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$
```

```
**FILE**ID**SYSSNDJBC                                    B 11
```

```
  SSSSSSSS  YY      YY    SSSSSSSS    SSSSSSSS  NN      NN  DDDDDDDD            JJ  BBBBBBBB      CCCCCCCC
  SSSSSSSS  YY      YY    SSSSSSSS    SSSSSSSS  NN      NN  DDDDDDDD            JJ  BBBBBBBB      CCCCCCCC
SS          YY      YY  SS          SS          NN      NN  DD      DD          JJ  BB      BB  CC
SS          YY      YY  SS          SS          NN      NN  DD      DD          JJ  BB      BB  CC
SS            YY  YY    SS          SS          NNNN    NN  DD      DD          JJ  BB      BB  CC
SS            YY  YY    SS          SS          NNNN    NN  DD      DD          JJ  BB      BB  CC
  SSSSSS        YY        SSSSSS      SSSSSS    NN  NN  NN  DD      DD          JJ  BBBBBBBB    CC
  SSSSSS        YY        SSSSSS      SSSSSS    NN  NN  NN  DD      DD          JJ  BBBBBBBB    CC
        SS      YY              SS          SS  NN    NNNN  DD      DD  JJ      JJ  BB      BB  CC
        SS      YY              SS          SS  NN    NNNN  DD      DD  JJ      JJ  BB      BB  CC
        SS      YY              SS          SS  NN      NN  DD      DD  JJ      JJ  BB      BB  CC              ....
        SS      YY              SS          SS  NN      NN  DD      DD  JJ      JJ  BB      BB  CC              ....
SSSSSSSS        YY      SSSSSSSS    SSSSSSSS    NN      NN  DDDDDDDD      JJJJJ     BBBBBBBB      CCCCCCCC      ....
SSSSSSSS        YY      SSSSSSSS    SSSSSSSS    NN      NN  DDDDDDDD      JJJJJ     BBBBBBBB      CCCCCCCC      ....

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
```

SYS
V04-

Table of contents

```
0000             .TITLE  SYSSNDJBC - SEND MESSAGE TO JOB CONTROLLER
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      ;       System services.
0000      ;
0000      ; ABSTRACT:
0000      ;       This module implements the Send to Job Controller ($SNDJBC) and the
0000      ;       Get Queue Information ($GETQUI) system services.
0000      ;
0000      ; AUTHOR: M. Jack, CREATION DATE: 29-Aug-1982
0000      ;
0000      ; MODIFIED BY:
0000      ;
0000      ;       V03-011 JAK0218         J A Krycka      10-Jul-1984
0000      ;               Update tables to support new $SNDJBC and $GETQUI item codes.
0000      ;
0000      ;       V03-010 JAK0203         J A Krycka      17-Apr-1984
0000      ;               Update tables to support new $SNDJBC item codes.
0000      ;
0000      ;       V03-009 TMK0001         Todd M. Katz    04-Apr-1984
0000      ;               Re-write the action routine TRANSLATE_OBJECT to:
0000      ;
0000      ;                  1. Replace the recursive $TRNLOGs with $TRNLNMs.
0000      ;
0000      ;                  2. Eliminate the code that removes tabs, blanks, and null
0000      ;                     characters from names before attempting to translate them.
0000      ;                     Logical names should be handled in a systematic fashion
0000      ;                     throughout the system, and nobody else fiddles with them in
0000      ;                     such a fashion. However, after the recursive translations
0000      ;                     complete, at this time format the final translation according
```

```
0000    ;                     to the syntax expected for queue names. This involves
0000    ;                     removing tabs, null characters, and spaces from the final
0000    ;                     translation, and then upcasing it. This upcasing is done by
0000    ;                     means of the DEC multi-national character upcasing table.
0000    ;
0000    ;                  3. Eliminate the code that upcases names before their
0000    ;                     translation because the $TRNLNMs will be done
0000    ;                     case-insensitive.
0000    ;
0000    ;                  4. Micro-optimize the action routine.
0000    ;
0000    ;          V03-008 ACG0354         Andrew C. Goldstein,    13-Sep-1983
0000    ;                  Change delete protection check to use alternate access
0000    ;                  rather than access-granted.
0000    ;
0000    ;          V03-007 MLJ0118         Martin L. Jack, 22-Aug-1983
0000    ;                  Guard against overlong resultant filename.  Update tables and
0000    ;                  limits for new $GETQUI and $SNDJBC items.
0000    ;
0000    ;          V03-006 MLJ0115         Martin L. Jack, 30-Jul-1983
0000    ;                  Changes for job controller baselevel.
0000    ;
0000    ;          V03-005 MLJ0114         Martin L. Jack, 23-Jun-1983
0000    ;                  Add support for $GETQUI and for new $SNDJBC items.
0000    ;
0000    ;          V03-004 MLJ0112         Martin L. Jack, 28-Apr-1983
0000    ;                  Update tables and limits for new items corresponding to job
0000    ;                  controller baselevel.
0000    ;
0000    ;          V03-003 CWH1002         CW Hobbs        24-Feb-1983
0000    ;                  Send extended pid and owner fields to the job controller.
0000    ;
0000    ;          V03-002 MLJ0106         Martin L. Jack, 1-Mar-1983
0000    ;                  Update tables and limits for new items corresponding to job
0000    ;                  controller baselevel.
0000    ;
0000    ;          V03-001 MLJ0103         Martin L. Jack, 7-Jan-1983
0000    ;                  Update tables and limits for new items corresponding to job
0000    ;                  controller baselevel.
0000    ;
0000    ;--
```

```
                 0000          .SBTTL  DATA DEFINITIONS
                 0000
                 0000   ;
                 0000   ; EXTERNAL SYMBOLS:
                 0000   ;
                 0000
                 0000          $ACBDEF                  ; Define AST control block offsets
                 0000          $ACMDEF                  ; Define accounting manager offsets
                 0000          $ARMDEF                  ; Define access rights mask
                 0000          $ATRDEF                  ; Define ACP attribute codes
                 0000          $FABDEF                  ; Define RMS file attributes block offsets
                 0000          $FATDEF                  ; Define RMS file attribute area offsets
                 0000          $FIBDEF                  ; Define file information block offsets
                 0000          $IODEF                   ; Define I/O function codes
                 0000          $LNMDEF                  ; Define logical name system service symbols
                 0000          $MSGDEF                  ; Define mailbox message types
                 0000          $NAMDEF                  ; Define RMS name block offsets
                 0000          $PCBDEF                  ; Define process control block offsets
                 0000          $PHDDEF                  ; Define process header offsets
                 0000          $PSLDEF                  ; Define processor status longword offsets
                 0000          $QUIDEF                  ; Define $GETQUI function and item codes
                 0000          $SJCDEF                  ; Define $SNDJBC function and item codes
                 0000          $SSDEF                   ; Define system status codes
                 0000
                 0000   ;
                 0000   ; LOCAL SYMBOLS:
                 0000   ;
                 0000   ; Extension to AST control block.  These definitions are also known to JOBCTL.
                 0000   ;
                 0000
                 0000          $DEFINI ACB
        0000001C 0000          .=ACB$L_KAST+4                   ; Position to end
                 001C   $DEF   ACB_L_IMGCNT    .BLKL   1        ; Image counter
                 0020   $DEF   ACB_L_EFN       .BLKL   1        ; Event flag number
                 0024   $DEF   ACB_L_IOSB      .BLKL   1        ; IOSB address
                 0028   $DEF   ACB_L_STATUS    .BLKL   1        ; Status for IOSB
                 002C   $DEF   ACB_W_ITEMCOUNT .BLKW   1        ; Count of data items
                 002E   $DEF   ACB_B_ITEMS     .BLKB   0        ; Base of item descriptors
                 002E          $DEFEND ACB
                 0000
                 0000   ;
                 0000   ; Argument list offsets (based on AP).
                 0000   ;
                 0000
        00000004 0000   EFN=           4                        ; Event flag number
        00000008 0000   FUNC=          8                        ; Function code
        0000000C 0000   NULARG=        12                       ; Reserved argument
        00000010 0000   ITMLST=        16                       ; Address of item descriptor list
        00000014 0000   IOSB=          20                       ; Address of I/O status block
        00000018 0000   ASTADR=        24                       ; Address of AST routine
        0000001C 0000   ASTPRM=        28                       ; AST parameter
                 0000
                 0000   ;
                 0000   ; Data table offsets (based on R11).
                 0000   ;
                 0000
        00000000 0000   MSG_CODE=      0                        ; Message function code
```

```
00000004  0000  MAX_FUNC=      4                      ; Highest function code minus one
00000008  0000  MAX_ITEM=      8                      ; Highest item code minus one
0000000C  0000  BOOL_ITEM=     12                     ; Pointer to boolean item table
00000010  0000  OUTPUT_ITEM=   16                     ; Pointer to output item table
00000014  0000  SPECIAL_TABLE= 20                     ; Pointer to special handling table
          0000
          0000  ;
          0000  ; Fixed work area offsets (above FP).
          0000  ;
          0000
FFFFFFFC  0000  FILE_ID=       -4                     ; Pointer to file identification item
FFFFFFF8  0000  FLAGS=         -8                     ; Pointer to miscellaneous status flags
00000000  0000  DELETE_FLAG=   0                      ; SJC$_DELETE_FILE seen
          0000
          0000  ;
          0000  ; Miscellaneous definitions.
          0000  ;
          0000
0000005A  0000  FIXED_AREA=    ACM$Q_SYSTIME+8 + 22   ; Size of fixed area of message
```

```
                0000
                0000    ;
                0000    ; LOCAL STORAGE:
                0000    ;
                0000    ; This table is indexed by item code (normalized to zero origin).  It
                0000    ; identifies items classified as Boolean.
                0000    ;
                0000
            00000000            .PSECT  Y$EXEPAGED
                0000    SNDJBC_BOOL_ITEM:
      99EAA198  0000            .LONG   ^B10011001111010101010000110011000
      B7005D3B  0004            .LONG   ^B10110111000000000101110100111011
      EB751BDF  0008            .LONG   ^B11101011011101010001101111011111
      C03B5DFC  000C            .LONG   ^B11000000001110110101110111111100
      0A9F3F09  0010            .LONG   ^B00001010100111110011111100001001
      0000001E  0014            .LONG   ^B00000000000000000000000000011110
      00000000  0018            .LONG   ^B00000000000000000000000000000000
      00000000  001C            .LONG   ^B00000000000000000000000000000000
                0020    GETQUI_BOOL_ITEM:
      00000000  0020            .LONG   ^B00000000000000000000000000000000
      00000000  0024            .LONG   ^B00000000000000000000000000000000
      00300000  0028            .LONG   ^B00000000001100000000000000000000
      00000000  002C            .LONG   ^B00000000000000000000000000000000
      00000000  0030            .LONG   ^B00000000000000000000000000000000
      00000000  0034            .LONG   ^B00000000000000000000000000000000
      00000000  0038            .LONG   ^B00000000000000000000000000000000
      00000000  003C            .LONG   ^B00000000000000000000000000000000
                0040
                0040    ;
                0040    ; This table is indexed by item code (normalized to zero origin).  It
                0040    ; identifies items classified as output.
                0040    ;
                0040
                0040    SNDJBC_OUTPUT_ITEM:
      40000400  0040            .LONG   ^B01000000000000000000010000000000
      00000000  0044            .LONG   ^B00000000000000000000000000000000
      00800000  0048            .LONG   ^B00000000100000000000000000000000
      00000000  004C            .LONG   ^B00000000000000000000000000000000
      00000000  0050            .LONG   ^B00000000000000000000000000000000
      00000600  0054            .LONG   ^B00000000000000000000011000000000
      00000000  0058            .LONG   ^B00000000000000000000000000000000
      00000000  005C            .LONG   ^B00000000000000000000000000000000
                0060    GETQUI_OUTPUT_ITEM:
      FFFFFFFF  0060            .LONG   ^B11111111111111111111111111111111
      FFFFFFFF  0064            .LONG   ^B11111111111111111111111111111111
      3F0FC7FF  0068            .LONG   ^B00111111000011111100011111111111
      00000000  006C            .LONG   ^B00000000000000000000000000000000
      00000000  0070            .LONG   ^B00000000000000000000000000000000
      00000000  0074            .LONG   ^B00000000000000000000000000000000
      00000000  0078            .LONG   ^B00000000000000000000000000000000
      00000000  007C            .LONG   ^B00000000000000000000000000000000
                0080
                0080    ;
                0080    ; This table identifies item codes that require special translation and the
                0080    ; routine that performs the translation.
                0080    ;
                0080
```

```
                                  0080   SNDJBC_SPECIAL_TABLE:
                       000C       0080           .WORD   SJC$_CHARACTERISTIC_NAME
                  000004B0'       0082           .LONG   TRANSLATE_OBJECT
                       001A       0086           .WORD   SJC$_DESTINATION_QUEUE
                  000004B0'       0088           .LONG   TRANSLATE_OBJECT
                       0027       008C           .WORD   SJC$_FILE_IDENTIFICATION
                  00000369'       008E           .LONG   FILE_IDENTIFICATION
                       002A       0092           .WORD   SJC$_FILE_SPECIFICATION
                  000002F1'       0094           .LONG   FILE_SPECIFICATION
                       0036       0098           .WORD   SJC$_FORM_NAME
                  000004B0'       009A           .LONG   TRANSLATE_OBJECT
                       0046       009E           .WORD   SJC$_GENERIC_TARGET
                  000004B0'       00A0           .LONG   TRANSLATE_OBJECT
                       0061       00A4           .WORD   SJC$_LOG_QUEUE
                  000004B0'       00A6           .LONG   TRANSLATE_OBJECT
                       0086       00AA           .WORD   SJC$_QUEUE
                  000004B0'       00AC           .LONG   TRANSLATE_OBJECT
                       0000       00B0           .WORD   0
                                  00B2   GETQUI_SPECIAL_TABLE:
                       004D       00B2           .WORD   QUI$_SEARCH_NAME
                  000004B0'       00B4           .LONG   TRANSLATE_OBJECT
                       0000       00B8           .WORD   0
                                  00BA
                                  00BA   ;
                                  00BA   ; The following values are needed as arguements to the $TRNLNMs performed by
                                  00BA   ; the action routine TRANSLATE_OBJECT.
                                  00BA   ;
                                  00BA
                                  00BA   TRNLNM_ATTR:                                    ; Optional attributes for $TRNLNMs
                   02000000       00BA           .LONG   LNM$M_CASE_BLIND                ; Translations are done case-insensitive
                                  00BE
                                  00BE   TRNLNM_TABLE:                                   ; Tables in which to do the translations
49 46 24 4D 4E 4C 000000C6'010E0000'  00BE       .ASCID  /LNM$FILE_DEV/
            56 45 44 5F 45 4C     00CC
```

```
                    00D2          .SBTTL  EXE$SNDJBC - Send message to job controller
                    00D2
                    00D2    ;++
                    00D2    ;
                    00D2    ; EXE$SNDJBC - SEND MESSAGE TO JOB CONTROLLER
                    00D2    ; EXE$GETQUI - GET QUEUE INFORMATION
                    00D2    ;
                    00D2    ; FUNCTIONAL DESCRIPTION:
                    00D2    ;
                    00D2    ;         This routine provides the send to job controller and get queue
                    00D2    ;         information system services.  The action is to build a message from the
                    00D2    ;         user's input data and send it to the job controller mailbox.  At request
                    00D2    ;         completion, the job controller queues a special kernel AST to routine
                    00D2    ;         EXE$JBCRSP to return status to this process.
                    00D2    ;
                    00D2    ; INPUTS:
                    00D2    ;         EFN(AP)          = Number of the event flag to set at request completion
                    00D2    ;         FUNC(AP)         = Function code
                    00D2    ;         NULARG(AP)       = Reserved argument, must be zero
                    00D2    ;         ITMLST(AP)       = Address of a list of item descriptors
                    00D2    ;         IOSB(AP)         = Address of a quadword to receive completion status
                    00D2    ;         ASTADR(AP)       = Address of an AST routine to be called at request
                    00D2    ;                            completion
                    00D2    ;         ASTPRM(AP)       = Longword AST parameter
                    00D2    ;
                    00D2    ; OUTPUTS:
                    00D2    ;         R0               = Status of the operation
                    00D2    ;
                    00D2    ; STATUS CODES RETURNED:
                    00D2    ;         SS$_NORMAL       Successful operation
                    00D2    ;
                    00D2    ;         SS$_ACCVIO       Unable to write IOSB, read ITMLST, read or write item
                    00D2    ;                          buffer, write return length buffer
                    00D2    ;         SS$_BADPARAM     Invalid FUNC, nonzero NULARG, invalid item code,
                    00D2    ;                          invalid zero or nonzero field in item descriptor
                    00D2    ;         SS$_DEVOFFLINE   No job controller
                    00D2    ;         SS$_EXASTLM      Exceeded ASTLM quota
                    00D2    ;         SS$_ILLEFC       Illegal event flag number
                    00D2    ;         SS$_INSFMEM      Insufficient system memory to complete request
                    00D2    ;         SS$_MBFULL       Job controller mailbox full
                    00D2    ;         SS$_MBTOOSML     Message too large for job controller mailbox
                    00D2    ;         SS$_UNASEFC      Unassociated event flag cluster
                    00D2    ;--
                    00D2
                    00D2    GETQUI_DATA:
        00000010    00D2          .LONG   MSG$_GETQUI               ; Message function code
        00000008    00D6          .LONG   QUI$_RESERVED_FUNC_2-1    ; Highest function code minus one
        0000005D    00DA          .LONG   QUI$_RESERVED_OUTPUT_6-1; Highest item code minus one
        00000020'   00DE          .LONG   GETQUI_BOOL_ITEM          ; Pointer to boolean item table
        00000060'   00E2          .LONG   GETQUI_OUTPUT_ITEM        ; Pointer to output item table
        000000B2'   00E6          .LONG   GETQUI_SPECIAL_TABLE      ; Pointer to special handling table
                    00EA
                    00EA    SNDJBC_DATA:
        0000000F    00EA          .LONG   MSG$_SNDJBC               ; Message function code
        0000001F    00EE          .LONG   SJC$_RESERVED_FUNC_2-1    ; Highest function code minus one
        000000AA    00F2          .LONG   SJC$_RESERVED_OUTPUT_2-1; Highest item code minus one
        00000000'   00F6          .LONG   SNDJBC_BOOL_ITEM          ; Pointer to boolean item table
```

```
                   00000040'  00FA          .LONG   SNDJBC_OUTPUT_ITEM      ; Pointer to output item table
                   00000080'  00FE          .LONG   SNDJBC_SPECIAL_TABLE    ; Pointer to special handling table
                              0102
                              0102          .ENABL  LSB
                              0102
                              0102  ACCVIO:
               50   0C   D0   0102          MOVL    #SS$_ACCVIO,R0          ; Set access violation status
                    0A   11   0105          BRB     10$
                              0107
                              0107  BADPARAM:
               50   14   D0   0107          MOVL    #SS$_BADPARAM,R0        ; Set bad parameter status
                    05   11   010A          BRB     10$
                              010C
                              010C  INSFMEM:
          50   0124 8F   3C   010C          MOVZWL  #SS$_INSFMEM,R0         ; Set insufficient memory status
                  0472   31   0111  10$:    BRW     ERROR
                              0114
                              0114
                              0114  EXE$GETQUI::                            ; Get queue information
                       0FFC   0114          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
               5B   B9 AF   9E   0116       MOVAB   GETQUI_DATA,R11         ; Point to $GETQUI data table
                    06   11   011A          BRB     20$                     ; Join common code
                              011C
                              011C
                              011C  EXE$SNDJBC::                            ; Send to job controller
                       0FFC   011C          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
               5B   C9 AF   9E   011E       MOVAB   SNDJBC_DATA,R11         ; Point to $SNDJBC data table
                              0122
                              0122  ;
                              0122  ; Point to lowest usable stack address.
                              0122  ;
                              0122
                    7E   7C   0122  20$:    CLRQ    -(SP)                   ; Allocate fixed work area
                    50   DC   0124          MOVPSL  R0                      ; Get PSL
          50   50   02   18   EF   0126     EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R0,R0 ; Get current mode
 00000000'EF40   0000008E 8F   C1   012B    ADDL3   #FIXED_AREA+52, -       ; Allow slop for fixed message area plus
                    5A        0137                  CTL$AL_STACKLIM[R0], -  ; 52 bytes for $CMKRNL frame and
                              0138                  R10                     ; parameters
                              0138
                              0138  ;
                              0138  ; Check for and clear I/O status block.
                              0138  ;
                              0138
               50   14 AC   D0   0138       MOVL    IOSB(AP),R0             ; Get IOSB address
                    08   13   013C          BEQL    30$                     ; Branch if none
                              013E          IFNOWRT #8,(R0),ACCVIO          ; Check write access to IOSB
                    60   7C   0144          CLRQ    (R0)                    ; Clear IOSB
                              0146
                              0146  ;
                              0146  ; Validate function code.
                              0146  ;
                              0146
          52   08 AC   01   C3   0146  30$:  SUBL3   #1,FUNC(AP),R2          ; Get function code and subtract out
                              014B                                          ; smallest value to get zero origin
               04 AB   52   D1   014B       CMPL    R2,MAX_FUNC(R11)        ; Check against largest value
                    B6   1A   014F          BGTRU   BADPARAM                ; Branch if invalid value
                              0151
```

```
                        0151  ;
                        0151  ; Validate unused argument (must be zero).
                        0151  ;
                        0151
         0C AC    D5    0151           TSTL    NULARG(AP)                 ; Unused argument zero?
            B1    12    0154           BNEQ    BADPARAM                   ; Branch if not zero
                        0156
                        0156  ;
                        0156  ; Process the item list to build the job controller message.  During this loop:
                        0156  ;
                        0156  ;        R5  = buffer size
                        0156  ;        R6  = item code
                        0156  ;        R7  = buffer address
                        0156  ;        R8  = return length address
                        0156  ;        R9  = pointer to item list
                        0156  ;        R10 = pointer to lowest available stack address
                        0156  ;        R11 = pointer to service-specific data area
                        0156  ;
                        0156
   59    10 AC    D0    J156           MOVL    ITMLST(AP),R9              ; Get item descriptor list address
            0E    13    015A           BEQL    40$                        ; Branch if no item list
                        015C           IFNORD  #4,(R9),90$                ; Check read access to first longword
      55    89    3C    0162  ITEM:    MOVZWL  (R9)+,R5                   ; Get uuffer size value
      56    89    3C    0165           MOVZWL  (R9)+,R6                   ; Get item code value
            03    12    ^168           BNEQ    50$                        ; Branch if nonzero, list not ended
          00B0    31    u16A  40$:     BRW     FINISH_MESSAGE             ; Branch if zero, list ended
   52    56       01    C3    016D  50$:     SUBL3   #1,R6,R2                   ; Subtract out smallest value to get
                        0171                                              ; zero origin
   08 AB    52    D1    0171           CMPL    R2,MAX_ITEM(R11)           ; Check against largest value
            59    1A    0175           BGTRU   100$                       ; Branch if invalid  alue
                        0177           IFNORD  #12,(R9),90$               ; Check read access to second and third
                        017D                                              ; longwords of this item and first
                        017D                                              ; longword of next item
      57    89    7D    017D           MOVQ    (R9)+,R7                   ; Get buffer address and return length
                        0180                                              ; address
                        0180  ;
                        0180  ; Boolean item.  Store the item code.
                        0180  ;
                        0180
17 0C BB    52    E1    0180           BBC     R2,@BOOL_ITEM(R11),60$     ; Branch if not boolean item
   50    5E    02    C3    0185           SUBL3   #2,SP,R0                   ; Get lowest address that will be used
      5A    50    D1    0189           CMPL    R0,R10                     ; Compare against that available
            45    1F    018C           BLSSU   110$                       ; Branch if space exceeded
      7E    56    B0    018E           MOVW    R6,-(SP)                   ; Store item code
      18    56    B1    0191           CMPW    R6,#SJC$_DELETE_FILE       ; Check for file deletion
            CC    12    0194           BNEQ    ITEM
   F8 AD    01    88    0196           BISB    #1@DELETE_FLAG,FLAGS(FP); Note file deletion for postprocessing
            C6    11    019A           BRB     ITEM                       ; Branch to process next item
                        019C
                        019C  ;
                        019C  ; Input or output item.  Set up to call EXE$PROBEx.
                        019C  ;
                        019C
      50    57    D0    019C  60$:     MOVL    R7,R0                      ; R0 = buffer address
      51    55    D0    019F           MOVL    R5,R1                      ; R1 = buffer length
            53    D4    01A2           CLRL    R3                         ; R3 = probe against previous mode
            55    DD    01A4           PUSHL   R5                         ; Save R5 across call
```

```
                            01A6
                            01A6    ;
                            01A6    ; Input item.  Ensure that the buffer is accessible.
                            01A6    ;
                            01A6
  47 10 BB   52     E0      01A6            BBS     R2,@OUTPUT_ITEM(R11),120$ ; Branch if output item
   00000000'EF      16      01AB            JSB     EXE$PROBER                ; Probe read access to buffer
          55      8ED0      01B1            POPL    R5                        ; Restore R5
       16 50        E9      01B4            BLBC    R0,90$                    ; Branch if no access
                            01B7
                            01B7    ;
                            01B7    ; Test for items that receive special translation.
                            01B7    ;
                            01B7
    50    14 AB     D0      01B7            MOVL    SPECIAL_TABLE(R11),R0     ; Point to special handling table
       51    60     3C      01BB    70$:    MOVZWL  (R0),R1                   ; Pick up item code and test if ended
             16     13      01BE            BEQL    INPUT_ITEM                ; Branch if ended
       51    56     D1      01C0            CMPL    R6,R1                     ; Correct item code?
             03     12      01C3            BNEQ    80$                       ; Branch if not
          02 B0     17      01C5            JMP     @2(R0)                    ; Jump to processing routine
       50    06     C0      01C8    80$:    ADDL2   #6,R0                     ; Increment to next table entry
             EE     11      01CB            BRB     70$                       ; Loop to compare next
                            01CD
                            01CD    ;
                            01CD    ; Helper branches.
                            01CD    ;
                            01CD
           FF32     31      01CD    90$:    BRW     ACCVIO
           FF34     31      01D0    100$:   BRW     BADPARAM
           FF36     31      01D3    110$:   BRW     INSFMEM
                            01D6
                            01D6    ;
                            01D6    ; Ordinary input item.  Store the item code, buffer length, and contents.
                            01D6    ;
                            01D6
                            01D6    INPUT_ITEM:
    50    5E    55  C3      01D6            SUBL3   R5,SP,R0                  ; Get lowest address that will be used
          50    04  C2      01DA            SUBL2   #4,R0                     ;
          5A    50  D1      01DD            CMPL    R0,R10                    ; Compare against that available
                F1  1F      01E0            BLSSU   110$                      ; Branch if space exceeded
          5E    50  D0      01E2            MOVL    R0,SP                     ; Allocate the space
          80    56  B0      01E5            MOVW    R6,(R0)+                  ; Store item code
          80    55  B0      01E8            MOVW    R5,(R0)+                  ; Store item length
    60    67    55  28      01EB            MOVC3   R5,(R7),(R0)              ; Store item value
           FF70     31      01EF            BRW     ITEM                      ; Branch to process next item
                            01F2
                            01F2    ;
                            01F2    ; Output item.  Ensure that the buffer, and return length if specified, are
                            01F2    ; accessible, and store the item code, buffer length, buffer address, and return
                            01F2    ; length address.
                            01F2    ;
                            01F2
   00000000'EF      16      01F2    120$:   JSB     EXE$PROBEW                ; Probe write access to buffer
          55      8ED0      01F8            POPL    R5                        ; Restore R5
       CF 50        E9      01FB            BLBC    R0,90$                    ; Branch if no access
          58        D5      01FE            TSTL    R8                        ; Test if return length specified
          06        13      0200            BEQL    130$                      ; Branch if not specified
```

```
                         0202               IFNOWRT #2,(R8),90$             ; Probe write access to length word
      50    5E    0C  C3 0208       130$:   SUBL3   #12,SP,R0               ; Get lowest address that will be used
            5A    50  D1 020C               CMPL    R0,R10                  ; Compare against that available
                  C2  1F 020F               BLSSU   110$                    ; Branch if space exceeded
            7E    57  7D 0211               MOVQ    R7,-(SP)                ; Store item buffer addresses
            7E    55  B0 0214               MOVW    R5,-(SP)                ; Store item length
            7E    56  B0 0217               MOVW    R6,-(SP)                ; Store item code
                FF45  31 021A               BRW     ITEM                    ; Branch to process next item
                         021D
                         021D       ;
                         021D       ; To here when all items have been processed. Do necessary postprocessing
                         021D       ; and finish the message.
                         021D       ;
                         021D
                         021D       FINISH_MESSAGE:
         5A     FC AD  D0 021D              MOVL    FILE_ID(FP),R10         ; Get file ID item, if any
                   03  13 0221              BEQL    140$                    ; Branch if none
                 017C  30 0223              BSBW    POSTPROCESS_FID         ; Deal with it
                          0226
                          0226      ;
                          0226      ; Build the message header.
                          0226      ;
                          0226
  56    00000000'9F       D0 0226   140$:   MOVL    @#CTL$GL_PCB,R6         ; Get PCB address
  57    00000000'9F       D0 022D           MOVL    @#CTL$GL_PHD,R7         ; Get PHD address
            7E    08 AC   B0 0234           MOVW    FUNC(AP),-(SP)          ; Store function code
            7E    18 AC   7D 0238           MOVQ    ASTADR(AP),-(SP)        ; Store AST address and parameter
                  14 AC   DD 023C           PUSHL   IOSB(AP)                ; Store IOSB address
            7E    04 AC   9A 023F           MOVZBL  EFN(AP),-(SP)           ; Store event flag number
                00F4 C7   DD 0243           PUSHL   PHD$L_IMGCNT(R7)        ; Store image counter
                 5E   08  C2 0247           SUBL2   #8,SP                   ; Make space for system time
  6E    00000000'EF       7D 024A   150$:   MOVQ    EXE$GQ_SYSTIME,(SP)     ; Store current time
  6E    00000000'EF       D1 0251           CMPL    EXE$GQ_SYSTIME,(SP)     ; Verify that value acquired was not
                      F0  12 0258           BNEQ    150$                    ;  being modified at the same time
04 AE   00000004'EF       D1 025A           CMPL    EXE$GQ_SYSTIME+4,4(SP)  ;  and store it again if it changed
                      E6  12 0262           BNEQ    150$                    ;
            7E    44 A6   7D 0264           MOVQ    PCB$T_TERMINAL(R6),-(SP); Store terminal name
                  68 A6   DD 0268           PUSHL   PCB$L_EOWNER(R6)        ; Store extended owner process ID
                  24 A6   DD 026B           PUSHL   PCB$L_STS(R6)           ; Store process status
                  64 A6   DD 026E           PUSHL   PCB$L_EPID(R6)          ; Store extended process ID
                     7E   B4 0271           CLRW    -(SP)                   ; Clear spare word
                     50   DC 0273           MOVPSL  R0                      ; Get PSL
   50    50    02    16   EF 0275           EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R0,R0 ; Get previous mode
                7E   50   90 027A           MOVB    R0,-(SP)                ; Store requester's mode
      7E    1F    2F A6   83 027D           SUBB3   PCB$B_PRIB(R6),#31,-(SP); Store base priority
                5E   14   C2 0282           SUBL2   #20,SP                  ; Allocate space for next field
6E   00000000'9F     14   28 0285           MOVC3   #20,@#CTL$T_USERNAME,(SP) ; Store username and account name
                00BC C6   DD 028D           PUSHL   PCB$L_UIC(R6)           ; Store UIC
                7E   67   7D 0291           MOVQ    PHD$Q_PRIVMSK(R7),-(SP) ; Store privileges
                7E   6B   3C 0294           MOVZWL  MSG_CODE(R11),-(SP)     ; Store message type, clear mailbox
                          0297
                          0297      ;
                          0297      ; Finished building the message.  Push the address of the service argument
                          0297      ; list, and the address and length of the message, and enter kernel mode to
                          0297      ; complete argument list processing and write the message.
                          0297      ;
                          0297
                 5E   DD  0297             PUSHL   SP                          ; Push address of message
     7E    5D    6E   C3  0299             SUBL3   (SP),FP,-(SP)               ; Push length of message
           6E    08   C2  029D             SUBL    #8,(SP)                     ; Deduct fixed work area
                 5C   DD  02A0             PUSHL   AP                          ; Push service argument list
                          02A2             $CMKRNL_S B^170$,(SP)               ; Finish in kernel mode
              01 50   E9  02AE             BLBC    R0,160$                     ; Branch if error
                      04  02B1             RET                                 ; Return
              02D1    31  02B2     160$:   BRW     ERROR                       ; Helper branch
                          02B5
                          02B5     ;
                          02B5     ; Kernel mode routine to finish processing.
                          02B5     ;
                          02B5
                    007C  02B5     170$:   .WORD   ^M<R2,R3,R4,R5,R6>          ; Entry mask
                          02B7
                          02B7     ;
                          02B7     ; Get parameter list address and PCB address.
                          02B7     ;
                          02B7
 54   00000000'9F     D0  02B7             MOVL    @#CTL$GL_PCB,R4             ; Get PCB address
           56    8C   D0  02BE             MOVL    (AP)+,R6                    ; Get service parameter list address
                          02C1
                          02C1     ;
                          02C1     ; Clear event flag.
                          02C1     ;
                          02C1
       53    04 A6    9A  02C1             MOVZBL  EFN(R6),R3                  ; Get event flag number
      00000000'EF     16  02C5             JSB     SCH$CLREF                   ; Clear this event flag
              22 50   E9  02CB             BLBC    R0,190$                     ; Return on errors
                          02CE
                          02CE     ;
                          02CE     ; Check and charge AST quota.
                          02CE     ;
                          02CE
              18 A6   D5  02CE             TSTL    ASTADR(R6)                  ; AST routine specified?
                 0D   13  02D1             BEQL    180$                        ; Branch if none
     50    2A04 8F    3C  02D3             MOVZWL  #SS$_EXASTLM,R0             ; Assume AST quota exceeded status
              38 A4   B5  02D8             TSTW    PCB$W_ASTCNT(R4)            ; Test for quota exceeded
                 13   15  02DB             BLEQ    190$                        ; Branch if exceeded
              38 A4   B7  02DD             DECW    PCB$W_ASTCNT(R4)            ; Charge AST quota
                          02E0
                          02E0     ;
                          02E0     ; Send the message.
                          02E0     ;
                          02E0
       53    6C       7D  02E0     180$:   MOVQ    (AP),R3                     ; R3 = size, R4 = address of message
 55   00000000'EF     9E  02E3             MOVAB   SYS$GL_JOBCTLMB,R5          ; R5 = mailbox UCB address
      00000000'EF     16  02EA             JSB     EXE$SENDMSG                 ; Send message
                      04  02F0     190$:   RET                                 ; Return
                          02F1             .DSABL  LSB
```

```
                       02F1
                       02F1      ;
                       02F1      ; Stack work area offsets for next routine.
                       02F1      ;
                       02F1
           00000000    02F1      FWA_DVI=        0                          ; DVI
           00000010    02F1      FWA_FID=        16                         ; FID
           00000016    02F1      FWA_DID=        22                         ; DID
           0000001C    02F1      FWA_FILE_SIZE=  28                         ; File size in blocks
                       02F1                                                 ; (spare longword)
           00000024    02F1      FWA_FILE_SPEC=  36                         ; File specification
           00000124    02F1      FWA_RECATTR=    292                        ; Record attributes
           00000144    02F1      FWA_CHAN=       324                        ; Channel assigned to device
           00000148    02F1      FWA_IOSB=       328                        ; I/O status block
                       02F1
           0000001C    02F1      FWA_FAB=        28                         ; FAB block
           0000006C    02F1      FWA_NAM=        FWA_FAB + FAB$C_BLN        ; NAM block
           000000CC    02F1      FWA_ESA=        FWA_NAM + NAM$C_BLN        ; Expanded string
           000001CB    02F1      FWA_SIZE=       FWA_ESA + NAM$C_MAXRSS     ; Length of area
                       02F1
           00000024    02F1      FWA_DVI_DESC=   36                         ; Descriptor for device name
                       02F1
           00000024    02F1      FWA_FIB_DESC=   36                         ; Descriptor for FIB
           0000002C    02F1      FWA_FIB=        44                         ; File information block
           0000006C    02F1      FWA_ATRLIST=    FWA_FIB+FIB$C_LENGTH       ; Attribute list
                       02F1
                       02F1              .ENABL  LSB
                       02F1      FILE_SPECIFICATION:                        ; Translate SJC$_FILE_SPECIFICATION
                       02F1
                       02F1      ;
                       02F1      ;       R5  = buffer size
                       02F1      ;       R6  = item code
                       02F1      ;       R7  = buffer address
                       02F1      ;       R10 = pointer to lowest available stack address
                       02F1      ;
                       02F1      ; Check that the parameter is the correct length.
                       02F1      ;
                       02F1
         FC AD   D5    02F1              TSTL    FILE_ID(FP)                ; See if there is already a filespec
            6D   12    02F4              BNEQ    20$                        ; Branch if so
   00FF 8F  55   B1    02F6              CMPW    R5,#255                    ; Ensure no longer than 255 bytes
            66   1A    02FB              BGTRU   20$                        ; Branch if incorrect
                       02FD
                       02FD      ;
                       02FD      ; Check for sufficient space to allocate the work area, and do so.
                       02FD      ;
                       02FD
   50    FE35 CE   9E  02FD              MOVAB   -FWA_SIZE(SP),R0           ; Get lowest address that will be used
        5A    50   D1  0302              CMPL    R0,R10                     ; Compare against that available
              5F   1F  0305              BLSSU   30$                        ; Branch if space exceeded
        5E    50   D0  0307              MOVL    R0,SP                      ; Allocate the space
                       030A
                       030A      ;
                       030A      ; Initialize the FAB and NAM blocks.
                       030A      ;
                       030A
              55   DD  030A              PUSHL   R5                         ; Save R5 across MOVC
```

```
SYS
Sym
PCB
PCB
PCB
PCB
PCB
PCB
PCB
PHD
PHD
POS
PSL
PSL
PSL
PSL
PSL
QUI
QUI
QUI
SCH
SCH
SCH
SJC
SJC
SJC
SJC
SJC
SJC
SJC
SJC
SJC
SJC
SJC
SND.
SND.
SND.
SND.
SPE
SS$.
SS$.
SS$.
SS$.
SS$.
SYS
SYS
SYS
SYS
SYS
SYS
SYS
SYS
SYS
SYS
TRA
TRN
TRN
```

```
20 AE   00B0 8F    00    6E    00    2C  030C   661          MOVC5    #0,(SP),#0,#<FAB$C_BLN+NAM$C_BLN>,FWA_FAB+4(SP) ; Clear FAB/NAM
                               55  8ED0  0315   662          POPL     R5                        ; Restore R5
                   52    1C AE     9E    0318   663          MOVAB    FWA_FAB(SP),R2            ; Point to FAB
                   53    50 A2     9E    031C   664          MOVAB    FAB$C_BLN(R2),R3          ; Point to NAM
                62     5003 8F     B0    0320   665          MOVW     #<FAB$C_BID!<FAB$C_BLN@8>>,FAB$B_BID(R2) ; Set FAB identifier
                34 A2      55      90    0325   666          MOVB     R5,FAB$B_FNS(R2)          ; Set file name length
                2C A2      57      D0    0329   667          MOVL     R7,FAB$L_FNA(R2)          ; Set file name address
                28 A2      63      9E    032D   668          MOVAB    (R3),FAB$L_NAM(R2)        ; Set NAM block address
                63     6002 8F     B0    0331   669          MOVW     #<NAM$C_BID!<NAM$C_BLN@8>>,NAM$B_BID(R3) ; Set NAM identifier
             0A A3       FF 8F     90    0336   670          MOVB     #NAM$C_MAXRSS,NAM$B_ESS(R3) ; Set ESA descriptor
          0C A3      00CC CE       9E    033B   671          MOVAB    FWA_ESA(SP),NAM$L_ESA(R3)
                     56    27      9A    0341   672          MOVZBL   #SJC$_FILE_IDENTIFICATION,R6 ; Set up item code
                   57    14 A3     9E    0344   673          MOVAB    NAM$T_DVI(R3),R7          ; Point to DVI/FID/DID area
                                         0348   674
                                         0348   675 ;
                                         0348   676 ; Execute a PARSE and SEARCH to get the DVI/FID/DID.
                                         0348   677 ;
                                         0348   678
                                         0348   679          $PARSE   FAB=(R2)                         ; Parse the file name
                      0C 50        E9    0351   680          BLBC     R0,10$                           ; Branch if error
                                         0354   681          $SEARCH  FAB=(R2)                         ; Search the file name
                      34 50        E8    035D   682          BLBS     R0,50$                           ; Branch to handle like FID item
                                         0360   683
                                         0360   684 ;
                                         0360   685 ; Helper branches.
                                         0360   686 ;
                                         0360   687
                       0223        31    0360   688 10$:     BRW      ERROR
                       FDA1        31    0363   689 20$:     BRW      BADPARAM
                       FDA3        31    0366   690 30$:     BRW      INSFMEM
                                         0369   691
                                         0369   692
                                         0369   693 FILE_IDENTIFICATION:                              ; Translate SJC$_FILE_IDENTIFICATION
                                         0369   694
                                         0369   695 ;
                                         0369   696 ;        R5  = buffer size
                                         0369   697 ;        R6  = item code
                                         0369   698 ;        R7  = buffer address
                                         0369   699 ;        R10 = pointer to lowest available stack address
                                         0369   700 ;
                                         0369   701 ; Check that the parameter is the correct length.  If it is not the expected
                                         0369   702 ; 28 bytes, and the previous mode is at least executive, assume that we have
                                         0369   703 ; been passed the entire expanded item and send it on as is.
                                         0369   704 ;
                                         0369   705
                      FC AD        D5    0369   706          TSTL     FILE_ID(FP)                      ; See if there is already a filespec
                         F5        12    036C   707          BNEQ     20$                              ; Branch if so
                   1C    55        B1    036E   708          CMPW     R5,#28                           ; Ensure parameter is 28 bytes
                         0F        13    0371   709          BEQL     40$                              ; Branch if correct
                         50        DC    0373   710          MOVPSL   R0                               ; Get PSL
        50    50   02    16        EF    0375   711          EXTZV    #PSL$V_PRVMOD,#PSL$S_PRVMOD,R0,R0 ; Get previous mode
                   01    50        D1    037A   712          CMPL     R0,#PS[$C_EXEC                   ; Previous mode exec or kernel?
                         E4        1A    037D   713          BGTRU    20$                              ; Branch if not
                       FE54        31    037F   714          BRW      INPUT_ITEM                       ; Branch to store item as is
                   0F    67        91    0382   715 40$:     CMPB     (R7),#15                         ; Ensure device no more than 15 bytes
                         DC        1A    0385   716          BGTRU    20$                              ; Branch if incorrect
                                         0387   717
```

```
                                  0387  ;
                                  0387  ; Check for sufficient space to allocate the work area, and do so.
                                  0387  ;
                                  0387
                50  FE35 CE   9E  0387          MOVAB   -FWA_SIZE(SP),R0        ; Get lowest address that will be used
                    5A   50   D1  038C          CMPL    R0,R10                  ; Compare against that available
                         D5   1F  038F          BLSSU   30$                     ; Branch if space exceeded
                    5E   50   D0  0391          MOVL    R0,SP                   ; Allocate the space
                                  0394
                                  0394  ;
                                  0394  ; Move the DVI/FID/DID to the work area.
                                  0394  ;
                                  0394
                6E   67   1C  28  0394  50$:    MOVC3   #28,(R7),FWA_DVI(SP)    ; Move the parameter to the work area
                  FC AD   5E  D0  0398          MOVL    SP,FILE_ID(FP)          ; Save location of file ID buffer
                    7E    56  3C  039C          MOVZWL  R6,-(SP)                ; Store item code, leave space for size
                       FDC0   31  039F          BRW     ITEM                    ; Remainder of processing comes later
                                  03A2
                                  03A2  ;
                                  03A2  ; The file specification, if any, must be post-processed after all items
                                  03A2  ; have been digested. Inputs:
                                  03A2  ;
                                  03A2  ;       R10 = address of file ID item
                                  03A2  ;
                                  03A2  ; Get a pointer to the DVI descriptor, and where the channel will be stored,
                                  03A2  ; and initialize the descriptor.
                                  03A2  ;
                                  03A2
                                  03A2  POSTPROCESS_FID:
2C AA   0040 8F   00   6E   00  2C  03A2          MOVC5   #0,(SP),#0,#FIB$C_LENGTH,FWA_FIB(R10) ; Initialize FIB
                       20 AA    D4  03AB          CLRL    32(R10)                 ; Clear unused longword
                  50   24 AA    9E  03AE          MOVAB   FWA_DVI_DESC(R10),R0    ; Point to DVI descriptor
                53   0144 CA    9E  03B2          MOVAB   FWA_CHAN(R10),R3        ; Point to channel
                  60     6A     9A  03B7          MOVZBL  FWA_DVI(R10),(R0)       ; Store device name length
              04 A0    01 AA    9E  03BA          MOVAB   FWA_DVI+1(R10),4(R0)    ; Store device name address
                                  03BF
                                  03BF  ;
                                  03BF  ; Assign a channel to the device.
                                  03BF  ;
                                  03BF
                                  03BF          $ASSIGN_S -                     ; Assign a channel
                                  03BF                  DEVNAM=(R0), -          ;  Device name
                                  03BF                  CHAN=(R3)               ;  Output channel number
                    91 50     E9  03CC          BLBC    R0,10$                  ; Branch if not assigned
                                  03CF
                                  03CF  ;
                                  03CF  ; Build the FIB, the FIB descriptor, and the ACP attributes list.
                                  03CF  ;
                                  03CF
                50   6C AA    9E  03CF          MOVAB   FWA_ATRLIST(R10),R0     ; Point to attribute list
                51   24 AA    9E  03D3          MOVAB   FWA_FIB_DESC(R10),R1    ; Point to FIB descriptor
              52   0148 CA    9E  03D7          MOVAB   FWA_IOSB(R10),R2        ; Point to IOSB
                54   2C AA    9E  03DC          MOVAB   FWA_FIB(R10),R4         ; Point to FIB
                                  03E0
              04 A4   10 AA   D0  03E0          MOVL    FWA_FID(R10),FIB$W_FID(R4) ; Store file ID
              08 A4   14 AA   B0  03E5          MOVW    FWA_FID+4(R10),FIB$W_FID+4(R4)
           18 F8 AD     00    E1  03EA          BBC     #DELETE_FLAG,FLAGS(FP),55$ ; Branch if not deleting file
```

```
        0A A4    16 AA   D0  03EF    775          MOVL    FWA_DID(R10),FIB$W_DID(R4) ; Also store directory ID
        0E A4    1A AA   B0  03F4    776          MOVW    FWA_DID+4(R10),FIB$W_DID+4(R4)
     14 A4     0800 8F   A8  03F9    777          BISW    #FIB$M_FINDFID,FIB$W_NMCTL(R4)
         38 A4     01    C8  03FF    778          BISL    #FIB$M_ALT_REQ,FIB$L_STATUS(R4) ; Alternate access required
         3C A4     08    D0  0403    779          MOVL    #ARM$M_DELETE,FIB$L_ALT_ACCESS(R4) ; Check for delete access
                             0407    780
    61   00000040 8F     D0  0407    781 55$:     MOVL    #FIB$C_LENGTH,(R1)          ; Initialize FIB descriptor
           04 A1   64    9E  040E    782          MOVAB   (R4),4(R1)                  ;
                             0412    783
    60   00040020 8F     D0  0412    784          MOVL    #<ATR$S_RECATTR+<ATR$C_RECATTR@16>>,(R0)
     04 A0   0124 CA     9E  0419    785          MOVAB   FWA_RECATTR(R10),4(R0)
 08 A0   002E0100 8F     D0  041F    786          MOVL    #<256+<ATR$C_FILE_SPEC@16>>,8(R0)
       0C A0    24 AA    9E  0427    787          MOVAB   FWA_FILE_SPEC(R10),12(R0)
               10 A0     D4  042C    788          CLRL    16(R0)
                             042F    789
                             042F    790 ;
                             042F    791 ; Access the file to get necessary information.
                             042F    792 ;
                             042F    793
                             042F    794          $QIOW_S -                           ; Issue QIO to obtain file attributes
                             042F    795                  EFN=EFN(AP), -              ;  User's event flag
                             042F    796                  CHAN=(R3), -                ;  Channel number
                             042F    797                  FUNC=#IO$_ACCESS, -         ;  Read attributes function code
                             042F    798                  IOSB=(R2), -                ;  I/O status block
                             042F    799                  P1=(R1), -                  ;  Address of FIB descriptor
                             042F    800                  P5=R0                       ;  Address of attribute list
                  50     DD  044D    801          PUSHL   R0                          ; Save $QIOW status
                             044F    802          $DASSGN_S -                         ; Deassign the channel
                             044F    803                  CHAN=(R3)                   ;  Channel number
                  50 8ED0    0459    804          POPL    R0                          ; Restore status from access
               48 50     E9  045C    805          BLBC    R0,70$                      ; Branch if $QIOW failed
    50   0148 CA         3C  045F    806          MOVZWL  FWA_IOSB(R10),R0            ; Pick up status from IOSB
               40 50     E9  0464    807          BLBC    R0,70$                      ; Branch if operation failed
                             0467    808
                             0467    809 ;
                             0467    810 ; Compute the file size from the record attributes.
                             0467    811 ;
                             0467    812
 1C AA    012C CA   10   9C  0467    813          ROTL    #16, -                      ; Move EFBLK to file size area and
                             046E    814                  FWA_RECATTR+FAT$L_EFBLK(R10), -
                             046E    815                  FWA_FILE_SIZE(R10)          ; convert to unswapped
                  09     13  046E    816          BEQL    60$                         ; Branch if EFBLK is zero
             0130 CA     B5  0470    817          TSTW    FWA_RECATTR+FAT$W_FFBYTE(R10) ; Test first free byte
                  03     12  0474    818          BNEQ    60$                         ; Branch if nonzero
               1C AA     D7  0476    819          DECL    FWA_FILE_SIZE(R10)          ; Adjust EFBLK
                             0479    820
                             0479    821 ;
                             0479    822 ; Slide the real data up adjacent to the previous item on the stack, and
                             0479    823 ; finish it by adding the length and item code.
                             0479    824 ;
                             0479    825
    57      24 AA        3C  0479    826 60$:     MOVZWL  FWA_FILE_SPEC(R10),R7       ; Get file specification length
  00FE 8F      57        B1  047D    827          CMPW    R7,#254                     ; Check against maximum supported length
                  05     1B  0482    828          BLEQU   65$                         ; Branch if in range
    57    00FE 8F        3C  0484    829          MOVZWL  #254,R7                     ; Shorten to maximum
            57    26     C0  0489    830 65$:     ADDL    #FWA_FILE_SPEC+2,R7         ; Add fixed portion
       FE AA      57     B0  048C    831          MOVW    R7,-2(R10)                  ; Store length in message
```

```
      58  000001CB 8F   57   C3  0490   832          SUBL3   R7,#FWA_SIZE,R8          ; Compute bias
                   57   5A   C0  0498   833          ADDL    R10,R7                   ; Compute size of area above filespec
                   57   5E   C2  049B   834          SUBL    SP,R7                    ; = R10 - SP + R7
            6E48   6E   57   28  049E   835          MOVC3   R7,(SP),(SP)[R8]         ; Squish out unused space
                   5E   58   C0  04A3   836          ADDL    R8,SP                    ; Delete unused stack
                             05  04A6   837          RSB                              ; Done with file spec
                                 04A7   838
                                 04A7   839 ;
                                 04A7   840 ; Helper branches.
                                 04A7   841 ;
                                 04A7   842
                   00DC      31  04A7   843 70$:     BRW     ERROR
                                 04AA   844
                                 04AA   845          .DSABL  LSB
```

```
                              04AA
                              04AA    ;
                              04AA    ; Stack work area offsets for next routine.
                              04AA    ;
                              04AA
                    00000000  04AA    LWA_BUFFER=     0                       ; Logical name buffer
                    00000100  04AA    LWA_LOGNAM=     256                     ; Logical name descriptor
                    00000108  04AA    LWA_ITMLST=     264                     ; $TRNLNM item list
                    00000124  04AA    LWA_RSLLEN=     292                     ; Translation length buffer
                    00000128  04AA    LWA_ATTRBUF=    296                     ; Translation attributes buffer
                    0000012C  04AA    LWA_SIZE=       300                     ; Work area length
                              04AA
                              04AA            .ENABL  LSB
                  FC5A    31  04AA    10$:    BRW     BADPARAM
                  FC5C    31  04AD    20$:    BRW     INSFMEM
                              04B0
                              04B0    TRANSLATE_OBJECT:                       ; Translate object names
                              04B0
                              04B0    ;
                              04B0    ;       R5  = buffer size
                              04B0    ;       R6  = item code
                              04B0    ;       R7  = buffer address
                              04B0    ;       R10 = pointer to lowest available stack address
                              04B0    ;
                              04B0    ; Check that the parameter is the correct length and that there is sufficient
                              04B0    ; space to allocate the work area (then do so).
                              04B0    ;
                              04B0
        00FF 8F      55   B1  04B0            CMPW    R5,#255                 ; Ensure no more than 255 bytes
                     F3   1A  04B5            BGTRU   10$                     ; Branch if incorrect
                              04B7
        53   FED4    CE   9E  04B7            MOVAB   -LWA_SIZE(SP),R3        ; Get lowest address that will be used
                5A   53   D1  04BC            CMPL    R3,R10                  ; Compare against that available
                EC   1F       04BF            BLSSU   20$                     ; Branch if space exceeded
                5E   53   D0  04C1            MOVL    R3,SP                   ; Allocate the space
                              04C4
                              04C4    ;
                              04C4    ; Prepare to perform the iterative translations by initializing the logical name
                              04C4    ; descriptor and the item list utilized by the recursive $TRNLNMs.
                              04C4    ;
                              04C4
                     55   DD  04C4            PUSHL   R5                      ; Save the input string length
        63   67      55   28  04C6            MOVC3   R5,(R7),(R3)            ; Move input string into scratch buffer
                     55 8ED0  04CA            POPL    R5                      ; Restore the input string length
             57      5E   D0  04CD            MOVL    SP,R7                   ; Restore scratch buffer address
                              04D0
        51   0104 C7      9E  04D0            MOVAB   LWA_LOGNAM+4(R7),R1     ; Addr of area requiring initialization
             81   67      9E  04D5            MOVAB   (R7),(R1)+              ; Init log name descriptor buffer addr
                              04D8
  81    000200FF 8F       D0  04D8            MOVL    #<LNM$_STRING @ 16+-    ; Init string item list item type
                              04DF                    255>,(R1)+              ; and string buffer length
             81   67      9E  04DF            MOVAB   (R7),(R1)+              ; Init string item buffer address
        81   0124 C7      9E  04E2            MOVAB   LWA_RSLLEN(R7),(R1)+    ; Init string item return buffer address
  81    00030004 8F       D0  04E7            MOVL    #<LNM$_ATTRIBUTES @ 16+-; Init attributes item list item type
                              04EE                    4>,(R1)+                ; and attributes buffer length
        81   0128 C7      9E  04EE            MOVAB   LWA_ATTRBUF(R7),(R1)+   ; Init attributes item buffer address
                  61      7C  04F3            CLRQ    (R1)                    ; Init attributes item return buffer
```

```
                          04F5    904                                          ; address and end of item list marker
                          04F5    905
           54    0A    D0 04F5    906          MOVL    #LNM$C_MAXDEPTH,R4      ; Initialize loop counter
```

```
                        04F8   908
                        04F8   909 ;
                        04F8   910 ; Loop to iterate over translations.
                        04F8   911 ;
                        04F8   912 ;       R4  = Current translation count
                        04F8   913 ;       R5  = Current input string length
                        04F8   914 ;       R6  = Item code
                        04F8   915 ;       R7  = Current input string address, and
                        04F8   916 ;             Address of work area
                        04F8   917 ;
                        04F8   918 ; The iterations successfully terminate when:
                        04F8   919 ;
                        04F8   920 ; 1. The maximum translation recursion depth is exceeded.
                        04F8   921 ; 2. The current translation succeeds but the translation is marked terminal.
                        04F8   922 ; 3. The current translation fails with an error of SS$_NOLOGNAM.
                        04F8   923 ;
                        04F8   924 ; The iterations unsuccessfully terminate when:
                        04F8   925 ;
                        04F8   926 ; 1. The current translation fails with some error other than SS$_NOLOGNAM.
                        04F8   927 ; 2. The current translation exceeds but the equivalence string is either of
                        04F8   928 ;    null length or does not exist.
                        04F8   929 ;
                        04F8   930
0100 C7    55     D0    04F8   931 30$:     MOVL    R5,LWA_LOGNAM(R7)          ; Store name length in descriptor
                        04FD   932
                        04FD   933          $TRNLNM_S        -                  ; Attempt to translate the name
                        04FD   934                  ATTR    = TRNLNM_ATTR,-           ; Case-insensitive translation
                        04FD   935                  ITMLST  = LWA_ITMLST(R7),-        ; Address of item list
                        04FD   936                  LOGNAM  = LWA_LOGNAM(R7),-        ; Address of name descriptor
                        04FD   937                  TABNAM  = TRNLNM_TABLE            ; Addr LNM$FILE_DEV descriptor
        19 50     E9    0516   938          BLBC    R0,45$                     ; Done if translation fails
                        0519   939
55    0124 C7     3C    0519   940          MOVZWL  LWA_RSLLEN(R7),R5          ; Retrieve length of equivalence string
           0B     13    051E   941          BEQL    40$                        ; Go return error if length is 0
                        0520   942                                             ; (null or non-existant translation)
                        0520   943
           09     E0    0520   944          BBS     #LNM$V_TERMINAL,-          ; Is the translation marked terminal?
     13 0128 C7         0522   945                  LWA_ATTRBUF(R7),50$        ; terminate iterative translations if so
        CF 54     F5    0526   946          SOBGTR  R4,30$                     ; Continue if more translations possible
           0E     11    0529   947          BRB     50$                        ; Else, done if xlation count exhausted
                        052B   948
50    0154 8F     3C    052B   949 40$:     MOVZWL  #SS$_IVLOGNAM,R0           ; Return an error for null length or
           54     11    0530   950          BRB     ERROR                      ; non-existant translation
                        0532   951
01BC 8F    50     B1    0532   952 45$:     CMPW    R0,#SS$_NOLOGNAM           ; If the translation failed for a reason
           4D     12    0537   953          BNEQ    ERROR                      ; other than the logical name did not
                        0539   954                                             ; exist then go return the error
```

```
                                 0539   956
                                 0539   957 ;
                                 0539   958 ; Recursive translations have completed. Format the final translation by
                                 0539   959 ; removing blanks, tabs, null characters, and a trailing colon if there is one,
                                 0539   960 ; and upcasing the name using the DEC multi-national character upcasing table.
                                 0539   961 ;
                                 0539   962 ;       R1 = Current character
                                 0539   963 ;       R2 = Current character index
                                 0539   964 ;       R3 = Cursor to output buffer
                                 0539   965 ;       R5 = Length of input string
                                 0539   966 ;       R7 = Address of input string
                                 0539   967 ;
                                 0539   968
            52    01      CE     0539   969 50$:    MNEGL   #1,R2                   ; Initialize the loop index
            53    57      D0     053C   970         MOVL    R7,R3                   ; Initialize output buffer cursor
                  18      11     053F   971         BRB     70$                     ; Branch to enter the loop
                                 0541   972
        51      6742      9A     0541   973 60$:    MOVZBL  (R7)[R2],R1             ; Pick up the current character
                  12      13     0545   974         BEQL    70$                     ; Remove it if it is null
          20      51      91     0547   975         CMPB    R1,#^A' '               ; Is the current character a blank?
                  0D      13     054A   976         BEQL    70$                     ; Remove it if it is
          09      51      91     054C   977         CMPB    R1,#^O011               ; Is the current character a tab?
                  08      13     054F   978         BEQL    70$                     ; Remove it if it is
    83    00000000'GF41   90     0551   979         MOVB    G^EXE$UPCASE_DAT[R1],-  ; Move upcased character into output
                                 0559   980                 (R3)+                   ; buffer
         E4 52    55      F2     0559   981 70$:    AOBLSS  R5,R2,60$               ; Continue loop until done
                                 055D   982
      57    53    57      C3     055D   983         SUBL3   R7,R3,R7                ; Computes name's compressed length
                  C8      13     0561   984         BEQL    40$                     ; Return an error if its zero
        3A     FF A3      91     0563   985         CMPB    -1(R3),#^A':'           ; Is there a trailing colon?
                  04      12     0567   986         BNEQ    80$                     ; Branch if there isn't
                  57      D7     0569   987         DECL    R7                      ; Otherwise remove it
                  BE      13     056B   988         BEQL    40$                     ; Return an error if name length is 0
                                 056D   989
                                 056D   990 ;
                                 056D   991 ; Slide the name up the stack so that it is adjacent to the previous item on
                                 056D   992 ; the stack. Then complete the formation of the item by adding the name length
                                 056D   993 ; and item code.
                                 056D   994 ;
                                 056D   995
  58    0000012C 8F       57      C3     056D   996 80$:    SUBL3   R7,#LWA_SIZE,R8         ; Compute bias
           6E48    6E     57      28     0575   997         MOVC3   R7,(SP),(SP)[R8]        ; Slide item up
                   5E     58      C0     057A   998         ADDL2   R8,SP                   ; Delete unused stack
                   7E     57      B0     057D   999         MOVW    R7,-(SP)                ; Store item length
                   7E     56      B0     0580  1000         MOVW    R6,-(SP)                ; Store item code
                        FBDC      31     0583  1001         BRW     ITEM                    ; Return to item list processing
                                         0586  1002         .DSABL  LSB
```

```
                              0586
                              0586  ;
                              0586  ; Synchronous error return path.  Store status in the IOSB, set the event flag,
                              0586  ; and declare the AST, if specified.
                              0586  ;
                              0586
                     50   DD  0586  ERROR:  PUSHL   R0                          ; Save completion status
   00000000'GF       6C   FA  0588          CALLG   (AP),G^SYS$SETEF            ; Set specified event flag
           51    14  AC   D0  058F          MOVL    IOSB(AP),R1                 ; Get address of IOSB
                     09   13  0593          BEQL    10$                         ; Branch if none
                              0595          IFNOWRT #8,(R1),10$                 ; Branch if no write access
                 61  6E   D0  059B          MOVL    (SP),(R1)                   ; Store completion status
           51    18  AC   D0  059E  10$:    MOVL    ASTADR(AP),R1               ; Get address of AST routine
                     15   13  05A2          BEQL    20$                         ; Branch if none
                     50   DC  05A4          MOVPSL  R0                          ; Get PSL
  50    50   02      16   EF  05A6          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R0,R0 ; Get previous mode
                              05AB          $DCLAST_S (R1),ASTPRM(AP),R0        ; Declare completion AST
                     50 8ED0  05B9  20$:    POPL    R0                          ; Restore completion status
                          04  05BC          RET                                 ; Return with error status
```

```
                        05BD         .SBTTL  EXE$JBCRSP - Store response from job controller
                        05BD
                        05BD ;++
                        05BD ;
                        05BD ; EXE$JBCRSP - STORE RESPONSE FROM JOB CONTROLLER
                        05BD ;
                        05BD ; FUNCTIONAL DESCRIPTION:
                        05BD ;
                        05BD ;       This routine is called as a special kernel AST routine to return status
                        05BD ;       from the send to job controller system service to the requesting
                        05BD ;       process.  It ensures that the same image is executing and then sets the
                        05BD ;       specified event flag, stores a status value in the IOSB if specified,
                        05BD ;       stores data in any output buffer items that were in the original
                        05BD ;       request, and declares the completion AST if specified.  If appropriate,
                        05BD ;       the ACB is deallocated.
                        05BD ;
                        05BD ; INPUTS:
                        05BD ;       R0-R3           = scratch
                        05BD ;       R4              = PCB address
                        05BD ;       R5              = ACB address
                        05BD ;
                        05BD ; OUTPUTS:
                        05BD ;       See above.
                        05BD ;--
                        05BD
                        05BD EXE$JBCRSP::                                ; Response from job controller
                        05BD
                        05BD ;
                        05BD ; Compare the image count when the request was queued with the current image
                        05BD ; count.  If different, a new image is running - do not store anything.
                        05BD ;
                        05BD
 53   00000000'9F   D0  05BD         MOVL    @#CTL$GL_PHD,R3             ; Get PHD address
  1C A5   00F4 C3   D1  05C4         CMPL    PHD$L_IMGCNT(R3),ACB_L_IMGCNT(R5) ; See if image count correct
               03   13  05CA         BEQL    10$                         ; Branch if correct
             0085   31  05CC         BRW     70$                         ; Join code to deallocate ACB
                        05CF
                        05CF ;
                        05CF ; Loop over the return item descriptors storing information in the user's
                        05CF ; output buffers.  During this loop:
                        05CF ;
                        05CF ;       R5  = user buffer address
                        05CF ;       R6  = pointer to item descriptors in ACB
                        05CF ;       R7  = user buffer size
                        05CF ;       R8  = actual data size
                        05CF ;       R9  = requester's access mode
                        05CF ;       R10 = item count
                        05CF ;
                        05CF
            2C A5   B5  05CF 10$:    TSTW    ACB_W_ITEMCOUNT(R5)         ; Any items to return?
               52   13  05D2         BEQL    50$                         ; Branch if none
          07F0 8F   BB  05D4         PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10>  ; Save registers
     56     2E A5   9E  05D8         MOVAB   ACB_B_ITEMS(R5),R6          ; Point to items
     59     0B A5   9A  05DC         MOVZBL  ACB$B_RMOD(R5),R9           ; Get requester's mode
     5A     2C A5   3C  05E0         MOVZWL  ACB_W_ITEMCOUNT(R5),R10     ; Get item count
     57        86   3C  05E4 20$:    MOVZWL  (R6)+,R7                    ; Get user buffer size
     58        86   3C  05E7         MOVZWL  (R6)+,R8                    ; Get actual size
```

```
                 55   86   D0  05EA  1081           MOVL    (R6)+,R5                  ; Get data buffer address
                 50   55   D0  05ED  1082           MOVL    R5,R0                     ; R0 = buffer address
                 51   57   D0  05F0  1083           MOVL    R7,R1                     ; R1 = buffer length
                 53   59   D0  05F3  1084           MOVL    R9,R3                     ; R3 = requester's mode
           00000000'EF     16  05F6  1085           JSB     EXE$PROBEW                ; Probe for write access
                66   50    E9  05FC  1086           BLBC    R0,90$                    ; Branch if inaccessible
 65   57   00   04 A6   58 2C  05FF  1087           MOVC5   R8,4(R6),#0,R7,(R5)       ; Move data to user buffer
                 55   86   D0  0606  1088           MOVL    (R6)+,R5                  ; Get return length address
                      11   13  0609  1089           BEQL    40$                       ; Branch if none
                               060B  1090           IFNOWRT #2,(R5),90$,R9            ; Probe for write access
                 57   58   D1  0611  1091           CMPL    R8,R7                     ; Minimize user and actual length
                      03   1E  0614  1092           BGEQU   30$                       ; Branch if actual length larger
                 57   58   D0  0616  1093           MOVL    R8,R7                     ; Get actual length as minimum
                 65   57   B0  0619  1094 30$:      MOVW    R7,(R5)                   ; Return buffer length
                 56   58   C0  061C  1095 40$:      ADDL2   R8,R6                     ; Advance over data
                   C2 5A   F5  061F  1096           SOBGTR  R10,20$                   ; Loop for all items
                 07F0 8F   BA  0622  1097           POPR    #^M<R4,R5,R6,R7,R8,R9,R10> ; Restore registers
                               0626  1098
                               0626  1099 ;
                               0626  1100 ; Output buffers stored.  Set the specified event flag, return status to the
                               0626  1101 ; IOSB, and declare the completion AST if specified.  If no AST specified,
                               0626  1102 ; deallocate the ACB.
                               0626  1103 ;
                               0626  1104
           51    60 A4     D0  0626  1105 50$:      MOVL    PCB$L_PID(R4),R1          ; R1 = PID
                      52   D4  062A  1106           CLRL    R2                        ; R2 = null priority increment
           53    20 A5     9A  062C  1107           MOVZBL  ACB_L_EFN(R5),R3          ; R3 = event flag number
         00000000'EF       16  0630  1108           JSB     SCH$POSTEF                ; Set specified event flag
           51    24 A5     D0  0636  1109           MOVL    ACB_L_IOSB(R5),R1         ; Get IOSB address
                      0B   13  063A  1110           BEQL    60$                       ; Branch if none
                               063C  1111           IFNOWRT #4,(R1),60$,ACB$B_RMOD(R5) ; Probe for write access
           61    28 A5     D0  0643  1112           MOVL    ACB_L_STATUS(R5),(R1)     ; Return status
                 10 A5     D5  0647  1113 60$:      TSTL    ACB$L_AST(R5)             ; Completion AST specified?
                      08   13  064A  1114           BEQL    70$                       ; Branch if no to deallocate ACB
                      52   D4  064C  1115           CLRL    R2                        ; R2 = null priority increment
         00000000'EF       17  064E  1116           JMP     SCH$QAST                  ; Queue completion AST and return
                               0654  1117
                               0654  1118 ;
                               0654  1119 ; Processing finished.  Return AST quota if charged, and deallocate the ACB.
                               0654  1120 ; (No byte count quota is charged for this ACB because it is allocated by the
                               0654  1121 ; job controller.)
                               0654  1122 ;
                               0654  1123
        03 0B A5      06   E1  0654  1124 70$:      BBC     #ACB$V_QUOTA,ACB$B_RMOD(R5),80$ ; Branch if no AST quota charged
                 38 A4     B6  0659  1125           INCW    PCB$W_ASTCNT(R4)          ; Return AST quota
                 50   55   D0  065C  1126 80$:      MOVL    R5,R0                     ; R0 = ACB address
         00000000'EF       17  065F  1127           JMP     EXE$DEANONPAGED           ; Deallocate ACB and return
                               0665  1128
                               0665  1129 ;
                               0665  1130 ; Memory is inaccessible.  Attempt to return an access violation status to
                               0665  1131 ; the IOSB.
                               0665  1132 ;
                               0665  1133
                 07F0 8F   BA  0665  1134 90$:      POPR    #^M<R4,R5,R6,R7,R8,R9,R10> ; Restore registers
           28 A5      0C   D0  0669  1135           MOVL    #SS$_ACCVIO,ACB_L_STATUS(R5) ; Force status to ACCVIO
                      B7   11  066D  1136           BRB     50$                       ; Go to return EFN and IOSB
                               066F  1137           .END
```

```
$$.TMP1                         = 00000001                   FILE_SPECIFICATION               000002F1 R       02
$$.TMP2                         = 00000062                   FINISH_MESSAGE                   0000021D R       02
$$T1                            = 00000000                   FIXED_AREA                     = 0000005A
ACB$B_RMOD                      = 0000000B                   FLAGS                          = FFFFFFF8
ACB$L_AST                       = 00000010                   FUNC                           = 00000008
ACB$L_KAST                      = 00000018                   FWA_ATRLIST                    = 0000006C
ACB$V_QUOTA                     = 00000006                   FWA_CHAN                       = 00000144
ACB_B_ITEMS                       0000002E                   FWA_DID                        = 00000016
ACB_L_EFN                         00000020                   FWA_DVI                        = 00000000
ACB_L_IMGCNT                      0000001C                   FWA_DVI_DESC                   = 00000024
ACB_L_IOSB                        00000024                   FWA_ESA                        = 000000CC
ACB_L_STATUS                      00000028                   FWA_FAB                        = 0000001C
ACB_W_ITEMCOUNT                   0000002C                   FWA_FIB                        = 0000002C
ACCVIO                            00000102 R       02        FWA_FIB_DESC                   = 00000024
ACM$Q_SYSTIME                   = 0000003C                   FWA_FID                        = 00000010
ARM$M_DELETE                    = 00000008                   FWA_FILE_SIZE                  = 0000001C
ASTADR                          = 00000018                   FWA_FILE_SPEC                  = 00000024
ASTPRM                          = 0000001C                   FWA_IOSB                       = 00000148
ATR$C_FILE_SPEC                 = 0000002E                   FWA_NAM                        = 0000006C
ATR$C_RECATTR                   = 00000004                   FWA_RECATTR                    = 00000124
ATR$S_RECATTR                   = 00000020                   FWA_SIZE                       = 000001CB
BADPARAM                          00000107 R       02        GETQUI_BOOL_ITEM                 00000020 R       02
BOOL_ITEM                       = 0000000C                   GETQUI_DATA                      000000D2 R       02
CTL$AL_STACKLIM                   ********   X     02        GETQUI_OUTPUT_ITEM               00000060 R       02
CTL$GL_PCB                        ********   X     02        GETQUI_SPECIAL_TABLE             000000B2 R       02
CTL$GL_PHD                        ********   X     02        INPUT_ITEM                       000001D6 R       02
CTL$T_USERNAME                    ********   X     02        INSFMEM                          0000010C R       02
DELETE_FLAG                     = 00000000                   IO$_ACCESS                     = 00000032
EFN                             = 00000004                   IOSB                           = 00000014
ERROR                             00000586 R       02        ITEM                             00000162 R       02
EXE$DEANONPAGED                   ********   X     02        ITMLST                         = 00000010
EXE$GETQUI                        00000114 RG      02        LNM$C_MAXDEPTH                 = 0000000A
EXE$GQ_SYSTIME                    ********   X     02        LNM$M_CASE_BLIND               = 02000000
EXE$JBCRSP                        000005BD RG      02        LNM$V_TERMINAL                 = 00000009
EXE$PROBER                        ********   X     02        LNM$_ATTRIBUTES                = 00000003
EXE$PROBEW                        ********   X     02        LNM$_STRING                    = 00000002
EXE$SENDMSG                       ********   X     02        LWA_ATTRBUF                    = 00000128
EXE$SNDJBC                        0000011C RG      02        LWA_BUFFER                     = 00000000
EXE$UPCASE_DAT                    ********   X     02        LWA_ITMLST                     = 00000108
FAB$B_BID                       = 00000000                   LWA_LOGNAM                     = 00000100
FAB$B_FNS                       = 00000034                   LWA_RSLLEN                     = 00000124
FAB$C_BID                       = 00000003                   LWA_SIZE                       = 0000012C
FAB$C_BLN                       = 00000050                   MAX_FUNC                       = 00000004
FAB$L_FNA                       = 0000002C                   MAX_ITEM                       = 00000008
FAB$L_NAM                       = 00000028                   MSG$_GETQUI                    = 00000010
FAT$L_EFBLK                     = 00000008                   MSG$_SNDJBC                    = 0000000F
FAT$W_FFBYTE                    = 0000000C                   MSG_CODE                       = 00000000
FIB$C_LENGTH                    = 00000040                   NAM$B_BID                      = 00000000
FIB$L_ALT_ACCESS                = 0000003C                   NAM$B_ESS                      = 0000000A
FIB$L_STATUS                    = 00000038                   NAM$C_BID                      = 00000002
FIB$M_ALT_REQ                   = 00000001                   NAM$C_BLN                      = 00000060
FIB$M_FINDFID                   = 00000800                   NAM$C_MAXRSS                   = 000000FF
FIB$W_DID                       = 0000000A                   NAM$L_ESA                      = 0000000C
FIB$W_FID                       = 00000004                   NAM$T_DVI                      = 00000014
FIB$W_NMCTL                     = 00000014                   NULARG                         = 0000000C
FILE_ID                         = FFFFFFFC                   OUTPUT_ITEM                    = 00000010
FILE_IDENTIFICATION               00000369 R       02        PCB$B_PRIB                     = 0000002F
```

```
PCB$L_EOWNER                   = 00000068
PCB$L_EPID                     = 00000064
PCB$L_PID                      = 00000060
PCB$L_STS                      = 00000024
PCB$L_UIC                      = 000000BC
PCB$T_TERMINAL                 = 00000044
PCB$W_ASTCNT                   = 00000038
PHD$L_IMGCNT                   = 000000F4
PHD$Q_PRIVMSK                  = 00000000
POSTPROCESS_FID                  000003A2 R      02
PSL$C_EXEC                     = 00000001
PSL$S_CURMOD                   = 00000002
PSL$S_PRVMOD                   = 00000002
PSL$V_CURMOD                   = 00000018
PSL$V_PRVMOD                   = 00000016
QUI$_RESERVED_FUNC_2           = 00000009
QUI$_RESERVED_OUTPUT_6         = 0000005E
QUI$_SEARCH_NAME               = 0000004D
SCH$CLREF                        ********   X    02
SCH$POSTEF                       ********   X    02
SCH$QAST                         ********   X    02
SJC$_CHARACTERISTIC_NAME       = 0000000C
SJC$_DELETE_FILE               = 00C00018
SJC$_DESTINATION_QUEUE         = 0000001A
SJC$_FILE_IDENTIFICATION       = 00000027
SJC$_FILE_SPECIFICATION        = 0000002A
SJC$_FORM_NAME                 = 00000036
SJC$_GENERIC_TARGET            = 00000046
SJC$_LOG_QUEUE                 = 00000061
SJC$_QUEUE                     = 00000086
SJC$_RESERVED_FUNC_2           = 00000020
SJC$_RESERVED_OUTPUT_2         = 000000AB
SNDJBC_BOOL_ITEM                 00000000 R      02
SNDJBC_DATA                      000000EA R      02
SNDJBC_OUTPUT_ITEM               00000040 R      02
SNDJBC_SPECIAL_TABLE             00000080 R      02
SPECIAL_TABLE                  = 00000014
SS$_ACCVIO                     = 0000000C
SS$_BADPARAM                   = 00000014
SS$_EXASTLM                    = 00002A04
SS$_INSFMEM                    = 00000124
SS$_IVLOGNAM                   = 00000154
SS$_NOLOGNAM                   = 000001BC
SYS$ASSIGN                       ********  GX    02
SYS$CMKRNL                       ********  GX    02
SYS$DASSGN                       ********  GX    02
SYS$DCLAST                       ********  GX    02
SYS$GL_JOBCTLMB                  ********   X    02
SYS$PARSE                        ********  GX    02
SYS$QIOW                         ********  GX    02
SYS$SEARCH                       ********  GX    02
SYS$SETEF                        ********   X    02
SYS$TRNLNM                       ********  GX    02
TRANSLATE_OBJECT                 000004B0 R      02
TRNLNM_ATTR                      000000BA R      02
TRNLNM_TABLE                     000000BE R      02
```

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

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           0000002E  (    46.)  01 (  1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
Y$EXEPAGED                      0000066F  ( 1647.)  02 (  2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE

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

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                      29    00:00:00.08    00:00:00.26
Command processing                 110    00:00:00.54    00:00:01.22
Pass 1                             566    00:00:23.66    00:00:27.07
Symbol table sort                    0    00:00:03.96    00:00:04.13
Pass 2                             208    00:00:04.77    00:00:05.38
Symbol table output                 20    00:00:00.18    00:00:00.20
Psect synopsis output                3    00:00:00.03    00:00:00.03
Cross-reference output               0    00:00:00.00    00:00:00.00
Assembler run totals               938    00:00:33.22    00:00:38.29

The working set limit was 1800 pages.
135859 bytes (266 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2520 non-local and 48 local symbols.
1137 source lines were read in Pass 1, producing 17 object records in Pass 2.
40 pages of virtual memory were used to define 38 macros.

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

Macro library name                           Macros defined
------------------                           --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       7
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   28
TOTALS (all libraries)                              35

2706 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

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

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY