D0  0006E 3$: MOVL    R6, @BLOCK                              ; 0203
                                 50              01  D0  00072     MOVL    #1, R0                                  ; 0204
                                                     04  00075 4$:  RET                                             ; 0205

; Routine Size:  118 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  208          0206  1 GLOBAL ROUTINE FIND_OCD (SCOPE, UIC, BLOCK) =
;  209          0207  1
;  210          0208  1 !++
;  211          0209  1 ! Functional description:
;  212          0210  1 !
;  213          0211  1 !       This routine will scan through the OCD data base and locate the
;  214          0212  1 !       OCD described by the given SCOPE and UIC.
;  215          0213  1 !
;  216          0214  1 ! Input:
;  217          0215  1 !
;  218          0216  1 !       SCOPE   : The SCOPE of the target OCD.
;  219          0217  1 !       UIC     : The UIC of the target OCD.
;  220          0218  1 !
;  221          0219  1 ! Implicit Input:
;  222          0220  1 !
;  223          0221  1 !       None.
;  224          0222  1 !
;  225          0223  1 ! Output:
;  226          0224  1 !
;  227          0225  1 !       BLOCK   : Address of a longword to receive the address of the OCD.
;  228          0226  1 !                 If the OCD is not found, then the address will be zero.
;  229          0227  1 !
;  230          0228  1 ! Implict output:
;  231          0229  1 !
;  232          0230  1 !       None.
;  233          0231  1 !
;  234          0232  1 ! Side effects:
;  235          0233  1 !
;  236          0234  1 !       None.
;  237          0235  1 !
;  238          0236  1 ! Routine value:
;  239          0237  1 !
;  240          0238  1 !       TRUE    : If the OCD is found
;  241          0239  1 !       FALSE   : If the OCD is not found
;  242          0240  1 !--
;  243          0241  1
;  244          0242  2 BEGIN                                           ! Start of FIND_OCD
;  245          0243  2
;  246          0244  2 MAP
;  247          0245  2         UIC             : $ref_bblock;                  ! Target UIC
;  248          0246  2
;  249          0247  2 EXTERNAL LITERAL
;  250          0248  2         MIN_SCOPE,                              ! Minimum (most privileged) scope
;  251          0249  2         MAX_SCOPE;                              ! Maximum (least privileged) scope
;  252          0250  2
;  253          0251  2 EXTERNAL
;  254          0252  2         OCD_VECTOR      : VECTOR;               ! Pointers to OCD list heads
;  255          0253  2
;  256          0254  2 LOCAL
;  257          0255  2         OCD             : $ref_bblock,          ! OCd data structure
;  258          0256  2         OCD_COUNT       : LONG,                 ! Count of OCDs in the OCD list
;  259          0257  2         FOUND           : LONG;                 ! Used as loop control and routine value holder
;  260          0258  2
;  261          0259  2 !
;  262          0260  2 ! Seach through the specified list of OCDs to locate
;  263          0261  2 ! an OCD that matches the specified UIC.  The OCD list
;  264          0262  2 ! scanned depends on the SCOPE.
```

```
;  265     0263  2 !
;  266     0264  2 .BLOCK = 0;                                     ! Zero address
;  267     0265  2 SELECTONE .SCOPE OF
;  268     0266  2     SET
;  269     0267  3     [OPC$K_SYSTEM]      : BEGIN
;  270     0268  3                           !
;  271     0269  3                           ! The SYSTEM OCD list is a special case, as the list
;  272     0270  3                           ! length will either be 1 or 0.  Get the address of
;  273     0271  3                           ! the OCD list header, and return the list count as
;  274     0272  3                           ! the routine value.
;  275     0273  3                           !
;  276     0274  3                           .BLOCK = .OCD_VECTOR [(.SCOPE - 1)*2];
;  277     0275  3                           FOUND = .OCD_VECTOR [(.SCOPE - 1)*2+1];
;  278     0276  2                           END;
;  279     0277  2
;  280     0278  3     [OPC$K_GROUP]       : BEGIN
;  281     0279  3                           !
;  282     0280  3                           ! Scan through the GROUP OCD list seaching for an OCD
;  283     0281  3                           ! with a UIC group field that matched the one specified.
;  284     0282  3                           !
;  285     0283  3                           OCD = .OCD_VECTOR [(.SCOPE - 1)*2];           ! Get address of first OCD in list
;  286     0284  3                           OCD_COUNT = .OCD_VECTOR [(.SCOPE - 1)*2+1];   ! Get # of OCDs in the list
;  287     0285  3                           FOUND = FALSE;                                ! Assume not found
;  288     0286  3                           WHILE (.OCD_COUNT GTR 0) AND (NOT .FOUND) DO
;  289     0287  3                               IF .UIC [0,16,16,0] EQL .OCD [$BYTEOFFSET (OCD_L_UIC),16,16,0]
;  290     0288  3                               THEN
;  291     0289  4                                    BEGIN                                ! We found it
;  292     0290  4                                    .BLOCK = .OCD;                       ! Save addres of OCD
;  293     0291  4                                    FOUND = TRUE;                        ! Note that we found it
;  294     0292  4                                    END
;  295     0293  3                               ELSE
;  296     0294  4                                    BEGIN
;  297     0295  4                                    OCD_COUNT = .OCD_COUNT - 1;          ! Decrement OCD count
;  298     0296  4                                    OCD = .OCD [OCD_L_FLINK];            ! Get address of next OCD
;  299     0297  3                                    END;
;  300     0298  2                           END;
;  301     0299  2
;  302     0300  3     [OPC$K_USER]        : BEGIN
;  303     0301  3                           !
;  304     0302  3                           ! Scan through the list of USER OCDs looking for an
;  305     0303  3                           ! OCD with the SAME UIC as the specified UIC.
;  306     0304  3                           !
;  307     0305  3                           OCD = .OCD_VECTOR [(.SCOPE - 1)*2];           ! Get address of first OCD in list
;  308     0306  3                           OCD_COUNT = .OCD_VECTOR [(.SCOPE - 1)*2+1];   ! Get # of OCDs in the list
;  309     0307  3                           FOUND = FALSE;                                ! Assume not found
;  310     0308  3                           WHILE (.OCD_COUNT GTR 0) AND (NOT .FOUND) DO
;  311     0309  3                               IF .UIC EQL .OCD [OCD_L_UIC]
;  312     0310  3                               THEN
;  313     0311  4                                    BEGIN                                ! We found it
;  314     0312  4                                    .BLOCK = .OCD;                       ! Save addres of OCD
;  315     0313  4                                    FOUND = TRUE;                        ! Note that we found it
;  316     0314  4                                    END
;  317     0315  3                               ELSE
;  318     0316  4                                    BEGIN
;  319     0317  4                                    OCD_COUNT = .OCD_COUNT - 1;          ! Decrement OCD count
;  320     0318  4                                    OCD = .OCD [OCD_L_FLINK];            ! Get address of next OCD
;  321     0319  3                                    END;
```

```
;  322     0320  2                        END;
;  323     0321  2
;  324     0322  2      [OTHERWISE]         : FOUND = FALSE;                                ! Bogus SCOPE
;  325     0323  2      TES;
;  326     0324  2
;  327     0325  2 RETURN (.FOUND);
;  328     0326  2
;  329     0327  1 END;                                                 ! End of FIND_OCD


                                                        .EXTRN  OCD_VECTOR

                                                001C 00000      .ENTRY  FIND_OCD, Save R2,R3,R4                 ; 0206
                                          54    0000G  CF  9E 00002     MOVAB   OCD_VECTOR-8, R4
                                                  0C   BC  D4 00007     CLRL    @BLOCK                          ; 0264
                                          51      04   AC  D0 0000A     MOVL    SCOPE, R1                       ; 0265
                                          01            51  D1 0000E     CMPL    R1, #1                          ; 0267
                                                        10  12 00011     BNEQ    1$
                                    50    51            01  78 00013     ASHL    #1, R1, R0                      ; 0274
                                                6440    D0 00017        MOVL    OCD_VECTOR-8[R0], @BLOCK
                                    0C    BC      A440  D0 0001C        MOVL    OCD_VECTOR-4[R0], FOUND         ; 0275
                                                  6B  11 00021        BRB     8$                              ; 0265
                                          02            51  D1 00023 1$:  CMPL    R1, #2                          ; 0278
                                                        32  12 00026     BNEQ    4$
                                    50    51            01  78 00028     ASHL    #1, R1, R0                      ; 0283
                                          51        6440  D0 0002C        MOVL    OCD_VECTOR-8[R0], OCD
                                          52    04  A440  D0 00030        MOVL    OCD_VECTOR-4[R0], OCD_COUNT     ; 0284
                                                        53  D4 00035     CLRL    FOUND                           ; 0285
                                                        52  D5 00037 2$:  TSTL    OCD_COUNT                       ; 0286
                                                        53  15 00039     BLEQ    8$
                                          50            53  E8 0003B     BLBS    FOUND, 8$
                                          50        26  A1  3C 0003E     MOVZWL  38(OCD), R0                     ; 0287
                          50        08    BC          10  ED 00042     CMPZV   #16, #16, @UIC, R0
                                                        09  12 00048     BNEQ    3$
                                                0C  BC      51  D0 0004A     MOVL    OCD, @BLOCK                     ; 0290
                                          53            01  D0 0004E     MOVL    #1, FOUND                       ; 0291
                                                        E4  11 00051     BRB     2$                              ; 0287
                                                        52  D7 00053 3$:  DECL    OCD_COUNT                       ; 0295
                                          51            61  D0 00055     MOVL    (OCD), OCD                      ; 0296
                                                        DD  11 00058     BRB     2$                              ; 0287
                                          03            51  D1 0005A 4$:  CMPL    R1, #3                          ; 0300
                                                        2D  12 0005D     BNEQ    7$
                                    50    51            01  78 0005F     ASHL    #1, R1, R0                      ; 0305
                                          51        6440  D0 00063        MOVL    OCD_VECTOR-8[R0], OCD
                                          52    04  A440  D0 00067        MOVL    OCD_VECTOR-4[R0], OCD_COUNT     ; 0306
                                                        53  D4 0006C     CLRL    FOUND                           ; 0307
                                                        52  D5 0006E 5$:  TSTL    OCD_COUNT                       ; 0308
                                                        1C  15 00070     BLEQ    8$
                                          19            53  E8 00072     BLBS    FOUND, 8$
                                                24  A1  08  AC  D1 00075     CMPL    UIC, 36(OCD)                    ; 0309
                                                        09  12 0007A     BNEQ    6$
                                                0C  BC      51  D0 0007C     MOVL    OCD, @BLOCK                     ; 0312
                                          53            01  D0 00080     MOVL    #1, FOUND                       ; 0313
                                                        E9  11 00083     BRB     5$                              ; 0309
                                                        52  D7 00085 6$:  DECL    OCD_COUNT                       ; 0317
                                          51            61  D0 00087     MOVL    (OCD), OCD                      ; 0318
```

```
                                E2  11 0008A          BRB     5$                              ; 0309
                                53  D4 0008C 7$:      CLRL    FOUND                           ; 0322
                          50    53  D0 0008E 8$:      MOVL    FOUND, R0                       ; 0325
                                    04 00091          RET                                     ; 0327

; Routine Size:  146 bytes,    Routine Base:  $CODE$ + 0076
```

```
;  331     0328  1 GLOBAL ROUTINE FORMAT_MESSAGE (RQCB, MESSAGE_VECTOR) =
;  332     0329  1
;  333     0330  1 !++
;  334     0331  1 ! Functional description:
;  335     0332  1 !
;  336     0333  1 !       Given an RQCB address and a message code, this routine will
;  337     0334  1 !       format the message and build an MCB to describe the message.
;  338     0335  1 !
;  339     0336  1 ! Input:
;  340     0337  1 !
;  341     0338  1 !       RQCB            : Address of an RQCB.
;  342     0339  1 !       MESSAGE_VECTOR  : Address of a longword vector
;  343     0340  1 !
;  344     0341  1 ! Implicit Input:
;  345     0342  1 !
;  346     0343  1 !       The format of the message vector is as follows:
;  347     0344  1 !
;  348     0345  1 !               +--------------------------------+
;  349     0346  1 !               |          message number        |
;  350     0347  1 !               +--------------------------------+
;  351     0348  1 !               |          arguement count       |
;  352     0349  1 !               +--------------------------------+
;  353     0350  1 !               |          first arguement       |
;  354     0351  1 !               +--------------------------------+
;  355     0352  1 !               |                .
;  356     0353  1 !                              .
;  357     0354  1 !                              .
;  358     0355  1 !               |          Nth arguement         |
;  359     0356  1 !               +--------------------------------+
;  360     0357  1 !
;  361     0358  1 ! Output:
;  362     0359  1 !
;  363     0360  1 !       None.
;  364     0361  1 !
;  365     0362  1 ! Implict output:
;  366     0363  1 !
;  367     0364  1 !       None.
;  368     0365  1 !
;  369     0366  1 ! Side effects:
;  370     0367  1 !
;  371     0368  1 !       None.
;  372     0369  1 !
;  373     0370  1 ! Routine value:
;  374     0371  1 !
;  375     0372  1 !       TRUE            : If the message was formatted and an MCB created
;  376     0373  1 !       <anything else> : The message could not be formatted and an MCB created.
;  377     0374  1 !--
;  378     0375  1
;  379     0376  2 BEGIN
;  380     0377  2
;  381     0378  2 MAP
;  382     0379  2         MESSAGE_VECTOR  : REF VECTOR,                   ! Message info
;  383     0380  2         RQCB            : $ref_bblock;                  ! RQCB data structure
;  384     0381  2
;  385     0382  2 EXTERNAL ROUTINE
;  386     0383  2         ALLOCATE_DS,                                    ! Allocate a data structure
;  387     0384  2         DEALLOCATE_DS;                                  ! Deallocate a data structure
```

```
;  388          0385  2
;  389          0386  2 EXTERNAL LITERAL
;  390          0387  2         RQCB_K_TYPE,
;  391          0388  2         MCB_K_TYPE;
;  392          0389  2
;  393          0390  2 LITERAL
;  394          0391  2         ASCII_BELL      = 7,                    ! ASCII 'bell' character
;  395          0392  2         LINE_FEED       = 10,                   ! ASCII line feed
;  396          0393  2         MSG_FLAGS1      = 9,                    ! Get message facility and text.
;  397          0394  2         MSG_FLAGS2      = 1,                    ! Get message text only.
;  398          0395  2         TEXT_LENGTH     = 256;                  ! Max VMS message text size
;  399          0396  2
;  400          0397  2 OWN
;  401          0398  2         HEADER_BUF      : VECTOR [100],         ! Buffer to hold header FAO string
;  402          0399  2         HEADER_DESC     : VECTOR [2, LONG]      ! Descriptor for header string
;  403          0400  2                         PRESET ([1] = HEADER_BUF);
;  404          0401  2
;  405          0402  2 LOCAL
;  406          0403  2         OUT_LENGTH      : LONG,                 ! Message text length
;  407          0404  2         OUT_ARRAY       : LONG,                 ! Output for $GETMSG
;  408          0405  2         TEXT_DESC       : $desc_block,          ! Message text descriptor
;  409          0406  2         TEXT_BUFFER     : $bblock [TEXT_LENGTH],! Maximum message size
;  410          0407  2         TEXT_BUFFER2    : $bblock [TEXT_LENGTH+100],! Reformatted message ctrstr
;  411          0408  2         MESSAGE_DESC    : $desc_block,          ! Formatted message descriptor
;  412          0409  2         MESSAGE_BUFFER  : $bblock [OPC$K_MAXMESSAGE],! Formatted message buffer
;  413          0410  2         BUFFER_POINTER  : $ref_bvector,         ! Holds address of a VM block
;  414          0411  2         MCB             : $ref_bblock,          ! MCB data structure
;  415          0412  2         BLOCK           : $ref_bblock,          ! VM block for message
;  416          0413  2         STATUS          : LONG;
;  417          0414  2 !
;  418          0415  2 ! Set up buffer descriptors
;  419          0416  2 !
;  420          0417  2 TEXT_DESC [0,0,32,0] = TEXT_LENGTH;             ! Set text buffer size
;  421          0418  2 TEXT_DESC [DSC$A_POINTER] = TEXT_BUFFER;        ! Set text buffer address
;  422          0419  2 MESSAGE_DESC [0,0,32,0] = OPC$K_MAXMESSAGE;    ! Clear message size
;  423          0420  2 MESSAGE_DESC [DSC$A_POINTER] = MESSAGE_BUFFER; ! Set message buffer address
;  424          0421  2
;  425          0422  2 !
;  426          0423  2 ! Get the message text via $GETMSG.
;  427          0424  2 !
;  428        P 0425  3 IF NOT (STATUS = $GETMSG (MSGID  = .MESSAGE_VECTOR [0],
;  429        P 0426  3                           MSGLEN =  OUT_LENGTH,
;  430        P 0427  3                           BUFADR =  TEXT_DESC,
;  431        P 0428  3                           FLAGS  =  (IF .MESSAGE_VECTOR [0] EQL OPC$_RQSTCMPLTE
;  432        P 0429  3                                      OR .MESSAGE_VECTOR [0] EQL OPC$_RQSTABORT
;  433        P 0430  3                                      OR .MESSAGE_VECTOR [0] EQL OPC$_RQSTPEND
;  434        P 0431  3                                      OR .MESSAGE_VECTOR [0] EQL OPC$_BLANKTAPE
;  435        P 0432  3                                      OR .MESSAGE_VECTOR [0] EQL OPC$_INITAPE
;  436        P 0433  3                                      THEN
;  437        P 0434  3                                          MSG_FLAGS2
;  438        P 0435  3                                      ELSE
;  439        P 0436  3                                          MSG_FLAGS1
;  440        P 0437  3                                     ),
;  441        P 0438  3                           OUTADR =  OUT_ARRAY
;  442          0439  3                          ))
;  443          0440  2 THEN
;  444          0441  2     $signal_stop (.STATUS);
```

```
;  445          0442  2 TEXT_DESC [DSC$W_LENGTH] = .OUT_LENGTH;
;  446          0443  2 !
;  447          0444  2 ! If the message control string starts with "%OPCOM, %D%", then perform a little substitution
;  448          0445  2 !
;  449          0446  2 IF CH$EQL (11, UPLIT BYTE ('%OPCOM, %D%'), MINU (11, .OUT_LENGTH), TEXT_BUFFER, 0)
;  450          0447  2 THEN
;  451          0448  3     BEGIN
;  452          0449  3     LOCAL
;  453          0450  3         ADJ;
;  454          0451  3     !
;  455          0452  3     ! On the first reference, get the header text from the message file
;  456          0453  3     !
;  457          0454  3     IF .HEADER_DESC [0] EQL 0
;  458          0455  3     THEN
;  459          0456  4         BEGIN
;  460          0457  4         HEADER_DESC [0] = 100;
;  461          0458  5         IF NOT (STATUS = $GETMSG (MSGID=OPC$_HEADER, MSGLEN=HEADER_DESC, BUFADR=HEADER_DESC, FLAGS=1))
;  462          0459  4         THEN
;  463          0460  4             $signal_stop (.STATUS);
;  464          0461  3         END;
;  465          0462  3     ADJ = .HEADER_DESC [0] - 11;
;  466          0463  3     CH$COPY (.HEADER_DESC [0], .HEADER_DESC [1],
;  467          0464  3              .OUT_LENGTH-11, TEXT_BUFFER+11, 0, .OUT_LENGTH + .ADJ, TEXT_BUFFER2);
;  468          0465  3     TEXT_DESC [DSC$W_LENGTH] = .OUT_LENGTH + .ADJ;
;  469          0466  3     TEXT_DESC [DSC$A_POINTER] = TEXT_BUFFER2;
;  470          0467  2     END;
;  471          0468  2 !
;  472          0469  2 ! Format the message via FAOL.
;  473          0470  2 !
;  474          0471  2 OUT_LENGTH = 0;
;  475        P 0472  3 IF NOT (STATUS = $FAOL  (CTRSTR = TEXT_DESC,
;  476        P 0473  3                          OUTLEN = OUT_LENGTH,
;  477        P 0474  3                          OUTBUF = MESSAGE_DESC,
;  478        P 0475  3                          PRMLST = MESSAGE_VECTOR [1]
;  479          0476  3                         ))
;  480          0477  2 THEN
;  481          0478  2     $signal_stop (.STATUS);
;  482          0479  2 MESSAGE_DESC [DSC$W_LENGTH] = .OUT_LENGTH;
;  483          0480  2
;  484          0481  2 !
;  485          0482  2 ! Allocate a block of memory to hold the formatted message text,
;  486          0483  2 ! and copy the message text to it.  Tack a 'bell' onto
;  487          0484  2 ! the front of the message.  Add an extra line feed to the end.
;  488          0485  2 !
;  489          0486  2 OUT_LENGTH = .OUT_LENGTH + 2;
;  490          0487  3 IF NOT (STATUS = OPC$GET_VM (OUT_LENGTH, BUFFER_POINTER))
;  491          0488  2 THEN
;  492          0489  2     RETURN (.STATUS);
;  493          0490  2 BUFFER_POINTER [0] = ASCII_BELL;
;  494          0491  2 CH$MOVE (.OUT_LENGTH-2, .MESSAGE_DESC [DSC$A_POINTER], .BUFFER_POINTER+1);
;  495          0492  2 BUFFER_POINTER [.OUT_LENGTH-1] = LINE_FEED;
;  496          0493  2
;  497          0494  2 !
;  498          0495  2 ! Allocate an MCB for the message.
;  499          0496  2 ! If one is already present on the RQCB, then
;  500          0497  2 ! deallocate it and allocate a new one.
;  501          0498  2 ! Fill in the required MCB fields and attach it
```

```
;  502          0499  2 ! to the RQCB.  If the allocate fails, then free
;  503          0500  2 ! the message buffer block that was just allocated.
;  504          0501  2 !
;  505          0502  2 IF .RQCB [RQCB_L_MCB] NEQ 0
;  506          0503  2 THEN
;  507          0504  2     DEALLOCATE_DS (.RQCB [RQCB_L_MCB]);
;  508          0505  2
;  509          0506  3 IF NOT (STATUS = ALLOCATE_DS (MCB_K_TYPE, MCB))
;  510          0507  2 THEN
;  511          0508  2     RETURN (.STATUS);
;  512          0509  2 MCB [MCB_L_MSGID] = .MESSAGE_VECTOR [0];        ! Set message code
;  513          0510  2 MCB [MCB_L_TEXTLEN] = .OUT_LENGTH;              ! Set the message length
;  514          0511  2 MCB [MCB_L_TEXTPTR] = .BUFFER_POINTER;          ! Set message address
;  515          0512  2 MCB [MCB_L_RQCB] = RQCB [HDR_L_FLINK];          ! Set RQCB address
;  516          0513  2 RQCB [RQCB_L_MCB] = .MCB;                       ! Set MCB address
;  517          0514  2
;  518          0515  2 RETURN (TRUE);
;  519          0516  2
;  520          0517  1 END;                                            ! End of FORMAT_MESSAGE


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2
                25 44 25 20 2C 4D 4F 43 50 4F 25  00000 P.AAA:  .ASCII  \%OPCOM, %D%\                          ;

                                                        .PSECT  $OWN$,NOEXE,2

                                                  00000 HEADER_BUF:
                                                        .BLKB   400
                                                  00# 00190 HEADER_DESC:
                                                        .BYTE   0[4]                                          ;
                                            00000000' 00194     .ADDRESS HEADER_BUF                           ;

                                                        .EXTRN  DEALLOCATE_DS, MCB_K_TYPE
                                                        .EXTRN  SYS$GETMSG, LIB$STOP
                                                        .EXTRN  SYS$FAOL, OPC$GET_VM

                                                        .PSECT  $CODE$,NOWRT,2

                                                   0FFC 00000     .ENTRY  FORMAT_MESSAGE, Save R2,R3,R4,R5,R6,R7,R8,- ; 0328
                                                                          R9,R10,R11                          ;
                    5B      0000'   CF   9E 00002               MOVAB   HEADER_DESC, R11                    ;
                    5E      F57C    CE   9E 00007               MOVAB   -2692(SP), SP                       ;
                F8  AD      0100    8F   3C 0000C               MOVZWL  #256, TEXT_DESC                     ; 0417
                FC  AD      FEF8    CD   9E 00012               MOVAB   TEXT_BUFFER, TEXT_DESC+4            ; 0418
              FD8C  CD      0800    8F   3C 00018               MOVZWL  #2048, MESSAGE_DESC                 ; 0419
              FD90  CD        10    AE   9E 0001F               MOVAB   MESSAGE_BUFFER, MESSAGE_DESC+4      ; 0420
                                    5E   DD 00025               PUSHL   SP                                  ; 0439
                    50        08    BC   D0 00027               MOVL    @MESSAGE_VECTOR, R0                 ;
          00058029  8F              50   D1 0002B               CMPL    R0, #360489                         ;
                                    24   13 00032               BEQL    1$                                  ;
          0005801C  8F              50   D1 00034               CMPL    R0, #360476                         ;
                                    1B   13 00038               BEQL    1$                                  ;
          00058021  8F              50   D1 0003D               CMPL    R0, #360481                         ;
                                    12   13 00044               BEQL    1$                                  ;
          000581E3  8F              50   D1 00046               CMPL    R0, #360931                         ;
```

```
                                   09 13 0004D       BEQL    1$                                          ;
             000581D3  8F          50 D1 0004F       CMPL    R0, #360915                                 ;
                                   04 12 00056       BNEQ    2$                                          ;
                                   01 DD 00058 1$:   PUSHL   #1                                          ;
                                   02 11 0005A       BRB     3$                                          ;
                                   09 DD 0005C 2$:   PUSHL   #9                                          ;
                             F8    AD 9F 0005E 3$:   PUSHAB  TEXT_DESC                                   ;
                             14    AE 9F 00061       PUSHAB  OUT_LENGTH                                  ;
                                   50 DD 00064       PUSHL   R0                                          ;
             00000000G  00         05 FB 00066       CALLS   #5, SYS$GETMSG                              ;
                        5A         50 D0 0006D       MOVL    R0, STATUS                                  ;
                        3C         5A E9 00070       BLBC    STATUS, 5$                                  ;
                   F8   AD    08   AE B0 00073       MOVW    OUT_LENGTH, TEXT_DESC                       ; 0442
                        50    08   AE D0 00078       MOVL    OUT_LENGTH, R0                              ; 0446
                        0B         50 D1 0007C       CMPL    R0, #11                                     ;
                                   03 1B 0007F       BLEQU   4$                                          ;
                        50         0B D0 00081       MOVL    #11, R0                                     ;
  50          00   0000'  CF       0B 2D 00084 4$:   CMPC5   #11, P.AAA, #0, R0, TEXT_BUFFER             ;
                           FEF8    CD    0008B                                                           ;
                                   59 12 0008E       BNEQ    8$                                          ;
                                   6B D5 00090       TSTL    HEADER_DESC                                 ; 0454
                                   1E 12 00092       BNEQ    6$                                          ;
                        6B    64   8F 9A 00094       MOVZBL  #100, HEADER_DESC                           ; 0457
                        7E         01 7D 00098       MOVQ    #1, -(SP)                                   ; 0458
                                   5B DD 0009B       PUSHL   R11                                         ;
                                   5B DD 0009D       PUSHL   R11                                         ;
                         000582B3  8F DD 0009F       PUSHL   #361139                                     ;
             00000000G  00         05 FB 000A5       CALLS   #5, SYS$GETMSG                              ;
                        5A         50 D0 000AC       MOVL    R0, STATUS                                  ;
                        56         5A E9 000AF 5$:   BLBC    STATUS, 9$                                  ;
            50          6B         0B C3 000B2 6$:   SUBL3   #11, HEADER_DESC, ADJ                       ; 0462
            59     08   AE         0B C3 000B6       SUBL3   #11, OUT_LENGTH, R9                         ; 0464
            56          50    08   AE C1 000BB       ADDL3   OUT_LENGTH, ADJ, R6                         ;
                        58         56 D0 000C0       MOVL    R6, R8                                      ;
                        57   FD94  CD 9E 000C3       MOVAB   TEXT_BUFFER2, R7                            ; 0463
  58          00     04 BB         6B 2C 000C8       MOVC5   HEADER_DESC, @HEADER_DESC+4, #0, R8, (R7)   ;
                                   67    000CE                                                           ;
                                   0E 18 000CF       BGEQ    7$                                          ;
                        57         6B C0 000D1       ADDL2   HEADER_DESC, R7                             ;
                        58         6B C2 000D4       SUBL2   HEADER_DESC, R8                             ;
  58          00   FF03  CD        59 2C 000D7       MOVC5   R9, TEXT_BUFFER+11, #0, R8, (R7)            ;
                                   67    000DE                                                           ;
                   F8   AD         56 B0 000DF 7$:   MOVW    R6, TEXT_DESC                               ; 0465
                   FC   AD   FD94  CD 9E 000E3       MOVAB   TEXT_BUFFER2, TEXT_DESC+4                   ; 0466
                             08    AE D4 000E9 8$:   CLRL    OUT_LENGTH                                  ; 0471
            7E     08   AC         04 C1 000EC       ADDL3   #4, MESSAGE_VECTOR, -(SP)                   ; 0476
                           FD8C    CD 9F 000F1       PUSHAB  MESSAGE_DESC                                ;
                             10    AE 9F 000F5       PUSHAB  OUT_LENGTH                                  ;
                             F8    AD 9F 000F8       PUSHAB  TEXT_DESC                                   ;
             00000000G  00         04 FB 000FB       CALLS   #4, SYS$FAOL                                ;
                        5A         50 D0 00102       MOVL    R0, STATUS                                  ;
                        0A         5A E8 00105       BLBS    STATUS, 10$                                 ;
                                   5A DD 00108 9$:   PUSHL   STATUS                                      ; 0478
             00000000G  00         01 FB 0010A       CALLS   #1, LIB$STOP                                ;
                                      04 00111       RET                                                 ;
                 FD8C   CD    08   AE B0 00112 10$:  MOVW    OUT_LENGTH, MESSAGE_DESC                    ; 0479
                   08   AE         02 C0 00118       ADDL2   #2, OUT_LENGTH                              ; 0486
```

```
                                                04 AE  9F 0011C          PUSHAB  BUFFER_POINTER                          ; 0487
                                                0C AE  9F 0011F          PUSHAB  OUT_LENGTH                              ;
                                  0000G   CF            02 FB 00122      CALLS   #2, OPC$GET_VM                          ;
                                          5A            50 D0 00127      MOVL    R0, STATUS                              ;
                                          41            5A E9 0012A      BLBC    STATUS, 12$                             ;
                                          57        04  AE D0 0012D      MOVL    BUFFER_POINTER, R7                      ; 0490
                                          67            07 90 00131      MOVB    #7, (R7)                                ;
                                50        08  AE        02 C3 00134      SUBL3   #2, OUT_LENGTH, R0                      ; 0491
                          01    A7  FD90    DD        50 28 00139      MOVC3   R0, @MESSAGE_DESC+4, 1(R7)              ;
                                50            57    08  AE C1 00140      ADDL3   OUT_LENGTH, R7, R0                      ; 0492
                                  FF        A0        0A 90 00145      MOVB    #10, -1(R0)                             ;
                                          52        04  AC D0 00149      MOVL    RQCB, R2                                ; 0502
                                                    6C  A2 D5 0014D      TSTL    108(R2)                                 ;
                                                        08 13 00150      BEQL    11$                                     ;
                                                    6C  A2 DD 00152      PUSHL   108(R2)                                 ; 0504
                                  0000G   CF            01 FB 00155      CALLS   #1, DEALLOCATE_DS                       ;
                                                    0C  AE 9F 0015A 11$: PUSHAB  MCB                                    ; 0506
                                        00000000G       8F DD 0015D      PUSHL   #MCB_K_TYPE                             ;
                                  0000G   CF            02 FB 00163      CALLS   #2, ALLOCATE_DS                         ;
                                          5A            50 D0 00168      MOVL    R0, STATUS                              ;
                                          04            5A E8 0016B      BLBS    STATUS, 13$                             ;
                                          50            5A D0 0016E 12$: MOVL    STATUS, R0                              ; 0508
                                                        04 00171         RET                                             ;
                                          50        0C  AE D0 00172 13$: MOVL    MCB, R0                                 ; 0509
                                    2C    A0        08  BC D0 00176      MOVL    @MESSAGE_VECTOR, 44(R0)                 ;
                                    30    A0        08  AE D0 0017B      MOVL    OUT_LENGTH, 48(R0)                      ; 0510
                                    34    A0            57 D0 00180      MOVL    R7, 52(R0)                              ; 0511
                                    24    A0            52 D0 00184      MOVL    R2, 36(R0)                              ; 0512
                                    6C    A2            50 D0 00188      MOVL    R0, 108(R2)                             ; 0513
                                          50            01 D0 0018C      MOVL    #1, R0                                  ; 0515
                                                        04 0018F         RET                                             ; 0517

; Routine Size:  400 bytes,    Routine Base:  $CODE$ + 0108
```

```
;  522     0518  1 GLOBAL ROUTINE INTERPRET_MASK (BIT_MASK, STRING_DESC, OUTLEN)  =
;  523     0519  1
;  524     0520  1 !++
;  525     0521  1 ! Functional description:
;  526     0522  1 !
;  527     0523  1 !       This routine will take a 64 bit mask, and format an ASCII string that will name each
;  528     0524  1 !       set bit in the mask.
;  529     0525  1 !
;  530     0526  1 ! Input:
;  531     0527  1 !
;  532     0528  1 !       BIT_MASK        : Address of a 64 element bit vector
;  533     0529  1 !       STRING_DESC     : Address of a string descriptor to receive the output
;  534     0530  1 !                         string that names the bits in the mask.
;  535     0531  1 !
;  536     0532  1 ! Implicit Input:
;  537     0533  1 !
;  538     0534  1 !       None.
;  539     0535  1 !
;  540     0536  1 ! Output:
;  541     0537  1 !
;  542     0538  1 !       OUTLEN          : Address of a longword to receive the length of the
;  543     0539  1 !                         output string.
;  544     0540  1 !
;  545     0541  1 ! Implict output:
;  546     0542  1 !
;  547     0543  1 !       None.
;  548     0544  1 !
;  549     0545  1 ! Side effects:
;  550     0546  1 !
;  551     0547  1 !       None.
;  552     0548  1 !
;  553     0549  1 ! Routine value:
;  554     0550  1 !
;  555     0551  1 !       TRUE            : Successful completion
;  556     0552  1 !       <anything else> : The operation failed
;  557     0553  1 !--
;  558     0554  1
;  559     0555  2 BEGIN                                           ! Start of INTERPRET_MASK
;  560     0556  2
;  561     0557  2 MAP
;  562     0558  2         STRING_DESC     : $ref_bblock,          ! String descriptor
;  563     0559  2         BIT_MASK        : REF VECTOR;           ! Quadword mask
;  564     0560  2
;  565     0561  2 EXTERNAL
;  566     0562  2         OPER_NAME       : VECTOR [, LONG];      ! Names for bits, from OPCOMDATA.MAR
;  567     0563  2
;  568     0564  2 LITERAL
;  569     0565  2         CR              = %X'0D',               ! ASCII carriage return
;  570     0566  2         LF              = %X'0A',               ! ASCII line feed
;  571     0567  2         MSG_FLAGS       = 1,                    ! Only get message text
;  572     0568  2         MAX_NAM_LEN     = 40,                   ! Maximum bit name length
;  573     0569  2         MAX_LINE        = 80;                   ! Maximum length of an in
;  574     0570  2
;  575     0571  2 OWN
;  576     0572  2         SEPARATOR_DESC  : $string_desc (', '),  ! Name separator
;  577     0573  2         NEWLINE_DESC    : $string_desc (%CHAR (',',CR,LF));
;  578     0574  2
```

```
;  579          0575  2 LOCAL
;  580          0576  2         K                   : LONG,                     ! Index
;  581          0577  2         MASK                : BITVECTOR [32],           ! 32 bits of the quadword mask
;  582          0578  2         NAME                : REF VECTOR [, LONG],      ! Pointer to name string
;  583          0579  2         WORK_DESC           : $desc_block,              ! Descriptor of work buffer
;  584          0580  2         WORK_BUF            : $bblock [OPC$K_MAXREAD],! Work buffer
;  585          0581  2         NAME_DESC           : $desc_block,              ! Descriptor for bit name
;  586          0582  2         NAME_BUF            : $bblock [MAX_NAM_LEN],! Buffer for bit name
;  587          0583  2         PTR                 : LONG,                     ! Pointer to current point in buffer
;  588          0584  2         LEN                 : LONG,                     ! Length of current string
;  589          0585  2         STATUS              : LONG;
;  590          0586  2
;  591          0587  2 !
;  592          0588  2 ! Return FALSE if the output buffer descriptor is strange.
;  593          0589  2 !
;  594          0590  3 IF (.STRING_DESC [DSC$W_LENGTH] LEQ 0) OR (.STRING_DESC [DSC$A_POINTER] EQL 0)
;  595          0591  2 THEN
;  596          0592  2     RETURN (FALSE);
;  597          0593  2 !
;  598          0594  2 ! Set up the work buffer descriptor.
;  599          0595  2 !
;  600          0596  2 WORK_DESC [0,0,32,0] = 0;
;  601          0597  2 WORK_DESC [DSC$A_POINTER] = WORK_BUF;
;  602          0598  2 !
;  603          0599  2 ! If the bit mask is not zero, then for each bit set in the mask get the name of that bit from
;  604          0600  2 ! the name vector.  The bit name is obtained by using the bit position as an index into the OPER_NAME
;  605          0601  2 ! vector.  Each entry in the name vector is a longword pointer to a string descriptor for the name.
;  606          0602  2 ! If the name is undefined, the pointer will be zero.
;  607          0603  2 !
;  608          0604  2 PTR = .WORK_DESC [DSC$A_POINTER];
;  609          0605  2 LEN = 0;
;  610          0606  2 INCR I FROM 0 TO 1 DO
;  611          0607  3     BEGIN
;  612          0608  3     !
;  613          0609  3     ! Go through the loop twice, each pass checking 32 bits.
;  614          0610  3     ! As an optimization, avoid the loop if the mask is zero.
;  615          0611  3     !
;  616          0612  3     IF .BIT_MASK [.I] NEQ 0
;  617          0613  3     THEN
;  618          0614  4         BEGIN
;  619          0615  4         MASK = .BIT_MASK [.I];                  ! Get 32 bits to check
;  620          0616  4         INCR J FROM 0 TO 31 DO
;  621          0617  5             BEGIN
;  622          0618  5             IF .MASK [.J]
;  623          0619  5             THEN
;  624          0620  6                 BEGIN
;  625          0621  6                 NAME = .OPER_NAME [((.I * 32) + .J)];   ! Calculate byte position from bit position
;  626          0622  6                 IF .NAME EQL 0                          ! If null, use special name
;  627          0623  6                 THEN
;  628          0624  7                     BEGIN
;  629          0625  7                     NAME_DESC [0,0,32,0] = 1;
;  630          0626  7                     NAME_DESC [DSC$A_POINTER] = UPLIT BYTE ('?');
;  631          0627  7                     END
;  632          0628  6                 ELSE
;  633          0629  7                     BEGIN
;  634          0630  7                     NAME_DESC [0,0,32,0] = .NAME [0];
;  635          0631  7                     NAME_DESC [DSC$A_POINTER] = .NAME [1];
```

```
;  636     0632  6                END;
;  637     0633  6            !
;  638     0634  6            ! If the name will not fit on the line, start a new line.
;  639     0635  6            ! Allow for the ", " that preceeds each new name.  Allow
;  640     0636  6            ! one space at the end of the line to hold a ",".
;  641     0637  6            !
;  642     0638  7            IF (.LEN + .NAME_DESC [DSC$W_LENGTH] + 2) GTR (MAX_LINE - 1)
;  643     0639  6            THEN
;  644     0640  7                BEGIN
;  645     0641  7                CH$MOVE (.NEWLINE_DESC [DSC$W_LENGTH], .NEWLINE_DESC [DSC$A_POINTER], .PTR);
;  646     0642  7                PTR = .PTR + .NEWLINE_DESC [DSC$W_LENGTH];
;  647     0643  7                LEN = 0;
;  648     0644  6                END;
;  649     0645  6            !
;  650     0646  6            ! If not at the beginning of a line, preceed the name with ", ".
;  651     0647  6            !
;  652     0648  6            IF .LEN NEQ 0
;  653     0649  6            THEN
;  654     0650  7                BEGIN
;  655     0651  7                CH$MOVE (.SEPARATOR_DESC [DSC$W_LENGTH], .SEPARATOR_DESC [DSC$A_POINTER], .PTR);
;  656     0652  7                PTR = .PTR + .SEPARATOR_DESC [DSC$W_LENGTH];
;  657     0653  7                LEN = .LEN + .SEPARATOR_DESC [DSC$W_LENGTH];
;  658     0654  6                END;
;  659     0655  6            !
;  660     0656  6            ! Append the name to the end of the work buffer.
;  661     0657  6            !
;  662     0658  6            CH$MOVE (.NAME_DESC [DSC$W_LENGTH], .NAME_DESC [DSC$A_POINTER], .PTR);
;  663     0659  6            PTR = .PTR + .NAME_DESC [DSC$W_LENGTH];
;  664     0660  6            LEN = .LEN + .NAME_DESC [DSC$W_LENGTH];
;  665     0661  5            END;
;  666     0662  4        END;
;  667     0663  3        END;
;  668     0664  3    !
;  669     0665  3    ! Determine how much text was copied to the work buffer.
;  670     0666  3    !
;  671     0667  3    WORK_DESC [DSC$W_LENGTH] = .PTR - .WORK_DESC [DSC$A_POINTER];
;  672     0668  2    END;
;  673     0669  2 !
;  674     0670  2 ! If no bits were named, then return a special message.
;  675     0671  2 !
;  676     0672  2 IF .WORK_DESC [DSC$W_LENGTH] EQL 0
;  677     0673  2 THEN
;  678     0674  3     BEGIN
;  679     0675  3     WORK_DESC [DSC$W_LENGTH] = OPC$K_MAXREAD;
;  680     0676  4     IF NOT (STATUS = $GETMSG (MSGID=OPC$_NOTENABLED, MSGLEN=WORK_DESC, BUFADR=WORK_DESC, FLAGS=MSG_FLAGS))
;  681     0677  3     THEN
;  682     0678  3         RETURN (.STATUS);
;  683     0679  2     END;
;  684     0680  2 !
;  685     0681  2 ! Copy the work buffer to the output buffer.  If the output
;  686     0682  2 ! buffer is not large enough, fill the output buffer and return
;  687     0683  2 ! SS$_BUFFEROVF to indicate a buffer overflow.
;  688     0684  2 !
;  689     0685  2 STATUS = TRUE;                                  ! Assume success
;  690     0686  2 IF .STRING_DESC [DSC$W_LENGTH] LSS .WORK_DESC [DSC$W_LENGTH]
;  691     0687  2 THEN
;  692     0688  3     BEGIN
```

```
;  693          0689  3     WORK_DESC [DSC$W_LENGTH] = .STRING_DESC [DSC$W_LENGTH];
;  694          0690  3     STATUS = SS$_BUFFEROVF;
;  695          0691  2     END;
;  696          0692  2 CH$MOVE (.WORK_DESC [DSC$W_LENGTH], .WORK_DESC [DSC$A_POINTER], .STRING_DESC [DSC$A_POINTER]);
;  697          0693  2 .OUTLEN = .WORK_DESC [DSC$W_LENGTH];
;  698          0694  2 RETURN (.STATUS);
;  699          0695  2
;  700          0696  1 END;                                    ! End of INTERPRET_MASK


                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                              20  2C  0000B P.AAB:   .ASCII  \, \                                    ;
                           0A 0D  2C  0000D P.AAC:   .ASCII  \,\<13><10>                            ;
                                 3F  00010 P.AAD:   .ASCII  \?\                                    ;

                                        .PSECT  $OWN$,NOEXE,2

                                0002  00198 SEPARATOR_DESC:
                                            .WORD   2                                             ;
                                  01  0E  0019A             .BYTE   14, 1                         ;
                                00000000' 0019C             .ADDRESS P.AAB                        ;
                                0003  001A0 NEWLINE_DESC:
                                            .WORD   3                                             ;
                                  01  0E  001A2             .BYTE   14, 1                         ;
                                00000000' 001A4             .ADDRESS P.AAC                        ;

                                        .EXTRN  OPER_NAME

                                        .PSECT  $CODE$,NOWRT,2

                                      0FFC 00000            .ENTRY  INTERPRET_MASK, Save R2,R3,R4,R5,R6,R7,R8,- ; 0518
                                                                    R9,R10,R11                    ;
                                5E      F5BC    CE  9E 00002        MOVAB   -2628(SP), SP         ;
                                5B        08    AC  D0 00007        MOVL    STRING_DESC, R11      ; 0590
                                              6B  B5 0000B        TSTW    (R11)                 ;
                                              03  13 0000D        BEQL    1$                    ;
                                        04    AB  D5 0000F        TSTL    4(R11)                ;
                                              03  12 00012 1$:    BNEQ    2$                    ;
                                          00E9  31 00014        BRW     14$                   ;
                                          F8    AD  D4 00017 2$:    CLRL    WORK_DESC             ; 0596
                                  FC    AD      3C  AE 9E 0001A     MOVAB   WORK_BUF, WORK_DESC+4 ; 0597
                                5A        FC    AD  D0 0001F        MOVL    WORK_DESC+4, PTR      ; 0604
                                              56  7C 00023        CLRQ    I                     ; 0612
                                50        04 BC46  D0 00025 3$:    MOVL    @BIT_MASK[I], R0      ;
                                              7F  13 0002A        BEQL    10$                   ;
                                08    AE        50  D0 0002C        MOVL    R0, MASK              ; 0615
                                        04    AE  D4 00030        CLRL    J                     ; 0616
                           6D   08    AE        04  AE E1 00033 4$:    BBC     J, MASK, 9$           ; 0618
                           50           56      05  78 00039        ASHL    #5, I, R0             ; 0621
                                        50    04  AE  C0 0003D        ADDL2   J, R0                 ;
                                        6E    0000GCF40  D0 00041     MOVL    OPER_NAME[R0], NAME   ;
                                              0C  12 00047        BNEQ    5$                    ; 0622
                                34    AE        01  D0 00049        MOVL    #1, NAME_DESC         ; 0625
                                38    AE  0000' CF  9E 0004D        MOVAB   P.AAD, NAME_DESC+4    ; 0626
                                              0D  11 00053        BRB     6$                    ; 0622
```

```
                                34   AE        00   BE   D0 00055 5$:     MOVL    @NAME, NAME_DESC                                        ; 0630
                            50  6E                04   C1 0005A         ADDL3   #4, NAME, R0                                            ; 0631
                                38   AE             60   D0 0005E         MOVL    (R0), NAME_DESC+4                                       ;
                                58        34   AE   3C 00062 6$:     MOVZWL  NAME_DESC, R8                                           ; 0638
                                50        02 A847   9E 00066         MOVAB   2(R8)[LEN], R0                                          ;
                           0000004F   8F             50   D1 0006B         CMPL    R0, #79                                                 ;
                                                     12   15 00072         BLEQ    7$                                                      ;
                            6A  0000'  DF   0000'     CF   28 00074         MOVC3   NEWLINE_DESC, @NEWLINE_DESC+4, (PTR)                    ; 0641
                                50   0000'     CF   3C 0007C         MOVZWL  NEWLINE_DESC, R0                                        ; 0642
                                5A             50   C0 00081         ADDL2   R0, PTR                                                 ;
                                               57   D4 00084         CLRL    LEN                                                     ; 0643
                                               57   D5 00086 7$:     TSTL    LEN                                                     ; 0648
                                               11   13 00088         BEQL    8$                                                      ;
                                59   0000'     CF   3C 0008A         MOVZWL  SEPARATOR_DESC, R9                                      ; 0651
                            6A  0000'  DF             59   28 0008F         MOVC3   R9, @SEPARATOR_DESC+4, (PTR)                            ;
                                5A             59   C0 00095         ADDL2   R9, PTR                                                 ; 0652
                                57             59   C0 00098         ADDL2   R9, LEN                                                 ; 0653
                            6A  38   BE           58   28 0009B 8$:     MOVC3   R8, @NAME_DESC+4, (PTR)                                 ; 0658
                                5A             58   C0 000A0         ADDL2   R8, PTR                                                 ; 0659
                                57             58   C0 000A3         ADDL2   R8, LEN                                                 ; 0660
                            88  04   AE           1F   F3 000A6 9$:     AOBLEQ  #31, J, 4$                                              ; 0616
                        F8  AD             5A        FC   AD A3 000AB 10$:    SUBW3   WORK_DESC+4, PTR, WORK_DESC                     ; 0667
                    FF6E    56             01             01   F1 000B1         ACBL    #1, #1, I, 3$                                   ; 0606
                                                    F8   AD B5 000B7         TSTW    WORK_DESC                                               ; 0672
                                                         22 12 000BA         BNEQ    11$                                                     ;
                                           F8   AD   0A00   8F B0 000BC         MOVW    #2560, WORK_DESC                                        ; 0675
                                           7E             01   7D 000C2         MOVQ    #1, -(SP)                                               ; 0676
                                                    F8   AD 9F 000C5         PUSHAB  WORK_DESC                                               ;
                                                    F8   AD 9F 000C8         PUSHAB  WORK_DESC                                               ;
                                               000580F3   8F DD 000CB         PUSHL   #360691                                                 ;
                                  00000000G    00             05   FB 000D1         CALLS   #5, SYS$GETMSG                                  ;
                                               57             50   D0 000D8         MOVL    R0, STATUS                                      ;
                                               1E             57   E9 000DB         BLBC    STATUS, 13$                                     ;
                                               57             01   D0 000DE 11$:     MOVL    #1, STATUS                                      ; 0685
                                           F8   AD             6B   B1 000E1         CMPW    (R11), WORK_DESC                                ; 0686
                                                               09   1E 000E5         BGEQU   12$                                             ;
                                           F8   AD             6B   B0 000E7         MOVW    (R11), WORK_DESC                                ; 0689
                                               57         0601   8F   3C 000EB         MOVZWL  #1537, STATUS                           ; 0690
                                04   BB         FC   BD     F8   AD   28 000F0 12$:     MOVC3   WORK_DESC, @WORK_DESC+4, @4(R11)        ; 0692
                                           0C   BC     F8   AD   3C 000F7         MOVZWL  WORK_DESC, @OUTLEN                              ; 0693
                                               50             57   D0 000FC 13$:     MOVL    STATUS, R0                                      ; 0694
                                                               04   00FF         RET                                                     ;
                                                               50   D4 00100 14$:     CLRL    R0                                              ; 0696
                                                                    04 00102         RET                                                     ;

; Routine Size:  259 bytes,    Routine Base:  $CODE$ + 0298
```

```
;  702     0697  1 GLOBAL ROUTINE SEND_REPLY (RQCB) =
;  703     0698  1
;  704     0699  1 !++
;  705     0700  1 ! Functional description:
;  706     0701  1 !
;  707     0702  1 !       This routine will send a reply to a specified mailbox.
;  708     0703  1 !
;  709     0704  1 !
;  710     0705  1 ! Input:
;  711     0706  1 !
;  712     0707  1 !       RQCB            : Address of an RQCB data structure
;  713     0708  1 !
;  714     0709  1 !
;  715     0710  1 ! Implicit Input:
;  716     0711  1 !
;  717     0712  1 !       There is an MCB attached to the RQCB.
;  718     0713  1 !
;  719     0714  1 ! Outpu::
;  720     0715  1 !
;  721     0716  1 !       None.
;  722     0717  1 !
;  723     0718  1 ! Implict output:
;  724     0719  1 !
;  725     0720  1 !       A formatted message is sent to the mailbox specified by the RQCB.
;  726     0721  1 !
;  727     0722  1 ! Side effects:
;  728     0723  1 !
;  729     0724  1 !       None.
;  730     0725  1 !
;  731     0726  1 ! Routine value:
;  732     0727  1 !
;  733     0728  1 !       TRUE    : If a reply was sent.
;  734     0729  1 !       FALSE   : If a reply could not be sent.
;  735     0730    !--
;  736     0731  1
;  737     0732  2 BEGIN                                           ! Start of SEND_REPLY
;  738     0733  2
;  739     0734  2 MAP
;  740     0735  2         RQCB            : $ref_bblock;          ! RQCB data structure
;  741     0736  2
;  742     0737  2 EXTERNAL LITERAL
;  743     0738  2         MCB_K_TYPE;                             ! MCB structure type
;  744     0739  2
;  745     0740  2 EXTERNAL
;  746     0741  2         BYPASS_PRIV     : $bblock,              ! Mask with BYPASS priv set
;  747     0742  2         MBX_FAO         : $bblock,              ! FAO control string descriptor
;  748     0743  2         GLOBAL_STATUS : BITVECTOR [32],
;  749     0744  2         LCL_NOD : $ref_bblock,
;  750     0745  2         LCL_CSID;
;  751     0746  2
;  752     0747  2 EXTERNAL ROUTINE
;  753     0748  2         CLUSUTIL_SYSTEMID_EQUAL : JSB_R0R1,
;  754     0749  2         DUMP_LOG_FILE : NOVALUE,
;  755     0750  2         SHARE_FAO_BUFFER,
;  756     0751  2         WRITE_LOG_FILE : NOVALUE;
;  757     0752  2
;  758     0753  2 LOCAL
```

```
;  759          0754  2          MCB             : $ref_bblock,          ! MCB data structure
;  760          0755  2          IO_STATUS       : $bblock [8],          ! I/O status block
;  761          0756  2          MSG_SIZE        : WORD,                 ! Size of reply message
;  762          0757  2          MSG_BUF         : $bblock [OPC$K_MAXREAD],! Message buffer
;  763          0758  2          MBX_CHAN        : WORD,                 ! Channel to reply mailbox
;  764          0759  2          MBX_NAME        : $bblock [MAX_DEV_NAM],! Mailbox device name buffer
;  765          0760  2          MBX_DESC        : $desc_block,          ! Mailbox name descriptor
;  766          0761  2          DEV_CHAR        : $bblock [DIB$K_LENGTH],! Device characteristics buffer
;  767          0762  2          DEV_DESC        : $desc_block,          ! Dev. char. buffer descriptor
;  768          0763  2          STATUS          : LONG;
;  769          0764  2
;  770          0765  2      !
;  771          0766  2      ! If operator is on another node, return success
;  772          0767  2      !
;  773          0768  2      IF .GLOBAL_STATUS [GBLSTS_K_IN_VAXcluster]
;  774          0769  2      THEN
;  775          0770  2          IF NOT CLUSUTIL_SYSTEMID_EQUAL (RQCB [RQCB_T_SYSTEMID], LCL_NOD [NOD_T_NODE_SYSTEMID])
;  776          0771  2          THEN
;  777          0772  2              RETURN TRUE;
;  778          0773  2      !
;  779          0774  2      ! If no reply mailbox is specified, then return success.
;  780          0775  2      !
;  781          0776  2      IF .RQCB [RQCB_W_REPLYMBX] EQL 0
;  782          0777  2      THEN
;  783          0778  2          RETURN (FALSE);
;  784          0779  2      !
;  785          0780  2      ! Check for a valid MCB.  If none, then return false.
;  786          0781  2      !
;  787          0782  2      MCB = .RQCB [RQCB_L_MCB];
;  788          0783  3      IF (.MCB EQL 0) OR (.MCB [MCB_B_TYPE] NEQ MCB_K_TYPE)
;  789          0784  2      THEN
;  790          0785  2          RETURN (FALSE);
;  791          0786  2      !
;  792          0787  2      ! Format the reply mailbox name.
;  793          0788  2      !
;  794          0789  2      MBX_DESC [0,0,32,0] = MAX_DEV_NAM;
;  795          0790  2      MBX_DESC [DSC$A_POINTER] = MBX_NAME;
;  796          0791  3      IF NOT $FAO (MBX_FAO, MBX_DESC, MBX_DESC, .RQCB [RQCB_W_REPLYMBX])
;  797          0792  2      THEN
;  798          0793  2          RETURN (FALSE);
;  799          0794  2      !
;  800          0795  2      ! Enable BYPASS privilege, so that we will be able to access the user's
;  801          0796  2      ! mailbox regardless of whatever strange protection he might have decided
;  802          0797  2      ! to give it.
;  803          0798  2      !
;  804          0799  3      IF NOT (STATUS = $SETPRV (ENBFLG=1, PRVADR=BYPASS_PRIV))
;  805          0800  2      THEN
;  806          0801  2          $signal_stop (.STATUS);
;  807          0802  2      !
;  808          0803  2      ! Declare a loop now that we have enabled the privs.  We will use this
;  809          0804  2      ! as a device to make sure that we disable the enhanced privs on any error.
;  810          0805  2      !
;  811          0806  2      WHILE 1
;  812          0807  2      DO
;  813          0808  3          BEGIN
;  814          0809  3          !
;  815          0810  3          ! Create a buffer descriptor for the device characteristics,
```

```
;  816          0811  3      ! and call $GETDEV to get the device characteristics.
;  817          0812  3      !
;  818          0813  3      DEV_DESC [0,0,32,0] = DIB$K_LENGTH;
;  819          0814  3      DEV_DESC [DSC$A_POINTER] = DEV_CHAR;
;  820          0815  4      IF NOT (STATUS = $GETDEV (DEVNAM=MBX_DESC, PRIBUF=DEV_DESC))
;  821          0816  3      THEN
;  822          0817  3          EXITLOOP;
;  823          0818  3      !
;  824          0819  3      ! Build the reply message.
;  825          0820  3      !
;  826          0821  3      MSG_BUF [OPC$B_MS_TYPE] = MSG$_OPREPLY;                    ! Set message type code
;  827          0822  3      MSG_BUF [OPC$W_MS_STATUS] = .MCB [MCB_L_MSGID];            ! Set reply status
;  828          0823  3      MSG_BUF     [OPC$[_MS_RPLYID] = .RQCB [RQCB_L_RQSTID];     ! Set user request id
;  829          0824  3      MSG_SIZE = $BYTEOFFSET (OPC$L_MS_TEXT);
;  830          0825  3      IF (.MCB [MCB_L_TEXTLEN]-1) GTR 0
;  831          0826  3      THEN
;  832          0827  4          BEGIN
;  833          0828  4          CH$MOVE (.MCB [MCB_L_TEXTLEN]-1,                       ! Copy reply text to buffer
;  834          0829  4                   .MCB [MCB_L_TEXTPTR]+1,                    ! but do not copy the
;  835          0830  4                   MSG_BUF [OPC$L_MS_TEXT]                    ! 'bell' character.
;  836          0831  4                  );
;  837          0832  4          MSG_SIZE = .MSG_SIZE + (.MCB [MCB_L_TEXTLEN]-1);
;  838          0833  3          END;
;  839          0834  3      !
;  840          0835  3      ! Check the mailbox size against the reply message size.
;  841          0836  3      ! If the mailbox is too small to accomodate the entire message,
;  842          0837  3      ! truncate the message to the mailbox size.
;  843          0838  3      !
;  844          0839  4      IF (.MSG_SIZE GTR .DEV_CHAR [DIB$W_DEVBUFSIZ])
;  845          0840  3      THEN
;  846          0841  3          MSG_SIZE = .DEV_CHAR [DIB$W_DEVBUFSIZ];
;  847          0842  3      !
;  848          0843  3      ! Assign a channel to the mailbox.
;  849          0844  3      !
;  850          0845  4      IF NOT (STATUS = $ASSIGN (DEVNAM=MBX_DESC, CHAN=MBX_CHAN))
;  851          0846  3      THEN
;  852          0847  3          EXITLOOP;
;  853          0848  3      !
;  854          0849  3      ! If debugging, check the logical and maybe write the message into the log file
;  855          0850  3      !
;  856        L 0851  3      %IF %VARIANT NEQ 0
;  857        U 0852  3      %THEN
;  858        U 0853  3          BEGIN
;  859        U 0854  3          LOCAL
;  860        U 0855  3              DBG_BUF      : $BVECTOR [16],
;  861        U 0856  3              DBG_DSC      : VECTOR [2, LONG];
;  862        U 0857  3          DBG_DSC [0] = 16;
;  863        U 0858  3          DBG_DSC [1] = DBG_BUF;
;  864        U 0859  3          IF ($TRNLOG (LOGNAM=%ASCID 'OPC$DUMP_MAILBOX', RSLBUF=DBG_DSC) EQL SS$_NORMAL)
;  865        U 0860  3          THEN
;  866        U 0861  3              BEGIN
;  867        U 0862  3              DBG_DSC [0] = .MSG_SIZE;
;  868        U 0863  3              DBG_DSC [1] = MSG_BUF;
;  869        U 0864  3              DUMP_LOG_FILE (DBG_DSC, %ASCID '-  Sending reply message to user mailbox');
;  870        U 0865  3              END;
;  871        U 0866  3          END;
;  872          0867  3      %FI
```

```
;  873          0868  3     !
;  874          0869  3     ! Send the message to the mailbox.
;  875          0870  3     !
;  876        P 0871  4     IF (STATUS = $QIOW (FUNC = (IO$_WRITEVBLK OR IO$M_NOW),
;  877        P 0872  4                         CHAN = .MBX_CHAN,
;  878        P 0873  4                         IOSB = IO_STATUS,
;  879        P 0874  4                         P1   = MSG_BUF,
;  880        P 0875  4                         P2   = .MSG_SIZE
;  881          0876  4                        ))
;  882          0877  3     THEN
;  883          0878  3         STATUS = .IO_STATUS [0,0,16,0];         ! Get I/O status
;  884          0879  3     !
;  885          0880  3     ! Display the status if debugging
;  886          0881  3     !
;  887        L 0882  3     %IF %VARIANT NEQ 0
;  888        U 0883  3     %THEN
;  889        U 0884  3         BEGIN
;  890        U 0885  3         LOCAL
;  891        U 0886  3             DBG_BUF      : $BVECTOR [16],
;  892        U 0887  3             DBG_DSC      : VECTOR [2, LONG];
;  893        U 0888  3         DBG_DSC [0] = 16;
;  894        U 0889  3         DBG_DSC [1] = DBG_BUF;
;  895        U 0890  3         IF (%STRNLOG (LOGNAM=%ASCID 'OPC$DUMP_MAILBOX', RSLBUF=DBG_DSC) EQL SS$_NORMAL)
;  896        U 0891  3         THEN
;  897        U 0892  3             WRITE_LOG_FILE (SHARE_FAO_BUFFER (%ASCID 'status from qio to user mailbox !XL', .status));
;  898        U 0893  3         END;
;  899          0894  3     %FI
;  900          0895  3     !
;  901          0896  3     ! Now, exit the loop anyway
;  902          0897  3     !
;  903          0898  3     EXITLOOP;
;  904          0899  2     END;
;  905          0900  2 !
;  906          0901  2 ! Deassign the channel to the mailbox.
;  907          0902  2 !
;  908          0903  2 $DASSGN (CHAN = .MBX_CHAN);
;  909          0904  2 !
;  910          0905  2 ! Remove bypass privilege
;  911          0906  2 !
;  912          0907  2 $SETPRV (ENBFLG=0, PRVADR=BYPASS_PRIV);
;  913          0908  2 !
;  914          0909  2 ! Return the I/O status as the return value.
;  915          0910  2 !
;  916          0911  2 RETURN (.STATUS);
;  917          0912  2
;  918          0913  1 END;                                    ! End of SEND_REPLY


                                .EXTRN  BYPASS_PRIV, MBX_FAO
                                .EXTRN  GLOBAL_STATUS, LCL_NOD
                                .EXTRN  LCL_CSID, CLUSUTIL_SYSTEMID_EQUAL
                                .EXTRN  DUMP_LOG_FILE, SHARE_FAO_BUFFER
                                .EXTRN  WRITE_LOG_FILE, SYS$FAO
                                .EXTRN  SYS$SETPRV, SYS$GETDEV
                                .EXTRN  SYS$ASSIGN, SYS$QIOW
                                .EXTRN  SYS$DASSGN
```

```
                                                 OFFC 00000          .ENTRY  SEND_REPLY, Save R2,R3,R4,R5,R6,R7,R8,R9,-      ; 0697
                                                                             R10,R11
                                              5B 00000000G   00  9E 00002    MOVAB   SYS$SETPRV, R11                        ;
                                              5E      F530   CE  9E 00009    MOVAB   -2768(SP), SP                          ;
                                              19      0000G  CF  E9 0000E    BLBC    GLOBAL_STATUS+1, 1$                    ; 0768
                                        51          0000G  CF  8F C1 00013    ADDL3   #80, LCL_NOD, R1                       ; 0770
                                        50            04   AC  1C C1 0001D    ADDL3   #28, RQCB, R0                          ;
                                                           0000G 30 00022    BSBW    CLUSUTIL_SYSTEMID_EQUAL                ;
                                              04              50  E8 00025    BLBS    R0, 1$                                 ;
                                              50              01  D0 00028    MOVL    #1, R0                                 ; 0772
                                                              04  0002B    RET                                            ;
                                              57          04  AC  D0 0002C 1$: MOVL    RQCB, R7                               ; 0776
                                                          2E  A7  B5 00030    TSTW    46(R7)                                 ;
                                                              12  13 00033    BEQL    2$                                     ;
                                              56          6C  A7  D0 00035    MOVL    108(R7), MCB                           ; 0782
                                                              0C  13 00039    BEQL    2$                                     ; 0783
00000000G  8F       0A  A6                    08                00  ED 0003B    CMPZV   #0, #8, 10(MCB), #MCB_K_TYPE           ;
                                                                  03  13 00045    BEQL    3$                                     ;
                                                                0OF5  31 00047 2$: BRW     8$                                     ;
                                                    0080  CE      40  8F 9A 0004A 3$: MOVZBL  #64, MBX_DESC                  ; 0789
                                                    0084  CE  0088  CE  9E 00050    MOVAB   MBX_NAME, MBX_DESC+4           ; 0790
                                                          7E    2E  A7  3C 00057    MOVZWL  46(R7), -(SP)                  ; 0791
                                                              0084  CE  9F 0005B    PUSHAB  MBX_DESC                       ;
                                                              0088  CE  9F 0005F    PUSHAB  MBX_DESC                       ;
                                                              0000G CF  9F 00063    PUSHAB  MBX_FAO                        ;
                                                   00000000G  00        04  FB 00067    CALLS   #4, SYS$FAO                    ;
                                                              D6        50  E9 0006E    BLBC    R0, 2$                         ;
                                                                        7E  7C 00071    CLRQ    -(SP)                          ; 0799
                                                              0000G CF  9F 00073    PUSHAB  BYPASS_PRIV                    ;
                                                                        01  DD 00077    PUSHL   #1                             ;
                                                                    6B  04  FB 00079    CALLS   #4, SYS$SETPRV                 ;
                                                                    59  50  D0 0007C    MOVL    R0, STATUS                     ;
                                                                    0A  59  E8 0007F    BLBS    STATUS, 4$                     ;
                                                                        59  DD 00082    PUSHL   STATUS                         ; 0801
                                                   00000000G  00        01  FB 00084    CALLS   #1, LIB$STOP                   ;
                                                                        04  0008B    RET                                    ;
                                                        04    AE    74  8F 9A 0008C 4$: MOVZBL  #116, DEV_DESC                 ; 0813
                                                        08    AE    0C  AE 9E 00091    MOVAB   DEV_CHAR, DEV_DESC+4           ; 0814
                                                                        7E 7C 00096    CLRQ    -(SP)                          ; 0815
                                                              0C    AE  9F 00098    PUSHAB  DEV_DESC                       ;
                                                                        7E  D4 0009B    CLRL    -(SP)                          ;
                                                              0090  CE  9F 0009D    PUSHAB  MBX_DESC                       ;
                                                   00000000G  00        05  FB 000A1    CALLS   #5, SYS$GETDEV                 ;
                                                                    59  50  D0 000A8    MOVL    R0, STATUS                     ;
                                                                    78  59  E9 000AB    BLBC    STATUS, 7$                     ;
                                                        00C8  CE        09  90 000AE    MOVB    #9, MSG_BUF                    ; 0821
                                                        00CA  CE    2C  A6  B0 000B3    MOVW    44(MCB), MSG_BUF+2             ; 0822
                                                        00CC  CE    64  A7  D0 000B9    MOVL    100(R7), MSG_BUF+4             ; 0823
                                                                    5A  08  B0 000BF    MOVW    #8, MSG_SIZE                   ; 0824
                                                                    01  30  A6  D1 000C2    CMPL    48(MCB), #1            ; 0825
                                                                            13  15 000C6    BLEQ    5$                             ;
                                                  58        30    A6      01  C3 000C8    SUBL3   #1, 48(MCB), R8        ; 0828
                                                                    50  34  A6  D0 000CD    MOVL    52(MCB), R0            ; 0829
                                              00D0  CE      01    A0      58  28 000D1    MOVC3   R8, 1(R0), MSG_BUF+8   ; 0830
                                                                    5A      58  A0 000D8    ADDW2   R8, MSG_SIZE           ; 0832
                                                        12    AE        5A  B1 000DB 5$: CMPW    MSG_SIZE, DEV_CHAR+6           ; 0839
```

```
                                04  1B 000DF          BLEQU   6$                                          ;
                             5A    12   AE  B0 000E1          MOVW    DEV_CHAR+6, MSG_SIZE                ; 0841
                                7E  7C 000E5 6$:       CLRQ    -(SP)                                       ; 0845
                                 08   AE  9F 000E7          PUSHAB  MBX_CHAN                            ;
                              008C    CE  9F 000EA          PUSHAB  MBX_DESC                            ;
                  00000000G   00        04  FB 000EE          CALLS   #4, SYS$ASSIGN                      ;
                              59        50  D0 000F5          MOVL    R0, STATUS                          ;
                              2B        59  E9 000F8          BLBC    STATUS, 7$                          ;
                                    7E  7C 000FB          CLRQ    -(SP)                                   ; 0876
                                    7E  7C 000FD          CLRQ    -(SP)                                   ;
                              7E        5A  3C 000FF          MOVZWL  MSG_SIZE, -(SP)                     ;
                              00DC      CE  9F 00102          PUSHAB  MSG_BUF                             ;
                                    7E  7C 00106          CLRQ    -(SP)                                   ;
                                 F8   AD  9F 00108          PUSHAB  IO_STATUS                           ;
                              7E    70   8F  9A 0010B          MOVZBL  #112, -(SP)                         ;
                              7E    28   AE  3C 0010F          MOVZWL  MBX_CHAN, -(SP)                     ;
                                    7E  D4 00113          CLRL    -(SP)                                   ;
                  00000000G   00        0C  FB 00115          CALLS   #12, SYS$QIOW                       ;
                              59        50  D0 0011C          MOVL    R0, STATUS                          ;
                              04        59  E9 0011F          BLBC    STATUS, 7$                          ;
                              59     F8   AD  3C 00122          MOVZWL  IO_STATUS, STATUS                   ; 0878
                              7E        6E  3C 00126 7$:       MOVZWL  MBX_CHAN, -(SP)                     ; 0903
                  00000000G   00        01  FB 00129          CALLS   #1, SYS$DASSGN                      ;
                                    7E  7C 00130          CLRQ    -(SP)                                   ; 0907
                              0000G     CF  9F 00132          PUSHAB  BYPASS_PRIV                         ;
                                    7E  D4 00136          CLRL    -(SP)                                   ;
                              6B        04  FB 00138          CALLS   #4, SYS$SETPRV                      ;
                              50        59  D0 0013B          MOVL    STATUS, R0                          ; 0911
                                        04 0013E          RET                                         ;
                                    50  D4 0013F 8$:       CLRL    R0                                  ; 0913
                                        04 00141          RET                                         ;

; Routine Size:  322 bytes,    Routine Base:  $CODE$ + 039B
```

```
;  920     0914  1 GLOBAL ROUTINE TRIM_LENGTH (IN_LEN, IN_PTR : REF VECTOR [,BYTE]) =
;  921     0915  1
;  922     0916  1 !++
;  923     0917  1 ! Functional description:
;  924     0918  1 !
;  925     0919  1 !       Return the length of a string without trailing blanks or nulls
;  926     0920  1 !
;  927     0921  1 ! Input:
;  928     0922  1 !
;  929     0923  1 !       IN_LEN  : Length of the string
;  930     0924  1 !       IN_PTR  : Address of the string
;  931     0925  1 !
;  932     0926  1 ! Implicit Input:
;  933     0927  1 !
;  934     0928  1 !       None.
;  935     0929  1 !
;  936     0930  1 ! Output:
;  937     0931  1 !
;  938     0932  1 !       None.
;  939     0933  1 !
;  940     0934  1 ! Implict output:
;  941     0935  1 !
;  942     0936  1 !       None.
;  943     0937  1 !
;  944     0938  1 ! Side effects:
;  945     0939  1 !
;  946     0940  1 !       None.
;  947     0941  1 !
;  948     0942  1 ! Routine value:
;  949     0943  1 !
;  950     0944  1 !       Return the length of a string without trailing blanks.
;  951     0945  1 !--
;  952     0946  1
;  953     0947  2 BEGIN                                           ! Start of TRIM_LENGTH
;  954     0948  2 LOCAL
;  955     0949  2     IDX;
;  956     0950  2
;  957     0951  2 IDX = .IN_LEN - 1;                              ! Vectors indexed 0:n-1
;  958     0952  2 WHILE .IDX GEQ 0
;  959     0953  2 DO
;  960     0954  3     BEGIN
;  961     0955  3     IF .IN_PTR [.IDX] NEQ %C' '
;  962     0956  3      AND
;  963     0957  3         .IN_PTR [.IDX] NEQ 0
;  964     0958  3     THEN
;  965     0959  3         EXITLOOP;
;  966     0960  3     IDX = .IDX - 1;
;  967     0961  2     END;
;  968     0962  2
;  969     0963  2 RETURN .IDX+1;
;  970     0964  1 END;                                            ! End of TRIM_LENGTH


                        0000 00000          .ENTRY  TRIM_LENGTH, Save nothing                   ; 0914
```

```
                50      04  AC          01  C3 00002          SUBL3   #1, IN_LEN, IDX                 ; 0951
                                        11  19 00007 1$:      BLSS    3$                              ; 0952
                        20      08 BC40 91  00009              CMPB    @IN_PTR[IDX], #32               ; 0955
                                    06  13  0000E              BEQL    2$                              ;
                                08 BC40 95  00010              TSTB    @IN_PTR[IDX]                    ; 0957
                                    04  12  00014              BNEQ    3$                              ;
                                    50  D7  00016 2$:          DECL    IDX                             ; 0960
                                    ED  11  00018              BRB     1$                              ; 0952
                                    50  D6  0001A 3$:          INCL    R0                              ; 0963
                                        04  0001C              RET                                     ; 0964

; Routine Size:  29 bytes,    Routine Base:  $CODE$ + 04DD
```

```
;  972          0965  1 END
;  973          0966  0 ELUDOM                    ! End of OPCOMUTIL
```

```
;                                  PSECT SUMMARY
;
;      Name                    Bytes                          Attributes
;
;  $CODE$                       1274  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                         424  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                         17  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        37         0        1000        00:01.8
;  _$255$DUA28:[OPCOM.OBJ]OPCOMLIB.L32;1    633        41         6          43        00:00.8
```

```
;                                   COMMAND QUALIFIERS

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

; Size:          1274 code + 441 data bytes
; Run Time:         00:27.2
; Elapsed Time:     01:17.7
; Lines/CPU Min:     2134
; Lexemes/CPU-Min: 21354
; Memory Used:  162 pages
; Compilation Complete
```

LOGFILE
LIS

OPCOMDEF
LIS

OPCOMDATA
LIS

OPCOMINI
LIS

OPCOMUTIL
LIS

OPCOMLIB
LIS

OPCDEFTMP
LIS

OPCOMMAIN
LIS

OPCOMOLD
LIS

OPCOMRPLY
LIS

OPCCRASH
LIS

OPERUTIL
LIS

OPCOMROST
LIS