L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
_$
Ps
--
YZ
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
```

FILEID**SYSSNDMSG

```
  SSSSSSSS  YY      YY    SSSSSSSS    SSSSSSSS  NN      NN  DDDDDDDD    MM      MM    SSSSSSSS    GGGGGGGG
  SSSSSSSS  YY      YY    SSSSSSSS    SSSSSSSS  NN      NN  DDDDDDDD    MM      MM    SSSSSSSS    GGGGGGGG
SS          YY      YY  SS          SS          NN      NN  DD      DD  MMMM  MMMM  SS          GG
SS          YY      YY  SS          SS          NN      NN  DD      DD  MMMM  MMMM  SS          GG
SS            YY  YY    SS          SS          NNNN    NN  DD      DD  MM  MM  MM  SS          GG
SS            YY  YY    SS          SS          NNNN    NN  DD      DD  MM  MM  MM  SS          GG
  SSSSSS        YY        SSSSSS      SSSSSS    NN  NN  NN  DD      DD  MM      MM    SSSSSS    GG
  SSSSSS        YY        SSSSSS      SSSSSS    NN  NN  NN  DD      DD  MM      MM    SSSSSS    GG
        SS      YY              SS          SS  NN    NNNN  DD      DD  MM      MM          SS  GG  GGGGGG
        SS      YY              SS          SS  NN    NNNN  DD      DD  MM      MM          SS  GG  GGGGGG
        SS      YY              SS          SS  NN      NN  DD      DD  MM      MM          SS  GG      GG       ....
        SS      YY              SS          SS  NN      NN  DD      DD  MM      MM          SS  GG      GG       ....
SSSSSSSS        YY      SSSSSSSS    SSSSSSSS    NN      NN  DDDDDDDD    MM      MM  SSSSSSSS      GGGGGG         ....
SSSSSSSS        YY      SSSSSSSS    SSSSSSSS    NN      NN  DDDDDDDD    MM      MM  SSSSSSSS      GGGGGG         ....

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

Table of contents

```
0000          .TITLE  SYSSNDMSG - SEND MESSAGE SYSTEM SERVICES
0000          .IDENT  'V04-000'
0000
0000    ;
0000    ;*****************************************************************************
0000    ;*                                                                           *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    ;*  ALL RIGHTS RESERVED.                                                     *
0000    ;*                                                                           *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    ;*  TRANSFERRED.                                                             *
0000    ;*                                                                           *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    ;*  CORPORATION.                                                             *
0000    ;*                                                                           *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    ;*                                                                           *
0000    ;*                                                                           *
0000    ;*****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY:
0000    ;
0000    ;       STARLET SYSTEM SERVICE
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       COMMON MODULE FOR SEND TO OPERATOR AND SYMBIONT MANAGER.
0000    ;
0000    ; AUTHOR: R.HEINEN, CREATION DATE: 11-JUL-77
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-010 LMP0185         L. Mark Pilant,         20-Jan-1984  10:57
0000    ;               Track interface changes to EXE$CHKxxxACCES.
0000    ;
0000    ;       V03-009 ACG0354         Andrew C. Goldstein,    19-Sep-1983  15:19
0000    ;               Use alternate access mask to validate delete access
0000    ;
0000    ;       V03-008 CWH3008         CW Hobbs                18-Sep-1983
0000    ;               KLUDGE - change ATR$x_ACCESS_MASK to DUMMY_0 to get the
0000    ;               build working.  Symbol was deleted.
0000    ;
0000    ;       V03-007 MLJ0118         Martin L. Jack, 22-Aug-1983  9:39
0000    ;               Guard against overlong resultant filename.
0000    ;
0000    ;       V03-006 MLJ0115         Martin L. Jack, 29-Jul-1983  14:30
0000    ;               Update for new file protection handling.
0000    ;
```

```
                    0000    ;       V03-005 CWH1002         Alan D. Eldridge        31-May-1983
                    0000    ;               Change BSBW to JSB in call to ERL$ALLOCMB and ERL$RELEASEMB.
                    0000    ;
                    0000    ;       V03-004 CWH1002         CW Hobbs                24-Feb-1983
                    0000    ;               Use extended pid and owner in the message.
                    0000    ;
                    0000    ;       V03-003 RNG0003         Rod N. Gamache          2-Feb-1983
                    0000    ;               Use longword displacements where needed.
                    0000    ;
                    0000    ;       V03-002 KDM0002         Kathleen D. Morse       28-Jun-1982
                    0000    ;               Added $PRDEF.
                    0000    ;
                    0000    ;
                    0000    ;--
                00000000            .PSECT  Y$EXEPAGED
                    0000    ;
                    0000    ; EXTERNAL SYMBOLS
                    0000    ;
                    0000            $ACMDEF                         ; DEFINE ACCOUNTING MESSAGE OFFSETS
                    0000            $ARMDEF                         ; Define access rights mask
                    0000            $ATRDEF                         ; Define ACP attribute codes
                    0000            $CCBDEF                         ; DEFINE CHANNEL CONTROL BLOCK
                    0000            $DEVDEF                         ; DEFINE DEVICE CHARACTERISTICS
                    0000            $EMBDEF SS                      ; DEFINE ERROR MESSAGE BUFFER OFFSETS
                    0000            $FATDEF                         ; Define RMS file attribute offsets
                    0000            $FIBDEF                         ; Define file information block offsets
                    0000            $IODEF                          ; Define I/O function codes
                    0000            $MSGDEF                         ; DEFINE MESSAGE TYPES
                    0000            $OPCDEF                         ; DEFINE OPERATOR MESSAGES
                    0000            $OPCMSG                         ; OPERATOR COMMUNICATIONS MESSAGES
                    0000            $PCBDEF                         ; DEFINE PCB
                    0000            $PHDDEF                         ; DEFINE PROCESS HEADER OFFSETS
                    0000            $PRDEF                          ; DEFINE PROCESSOR REGISTER NUMBERS
                    0000            $PRVDEF                         ; DEFINE PRIVILEGE MASK
                    0000            $SMRDEF                         ; Define $SNDSMB function codes
                    0000            $SSDEF                          ; DEFINE SYSTEM STATUS RETURN CODES
                    0000            $UCBDEF                         ; DEFINE UCB
                    0000            $IPLDEF                         ; DEFINE IPL CONSTANTS
                    0000            $RSNDEF                         ; DEFINE RESOURCE NUMBERS
                    0000    ;
                    0000    ; LOCAL SYMBOLS
                    0000    ;
00000004            0000    MSG=4
00000008            0000    MBX=8
                    0000    ;
                    0000    ; The messages sent by $SNDACC, $SNDERR, and $SNDOPR consist of
                    0000    ; a common header followed by the user specified message.  The
                    0000    ; common header has the following format:
                    0000    ;
                    0000    ;       .WORD   <message type>
                    0000    ;       .WORD   <reply mailbox channel #>
                    0000    ;       .QUAD   <sender's privilege mask>
                    0000    ;       .LONG   <sender's UIC>
                    0000    ;       .BLKB   <sender's USERNAME.  12 bytes, blank filled>
                    0000    ;       .BLKB   <sender's ACCOUNT.  8 bytes, blank filled>
                    0000    ;       .BYTE   <sender's base priority>
                    0000    ;       .BYTE   <unused>
```

```
                                   0000    115 ;
                         00000026  0000    116 COMMON_HDR=38                              ; Common header size
                                   0000    117
                                   0000    118
                                   0000    119 ;
                                   0000    120 ; Option size table to allow scanning of options list
                                   0000    121 ;
                                   0000    122 OPTION_SIZE:
04 01 02 00 00 00 00 00 00 00 00 00 0000   123         .BYTE   0,0,0,0,0,0,0,0,0,2,1,4,-1,0,0,0
                        00 00 00 FF 000C
00'00'00'00'00'00'00'00'00'00'00'00' 0010  124         .BYTE   0[16]
                        00'00'00'00' 001C
02 00 04 00 FF FE 00 01 01 01 00 08 0020   125         .BYTE   8,0,1,1,1,0,-2,-1,0,4,0,2,0,2,0,16
                        10 00 02 00 002C
00 00 00 02 00 00 00 00 00 00 FF FF 0030   126         .BYTE   -1,-1,0,0,0,0,0,0,2,0,0,0,0,0,0,0
                        00 00 00 00 003C
```

```
                         0040        .SBTTL  EXE$SNDACC - SEND MESSAGE TO ACCOUNT MANAGER
                         0040  ;++
                         0040  ; EXE$SNDACC - SEND MESSAGE TO ACCOUNT MANAGER
                         0040  ;
                         0040  ; FUNCTIONAL DESCRIPTION:
                         0040  ;
                         0040  ; THIS ROUTINE PROVIDES THE SEND TO ACCOUNT MANAGER MAILBOX SYSTEM SERVICE.
                         0040  ; THE ACTION IS TO BUILD A MESSAGE CONSISTING OF A COMMON HEADER
                         0040  ; AND THE USER SPECIFIED TEXT AND THEN SEND IT TO THE JOB CONTROLLER MAILBOX.
                         0040  ; THE SPECIFIED MESSAGE IS ADDRESSED CHECKED AND THE REQUEST REPLY MAILBOX
                         0040  ; IS CHECKED FOR BEING A MAILBOX AND ACCESSIBLE TO THE PROCESS.
                         0040  ;
                         0040  ; INPUTS:
                         0040  ;
                         0040  ;       MSG(AP) = ADDRESS OF THE QUADWORD DESC FOR THE MESSAGE TEXT.
                         0040  ;       MBX(AP) = CHANNEL NUMBER OF THE MAILBOX FOR THE REPLY.
                         0040  ;
                         0040  ; OUTPUTS:
                         0040  ;
                         0040  ;       R0 = STATUS OF THE OPERATION
                         0040  ;
                         0040  ; STATUS CODES RETURNED:
                         0040  ;
                         0040  ;       SS$_NORMAL - SUCCESSFUL OPERATION
                         0040  ;       SS$_INSFMEM - INSUFFICIENT MEMORY FOR THE REQUEST
                         0040  ;       SS$_ACCVIO - ACCESS VIOLATION ON BUFFER
                         0040  ;       SS$_NOPRIV - PROCESS DOES NOT HAVE READ ACCESS TO SPECIFIED MAILBOX
                         0040  ;       SS$_IVCHAN - SPECIFIED CHANNEL INVALID
                         0040  ;       SS$_DEVNOTMBX - SPECIFIED CHANNEL IS NOT TO MAILBOX
                         0040  ;       SS$_BADPARAM - MESSAGE SIZE ERROR
                         0040  ;       SS$_MBTOOSML - MESSAGE EXCEEDS MAILBOX SIZE
                         0040  ;       SS$_DEVOFFLIN - DEVICE OFFLINE
                         0040  ;--
                         0040  EXE$SNDACC::                          ; SEND TO ACCOUNTING MANAGER
                   0FFC  0040        .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
         5B    0A    9A  0042        MOVZBL  #MSG$_SNDACC,R11        ; SET MESSAGE TYPE CODE
55  00000000'EF      9E  0045        MOVAB   SYS$GL_JOBCTLMB,R5      ; ADDRESS TARGET MAILBOX
   57    00C8 8F     3C  004C        MOVZWL  #200,R7                 ; SET MAXIMUM MESSAGE SIZE
               2A    11  0051        BRB     BUILDMSG                ; CONTINUE IN COMMON
```

```
                         0053               .SBTTL  EXE$SNDSMB - SEND MESSAGE TO SYMBIONT MANAGER
                         0053      ;++
                         0053      ; EXE$SNDSMB - SEND MESSAGE TO SYMBIONT MANAGER
                         0053      ;
                         0053      ; FUNCTIONAL DESCRIPTION:
                         0053      ;
                         0053      ; THIS ROUTINE PROVIDES THE SEND TO SYMBIONT MANAGER MAILBOX SYSTEM SERVICE.
                         0053      ; THE ACTION IS TO BUILD A MESSAGE CONSISTING OF A COMMON HEADER
                         0053      ; AND THE USER SPECIFIED TEXT AND THEN SEND IT TO THE JOB CONTROLLER MAILBOX.
                         0053      ; THE SPECIFIED MESSAGE IS ADDRESSED CHECKED AND THE REQUEST REPLY MAILBOX
                         0053      ; IS CHECKED FOR BEING A MAILBOX AND ACCESSIBLE TO THE PROCESS.
                         0053      ;
                         0053      ; INPUTS:
                         0053      ;
                         0053      ;       MSG(AP) = ADDRESS OF THE QUADWORD DESC FOR THE MESSAGE TEXT.
                         0053      ;       MBX(AP) = CHANNEL NUMBER OF THE MAILBOX FOR THE REPLY.
                         0053      ;
                         0053      ; OUTPUTS:
                         0053      ;
                         0053      ;       R0 = STATUS OF THE OPERATION
                         0053      ;
                         0053      ; STATUS CODES RETURNED:
                         0053      ;
                         0053      ;       SS$_NORMAL - SUCCESSFUL OPERATION
                         0053      ;       SS$_INSFMEM - INSUFFICIENT MEMORY FOR THE REQUEST
                         0053      ;       SS$_ACCVIO - ACCESS VIOLATION ON BUFFER
                         0053      ;       SS$_NOPRIV - PROCESS DOES NOT HAVE READ ACCESS TO SPECIFIED MAILBOX
                         0053      ;       SS$_IVCHAN - SPECIFIED CHANNEL INVALID
                         0053      ;       SS$_DEVNOTMBX - SPECIFIED CHANNEL IS NOT TO MAILBOX
                         0053      ;       SS$_BADPARAM - MESSAGE SIZE ERROR
                         0053      ;       SS$_MBTOOSML - MESSAGE EXCEEDS MAILBOX SIZE
                         0053      ;       SS$_DEVOFFLIN - DEVICE OFFLINE
                         0053      ;--
                         0053      EXE$SNDSMB::                              ; SEND TO SYMBIONT MANAGER
                   0FFC  0053             .WORD    ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
          5B   04    9A  0055             MOVZBL   #MSG$_SNDSMB,R11          ; SET MESSAGE TYPE CODE
 55  00000000'EF     9E  0058             MOVAB    SYS$GL_JOBCTLMB,R5        ; SET ADDRESS OF MAILBOX
    57   03E8 8F     3C  005F             MOVZWL   #1000,R7                  ; SET MAXIMUM MESSAGE SIZE
               17    11  0064             BRB      BUILDMSG                  ; CONTINUE IN COMMON
```

```
                        0066          .SBTTL  EXE$SNDOPR - SEND MESSAGE TO OPERATOR MAILBOX
                        0066   ;++
                        0066   ; EXE$SNDOPR - SEND MESSAGE TO OPERATOR MAILBOX
                        0066   ;
                        0066   ; FUNCTIONAL DESCRIPTION:
                        0066   ;
                        0066   ; THIS ROUTINE PROVIDES THE SEND TO OPERATOR MAILBOX SYSTEM SERVICE.
                        0066   ; THE ACTION IS TO BUILD A MESSAGE CONSISTING OF A COMMON HEADER
                        0066   ; AND THE USER SPECIFIED TEXT AND THEN SEND IT TO THE OPERATOR MAILBOX.
                        0066   ; THE SPECIFIED MESSAGE IS ADDRESSED CHECKED AND THE REQUEST REPLY MAILBOX
                        0066   ; IS CHECKED FOR BEING A MAILBOX AND ACCESSIBLE TO THE PROCESS.
                        0066   ;
                        0066   ; INPUTS:
                        0066   ;
                        0066   ;       MSG(AP) = ADDRESS OF THE QUADWORD DESC FOR THE MESSAGE TEXT.
                        0066   ;       MBX(AP) = CHANNEL NUMBER OF THE MAILBOX FOR THE REPLY.
                        0066   ;
                        0066   ; OUTPUTS:
                        0066   ;
                        0066   ;       R0 = STATUS OF THE OPERATION
                        0066   ;
                        0066   ; STATUS CODES RETURNED:
                        0066   ;
                        0066   ;       SS$_NORMAL - SUCCESSFUL OPERATION
                        0066   ;       SS$_IVCHAN - SPECIFIED CHANNEL INVALID
                        0066   ;       SS$_MBTOOSML - MESSAGE EXCEEDS MAILBOX SIZE
                        0066   ;       OPC$_NOPERATOR - NO OPERATOR COVERAGE
                        0066   ;       SS$_INSFMEM - INSUFFICIENT MEMORY FOR THE REQUEST
                        0066   ;       SS$_ACCVIO - ACCESS VIOLATION ON BUFFER
                        0066   ;       SS$_NOPRIV - PROCESS DOES NOT HAVE READ ACCESS TO SPECIFIED MAILBOX
                        0066   ;       SS$_DEVNOTMBX - SPECIFIED CHANNEL IS NOT TO MAILBOX
                        0066   ;       SS$_BADPARAM - MESSAGE SIZE ERROR
                        0066   ;--
                        0066   EXE$SNDOPR::                             ; SEND TO OPERATOR
                  0FFC  0066          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
       5B   08     9A   0068          MOVZBL  #MSG$_OPRQST,R11         ; SET MESSAGE TYPE CODE
55  00000000'EF    9E   006B          MOVAB   SYS$GL_OPRMBX,R5         ; SET ADDRESS OF MAILBOX
    57   03DA 8F   3C   0072          MOVZWL  #<1024-COMMON_HDR>,R7    ; SET MAXIMUM USER MESSAGE SIZE
            04     11   0077          BRB     BUILDMSG                 ; CONTINUE IN COMMON
```

```
                                0079          .SBTTL  BUILD MESSAGE SUBROUTINE
                                0079  ;++
                                0079  ; BUILDMSG - BUILD MESSAGE ROUTINE FOR EXE$SNDOPR/EXE$SNDSMB
                                0079  ;
                                0079  ; FUNCTIONAL DESCRIPTION:
                                0079  ;
                                0079  ; THIS ROUTINE BUILDS THE REQUESTED MESSAGE ON THE EXEC STACK
                                0079  ; AND ENTERS A KERNEL MODE ROUTINE TO PERFORM THE MAILBOX VALIDATION
                                0079  ; AND SEND THE MESSAGE.
                                0079  ;
                                0079  ; INPUTS:
                                0079  ;
                                0079  ;       MSG(AP) = ADDRESS OF THE MESSAGE DESCRIPTER
                                0079  ;       MBX(AP) = CHANNEL NUMBER OF THE REPLY MAILBOX IF ANY
                                0079  ;       R5 = MAILBOX UCB ADDRESS
                                0079  ;       R7 = MAXIMUM MESSAGE SIZE
                                0079  ;       R11 = MESSAGE TYPE
                                0079  ;
                                0079  ; OUTPUTS:
                                0079  ;
                                0079  ;       R0 = STATUS OF THE OPERATION
                                0079  ;
                                0079  ; STATUS CODES RETURNED:
                                0079  ;
                                0079  ;       SS$_NORMAL - SUCCESSFUL OPERATION
                                0079  ;       SS$_IVCHAN - SPECIFIED CHANNEL INVALID
                                0079  ;       SS$_MBTOOSML - MESSAGE EXCEEDS MAILBOX SIZE
                                0079  ;       SS$_DEVOFFLIN - DEVICE OFFLINE
                                0079  ;       SS$_INSFMEM - INSUFFICIENT MEMORY FOR THE REQUEST
                                0079  ;       SS$_ACCVIO - ACCESS VIOLATION ON BUFFER
                                0079  ;       SS$_NOPRIV - PROCESS DOES NOT HAVE READ ACCESS TO SPECIFIED MAILBOX
                                0079  ;                    OR THE MESSAGE REQUEST TYPE REQUIRES THE OPERATOR PRIV.
                                0079  ;       SS$_DEVNOTMBX - SPECIFIED CHANNEL IS NOT TO MAILBOX
                                0079  ;       SS$_BADPARAM - MESSAGE SIZE ERROR
                                0079  ;--
                                0079  BADPARAM:
        50    14         3C     0079          MOVZWL  #SS$_BADPARAM,R0        ; SET BAD PARAM ERROR
                         04     007C  ERROR:  RET                             ; ERROR RETURN
                                007D  BUILDMSG:                               ; BUILD MESSAGE
                                007D          ;
                                007D          ; MINIMIZE THE ALLOWABLE MESSAGE SIZE WITH
                                007D          ; THE SYSTEM MAXBUF PARAMETER.
                                007D          ;
   57    00000000'EF     B1     007D          CMPW    IOC$GW_MAXBUF,R7        ; COMPARE MAX MSG SIZE AGAINST SYS MAX
                  07     1E     0084          BGEQU   10$                     ; BRANCH IF SYSTEM MAX GREATER
   57    00000000'EF     3C     0086          MOVZWL  IOC$GW_MAXBUF,R7        ; SET MAXBUF AS MSG LIMIT
                                008D          ;
                                008D          ; CHECK THE INPUT PARAMETERS
                                008D          ;
       51    04 AC       D0     008D  10$:    MOVL    MSG(AP),R1              ; GET MESSAGE DESCRIPTER
                  E6     13     0091          BEQL    BADPARAM                ; IF EQL THEN NO MESSAGE AND ERROR
         00000000'GF     16     0093          JSB     G^EXE$PROBER_DSC        ; PROBE DESCRIPTOR AND BUFFER
               E0 50     E9     0099          BLBC    R0,ERROR                ; BRANCH IF ERROR
                                009C          ;
                                009C          ; R1<0:15> = SIZE, R2 = ADDRESS OF BUFFER
                                009C          ;
        59    52         D0     009C          MOVL    R2,R9                   ; SAVE ADDRESS OF BUFFER
```

```
          58   51   3C  009F   305          MOVZWL  R1,R8                        ; GET SIZE OF MESSAGE
               D5   13  00A2   306          BEQL    BADPARAM                     ; IF EQL THEN ILLEGAL
          57   58   B1  00A4   307          CMPW    R8,R7                        ; LEGAL SIZE?
               D0   1A  00A7   308          BGTRU   BADPARAM                     ; IF GTRU THEN NO
          04   5B   B1  00A9   309          CMPW    R11,#MSG$_SNDSMB             ; $SNDSMB service?
               18   12  00AC   310          BNEQ    20$                          ; Br if not
          02   58   B1  00AE   311          CMPW    R8,#2                        ; Message at least 2 bytes?
               C6   1F  00B1   312          BLSSU   BADPARAM                     ; Br if not, invalid
          57   1E   D0  00B3   313          MOVL    #30,R7                       ; Assume offset past file ID
          0A   69   B1  00B6   314          CMPW    (R9),#SMR$K_ADDFIL           ; Add file request?
               08   13  00B9   315          BEQL    15$                          ; Br if yes
          57   2E   D0  00BB   316          MOVL    #46,R7                       ; Assume offset past file ID
          08   69   B1  00BE   317          CMPW    (R9),#SMR$K_ENTER            ; Enter file request?
               03   12  00C1   318          BNEQ    20$                          ; Br if no, no special handling
             0093   31  00C3   319 15$:     BRW     60$                          ; Go to file protection check code
     51   58   26   C1  00C6   320 20$:     ADDL3   #COMMON_HDR,R8,R1            ; CALC SIZE OF TOTAL MESSAGE
          5B   08   B1  00CA   321          CMPW    #MSG$_OPRQST,R11             ; SYMBIONT OR ACCOUNTING MGR MESSAGE?
               03   13  00CD   322          BEQL    30$                          ; IF EQL, NO
          51   1E   C0  00CF   323          ADDL2   #<ACM$Q_SYSTIME+8-ACM$B_PROCPRI-2>,R1
                        00D2   324                                               ; ALLOCATE SPACE FOR ID DATA
     50   0124 8F   3C  00D2   325 30$:     MOVZWL  #SS$_INSFMEM,R0              ; ASSUME NO STACK
     53   5E   51   C3  00D7   326          SUBL3   R1,SP,R3                     ; ADDRESS MESSAGE STORAGE
  00000000'9F  53   D1  00DB   327          CMPL    R3,@#CTL$AL_STACK            ; IN KERNEL STACK?
               98   1B  00E2   328          BLEQU   ERROR                        ; IF LEQU THEN YES
                        00E4   329 ;
                        00E4   330 ; BUILD THE MESSAGE ON THE EXEC STACK.
                        00E4   331 ;
          5E   53   D0  00E4   332          MOVL    R3,SP                        ; ALLOCATE THE SPACE
               28   BB  00E7   333          PUSHR   #^M<R3,R5>                   ; SAVE SIZE AND UCB ADDRESS
          83   5B   B0  00E9   334          MOVW    R11,(R3)+                    ; INSERT MESAGE TYPE
     83   08 AC     B0  00EC   335          MOVW    MBX(AP),(R3)+                ; INSERT REPLY MAILBOX CHANNEL NUMBER
  56   00000000'GF  D0  00F0   336          MOVL    G^CTL$GL_PCB,R6              ; GET ADDRESS OF PCB
                        00F7   337
                        00F7   338          ASSUME  PHD$Q_PRIVMSK EQ 0
                        00F7   339
     83   6C B6     7D  00F7   340          MOVQ    @PCB$L_PHD(R6),(R3)+         ; INSERT PRIVILEGE MASK
    83   008C C6    D0  00FB   341          MOVL    PCB$L_UIC(R6),(R3)+          ; INSERT UIC
63  00000000'9F 14  28  0100   342          MOVC3   #20,@#CTL$T_USERNAME,(R3); INSERT USER NAME AND ACCOUNT NAME
   83   1F   2F A6  83  0108   343          SUBB3   PCB$B_PRIB(R6),#31,(R3)+; INSERT BASE PRIORITY
               83   94  010D   344          CLRB    (R3)+                        ; CLEAR SPARE BYTE
          5B   08   B1  010F   345          CMPW    #MSG$_OPRQST,R11             ; ACCOUNTING OR SYMBIONT MESSAGE ?
               2F   13  0112   346          BEQL    50$                          ; IF EQL, NO
                        0114   347 ;************************************************************************
                        0114   348 ;
                        0114   349          ASSUME  ACM$W_MAILBOX EQ ACM$W_TYPE+2
                        0114   350          ASSUME  ACM$Q_PRVMSK EQ ACM$W_MAILBOX+2
                        0114   351          ASSUME  ACM$L_UIC EQ ACM$Q_PRVMSK+8
                        0114   352          ASSUME  ACM$T_USERNAME EQ ACM$L_UIC+4
                        0114   353          ASSUME  ACM$T_ACCOUNT EQ ACM$T_USERNAME+12
                        0114   354          ASSUME  ACM$B_PROCPRI EQ ACM$T_ACCOUNT+8
                        0114   355          ASSUME  ACM$L_PID EQ ACM$B_PROCPRI+4
                        0114   356          ASSUME  ACM$L_STS EQ ACM$L_PID+4
                        0114   357          ASSUME  ACM$L_OWNER EQ ACM$L_STS+4
                        0114   358          ASSUME  ACM$T_TERMINAL EQ ACM$L_OWNER+4
                        0114   359          ASSUME  ACM$Q_SYSTIME EQ ACM$T_TERMINAL+8
                        0114   360 ;
                        0114   361 ;************************************************************************
```

```
                  83  B4  0114        CLRW    (R3)+                    ; CLEAR SPARE BYTES
          83   64 A6  D0  0116        MOVL    PCB$L_EPID(R6),(R3)+     ; INSERT EXTENDED PROCESS ID
          83   24 A6  D0  011A        MOVL    PCB$L_STS(R6),(R3)+      ; INSERT PROCESS STATUS
          83   68 A6  D0  011E        MOVL    PCB$L_EOWNER(R6),(R3)+   ; INSERT EXTENDED OWNER PID (0 => NONE)
          83   44 A6  7D  0122        MOVQ    PCB$T_TERMINAL(R6),(R3)+; INSERT TERMINAL NAME
    63  00000000'EF  7D  0126  40$:  MOVQ    EXE$GQ_SYSTIME,(R3)      ; CURRENT SYSTEM TIME
    63  00000000'EF  D1  012D        CMPL    EXE$GQ_SYSTIME,(R3)      ; VERIFY THAT VALUE ACQUIRED WAS
                  F0  12  0134        BNEQ    40$                      ;  NOT BEING MODIFIED AT THE SAME
 04 A3  00000004'EF  D1  0136        CMPL    EXE$GQ_SYSTIME+4,4(R3)   ;  AT THE SAME TIME.  ACQUIRE TIME
                  E6  12  013E        BNEQ    40$                      ;  AGAIN IF IT CHANGED.
             53   08  C0  0140        ADDL    #8,R3                    ; POINT TO NEXT FIELD
       63    69   58  28  0143  50$:  MOVC3   R8,(R9),(R3)             ; COPY MESSAGE
       7E    5D   6E  C3  0147        SUBL3   (SP),FP,-(SP)            ; CALC MESSAGE SIZE
                      014B        $CMKRNL_S       W^SENDMSG,(SP)   ; SEND MESSAGE IN KERNEL MODE
                      04  0158        RET                              ; RETURN AND CLEAN STACK
```

```
                              0159   378            .SBTTL  File protection check
                              0159   379 ;
                              0159   380 ; Stack work area offsets for protection check routine.
                              0159   381 ;
                  00000000    0159   382 FWA_ATRLIST=     0                          ; Attribute list
                  00000014    0159   383 FWA_FIB=         20                         ; File information block
                  00000054    0159   384 FWA_CHAN=        FIB$C_LENGTH+20            ; Channel assigned to device
                  00000056    0159   385 FWA_DVI_DESC=    FIB$C_LENGTH+22            ; Descriptor for device name
                  0000005E    0159   386 FWA_FIB_DESC=    FIB$C_LENGTH+30            ; Descriptor for FIB
                  00000066    0159   387 FWA_IOSB=        FIB$C_LENGTH+38            ; I/O status block
                  0000006E    0159   388 FWA_RECATTR=     FIB$C_LENGTH+46            ; Record attributes
                              0159   389
                  0000008E    0159   390 FWA_FILE_SPEC=   FIB$C_LENGTH+78            ; File specification
                  0000018E    0159   391 FWA_FSPC_LEN=    FIB$C_LENGTH+334           ; File specification length
                  00000190    0159   392 FWA_DVI=         FIB$C_LENGTH+336           ; DVI
                  000001A0    0159   393 FWA_FID=         FIB$C_LENGTH+352           ; FID
                  000001A6    0159   394 FWA_DID=         FIB$C_LENGTH+358           ; DID
                  000001AC    0159   395 FWA_FILE_SIZE=   FIB$C_LENGTH+364           ; File size in blocks
                  000001B0    0159   396 FWA_SPARE=       FIB$C_LENGTH+368           ; Spare longword
                              0159   397
                  000001B4    0159   398 FWA_SIZE=        FIB$C_LENGTH+372           ; Length of area
                              0159   399 ;
                              0159   400 ; The SMR$K_ENTER and SMR$K_ADDFIL functions check the protection of the
                              0159   401 ; submitted file, and append information to the user's message buffer:
                              0159   402 ;
                              0159   403 ;            fixed area
                              0159   404 ;            user's message buffer
                              0159   405 ;            zero byte to stop options scan
                              0159   406 ;            file specification
                              0159   407 ;            1 word length of file specification
                              0159   408 ;            28 byte DVI/FID/DID
                              0159   409 ;            longword file size
                              0159   410 ;            longword allowed file access
                              0159   411 ;
                              0159   412 60$:                                        ; Check file protection
              57     58   B1  0159   413            CMPW    R8,R7               ; Message contains file ID?
                     6B   1F  015C   414            BLSSU   120$                ; Br if not, invalid
                              015E   415 ;
                              015E   416 ; Check for sufficient space to allocate the work area, and do so.
                              015E   417 ;
           50      0124 8F   3C  015E   418            MOVZWL  #SS$_INSFMEM,R0     ; Assume no space
           51      FE4C CE   9E  0163   419            MOVAB   -FWA_SIZE(SP),R1    ; Get lowest address that will be used
       00000000'9F     51   D1  0168   420            CMPL    R1,@#CTL$AL_STACK   ; Compare against that available
                     5B   1F  016F   421            BLSSU   130$                ; Br if space exceeded
              5E     51   D0  0171   422            MOVL    R1,SP               ; Allocate the space
                              0174   423 ;
                              0174   424 ; Move the DVI/FID/DID to the result area.
                              0174   425 ;
                     55   DD  0174   426            PUSHL   R5                  ; Save R5
      0194 CE    E4 A947  1C   28  0176   427            MOVC3   #28,-28(R9)[R7],FWA_DVI+4(SP) ; Move parameter to the work area
14 AE  0040 8F    00   6E  00   2C  017E   428            MOVC5   #0,(SP),#0,#FIB$C_LENGTH,FWA_FIB(SP) ; Initialize FIB
                  55 8ED0  0187   429            POPL    R5                  ; Restore R5
           0F      0190 CE   91  018A   430            CMPB    FWA_DVI(SP),#15     ; Check device name length
                     38   1A  018F   431            BGTRU   120$                ; Br if invalid
                              0191   432 ;
                              0191   433 ; Scan the options string for a delete request. If found, we have to
                              0191   434 ; check for delete access to the file.
```

```
SYS
Sym
$$T
ACM
ACM
ACM
ACM
ACM
ACM
ACM
ACM
ACM
ACM
ACM
ARM
ATR
ATR
ATR
BAD
BIT
BUI
CCB
COM
CTL
CTL
CTL
DEV
DEV
DEV
DEV
DEV
EMB
EMB
EMB
EMB
EMB
EMB
EMB
ERL
ERL
ERR
EXE
EXE
EXE
EXE
EXE
EXE
EXE
EXE
EXE
EXE
EXE
EXE
EXE
FAT
FAT
FIB
FIB
```

```
                                0191   ;
                      56   D4   0191            CLRL    R6                          ; Assume no delete wanted
           51    14 A947   9E   0193            MOVAB   20(R9)[R7],R1               ; Point to start of options string
           52    59   58   C1   0198            ADDL3   R8,R9,R2                    ; Get end of options string
                 52   51   D1   019C   80$:     CMPL    R1,R2                       ; Check if at end
                      2E   1E   019F            BGEQU   150$                        ; Branch if done
                 50   81   9A   01A1            MOVZBL  (R1)+,R0                    ; Get next option byte
              40 8F   50   91   01A4            CMPB    R0,#64                      ; Check option validity
                      1F   1E   01A8            BGEQU   120$                        ; Branch if out of range
                 01   50   91   01AA            CMPB    R0,#SMO$K_DELETE            ; Check for delete option
                      03   12   01AD            BNEQ    90$                         ; Branch if not
                 56   01   D0   01AF            MOVL    #1,R6                       ; Note delete wanted
         53   FE49 CF40   98   01B2   90$:     CVTBL   OPTION_SIZE[R0],R3          ; Get size of option text
                      05   18   01B8            BGEQ    110$                        ; Branch if fixed
                 53   81   9A   01BA   100$:    MOVZBL  (R1)+,R3                    ; Else counted string - get size
                      DD   13   01BD            BEQL    80$                         ; Branch if end of params
                 51   53   C0   01BF   110$:    ADDL    R3,R1                       ; Skip option text
                 26   50   91   01C2            CMPB    R0,#SMO$K_PARAMS            ; Check for parameter option
                      D5   12   01C5            BNEQ    80$                         ; Branch if not
                      F1   11   01C7            BRB     100$                        ; And loop for next parameter
                                01C9   ;
                                01C9   ; Helper branches.
                                01C9   ;
                    FEAD   31   01C9   120$:    BRW     BADPARAM
                    FEAD   31   01CC   130$:    BRW     ERROR
                                01CF   ;
                                01CF   ; Get a pointer to the DVI descriptor, and where the channel will be stored,
                                01CF   ; and initialize the descriptor.
                                01CF   ;
               01B0 CE   D4   01CF   150$:    CLRL    FWA_SPARE(SP)               ; Clear spare longword
            50    56 AE   9E   01D3            MOVAB   FWA_DVI_DESC(SP),R0         ; Point to DVI descriptor
            53    54 AE   9E   01D7            MOVAB   FWA_CHAN(SP),R3             ; Point to channel
         60     0190 CE   9A   01DB            MOVZBL  FWA_DVI(SP),(R0)            ; Store device name length
      04 A0     0191 CE   9E   01E0            MOVAB   FWA_DVI+1(SP),4(R0)         ; Store device name address
                                01E6   ;
                                01E6   ; Assign a channel to the device.
                                01E6   ;
                                01E6            $ASSIGN_S -
                                01E6                    DEVNAM=(R0), -              ; Device name
                                01E6                    CHAN=(R3)                   ; Output channel number
                   D6 50   E9   01F3            BLBC    R0,130$                     ; Br if not assigned
                                01F6   ;
                                01F6   ; Build the FIB, the FIB descriptor, and the ACP attributes list.
                                01F6   ;
               50    6E   9E   01F6            MOVAB   FWA_ATRLIST(SP),R0          ; Point to attribute list
            51    5E AE   9E   01F9            MOVAB   FWA_FIB_DESC(SP),R1         ; Point to FIB descriptor
            52    66 AE   9E   01FD            MOVAB   FWA_IOSB(SP),R2             ; Point to IOSB
            54    14 AE   9E   0201            MOVAB   FWA_FIB(SP),R4              ; Point to FIB
                                0205
                      64   D4   0205            CLRL    FIB$L_ACCTL(R4)             ; Clear access control
      04 A4     01A0 CE   D0   0207            MOVL    FWA_FID(SP),FIB$W_FID(R4)   ; Store file ID
      08 A4     01A4 CE   B0   020D            MOVW    FWA_FID+4(SP),FIB$W_FID+4(R4)
                   1A 56   E9   0213            BLBC    R6,160$                     ; Check if delete access needed
      0A A4     01A6 CE   D0   0216            MOVL    FWA_DID(SP),FIB$W_DID(R4)   ; Store directory ID
      0E A4     01AA CE   B0   021C            MOVW    FWA_DID+4(SP),FIB$W_DID+4(R4)
         3C A4      08   D0   0222            MOVL    #ARM$M_DELETE,FIB$L_ALT_ACCESS(R4) ; Store alternate access mask
         38 A4      01   D0   0226            MOVL    #FIB$M_ALT_REQ,FIB$L_STATUS(R4) ; Note alternate access required
      14 A4    0800 8F     B0  022A               MOVW     #FIB$M_FINDFID,FIB$W_NMCTL(R4) ; Search for file ID
                               0230
     61     00000040 8F    D0  0230        160$:  MOVL     #FIB$C_LENGTH,(R1)       ; Initialize FIB descriptor
              04 A1    64  9E  0237               MOVAB    (R4),4(R1)               ;
                               023B
     60     00040020 8F    D0  023B               MOVL     #<ATR$S_RECATTR+<ATR$C_RECATTR@16>>,(R0)
          04 A0    6E AE   9E  0242               MOVAB    FWA_RECATTR(SP),4(R0)
   08 A0    002E0100 8F    D0  0247               MOVL     #<256+<ATR$C_FILE_SPEC@16>>,8(R0)
     0C A0    008E CE      9E  024F               MOVAB    FWA_FILE_SPEC(SP),12(R0)
                 10 A0     D4  0255               CLRL     16(R0)
                               0258        ;
                               0258        ; Access the file to get necessary information.
                               0258        ;
                               0258               $QIOW_S -
                               0258                        EFN=#31, -              ; Event flag
                               0258                        CHAN=(R3), -            ; Channel number
                               0258                        FUNC=#IO$_ACCESS, -     ; Read attributes function
                               0258                        IOSB=(R2), -            ; I/O status block
                               0258                        P1=(R1), -              ; Address of FIB descriptor
                               0258                        P5=R0                   ; Address of attribute list
                      50   DD  0275               PUSHL    R0                      ; Save $QIOW status
                               0277               $DASSGN_S -
                               0277                        CHAN=(R3)               ; Deassign the channel
                    50 8ED0    0281               POPL     R0                      ; Restore status from access
                   50 50   E9  0284               BLBC     R0,190$                 ; Br if $QIOW failed
            50     66 AE   3C  0287               MOVZWL   FWA_IOSB(SP),R0         ; Pick up status from IOSB
                   49 50   E9  028B               BLBC     R0,190$                 ; Br if operation failed
                               028E        ;
                               028E        ; Compute the file size from the record attributes.
                               028E        ;
   01AC CE      76 AE   10  9C 028E               ROTL     #16, -                  ; Move EFBLK to file size area and
                               0295                        FWA_RECATTR+FAT$L_EFBLK(SP), - ; convert to unswapped
                               0295                        FWA_FILE_SIZE(SP)
                      09   13  0295               BEQL     170$                    ; Br if EFBLK is zero
                   7A AE   B5  0297               TSTW     FWA_RECATTR+FAT$W_FFBYTE(SP) ; Test first free byte
                      04   12  029A               BNEQ     170$                    ; Br if nonzero
                 01AC CE   D7  029C               DECL     FWA_FILE_SIZE(SP)       ; Adjust EFBLK
                               02A0        ;
                               02A0        ; Slide the file specification up adjacent to the count, and finish it by adding
                               02A0        ; the zero byte.
                               02A0        ;
        57    008E CE      3C  02A0        170$:  MOVZWL   FWA_FILE_SPEC(SP),R7    ; Get file specification length
        00FE 8F       57   B1  02A5               CMPW     R7,#254                 ; Check maximum supported length
                      05   1B  02AA               BLEQU    180$                    ; Br if in range
        57    00FE 8F      3C  02AC               MOVZWL   #254,R7                 ; Shorten
        018E CE       57   B0  02B1        180$:  MOVW     R7,FWA_FSPC_LEN(SP)     ; Set length in message
   56   000000FE 8F   57   C3  02B6               SUBL3    R7,#254,R6              ; Compute bias
                      55   DD  02BE               PUSHL    R5                      ; Save R5
 0094 CE46    0094 CE 57   28  02C0               MOVC3    R7, -                   ; Slide item up
                               02C9                        FWA_FILE_SPEC+6(SP), -
                               02C9                        FWA_FILE_SPEC+6(SP)[R6]
                    55 8ED0    02C9               POPL     R5                      ; Restore R5
        5E    008F CE46    9E  02CC               MOVAB    FWA_FILE_SPEC+1(SP)[R6],SP ; Delete unused stack
                      6E   94  02D2               CLRB     (SP)                    ; Zero byte to stop options scan
                    FDEF   31  02D4               BRW      20$                     ; Return to mainline processing
                               02D7        ;
                               02D7        ; Helper branches.
                 02D7   549 ;
   FDA2    31    02D7   550 190$:   BRW     ERROR
                 02DA   551
                 02DA   552         .DSABL  LSB
```

```
                                  02DA
                                  02DA          .SBTTL  SEND MESSAGE ROUTINE
                                  02DA  ;++
                                  02DA  ; SENDMSG - KERNEL MODE MESSAGE SEND ROUTINE
                                  02DA  ;
                                  02DA  ; FUNCTIONAL DESCRIPTION:
                                  02DA  ;
                                  02DA  ; THIS ROUTINE RUNS IN KERNEL MODE AND SENDS THE MESSAGE TO THE
                                  02DA  ; TARGET MAILBOX.
                                  02DA  ;
                                  02DA  ; INPUTS:
                                  02DA  ;
                                  02DA  ;       0(AP) = SIZE OF MESSAGE
                                  02DA  ;       4(AP) = ADDRESS OF THE MESSAGE
                                  02DA  ;       8(AP) = MAILBOX UCB ADDRESS
                                  02DA  ;
                                  02DA  ; OUTPUTS:
                                  02DA  ;
                                  02DA  ;       R0 = STATUS OF THE OPERTATION
                                  02DA  ;
                                  02DA  ; STATUS CODES RETURNED:
                                  02DA  ;
                                  02DA  ;       SS$_NORMAL - SUCCESSFUL OPERATION
                                  02DA  ;       SS$_IVCHAN - SPECIFIED CHANNEL INVALID
                                  02DA  ;       SS$_MBTOOSML - MESSAGE EXCEEDS MAILBOX SIZE
                                  02DA  ;       SS$_DEVOFFLIN - NO LISTENER FOR SYMBIONT OR JOB CONTROLLER
                                  02DA  ;       OPC$_NOPERATOR - NO LISTENER FOR OPERATOR REQUEST
                                  02DA  ;       SS$_INSFMEM - INSUFFICIENT MEMORY FOR THE REQUEST
                                  02DA  ;       SS$_NOPRIV - PROCESS DOES NOT HAVE READ ACCESS TO SPECIFIED MAILBOX
                                  02DA  ;       SS$_DEVNOTMBX - SPECIFIED CHANNEL IS NOT TO MAILBOX
                                  02DA  ;--
                                  02DA  SENDMSG:                                  ;
                          0000    02DA          .WORD   0                       ; SAVE NO REGISTERS
       55    04 AC          D0    02DC          MOVL    4(AP),R5                ; GET MESSAGE ADDRESS
       50    02 A5          3C    02E0          MOVZWL  2(R5),R0                ; GET CHANNEL NUMBER
                38          13    02E4          BEQL    10$                     ; IF EQL THEN NO REPLY
 54    00000000'EF          D0    02E6          MOVL    SCH$GL_CURPCB,R4        ; GET CURRENT PCB
       00000000'GF          16    02ED          JSB     G^IOC$VERIFYCHAN        ; CHECK OUT CHANNEL NUMBER
             31 50          E9    02F3          BLBC    R0,20$                  ; BR IF ERROR
       53    04 AC          D0    02F6          MOVL    4(AP),R3                ; GET MESSAGE ADDRESS
          55    61          D0    02FA          MOVL    CCB$L_UCB(R1),R5        ; GET UCB OF REPLY MAILBOX
     02 A3    54 A5         B0    02FD          MOVW    UCB$W_UNIT(R5),2(R3)    ; INSERT UNIT NUMBER OF MAILBOX
     50    0074 8F          3C    0302          MOVZWL  #SS$_DEVNOTMBX,R0       ; ASSUME DEVICE NOT MAILBOX
   1B 38 A5     14          E1    0307          BBC     #DEV$V_MBX,UCB$L_DEVCHAR(R5),20$; BR IF NOT MAILBOX
       00000000'GF          16    030C          JSB     G^EXE$CHKRDACCES        ; CHECK ACCESS
             12 50          E9    0312          BLBC    R0,20$                  ; BR IF ERROR IN ACCESS
       00000000'GF          16    0315          JSB     G^EXE$CHKWRTACCES       ; CHECK OUT FOR WRITE
             09 50          E9    0318          BLBC    R0,20$                  ; AND RETURN IF NO ACCESS
                                  031E  10$:
       55    08 AC          D0    031E          MOVL    8(AP),R5                ; ADDRESS UCB OF MAILBOX
          53    6C          7D    0322          MOVQ    (AP),R3                 ; GET SIZE AND ADDRESS OF MESSAGE
                01          10    0325          BSBB    EXE$SENDMSG             ; SEND MESSAGE
                                  0327  20$:
                            04    0327          RET
```

```
                                 0328  ;+
                                 0328  ; EXE$SENDMSG -- Send mail box message
                                 0328  ;
                                 0328  ; INPUTS:       R3 = message size
                                 0328  ;               R4 = message address
                                 0328  ;               R5 = UCB address
                                 0328  ;
                                 0328  ;-
                                 0328
                                 0328  EXE$SENDMSG::
                                 0328  ;
                                 0328  ; Check the reference count in the UCB to see if the
                                 0328  ; mailbox has a listener.  Note that both the JOB CONTROLER
                                 0328  ; mailbox and the OPERATOR mailbox have an initial ref. count
                                 0328  ; of 1.  Therefore, if there is a listener at the mailbox,
                                 0328  ; the reference count must be greater than 1.
                                 0328  ;
              01   5C A5   B1    0328          CMPW    UCB$W_REFC(R5),#1       ; DOES A LISTENER EXIST?
                      1F   1B    032C          BLEQU   10$                     ; BRANCH IF NOT
                                 032E  ;
                                 032E  ; The message must faulted in before calling EXE$WRTMAILBOX.
                                 032E  ; The manner in which this is done verges on the magical.
                                 032E  ; For a detailed explanation, see below.
                                 032E  ;
                                 032E  ; First round the address of the message (on the EXEC stack)
                                 032E  ; DOWN to the nearest page boundary.  Then raise IPL to ASTDEL
                                 032E  ; to block AST delivery.  This step is necessary to avoid
                                 032E  ; pagefaults incurred during the execution of the AST routine.
                                 032E  ; Since we now have a page-aligned base address of the message
                                 032E  ; in R2, we can use one instruction to fault in the pages,
                                 032E  ; two pages with each operand referenced.  This is done by
                                 032E  ; choosing the operand address and instruction operand context
                                 032E  ; so that the operand is split across a page boundary.
                                 032E  ; Note that if the size (rounded up) in pages of the message
                                 032E  ; is N, then the maximum number of pages that must be faulted
                                 032E  ; in is N+1.  If the message size is small, and the message
                                 032E  ; resides near the end of the EXEC stack, it is possible that
                                 032E  ; we may overrun the EXEC stack.  This is ok, because the
                                 032E  ; KERNEL stack follows the EXEC stack, and we're in KERNEL
                                 032E  ; mode right now (remember?).  Note that after the KERNEL
                                 032E  ; stack is a page that is inacessable.  As a result, the message
                                 032E  ; size (in pages) must not exceed the number of pages in the
                                 032E  ; KERNEL stack.
                                 032E  ;
   52    54   000001FF 8F  CB    032E          BICL3   #^X01FF,R4,R2           ; ROUND ADDRESS DOWN
                                 0336          SETIPL  #IPL$_ASTDEL            ; BLOCK AST DELIVERY
       03FF C2    01FF C2  B1    0339          CMPW    511(R2),1023(R2)        ; FAULT IN 4 CONTIGUOUS PAGES
              00000000'GF  16    0340          JSB     G^EXE$WRTMAILBOX        ; WRITE MESSAGE
                                 0346          SETIPL  #0                      ; ENABLE AST DELIVERY
                   01 50   E9    0349          BLBC    R0,10$                  ; BRANCH IF ERROR
                           05    034C          RSB
                                 034D  ;
                                 034D  ; SOMETHING IS WRONG.  EITHER THERE IS NO LISTENER FOR
                                 034D  ; THE SPECIFIED MAILBOX, OR THERE IS INSUFFICIENT NON-
                                 034D  ; PAGED POOL TO MAIL THE MESSAGE.  CHECK THE STATUS CODE
                                 034D  ; AND DO THE APPROPRIATE THING.
                                 034D  ;
       0124 8F    50    B1  034D   10$:    CMPW    R0,#SS$_INSFMEM          ; INSUFFICIENT MEMORY?
                  12    13  0352           BEQL    30$                      ; BRANCH IF YES
                            0354   ;
                            0354   ; NO LISTENER IS PRESENT.
                            0354   ;
                            0354   ; IF THE SPECIFIED MAILBOX IS THE OPERATOR MAILBOX THEN
                            0354   ; THE SENDER IS INFORMED THAT NO OPERATOR IS PRESENT.
                            0354   ;
       50     0084 8F       3C  0354           MOVZWL  #SS$_DEVOFFLINE,R0       ; ASSUME NO LISTENER
              64    08      91  0359           CMPB    #MSG$_OPRQST,(R4)        ; OPERATOR?
                    07      12  035C           BNEQ    20$                      ; IF NEQ THEN NO
 50     00058061 8F         D0  035E           MOVL    #OPC$_NOPERATOR,R0       ; SET NO OPERATOR SUCCESS STATUS
                            05  0365   20$:    RSB                              ; RETURN
                                0366   ;
                                0366   ; THERE WAS INSUFFICIENT NONPAGED POOL TO SEND THE MESSAGE.
                                0366   ; IF THE PROCESS HAS RESOURCE WAIT MODE ENABLED, WAIT FOR
                                0366   ; THE POOL TO FREE UP.  IF NOT, THEN RETURN THE ERROR STATUS.
                                0366   ;
                                0366   30$:
                    38      BB  0366           PUSHR   #^M<R3,R4,R5>            ; SAVE SIZE AND ADDRESS OF MESSAGE
                                0368                                            ; AND UCB ADDRESS
 54     00000000'EF         D0  0368           MOVL    SCH$GL_CURPCB,R4         ; GET CURRENT PROCESS PCB ADDRESS
                    0A      E0  036F           BBS     #PCB$V_SSRWAIT,-         ; IF SET, NO WAIT
            12 24   A4          0371                   PCB$L_STS(R4),40$        ;
            50      03      3C  0374           MOVZWL  #RSN$_NPDYNMEM,R0        ; SET RESOURCE WAIT NUMBER
                    7E      DC  0377           MOVPSL  -(SP)                    ; PUSH PSL ON STACK
                                0379           SETIPL  #IPL$_SYNCH              ; RAISE IPL TO SYNCH
        00000000'GF         16  037C           JSB     G^SCH$RWAIT              ; WAIT FOR NONPAGED MEMORY
                    38      BA  0382           POPR    #^M<R3,R4,R5>            ; RESTORE SIZE AND ADDRESS OF MESSAGE
                                0384                                            ; AND UCB ADDRESS
                    A2      11  0384           BRB     EXE$SENDMSG              ; TRY AGAIN
                                0386   40$:
                    38      BA  0386           POPR    #^M<R3,R4,R5>            ; RESTORE SIZE AND ADDRESS OF MESSAGE
                                0388                                            ; AND UCB ADDRESS
                            05  0388           RSB
```

```
               0389         .SBTTL  EXE$OPRSNDERL - OPERATOR SEND MESSAGE TO ERROR LOGGER
               0389  ;++
               0389  ; EXE$OPRSNDERL - OPERATOR SEND MESSAGE TO ERROR LOGGER
               0389  ;
               0389  ; FUNCTIONAL DESCRIPTION:
               0389  ;
               0389  ; THIS ROUTINE PROVIDES THE SEND TO ERROR LOGGER SYSTEM SERVICE
               0389  ; FOR THE OPERATOR COMMUNICATION PROCESS.
               0389  ; THE ACTION IS TO BUILD A MESSAGE CONSISTING OF A COMMON HEADER
               0389  ; AND THE USER SPECIFIED TEXT AND THEN SEND IT TO THE ERROR LOG FORMAT PROGRAM.
               0389  ; THE SPECIFIED MESSAGE IS ADDRESSED CHECKED.
               0389  ;
               0389  ; INPUTS:
               0389  ;
               0389  ;        MSG(AP) = ADDRESS OF THE MESSAGE DESCRIPTER
               0389  ;
               0389  ; OUTPUTS:
               0389  ;
               0389  ;        R0 = STATUS OF THE OPERATION
               0389  ;
               0389  ; STATUS CODES RETURNED:
               0389  ;
               0389  ;        SS$_NORMAL - SUCCESSFUL OPERATION
               0389  ;        SS$_INSFMEM - INSUFFICIENT MEMORY FOR THE REQUEST
               0389  ;        SS$_ACCVIO - ACCESS VIOLATION ON BUFFER
               0389  ;--
               0389         .ENABL  LSB
               0389
               0389  EXE$OPRSNDERL::                         ; SEND TO ERROR LOGGER
         003C  0389         .WORD   ^M<R2,R3,R4,R5>          ;
     29    DD  038B         PUSHL   S^#EMB$K_OM              ; SET MESSAGE TYPE TO OPERATOR MESSAGE
     13    11  038D         BRB     10$                      ; JOIN COMMON CODE
```

```
              038F          .SBTTL  EXE$NETSNDERL - NETWORK SEND MESSAGE TO ERROR LOGGER
              038F    ;++
              038F    ; EXE$NETSNDERL - NETWORK SEND MESSAGE TO ERROR LOGGER
              038F    ;
              038F    ; FUNCTIONAL DESCRIPTION:
              038F    ;
              038F    ; THIS ROUTINE PROVIDES THE SEND TO ERROR LOGGER SYSTEM SERVICE
              038F    ; FOR THE NETWORK COMMUNICATION PROCESS.
              038F    ; THE ACTION IS TO BUILD A MESSAGE CONSISTING OF A COMMON HEADER
              038F    ; AND THE USER SPECIFIED TEXT AND THEN SEND IT TO THE ERROR LOG FORMAT PROGRAM.
              038F    ; THE SPECIFIED MESSAGE IS ADDRESSED CHECKED.
              038F    ;
              038F    ; INPUTS:
              038F    ;
              038F    ;       MSG(AP) = ADDRESS OF THE MESSAGE DESCRIPTER
              038F    ;
              038F    ; OUTPUTS:
              038F    ;
              038F    ;       R0 = STATUS OF THE OPERATION
              038F    ;
              038F    ; STATUS CODES RETURNED:
              038F    ;
              038F    ;       SS$_NORMAL - SUCCESSFUL OPERATION
              038F    ;       SS$_INSFMEM - INSUFFICIENT MEMORY FOR THE REQUEST
              038F    ;       SS$_ACCVIO - ACCESS VIOLATION ON BUFFER
              038F    ;--
              038F
              038F   EXE$NETSNDERL::                          ; SEND TO ERROR LOGGER
        003C  038F          .WORD   ^M<R2,R3,R4,R5>           ;
   2A     DD  0391          PUSHL   S^#EMB$K_NM               ; SET MESSAGE TYPE TO NETWORK MESSAGE
   0D     11  0393          BRB     10$                       ; JOIN COMMON CODE
```

```
                          0395          .SBTTL  EXE$SNDERL - SEND MESSAGE TO ERROR LOGGER
                          0395     ;++
                          0395     ; EXE$SNDERR - SEND MESSAGE TO ERROR LOGGER
                          0395     ;
                          0395     ; FUNCTIONAL DESCRIPTION:
                          0395     ;
                          0395     ; THIS ROUTINE PROVIDES THE SEND TO ERROR LOGGER SYSTEM SERVICE.
                          0395     ; THE ACTION IS TO BUILD A MESSAGE CONSISTING OF A COMMON HEADER
                          0395     ; AND THE USER SPECIFIED TEXT AND THEN SEND IT TO THE ERROR LOG FORMAT PROGRAM.
                          0395     ; THE SPECIFIED MESSAGE IS ADDRESSED CHECKED.
                          0395     ;
                          0395     ; INPUTS:
                          0395     ;
                          0395     ;       MSG(AP) = ADDRESS OF THE MESSAGE DESCRIPTER
                          0395     ;       R4      = ADDRESS OF CURRENT PROCESS PCB - COURTESY OF CMKRNL
                          0395     ;
                          0395     ; OUTPUTS:
                          0395     ;
                          0395     ;       R0 = STATUS OF THE OPERATION
                          0395     ;
                          0395     ; STATUS CODES RETURNED:
                          0395     ;
                          0395     ;       SS$_NORMAL - SUCCESSFUL OPERATION
                          0395     ;       SS$_INSFMEM - INSUFFICIENT MEMORY FOR THE REQUEST
                          0395     ;       SS$_ACCVIO - ACCESS VIOLATION ON BUFFER
                          0395     ;       SS$_NOPRIV - PROCESS DOES NOT HAVE REQUIRED PRIVILEGE
                          0395     ;--
                          0395     EXE$SNDERR::                                    ; SEND TO ERROR LOGGER
                   003C   0395          .WORD   ^M<R2,R3,R4,R5>                 ; REGISTERS USED
          50   24    B0   0397          MOVW    #SS$_NOPRIV,R0                  ; ASSUME SUER HAS NO PRIVILEGE
                          039A          IFNPRIV BUGCHK,30$                      ; BR IF NO PRIVILEGE
               27    DD   03A0          PUSHL   S^#EMB$C_SS                     ; SET MESSAGE TYPE CODE
                          03A2     ;
                          03A2     ; 0(SP) = MESSAGE TYPE CODE
                          03A2     ;
       51   04 AC    D0   03A2     10$:  MOVL    MSG(AP),R1                      ; GET ADDRESS OF MESSAGE DESCRIPTOR
      00000000'GF    16   03A6          JSB     G^EXE$PROBER_DSC                ; CHECK ACCESS TO DESCRIPTOR AND BUFFER
            35 50    E9   03AC          BLBC    R0,30$                          ; BR IF CAN'T READ DESCRIPTOR OR BUFFER
                          03AF     ;
                          03AF     ; R1<0:15> = SIZE, R2 = ADDRESS OF BUFFER
                          03AF     ;
          51   51    3C   03AF          MOVZWL  R1,R1                           ; GET SIZE OF MESSAGE AS A WORD
          54   51    7D   03B2          MOVQ    R1,R4                           ; R4 = SIZE, R5 = ADDRESS OF BUFFER
          51   15    C0   03B5          ADDL    #EMB$K_SS_LENGTH+3,R1           ; SET SIZE OF MESSAGE BUFFER NEEDED
          51   03    CA   03B8          BICL    #3,R1                           ; MODULO 4 BYTES
      00000000'GF    16   03BB          JSB     G^ERL$ALLOCEMB                  ; ATTEMPT TO ALLOCATE A BUFFER
            1B 50    E9   03C1          BLBC    R0,20$                          ; BR IF CAN'T ALLOCATE BUFFER
       04 A2   8E    F7   03C4          CVTLW   (SP)+,EMB$W_SS_ENTRY(R2)  ; STORE MESSAGE TYPE
       10 A2   54    B0   03C8          MOVW    R4,EMB$W_SS_MSGSZ(R2)     ; STORE LENGTH OF DATA MESSAGE
               52    DD   03CC          PUSHL   R2                              ; SAVE STARTING ADDRESS OF BUFFER
   12 A2   65  54    28   03CE          MOVC    R4,(R5),EMB$B_SS_MSGTXT(R2)  ; INSERT USERS DATA INTO THE BUFFER
               04    BA   03D3          POPR    #^M<R2>                         ; RESTORE START OF MESSAGE
      00000000'GF    16   03D5          JSB     G^ERL$RELEASEMB                 ; RELEASE THE BUFFER TO THE ERROR LOGGER
          50   01    D0   03DB          MOVL    S^#SS$_NORMAL,R0                ; SET SUCCESSFUL STATUS
                     04   03DE          RET                                     ; SUCCESSFUL RETURN
   50     0124 8F    3C   03DF     20$:  MOVZWL  #SS$_INSFMEM,R0                 ; SET INSUFFICIENT MEMORY FLAG
                     04   03E4     30$:  RET                                     ; ERROR RETURN
```

```
     03E5   824          .DSABL  LSB
     03E5   825
```

```
                          03E5              .SBTTL  SETOPR - set OPR bit in device UCB
                          03E5      ;++
                          03E5      ; EXE$SETOPR
                          03E5      ;
                          03E5      ; Functional Descripton:
                          03E5      ;
                          03E5      ;       This routine will set or clear the OPR bit in a
                          03E5      ;       terminal, remote terminal, or mailbox UCB.
                          03E5      ;
                          03E5      ; Input:
                          03E5      ;
                          03E5      ;       DEVNAM(AP)      = Address of device name descriptor
                          03E5      ;       BIT_STATE(AP)   = Value of the OPR bit.  Must be 0 or 1.
                          03E5      ;
                          03E5      ; Implicit Inputs:
                          03E5      ;
                          03E5      ;       The caller has OPER privilege.
                          03E5      ;
                          03E5      ; Output:
                          03E5      ;
                          03E5      ;       None.
                          03E5      ;
                          03E5      ; Implict Outputs:
                          03E5      ;
                          03E5      ;       R1 = address of device UCB
                          03E5      ;
                          03E5      ; Routine value:
                          03E5      ;
                          03E5      ;       R0 = The status of the operation.  Possible values listed below.
                          03E5      ;
                          03E5      ;       SS$_NORMAL      - The operation was completed.
                          03E5      ;
                          03E5      ;       SS$_ACCVIO      - The device name descriptor could not be accessed.
                          03E5      ;       SS$_BADPARAM    - The bit state was not 0 or 1.
                          03E5      ;       SS$_IVDEVNAM    - The device name specified was not valid.
                          03E5      ;       SS$_NONLOCAL    - The specified device is not local.
                          03E5      ;       SS$_NOPRIV      - The caller does not have OPER privilege.
                          03E5      ;       SS$_NOSUCHDEV   - The specified device does not exist.
                          03E5      ;--
                          03E5
                          03E5      ;
                          03E5      ; Local symbols
                          03E5      ;
                          03E5
              00000004    03E5      DEVNAM = 4                              ; Offset to device descriptor parameter
              00000008    03E5      BIT_STATE = 8                           ; Offset to OPR bit state parameter
                          03E5
                  0FFC    03E5              .ENTRY  EXE$SETOPR,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                          03E7      ;
                          03E7      ; Check the input parameters.
                          03E7      ;
                          03E7      ;
                          03E7      ; Make sure the caller has OPER privilege.
                          03E7      ;
          50    24    3C  03E7              MOVZWL  #SS$_NOPRIV,R0          ; Assume insufficeint privilege
   54    00000000'GF  D0  03EA              MOVL    G^CTL$GL_PCB,R4         ; Get current process PCB address
                          03F1              IFNPRIV OPER,13$                ; Branch if no OPER privilege
```

```
                                 03F7     ;
                                 03F7     ; PROBE the device name descriptor for read access.
                                 03F7     ; The actual device name string is PROBEd by LOG$TRNSLOGNAME
                                 03F7     ; during processing by IOC$SEARCHDEV.
                                 03F7     ;
          53    04 AC      D0    03F7             MOVL    DEVNAM(AP),R3           ; Get the device descriptor address
             50    0C      3C    03FB             MOVZWL  #SS$_ACCVIO,R0          ; Assume not readable
                                 03FE             IFNORD  #8,(R3),13$             ; Check descriptor for readability
                                 0404     ;
                                 0404     ; Check the bit_state parameter.
                                 0404     ;
             50    14      3C    0404             MOVZWL  #SS$_BADPARAM,R0        ; Assume value not 0 or 1
    08 AC    FFFFFFFE 8F   D3    0407             BITL    #^C1,BIT_STATE(AP)      ; Test for all but low bit set
                   01      13    040F             BEQL    20$                     ; Branch if yes
                           04    0411     13$:    RET                             ; Exit with error
                                 0412     ;
                                 0412     ; Lock the I/O database for read access and search for the device.
                                 0412     ; If the device exists, and it is a terminal, remote terminal or
                                 0412     ; mailbox, then set the OPR bit as indicated.  A side effect of
                                 0412     ; locking the I/O database is that the IPL is raised to IPL$_ASTDEL.
                                 0412     ;
          00000000'GF      16    0412     20$:    JSB     G^SCH$IOLOCKR           ; Lock I/O database for read access
             51    53      D0    0418             MOVL    R3,R1                   ; Get device name descriptor address
          00000000'GF      16    041B             JSB     G^IOC$SEARCHDEV         ; Search for the device
                23 50      E9    0421             BLBC    R0,UNLOCK               ; Branch if error
                                 0424     ;
                                 0424     ; Check the device type.  R1 now contains the device UCB address.
                                 0424     ;
         50    0144 8F     3C    0424             MOVZWL  #SS$_IVDEVNAM,R0        ; Assume invalid device
          00100004 8F      D3    0429             BITL    #DEV$M_TRM!DEV$M_MBX,-  ; Check device type
                38 A1            042F                     UCB$L_DEVCHAR(R1)       ;
                   14      13    0431             BEQL    UNLOCK                  ; Branch if not an operator type device
                                 0433     ;
                                 0433     ; Set the OPR bit as indicated.
                                 0433     ;
             50    01      3C    0433             MOVZWL  #SS$_NORMAL,R0          ; Set normal return status
                08 AC      D5    0436             TSTL    BIT_STATE(AP)           ; Set or clear?
                   07      13    0439             BEQL    30$                     ; Branch if clear desired
                   07      E2    043B             BBSS    #DEV$V_OPR,-            ; Set the OPR bit
             00 38 A1            043D                     UCB$L_DEVCHAR(R1),25$   ;
                   05      11    0440     25$:    BRB     UNLOCK                  ; Exit
                   07      E5    0442     30$:    BBCC    #DEV$V_OPR,-            ; Clear the OPR bit
             00 38 A1            0444                     UCB$L_DEVCHAR(R1),UNLOCK;
                   50      DD    0447     UNLOCK: PUSHL   R0                      ; Save return status
          00000000'GF      16    0449             JSB     G^SCH$IOUNLOCK          ; Unlock the I/O database
                                 044F             SETIPL  #0                      ; Allow all interrupts
             50    8E      D0    0452             MOVL    (SP)+,R0                ; Restore return status
                           04    0455             RET                             ; Return
                                 0456
                                 0456             .END
```

```
                                                                 C 15
SYSSNDMSG                         - SEND MESSAGE SYSTEM SERVICES          16-SEP-1984 02:35:36  VAX/VMS Macro V04-00       Page  23
Symbol table                                                               5-SEP-1984 03:57:44  [SYS.SRC]SYSSNDMSG.MAR;1        (9)

$$T1                            = 00000001              FIB$L_ALT_ACCESS           = 0000003C
ACM$B_PROCPRI                   = 00000024              FIB$L_STATUS               = 00000038
ACM$L_OWNER                     = 00000030              FIB$M_ALT_REQ              = 00000001
ACM$L_PID                       = 00000028              FIB$M_FINDFID              = 00000800
ACM$L_STS                       = 0000002C              FIB$W_DID                  = 0000000A
ACM$L_UIC                       = 0000000C              FIB$W_FID                  = 00000004
ACM$Q_PRVMSK                    = 00000004              FIB$W_NMCTL                = 00000014
ACM$Q_SYSTIME                   = 0000003C              FWA_ATRLIST                = 00000000
ACM$T_ACCOUNT                   = 0000001C              FWA_CHAN                   = 00000054
ACM$T_TERMINAL                  = 00000034              FWA_DID                    = 000001A6
ACM$T_USERNAME                  = 00000010              FWA_DVI                    = 00000190
ACM$W_MAILBOX                   = 00000002              FWA_DVI_DESC               = 00000056
ACM$W_TYPE                      = 00000000              FWA_FIB                    = 00000014
ARM$M_DELETE                    = 00000008              FWA_FIB_DESC               = 0000005E
ATR$C_FILE_SPEC                 = 0000002E              FWA_FID                    = 000001A0
ATR$C_RECATTR                   = 00000004              FWA_FILE_SIZE              = 000001AC
ATR$S_RECATTR                   = 00000020              FWA_FILE_SPEC              = 0000008E
BADPARAM                          00000079 R      01    FWA_FSPC_LEN               = 0000018E
BIT_STATE                       = 00000008              FWA_IOSB                   = 00000066
BUILDMSG                          0000007D R      01    FWA_RECATTR                = 0000006E
CCB$L_UCB                       = 00000000              FWA_SIZE                   = 000001B4
COMMON_HDR                      = 00000026              FWA_SPARE                  = 000001B0
CTL$AL_STACK                      ********   X    01    IO$_ACCESS                 = 00000032
CTL$GL_PCB                        ********   X    01    IOC$GW_MAXBUF                ********   X    01
CTL$T_USERNAME                    ********   X    01    IOC$SEARCHDEV                ********   X    01
DEV$M_MBX                       = 00100000              IOC$VERIFYCHAN               ********   X    01
DEV$M_TRM                       = 00000004              IPL$_ASTDEL                = 00000002
DEV$V_MBX                       = 00000014              IPL$_SYNCH                 = 00000008
DEV$V_OPR                       = 00000007              MBX                        = 00000008
DEVNAM                          = 00000004              MSG                        = 00000004
EMB$B_SS_MSGTXT                 = 00000012              MSG$_OPRQST                = 00000008
EMB$C_SS                        = 00000027              MSG$_SNDACC                = 0000000A
EMB$K_NM                        = 0000002A              MSG$_SNDSMB                = 00000004
EMB$K_OM                        = 00000029              OPC$_NOPERATOR             = 00058061
EMB$K_SS_LENGTH                 = 00000012              OPTION_SIZE                  00000000 R      01
EMB$W_SS_ENTRY                  = 00000004              PCB$B_PRIB                 = 0000002F
EMB$W_SS_MSGSZ                  = 00000010              PCB$L_EOWNER               = 00000068
ERL$ALLOCEMB                      ********   X    01    PCB$L_EPID                 = 00000064
ERL$RELEASEMB                     ********   X    01    PCB$L_PHD                  = 0000006C
ERROR                             0000007C R      01    PCB$L_STS                  = 00000024
EXE$CHKRDACCES                    ********   X    01    PCB$L_UIC                  = 000000BC
EXE$CHKWRTACCES                   ********   X    01    PCB$Q_PRIV                 = 00000084
EXE$GQ_SYSTIME                    ********   X    01    PCB$T_TERMINAL             = 00000044
EXE$NETSNDERL                     0000038F RG     01    PCB$V_SSRWAIT              = 0000000A
EXE$OPRSNDERL                     00000389 RG     01    PHD$Q_PRIVMSK              = 00000000
EXE$PROBER_DSC                    ********   X    01    PR$_IPL                    = 00000012
EXE$SENDMSG                       00000328 RG     01    PRV$V_BUGCHK               = 00000017
EXE$SETOPR                        000003E5 RG     01    PRV$V_OPER                 = 00000012
EXE$SNDACC                        00000040 RG     01    RSN$_NPDYNMEM              = 00000003
EXE$SNDERR                        00000395 RG     01    SCH$GL_CURPCB                ********   X    01
EXE$SNDOPR                        00000066 RG     01    SCH$IOLOCKR                  ********   X    01
EXE$SNDSMB                        00000053 RG     01    SCH$IOUNLOCK                 ********   X    01
EXE$WRTMAILBOX                    ********   X    01    SCH$RWAIT                    ********   X    01
FAT$L_EFBLK                     = 00000008              SENDMSG                      000002DA R      01
FAT$W_FFBYTE                    = 0000000C              SMO$K_DELETE               = 00000001
FIB$C_LENGTH                    = 00000040              SMO$K_PARAMS               = 00000026
FIB$L_ACCTL                     = 00000000              SMR$K_ADDFIL               = 0000000A
```

```
SMR$K_ENTER                      = 00000008
SS$_ACCVIO                       = 0000000C
SS$_BADPARAM                     = 00000014
SS$_DEVNOTMBX                    = 00000074
SS$_DEVOFFLINE                   = 00000084
SS$_INSFMEM                      = 00000124
SS$_IVDEVNAM                     = 00000144
SS$_NOPRIV                       = 00000024
SS$_NORMAL                       = 00000001
SYS$ASSIGN                         ********   GX     01
SYS$CMKRNL                         ********   GX     01
SYS$DASSGN                         ********   GX     01
SYS$GL_JOBCTLMB                    ********    X     01
SYS$GL_OPRMBX                      ********    X     01
SYS$QIOW                           ********   GX     01
UCB$L_DEVCHAR                    = 00000038
UCB$W_REFC                       = 0000005C
UCB$W_UNIT                       = 00000054
UNLOCK                             00000447 R        01

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

PSECT name                        Allocation           PSECT No.  Attributes
----------                        ----------           ---------  ----------
.  ABS  .                         00000000  (      0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
Y$EXEPAGED                        00000456  (   1110.)  01 (   1.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$ABS$                             00000000  (      0.)  02 (   2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE

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

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                       35    00:00:00.07    00:00:00.39
Command processing                  128    00:00:00.57    00:00:01.20
Pass 1                              561    00:00:23.79    00:00:27.89
Symbol table sort                     0    00:00:03.95    00:00:04.17
Pass 2                              175    00:00:04.49    00:00:05.03
Symbol table output                  17    00:00:00.15    00:00:00.15
Psect synopsis output                 1    00:00:00.02    00:00:00.02
Cross-reference output                0    00:00:00.00    00:00:00.00
Assembler run totals                919    00:00:33.04    00:00:38.85

The working set limit was 1950 pages.
134934 bytes (264 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2544 non-local and 31 local symbols.
933 source lines were read in Pass 1, producing 20 object records in Pass 2.
43 pages of virtual memory were used to define 42 macros.
```

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

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         15
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      24
TOTALS (all libraries)                                 39

2757 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

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

0388 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYSSETPRA
LIS

SYSSETSSF
LIS

SYSSNDJBC
LIS

SYSSETIME
LIS

SYSSETPFM
LIS

SYSUPDSEC
LIS

SYSSNDMSG
LIS

SYSSETPRT
LIS

SYSSETPRV
LIS

SYSUNWIND
LIS

SYSSETSTK
LIS

SYSSETMOD
LIS

SYSSETEXV
LIS

SYSSETPRI
LIS

SYSSCHEVT
LIS