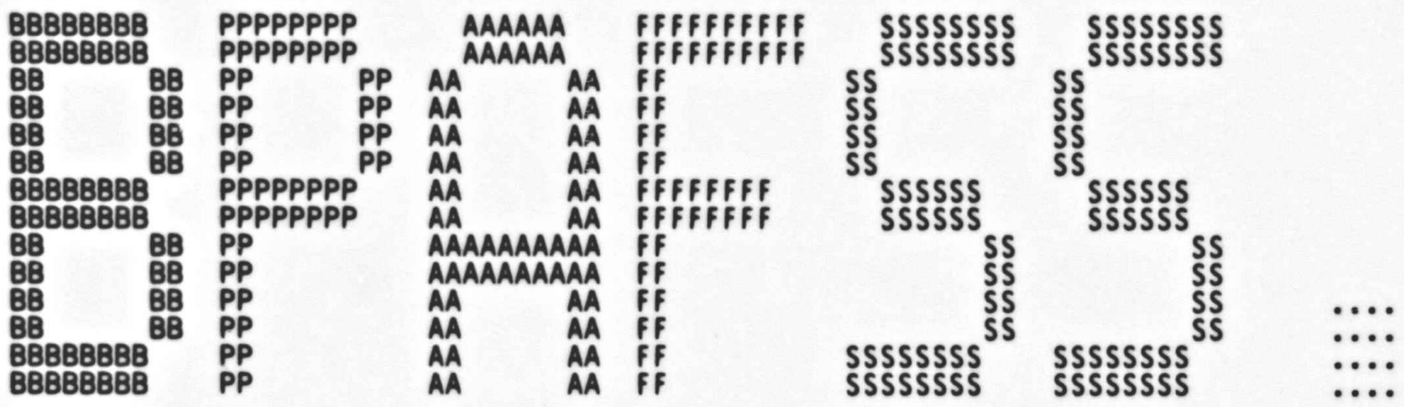
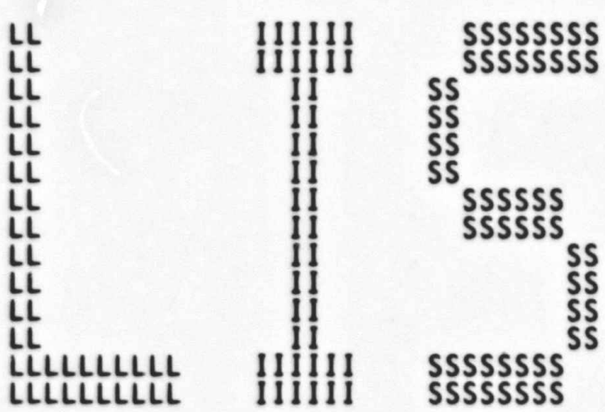
M 1

```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRRR     TTTTTTTTTTTTTTT LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRRR     TTTTTTTTTTTTTTT LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRRR     TTTTTTTTTTTTTTT LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT       LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT       LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT       LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT       LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT       LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT       LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRRR           TTT       LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRRR           TTT       LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRRR           TTT       LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR              TTT       LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR              TTT       LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR              TTT       LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR           TTT       LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR           TTT       LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR           TTT       LLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT       LLLLLLLLLLLLLLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT       LLLLLLLLLLLLLLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT       LLLLLLLLLLLLLLL
```

FILEID**BPAFSS





```
;     1          0001  0 MODULE bpa$fss (                       ! File string scan
;     2          0002  0                 IDENT = '1-006'       ! File: BPAFSS.B32, Edit: STAN1006
;     3          0003  0                 ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                             *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                             *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1 !<blf/uppercase_key>
;    30          0030  1 !<blf/lowercase_user>
;    31          0031  1 !
;    32          0032  1
;    33          0033  1 !++
;    34          0034  1 ! FACILITY: VAX-11 BASIC RTL SUPPORT
;    35          0035  1 !
;    36          0036  1 ! ABSTRACT:
;    37          0037  1 !
;    38          0038  1 !       Main routine BPA$FSS and associated routines parse a
;    39          0039  1 !       RSTS- or VMS-type filespec.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT: Native mode VAX processor, User mode.
;    42          0042  1 !
;    43          0043  1 ! AUTHOR: Jim Ibbett , CREATION DATE: 25-Oct-79
;    44          0044  1 !
;    45          0045  1 ! MODIFIED BY:
;    46          0046  1 !
;    47          0047  1 ! 1-002 - Repair version number, and interface to system build procedures.
;    48          0048  1 !         JBS 02-DEC-1979
;    49          0049  1 ! 1-003 - Replace MOVTUC builtin with a routine call, to avoid problems dealing
;    50          0050  1 !         directly with registers in BLISS.  JBS 03-DEC-1979
;    51          0051  1 ! 1-004 - Use RTL's standard PSECT names.  JBS 05-DEC-1979
;    52          0052  1 ! 1-005 - BPA$FSS should check if a wildcard was inserted in theFIRQB and
;    53          0053  1 !         not just if fqnam1 or fqnam2 is non-zero.  (RAD_50 wasnever
;    54          0054  1 !         being called, instead of being omitted just in the case
;    55          0055  1 !         of wildcards.)  Also, call RAD_50 twice, whether the filename is
;    56          0056  1 !         six characters or not, so that garbage isn't returned to the
;    57          0057  1 !         user.  PLL 20-Jun-81
```

```
;    58          0058  1 ! 1-006 - Remove informational error by making EXT_SIZE signed. STAN 24-Jul-1984.
;    59          0059  1 !
;    60          0060  1 !<blf/page>
```

```
;    62          0061  1 !
;    63          0062  1 ! SWITCHES:
;    64          0063  1 !
;    65          0064  1
;    66          0065  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    67          0066  1
;    68          0067  1 !
;    69          0068  1 ! TABLE OF CONTENTS:
;    70          0069  1 !
;    71          0070  1
;    72          0071  1 FORWARD ROUTINE
;    73          0072  1     bpa$fss,                                    ! Performs pre-parsing
;    74          0073  1     find_switch,                                ! Parses any switches
;    75          0074  1     rad_50,                                     ! Converts from ascii to rad50
;    76          0075  1     translate_dev,                              ! Convert RSTS device to VMS device
;    77          0076  1     bpa$file_scan;                              ! Performs parsing & fills O/P blocks
;    78          0077  1
;    79          0078  1 !
;    80          0079  1 ! INCLUDE FILES:
;    81          0080  1 !
;    82          0081  1
;    83          0082  1 LIBRARY 'RTLSTARLE';
;    84          0083  1
;    85          0084  1 REQUIRE 'RTLIN:BPASTRUCT';
;    86          0175  1
;    87          0176  1 REQUIRE 'RTLIN:BPAFSBDEF';
;    88          0307  1
;    89          0308  1 REQUIRE 'RTLIN:BPAFQBDEF';
;    90          0432  1
;    91          0433  1 REQUIRE 'RTLIN:RTLPSECT';
;    92          0528  1
;    93          0529  1 !
;    94          0530  1 ! MACROS:
;    95          0531  1 !
;    96          0532  1 !       NONE
;    97          0533  1 !
;    98          0534  1 ! EQUATED SYMBOLS:
;    99          0535  1 !
;   100          0536  1
;   101          0537  1 LITERAL
;   102          0538  1     bpa$k_def_prot = 60;                        ! Default protection code if none given
;   103          0539  1
;   104          0540  1 !
;   105          0541  1 ! PSECTS:
;   106          0542  1 !
;   107          0543  1 declare_psects (bpa);
;   108          0544  1 !
;   109          0545  1 ! OWN STORAGE:
;   110          0546  1 !
;   111          0547  1
;   112          0548  1 GLOBAL
;   113          0549  1     bpa$gb_usr_prot : BLOCK [1, BYTE],          ! default prot code
;   114          0550  1     bpa$gb_usr_real : BLOCK [1, BYTE],          ! prot_real if -1
;   115          0551  1     bpa$al_usrppn : VECTOR [2, LONG];           ! user ppn string descriptor
;   116          0552  1
;   117          0553  1 !
;   118          0554  1 ! EXTERNAL REFERENCES:
```

```
;  119          0555  1 !
;  120          0556  1
;  121          0557  1 EXTERNAL ROUTINE
;  122          0558  1     bpa$get_block,
;  123          0559  1     bpa$free_block,
;  124          0560  1     bpa$$movtuc : NOVALUE;
;  125          0561  1
;  126          0562  1 !+
;  127          0563  1 ! The following are the error codes used in this module:
;  128          0564  1 !-
;  129          0565  1
;  130          0566  1 EXTERNAL LITERAL
;  131          0567  1     bas$_illnum,                        ! Illegal number
;  132          0568  1     bas$_illswiusa,                     ! Illegal switch usage
;  133          0569  1     bas$_illfilnam;                     ! Illegal file name
;  134          0570  1
```

```
;  136     0571  1 GLOBAL ROUTINE bpa$fss (fqb_ptr, fsb_ptr, buf_ptr, byt_cnt) =
;  137     0572  1
;  138     0573  1 !++
;  139     0574  1 ! FUNCTIONAL DESCRIPTION:
;  140     0575  1 !
;  141     0576  1 !       On entry, a filespec is described by buf_ptr/byt_cnt.
;  142     0577  1 !       The string is pre-parsed (remove nulls/spaces, convert
;  143     0578  1 !       lowercase to uppercase, strip parity, etc) and a
;  144     0579  1 !       call made to BPA$FILE_SCAN which parses the string.
;  145     0580  1 !       On return, any switches are decoded and the FLAGWORDS,
;  146     0581  1 !       FIRQB and FSB are loaded. Status is returned to
;  147     0582  1 !       the caller.
;  148     0583  1 !
;  149     0584  1 ! FORMAL PARAMETERS:
;  150     0585  1 !
;  151     0586  1 !       fqb_ptr.rla.v  -  longword address of FIRQB
;  152     0587  1 !       fsb_ptr.rla.v  -  longword address of FSB
;  153     0588  1 !       buf_ptr.rla.v  -  longword address of filespec string
;  154     0589  1 !       byt_cnt.rlu.v  -  contains the length of the filespec string
;  155     0590  1 !
;  156     0591  1 ! IMPLICIT INPUTS:
;  157     0592  1 !
;  158     0593  1 !       bpa$al_usrppn    - descriptor for user ppn string
;  159     0594  1 !       bpa$gb_usr_prot  - default prot code
;  160     0595  1 !       bpa$gb_usr_real  - non-zero to indicate bpa$gb_usr_prot has
;  161     0596  1 !                          been initialised
;  162     0597  1 !
;  163     0598  1 ! IMPLICIT OUTPUTS:
;  164     0599  1 !
;  165     0600  1 !       FQB  -  the FIRQB, contains information as for
;  166     0601  1 !               the RSTS/E V7 .FSS call
;  167     0602  1 !
;  168     0603  1 !       FSB  -  contains descriptor for the parsed string
;  169     0604  1 !               and 'XRB' flagwords, as shown below.
;  170     0605  1 !               (see $fsb_def for details of fields/bits)
;  171     0606  1 !
;  172     0607  1 !
;  173     0608  1 !       32              24              16              8               0
;  174     0609  1 !       +---------------+---------------+---------------+---------------+
;  175     0610  1 !       !                          filespec length                      !   0
;  176     0611  1 !       +---------------+---------------+---------------+---------------+
;  177     0612  1 !       !                          filespec address                     !   4
;  178     0613  1 !       +---------------+---------------+---------------+---------------+
;  179     0614  1 !       !        position value         !   file_off    !   prot_real   !   8
;  180     0615  1 !       +---------------+---------------+---------------+---------------+
;  181     0616  1 !       !   size MSB    !          size LSB             !   prot code   !  12
;  182     0617  1 !       +---------------+---------------+---------------+---------------+
;  183     0618  1 !       !       clustersize value       !            mode value         !  16
;  184     0619  1 !       +---------------+---------------+---------------+---------------+
;  185     0620  1 !       ! dev_name 2    ! dev_name 1    ! dev_unit #    !   unit_real   !  20
;  186     0621  1 !       +---------------+---------------+---------------+---------------+
;  187     0622  1 !       !               status bits    (from NAM$L_FNB)                 !  24
;  188     0623  1 !       +---------------+---------------+---------------+---------------+
;  189     0624  1 !       ! offset to /   ! offset to <   ! offset to :   ! offset to ::  !  28
;  190     0625  1 !       +---------------+---------------+---------------+---------------+
;  191     0626  1 !       !       'XRB' flagword 1        !       'XRB' flagword 2        !  32
;  192     0627  1 !       +---------------+---------------+---------------+---------------+
```

```
;  193          0628  1 !
;  194          0629  1 !
;  195          0630  1 ! ROUTINE VALUE:
;  196          0631  1 !
;  197          0632  1 !       Errors as follows:
;  198          0633  1 !
;  199          0634  1 !       a) bas$_illfilnam     if parse error
;  200          0635  1 !       b) bas$_illnum        if illegal protection or switch value
;  201          0636  1 !       c) bas$_illswiusa     if illegal switch
;  202          0637  1 !
;  203          0638  1 !       else returns ss$_normal for success.
;  204          0639  1 !
;  205          0640  1 ! SIDE EFFECTS:
;  206          0641  1 !
;  207          0642  1 !       Allocates a block of dynamic memory for output string if
;  208          0643  1 !       successful. This block must be returned by the caller
;  209          0644  1 !       using BPA$FREE_BLOCK.
;  210          0645  1 !
;  211          0646  1 !--
;  212          0647  1
;  213          0648  2     BEGIN
;  214          0649  2
;  215          0650  2     MAP
;  216          0651  2         fqb_ptr : REF $fqb_def,                 ! address of FIRQB
;  217          0652  2         fsb_ptr : REF $fsb_def,                 ! address of FSB
;  218          0653  2         buf_ptr : REF VECTOR [, BYTE];          ! address of string buffer
;  219          0654  2
;  220          0655  2     LOCAL
;  221          0656  2         op_ptr : REF VECTOR [, BYTE],           ! address of MOVTUC dst buffer
;  222          0657  2         string : REF VECTOR [, BYTE],           ! input string work buffer
;  223          0658  2         copy : REF VECTOR [, BYTE],             ! xlated string buffer for $parse
;  224          0659  2         EXPAND : REF VECTOR [, BYTE],           ! expanded string  buffer
;  225          0660  2         buffer_address : REF VECTOR [, BYTE],   ! user I/P string buffer
;  226          0661  2         ctlr_val : WORD,                        ! value of VMS ctlr #
;  227          0662  2         unit_val : WORD,                        ! value of VMS unit #
;  228          0663  2         prot_val : WORD,                        ! calculated prot code value
;  229          0664  2         ppn_val : WORD,                         ! calculated ppn value
;  230          0665  2         xlate_length : BYTE,                    ! length of xlated string
;  231          0666  2         device_size : BYTE,                     ! length of device name
;  232          0667  2         file_size : BYTE,                       ! length of filename
;  233          0668  2         ext_size : SIGNED BYTE,                 ! length of ext
;  234          0669  2         log_dir_len : BYTE,                     ! logical directory string length
;  235          0670  2         node_ptr,                               ! offset to '::'
;  236          0671  2         dev_ptr,                                ! offset to ':'
;  237          0672  2         usr_dev_ptr,                            ! offset to user spec'd device
;  238          0673  2         ppn_ptr,                                ! offset to '[' or '<ppn'
;  239          0674  2         ppn_end_ptr,                            ! offset to 'ppn>'
;  240          0675  2         dot_ptr,                                ! offset to filename
;  241          0676  2         ver_ptr,                                ! offset to ';'
;  242          0677  2         prot_ptr,                               ! offset to '<prot'
;  243          0678  2         swit_ptr,                               ! offset to '/'
;  244          0679  2         end_part1,                              ! offset to end of VMS-type string
;  245          0680  2         next_posn,                              ! offset to next scan position
;  246          0681  2         next_char,                              ! offset to next char to examine
;  247          0682  2         log_dir_addr,                           ! logical directory string address
;  248          0683  2         parsed_address,                         ! address of parsed string
;  249          0684  2         parsed_length,                          ! length of parsed string
```

```
;  250          0685  2          parsed_end,                     ! offset to end (';' or length)
;  251          0686  2          dst_length,                     ! length of MOVTUC dst buffer
;  252          0687  2          byte_count,                     ! length of user I/P string
;  253          0688  2          sts;                            ! return status from routines
;  254          0689  2
;  255          0690  2      BIND
;  256          0691  2  !+
;  257          0692  2  ! The table below is a translation table and is used to pre-parse
;  258          0693  2  ! the incoming user string. The method of operation is that there
;  259          0694  2  ! are 256 characters in the table, associated with the 256 ascii
;  260          0695  2  ! character codes. Thus if an incoming character has the ascii
;  261          0696  2  ! code octal 50 (decimal 40 - a '(') it will be translated to the
;  262          0697  2  ! character at offset 40 in the table, in this case a '['.
;  263          0698  2  ! This table is set up to perform a number of conversions,
;  264          0699  2  ! specifically strip parity, change lower to upper case, change
;  265          0700  2  ! '()' to '[]', and change all non-printing characters to '0'.
;  266          0701  2  ! It is used in conjunction with the MOVTUC instruction,
;  267          0702  2  ! which is explained later in this code.
;  268          0703  2  !-
;  269          0704  2          table = CH$TRANSTABLE (                 ! translation table
;  270          0705  2          REP 33 OF (%O'0'),
;  271          0706  2          %C'!', %C'"', %C'#', %C'$', %C'%', %C'&', %C''', %C'[',
;  272          0707  2          %C']', %C'*', %C'+', %C',', %C'-', %C'.', %C'/', %C'0',
;  273          0708  2          %C'1', %C'2', %C'3', %C'4', %C'5', %C'6', %C'7', %C'8',
;  274          0709  2          %C'9', %C':', %C';', %C'<', %C'=', %C'>', %C'?', %C'@',
;  275          0710  2          %C'A', %C'B', %C'C', %C'D', %C'E', %C'F', %C'G', %C'H',
;  276          0711  2          %C'I', %C'J', %C'K', %C'L', %C'M', %C'N', %C'O', %C'P',
;  277          0712  2          %C'Q', %C'R', %C'S', %C'T', %C'U', %C'V', %C'W', %C'X',
;  278          0713  2          %C'Y', %C'Z', %C'[', %C'\', %C']', %C'^', %C'_', %C'`',
;  279          0714  2          %C'A', %C'B', %C'C', %C'D', %C'E', %C'F', %C'G', %C'H',
;  280          0715  2          %C'I', %C'J', %C'K', %C'L', %C'M', %C'N', %C'O', %C'P',
;  281          0716  2          %C'Q', %C'R', %C'S', %C'T', %C'U', %C'V', %C'W', %C'X',
;  282          0717  2          %C'Y', %C'Z', %C'{', %C'|', %C'}', %C'~', %O'0',
;  283          0718  2          REP 33 OF (%O'0'),
;  284          0719  2          %C'!', %C'"', %C'#', %C'$', %C'%', %C'&', %C''', %C'[',
;  285          0720  2          %C']', %C'*', %C'+', %C',', %C'-', %C'.', %C'/', %C'0',
;  286          0721  2          %C'1', %C'2', %C'3', %C'4', %C'5', %C'6', %C'7', %C'8',
;  287          0722  2          %C'9', %C':', %C';', %C'<', %C'=', %C'>', %C'?', %C'@',
;  288          0723  2          %C'A', %C'B', %C'C', %C'D', %C'E', %C'F', %C'G', %C'H',
;  289          0724  2          %C'I', %C'J', %C'K', %C'L', %C'M', %C'N', %C'O', %C'P',
;  290          0725  2          %C'Q', %C'R', %C'S', %C'T', %C'U', %C'V', %C'W', %C'X',
;  291          0726  2          %C'Y', %C'Z', %C'[', %C'\', %C']', %C'^', %C'_', %C'`',
;  292          0727  2          %C'A', %C'B', %C'C', %C'D', %C'E', %C'F', %C'G', %C'H',
;  293          0728  2          %C'I', %C'J', %C'K', %C'L', %C'M', %C'N', %C'O', %C'P',
;  294          0729  2          %C'Q', %C'R', %C'S', %C'T', %C'U', %C'V', %C'W', %C'X',
;  295          0730  2          %C'Y', %C'Z', %C'{', %C'|', %C'}', %C'~', %O'0');
;  296          0731  2
;  297          0732  2  !+
;  298          0733  2  ! logical directory translation strings for $, #, ! and &
;  299          0734  2  !-
;  300          0735  2
;  301          0736  2      BIND
;  302          0737  2          dollar_string = UPLIT BYTE('SYS$SYSTEM:'),
;  303          0738  2          hash_string = UPLIT BYTE('BPA$DIR_HASH:'),
;  304          0739  2          exclam_string = UPLIT BYTE('BPA$DIR_EXCLAM:'),
;  305          0740  2          amper_string = UPLIT BYTE('BPA$DIR_AMPER:'),
;  306          0741  2          end_string = UPLIT BYTE('0');
```

```
;  307          0742  2
;  308          0743  2 !+
;  309          0744  2 ! logical directory string lengths (from above)
;  310          0745  2 !-
;  311          0746  2
;  312          0747  2     LITERAL
;  313          0748  2         dollar_string_l = hash_string - dollar_string,
;  314          0749  2         hash_string_l = exclam_string - hash_string,
;  315          0750  2         exclam_string_l = amper_string - exclam_string,
;  316          0751  2         amper_string_l = end_string - amper_string;
;  317          0752  2
;  318          0753  2 !+
;  319          0754  2 ! Get parameters
;  320          0755  2 ! If length is zero, just return what we can to user.
;  321          0756  2 !-
;  322          0757  2
;  323          0758  2     IF .byt_cnt GTRU 0
;  324          0759  2     THEN
;  325          0760  3         BEGIN
;  326          0761  3         byte_count = .byt_cnt;
;  327          0762  3         buffer_address = .buf_ptr;
;  328          0763  3         END
;  329          0764  2     ELSE
;  330          0765  3         BEGIN
;  331          0766  3         CH$FILL (0, fsb$k_length, .fsb_ptr);
;  332          0767  3         fsb_ptr [fsb$a_fsa] = .buf_ptr;
;  333          0768  3         fsb_ptr [fsb$l_fsl] = 0;
;  334          0769  3         RETURN ss$_normal;
;  335          0770  2         END;
;  336          0771  2
;  337          0772  2 !+
;  338          0773  2 ! Copy string to our buffer and use translate table to
;  339          0774  2 ! convert lower- to upper-case, lose any spaces, nulls
;  340          0775  2 ! and non-printing characters, and convert () to [].
;  341          0776  2 !-
;  342          0777  2
;  343          0778  2     IF NOT (sts = bpa$get_block (.byte_count, string)) THEN RETURN SIGNAL (.sts);
;  344          0779  2
;  345          0780  2 !+
;  346          0781  2 ! The following code performs the translation or pre-parsing
;  347          0782  2 ! of the user string, via the translation table explained
;  348          0783  2 ! previously. It uses the MOVTUC (MOVe Translated Until
;  349          0784  2 ! Character) VAX machine instruction, which operates as follows:-
;  350          0785  2 ! Each incoming character is translated via the table. If
;  351          0786  2 ! the translation results in the character being converted to
;  352          0787  2 ! a code of 0, MOVTUC will complete with R0, R4 and R5
;  353          0788  2 ! containing values as described in the architecture handbook.
;  354          0789  2 ! The rogue character is skipped over (lost) and MOVTUC continued.
;  355          0790  2 ! This all continues until 'bytes_to_move' is zero (i.e. all translated).
;  356          0791  2 ! When all is done, the vector 'string' contains the final
;  357          0792  2 ! translated string.
;  358          0793  2 !-
;  359          0794  2     op_ptr = .string;                       ! O/P buffer address
;  360          0795  2     dst_length = nam$c_maxrss;              ! O/P buffer length
;  361          0796  2
;  362          0797  2     WHILE .byte_count GTRU 0 DO
;  363          0798  3         BEGIN
```

```
;  364     0799  3
;  365     0800  3          LOCAL
;  366     0801  3              esc_flag,                       ! 1 if MOVTUC stopped by escape
;  367     0802  3              esc_char,                       ! pointer to escape character
;  368     0803  3              bytes_to_move,                  ! number of chars not Xlated
;  369     0804  3              bytes_left,                     ! bytes left in O/P area
;  370     0805  3              next_dst_addr;                  ! addr of next byte in O/P area
;  371     0806  3
;  372     0807  3          bpa$$movtuc (byte_count, .buffer_address, %REF (0), table, dst_length, .op_ptr, esc_flag,
;  373     0808  3              bytes_to_move, esc_char, bytes_left, next_dst_addr);
;  374     0809  3          op_ptr = .next_dst_addr;            ! where to put next char
;  375     0810  3
;  376     0811  3          IF .esc_flag NEQU 0
;  377     0812  3          THEN                                ! '0' seen, so step over it & continue...
;  378     0813  4              BEGIN
;  379     0814  4              buffer_address = .esc_char + 1; ! next I/P char address
;  380     0815  4              byte_count = .bytes_to_move - 1; ! # of bytes left to do
;  381     0816  4              dst_length = .bytes_left;       ! # of bytes left in O/P buffer
;  382     0817  4              END
;  383     0818  3          ELSE
;  384     0819  3              byte_count = 0;                 ! All Xlated - force exit from 'WHILE...'
;  385     0820  3
;  386     0821  3          END;
;  387     0822  2
;  388     0823  2      xlate_length = .op_ptr - .string;       ! length of Xlated string
;  389     0824  2  !+
;  390     0825  2  ! Initialise fsb
;  391     0826  2  !-
;  392     0827  2      CH$FILL (0, fsb$k_length, .fsb_ptr);
;  393     0828  2      fsb_ptr [fsb$a_fsa] = .string;
;  394     0829  2      fsb_ptr [fsb$l_fsl] = .xlate_length;
;  395     0830  2  !+
;  396     0831  2  ! Find and mark any '::'
;  397     0832  2  !-
;  398     0833  2      node_ptr = CH$FIND_SUB (.xlate_length, .string, 2, UPLIT BYTE('::'));
;  399     0834  2
;  400     0835  2      IF .node_ptr NEQU 0
;  401     0836  2      THEN
;  402     0837  3          BEGIN
;  403     0838  3          node_ptr = .node_ptr - .string;
;  404     0839  3          fsb_ptr [fsb$b_node_off] = .node_ptr;
;  405     0840  3          usr_dev_ptr = .node_ptr + 2;
;  406     0841  3          END
;  407     0842  2      ELSE
;  408     0843  3          BEGIN
;  409     0844  3          fsb_ptr [fsb$b_node_off] = 0;
;  410     0845  3          usr_dev_ptr = 0;
;  411     0846  2          END;
;  412     0847  2
;  413     0848  2  !+
;  414     0849  2  ! Find and mark any ':' (following any '::')
;  415     0850  2  ! NOTE: the colon MAY relate to a switch, rather
;  416     0851  2  ! than be a device delimeter. We assume it is the
;  417     0852  2  ! later for now, and check its offset when we look
;  418     0853  2  ! for a '/'.
;  419     0854  2  !-
;  420     0855  2
```

```
;  421          0856  2      IF .fsb_ptr [fsb$b_node_off] NEQU 0
;  422          0857  2      THEN
;  423          0858  2          dev_ptr = CH$FIND_CH (.xlate_length - .fsb_ptr [fsb$b_node_off] - 2,
;  424          0859  2              .string + .fsb_ptr [fsb$b_node_off] + 2, %C':')
;  425          0860  2      ELSE
;  426          0861  2          dev_ptr = CH$FIND_CH (.xlate_length, .string, %C':');
;  427          0862  2
;  428          0863  2      IF .dev_ptr NEQU 0 THEN fsb_ptr [fsb$b_dev_off] = .dev_ptr - .string ELSE fsb_ptr [fsb$b_dev_off] = 0;
;  429          0864  2
;  430          0865  2  !+
;  431          0866  2  ! Find '<x' where x is numeric (i.e. protection)
;  432          0867  2  !-
;  433          0868  2      prot_ptr = CH$FIND_CH (.xlate_length, .string, %C'<');
;  434          0869  2
;  435          0870  2      IF .prot_ptr NEQU 0
;  436          0871  2      THEN                                    ! we found a '<', see if numeric follows
;  437          0872  3          BEGIN
;  438          0873  3          prot_ptr = .prot_ptr - .string;
;  439          0874  3
;  440          0875  4          IF (.string [.prot_ptr + 1] GEQU %C'0' AND .string [.prot_ptr + 1] LEQU %C'9')
;  441          0876  3          THEN                                ! '<prot>' found
;  442          0877  3              fsb_ptr [fsb$b_prot_off] = .prot_ptr
;  443          0878  3          ELSE
;  444          0879  3  !+
;  445          0880  3  ! '<' found was (probably) a directory, so scan for another '<'
;  446          0881  3  !-
;  447          0882  4              BEGIN
;  448          0883  4              prot_ptr = CH$FIND_CH (.xlate_length - .prot_ptr - 1, .string + .prot_ptr + 1, %C'<');
;  449          0884  4
;  450          0885  4              IF .prot_ptr NEQU 0
;  451          0886  4              THEN                            ! '<' found
;  452          0887  5                  BEGIN
;  453          0888  5                  prot_ptr = .prot_ptr - .string;
;  454          0889  5
;  455          0890  6                  IF (.string [.prot_ptr + 1] GEQU %C'0' AND .string [.prot_ptr + 1] LEQU %C'9')
;  456          0891  5                  THEN                        ! '<prot>' found
;  457          0892  5                      fsb_ptr [fsb$b_prot_off] = .prot_ptr
;  458          0893  5                  ELSE
;  459          0894  6                      BEGIN
;  460          0895  6                      bpa$free_block (.string, .byte_count);
;  461          0896  6                      RETURN SIGNAL (bas$_illnum);
;  462          0897  5                      END;
;  463          0898  5
;  464          0899  4                  END;
;  465          0900  4
;  466          0901  3              END;
;  467          0902  3
;  468          0903  2          END;
;  469          0904  2
;  470          0905  2  !+
;  471          0906  2  ! Find first '/' (if any)
;  472          0907  2  ! NOTE: If we found a ':' previously, we compare the
;  473          0908  2  ! offsets of the colon & slash. If the colon offset is
;  474          0909  2  ! greater then it could not have been a device
;  475          0910  2  ! delimeter so we zero its stored offset.
;  476          0911  2  !-
;  477          0912  2      swit_ptr = CH$FIND_CH (.xlate_length, .string, %C'/');
```

```
;  478          0913  2
;  479          0914  2      IF .swit_ptr NEQU 0
;  480          0915  2      THEN
;  481          0916  2          BEGIN
;  482          0917  2          fsb_ptr [fsb$b_swit_off] = .swit_ptr - .string;
;  483          0918  2
;  484          0919  2          IF .fsb_ptr [fsb$b_dev_off] GTRU .fsb_ptr [fsb$b_swit_off] THEN fsb_ptr [fsb$b_dev_off] = 0;
;  485          0920  2
;  486          0921  2          END
;  487          0922  2      ELSE
;  488          0923  2          fsb_ptr [fsb$b_swit_off] = 0;
;  489          0924  2
;  490          0925  2  !+
;  491          0926  2  ! Get ptr to end of VMS-type filespec and check that
;  492          0927  2  ! the <prot> is before the /switch (if both present)
;  493          0928  2  !-
;  494          0929  2
;  495          0930  3      IF (.fsb_ptr [fsb$b_prot_off] NEQU 0 AND .fsb_ptr [fsb$b_swit_off] NEQU 0)
;  496          0931  2      THEN                                        ! both present
;  497          0932  2
;  498          0933  2          IF .fsb_ptr [fsb$b_prot_off] LSSU .fsb_ptr [fsb$b_swit_off]
;  499          0934  2          THEN                                    ! '<' before '/'  OK
;  500          0935  2              end_part1 = .fsb_ptr [fsb$b_prot_off]
;  501          0936  2          ELSE
;  502          0937  3              BEGIN
;  503          0938  3              bpa$free_block (.string, .byte_count);
;  504          0939  3              RETURN SIGNAL (bas$_illswiusa);     ! '/' before '<'  ERROR
;  505          0940  3              END
;  506          0941  3
;  507          0942  2      ELSE
;  508          0943  2
;  509          0944  2          IF .fsb_ptr [fsb$b_prot_off] NEQU 0
;  510          0945  2          THEN                                    ! only '<' present  OK
;  511          0946  2              end_part1 = .fsb_ptr [fsb$b_prot_off]
;  512          0947  2          ELSE
;  513          0948  2
;  514          0949  2              IF .fsb_ptr [fsb$b_swit_off] NEQU 0
;  515          0950  2              THEN                                ! only '/' present  OK
;  516          0951  2                  end_part1 = .fsb_ptr [fsb$b_swit_off]
;  517          0952  2              ELSE
;  518          0953  2                  end_part1 = .xlate_length;      ! neither  -  end = length
;  519          0954  2
;  520          0955  2  !+
;  521          0956  2  ! Find position of next char after '::' or ':'
;  522          0957  2  ! (whichever is the greater)
;  523          0958  2  !-
;  524          0959  2
;  525          0960  2      IF .fsb_ptr [fsb$b_dev_off] GTRU .fsb_ptr [fsb$b_node_off]
;  526          0961  2      THEN
;  527          0962  2          next_posn = .fsb_ptr [fsb$b_dev_off]
;  528          0963  2      ELSE
;  529          0964  2
;  530          0965  3          IF (.fsb_ptr [fsb$b_node_off] NEQU 0)   !
;  531          0966  2          THEN
;  532          0967  2              next_posn = .fsb_ptr [fsb$b_node_off] + 1
;  533          0968  2          ELSE
;  534          0969  2              next_posn = 0;
```

```
;   535          0970  2
;   536          0971  2      next_char = .next_posn;
;   537          0972  2
;   538          0973  2      IF .next_char NEQU 0 THEN next_char = .next_char + 1;
;   539          0974  2
;   540          0975  2  !+
;   541          0976  2  ! Check integrity of [...] or <...> if present, and
;   542          0977  2  ! make next_char the char after the ']' or '>'
;   543          0978  2  !-
;   544          0979  2
;   545          0980  2      SELECTONEU .string [.next_char] OF
;   546          0981  2          SET
;   547          0982  2
;   548          0983  2          [%C'['] :
;   549          0984  3              BEGIN
;   550          0985  3              next_posn = CH$FIND_CH (.xlate_length - .next_char, .string + .next_char, %C']');
;   551          0986  3
;   552          0987  3              IF .next_posn NEQU 0
;   553          0988  3              THEN
;   554          0989  4                  BEGIN
;   555          0990  4                  next_posn = .next_posn - .string;
;   556          0991  4                  next_char = .next_posn + 1;
;   557          0992  4                  END
;   558          0993  3              ELSE
;   559          0994  4                  BEGIN
;   560          0995  4                  bpa$free_block (.string, .byte_count);
;   561          0996  4                  RETURN SIGNAL (bas$_illfilnam);
;   562          0997  3                  END;
;   563          0998  3
;   564          0999  2              END;
;   565          1000  2
;   566          1001  2          [%C'<'] :
;   567          1002  3              BEGIN
;   568          1003  3              next_posn = CH$FIND_CH (.xlate_length - .next_char, .string + .next_char, %C'>');
;   569          1004  3
;   570          1005  3              IF .next_posn EQLU 0
;   571          1006  3              THEN
;   572          1007  4                  BEGIN
;   573          1008  4                  bpa$free_block (.string, .byte_count);
;   574          1009  4                  RETURN SIGNAL (bas$_illfilnam);
;   575          1010  4                  END
;   576          1011  3              ELSE
;   577          1012  3
;   578          1013  4                  IF (.string [.next_char + 1] GEQU %C'A' AND .string [.next_char + 1] LEQU %C'Z')
;   579          1014  3                  THEN
;   580          1015  4                      BEGIN
;   581          1016  4                      next_posn = .next_posn - .string;
;   582          1017  4                      next_char = .next_posn + 1;
;   583          1018  3                      END;
;   584          1019  3
;   585          1020  2              END;
;   586          1021  2
;   587          1022  2          [OTHERWISE] :
;   588          1023  2              0;                                  ! nothing to do
;   589          1024  2          TES;
;   590          1025  2
;   591          1026  2  !+
```

```
;  592          1027  2 ! Set NFREP bit if a '%' in source string
;  593          1028  2 !-
;  594          1029  2     fsb_ptr [fsb$b_file_off] = .next_char;
;  595          1030  2     next_posn = .next_char;
;  596          1031  2
;  597          1032  3     IF (CH$FIND_CH (.end_part1 - .next_posn, .string + .next_posn, %C'%') NEQU 0)
;  598          1033  2     THEN
;  599          1034  2         fqb_ptr [fqb$b_nfrep] = fqb$k_nfrep;
;  600          1035  2
;  601          1036  2 !+
;  602          1037  2 ! Check to see if both '?' and '%' are present. If so error, else
;  603          1038  2 ! convert any '?' to '%'
;  604          1039  2 !-
;  605          1040  2
;  606          1041  3     IF (CH$FIND_CH (.end_part1 - .next_posn, .string + .next_posn, %C'?') NEQU 0)
;  607          1042  2     THEN
;  608          1043  3         BEGIN
;  609          1044  3
;  610          1045  4         IF (CH$FIND_CH (.end_part1 - .next_posn, .string + .next_posn, %C'%') NEQU 0)
;  611          1046  3         THEN
;  612          1047  4             BEGIN
;  613          1048  4             bpa$free_block (.string, .byte_count);
;  614          1049  4             RETURN SIGNAL (bas$_illfilnam);
;  615          1050  4             END
;  616          1051  3         ELSE
;  617          1052  4             BEGIN
;  618          1053  4
;  619          1054  4             LOCAL
;  620          1055  4                 ptr;
;  621          1056  4
;  622          1057  4             UNTIL CH$FAIL (ptr = CH$FIND_CH (.end_part1 - .next_posn, .string + .next_posn, %C'?')) DO
;  623          1058  4                 CH$WCHAR (%C'%', .ptr);
;  624          1059  4
;  625          1060  3             END;
;  626          1061  3
;  627          1062  2         END;
;  628          1063  2
;  629          1064  2 !+
;  630          1065  2 ! See if the next char is a logical directory specifier.
;  631          1066  2 ! If so, load the required translation string address/length
;  632          1067  2 ! and point next_posn to the following char
;  633          1068  2 !-
;  634          1069  2
;  635          1070  2     SELECTONEU .string [.next_char] OF
;  636          1071  2         SET
;  637          1072  2
;  638          1073  2         [%C'$'] :
;  639          1074  3             BEGIN
;  640          1075  3             log_dir_addr = dollar_string;
;  641          1076  3             log_dir_len = dollar_string_l;
;  642          1077  3             next_posn = .next_posn + 1;
;  643          1078  2             END;
;  644          1079  2
;  645          1080  2         [%C'#'] :
;  646          1081  3             BEGIN
;  647          1082  3             log_dir_addr = hash_string;
;  648          1083  3             log_dir_len = hash_string_l;
```

```
;  649          1084  3              next_posn = .next_posn + 1;
;  650          1085  2              END;
;  651          1086  2
;  652          1087  2          [%C'!'] :
;  653          1088  3              BEGIN
;  654          1089  3              log_dir_addr = exclam_string;
;  655          1090  3              log_dir_len = exclam_string_l;
;  656          1091  3              next_posn = .next_posn + 1;
;  657          1092  2              END;
;  658          1093  2
;  659          1094  2          [%C'&'] :
;  660          1095  3              BEGIN
;  661          1096  3              log_dir_addr = amper_string;
;  662          1097  3              log_dir_len = amper_string_l;
;  663          1098  3              next_posn = .next_posn + 1;
;  664          1099  2              END;
;  665          1100  2
;  666          1101  2          [%C'@'] :
;  667          1102  3              BEGIN
;  668          1103  3              log_dir_addr = .bpa$al_usrppn [1];
;  669          1104  3              log_dir_len = .bpa$al_usrppn [0];
;  670          1105  3              next_posn = .next_posn + 1;
;  671          1106  3
;  672          1107  3              IF .log_dir_addr EQLU 0
;  673          1108  3              THEN
;  674          1109  4                  BEGIN
;  675          1110  4                  log_dir_addr = end_string;
;  676          1111  4                  log_dir_len = 0;
;  677          1112  3                  END;
;  678          1113  3
;  679          1114  2              END;
;  680          1115  2
;  681          1116  2          [OTHERWISE] :
;  682          1117  2              log_dir_len = 0;
;  683          1118  2          TES;
;  684          1119  2
;  685          1120  2      !+
;  686          1121  2      ! If the above 5 chars appear further on in the string, then error
;  687          1122  2      !-
;  688          1123  2
;  689          1124  3          IF (CH$FIND_CH (.end_part1 - .next_posn, .string + .next_posn, %C'$') NEQU 0)        !
;  690          1125  3              OR (CH$FIND_CH (.end_part1 - .next_posn, .string + .next_posn, %C'#') NEQU 0)    !
;  691          1126  3              OR (CH$FIND_CH (.end_part1 - .next_posn, .string + .next_posn, %C'!') NEQU 0)    !
;  692          1127  3              OR (CH$FIND_CH (.end_part1 - .next_posn, .string + .next_posn, %C'&') NEQU 0)    !
;  693          1128  3              OR (CH$FIND_CH (.end_part1 - .next_posn, .string + .next_posn, %C'@') NEQU 0)
;  694          1129  2          THEN
;  695          1130  3              BEGIN
;  696          1131  3              bpa$free_block (.string, .byte_count);
;  697          1132  3              RETURN SIGNAL (bas$_illfilnam);
;  698          1133  2              END;
;  699          1134  2
;  700          1135  2      !+
;  701          1136  2      ! If we found a logical directory specifier, then merge the
;  702          1137  2      ! translation with the rest of the string
;  703          1138  2      ! Get the merge buffer from free core
;  704          1139  2      ! If the logical_directory_length is zero, just copy
;  705          1140  2      ! 'string' to 'copy' and leave 'lengths' as they are.
```

```
;   706          1141  2 !-
;   707          1142  2
;   708          1143  3     IF NOT (sts = bpa$get_block (nam$c_maxrss, copy))
;   709          1144  2     THEN
;   710          1145  3         BEGIN
;   711          1146  3         bpa$free_block (.string, .byte_count);
;   712          1147  3         RETURN SIGNAL (.sts);
;   713          1148  2         END;
;   714          1149  2
;   715          1150  2     IF .log_dir_len NEQU 0
;   716          1151  2     THEN
;   717          1152  3         BEGIN
;   718          1153  3         CH$COPY (.next_posn - 1, .string,          ! 1st bit upto specifier
;   719          1154  3             .log_dir_len, .log_dir_adar,           ! translation string
;   720          1155  3             .end_part1 - .next_posn, .string + .next_posn,      ! last bit after specifier
;   721          1156  3             0,                                     ! fill char
;   722          1157  3             nam$c_maxrss, .copy);                  ! destination
;   723          1158  3         xlate_length = .xlate_length + .log_dir_len - 1;        ! adjust total length
;   724          1159  3         end_part1 = .end_part1 + .log_dir_len - 1;      ! adjust 'VMS-part' length
;   725          1160  3         END
;   726          1161  2     ELSE
;   727          1162  2         CH$MOVE (.end_part1, .string, .copy);   ! just copy the string
;   728          1163  2
;   729          1164  2 !+
;   730          1165  2 ! Setup and call bpa$file_scan
;   731          1166  2 !-
;   732          1167  2     fsb_ptr [fsb$a_fsa] = .copy;
;   733          1168  2     fsb_ptr [fsb$l_fsl] = .end_part1;
;   734          1169  2
;   735          1170  3     IF NOT (sts = bpa$file_scan (.fsb_ptr))
;   736          1171  2     THEN
;   737          1172  3         BEGIN
;   738          1173  3         bpa$free_block (.string, .byte_count);
;   739          1174  3         bpa$free_block (.copy, nam$c_maxrss);
;   740          1175  3         RETURN SIGNAL (.sts);
;   741          1176  2         END;
;   742          1177  2
;   743          1178  2 !+
;   744          1179  2 ! Decode the <prot> if present
;   745          1180  2 !-
;   746          1181  2
;   747          1182  2     IF .fsb_ptr [fsb$b_prot_off] NEQU 0
;   748          1183  2     THEN
;   749          1184  3         BEGIN
;   750          1185  3         prot_ptr = .fsb_ptr [fsb$b_prot_off] + 1;
;   751          1186  3         prot_val = 0;
;   752          1187  3
;   753          1188  3         WHILE (.string [.prot_ptr] GEQU %C'0' AND .string [.prot_ptr] LEQU %C'9') DO
;   754          1189  4             BEGIN
;   755          1190  4             prot_val = (.prot_val*10) + (.string [.prot_ptr] - %C'0');
;   756          1191  4             prot_ptr = .prot_ptr + 1;
;   757          1192  3             END;
;   758          1193  3
;   759          1194  3         IF .string [.prot_ptr] NEQU %C'>'
;   760          1195  3         THEN
;   761          1196  4             BEGIN
;   762          1197  4             bpa$free_block (.string, .byte_count);
```

```
;   763          1198  4              bpa$free_block (.copy, nam$c_maxrss);
;   764          1199  4              RETURN SIGNAL (bas$_illnum);
;   765          1200  3              END;
;   766          1201  3
;   767          1202  3          IF .prot_val LEQU 127
;   768          1203  3          THEN
;   769          1204  4              BEGIN
;   770          1205  4              fsb_ptr [fsb$b_prot] = .prot_val;
;   771          1206  4              fsb_ptr [fsb$b_prot_real] = -1;
;   772          1207  4              END
;   773          1208  3          ELSE
;   774          1209  4              BEGIN
;   775          1210  4              bpa$free_block (.string, .byte_count);
;   776          1211  4              bpa$free_block (.copy, nam$c_maxrss);
;   777          1212  4              RETURN SIGNAL (bas$_illnum);
;   778          1213  3              END;
;   779          1214  3
;   780          1215  2          END;
;   781          1216  2
;   782          1217  2      !+
;   783          1218  2      ! Sort out switches (if any)
;   784          1219  2      !-
;   785          1220  2
;   786          1221  2          IF .fsb_ptr [fsb$b_swit_off] NEQU 0
;   787          1222  2          THEN
;   788          1223  2
;   789          1224  3              IF (sts = find_switch (.fsb_ptr, .string + .fsb_ptr [fsb$b_swit_off],
;   790          1225  2                      .xlate_length - .fsb_ptr [fsb$b_swit_off])) NEQU 1
;   791          1226  2              THEN
;   792          1227  2                  RETURN SIGNAL (.sts);
;   793          1228  2
;   794          1229  2      !+
;   795          1230  2      ! Load up 'XRB' flagwords in FSB
;   796          1231  2      ! (bpa$file_scan has already loaded some bits)
;   797          1232  2      !-
;   798          1233  2
;   799          1234  2          IF .fsb_ptr [fsb$b_prot_off] NEQU 0
;   800          1235  2          THEN
;   801          1236  2              fsb_ptr [fsb$v_prot_seen] = 1
;   802          1237  2          ELSE
;   803          1238  2
;   804          1239  2              IF .bpa$gb_usr_real NEQU 0 THEN fsb_ptr [fsb$v_def_prot] = 1;
;   805          1240  2
;   806          1241  2          IF .fsb_ptr [fsb$b_dev_off] NEQU 0 THEN fsb_ptr [fsb$v_coln_seen] = 1;
;   807          1242  2
;   808          1243  2          dot_ptr = CH$FIND_CH (.end_part1 - .fsb_ptr [fsb$b_file_off],       !
;   809          1244  2              .string + .fsb_ptr [fsb$b_file_off],    !
;   810          1245  2              %C'.');
;   811          1246  2
;   812          1247  2          IF .dot_ptr NEQU 0
;   813          1248  2          THEN
;   814          1249  3              BEGIN
;   815          1250  3              dot_ptr = .dot_ptr - .string;
;   816          1251  3              fsb_ptr [fsb$v_dot_seen] = 1;
;   817          1252  3              END
;   818          1253  2          ELSE
;   819          1254  2              dot_ptr = .end_part1;
```

```
;  820          1255  2
;  821          1256  3      IF (CH$FIND_CH (.end_part1 - .fsb_ptr [fsb$b_file_off], .string + .fsb_ptr [fsb$b_file_off], %C'%'))
;  822          1257  2      THEN
;  823          1258  2          fsb_ptr [fsb$v_name_ques] = 1;
;  824          1259  2
;  825          1260  2      IF (CH$FIND_CH (.end_part1 - .dot_ptr, .string + .dot_ptr, %C'%')) THEN fsb_ptr [fsb$v_ext_ques] = 1;
;  826          1261  2
;  827          1262  2  !+
;  828          1263  2  ! Copy duplicate bits from flagword 2 to flagword 1
;  829          1264  2  !-
;  830          1265  2
;  831          1266  2      IF .fsb_ptr [fsb$v_name_seen] THEN fsb_ptr [fsb$v_name_1] = 1;
;  832          1267  2
;  833          1268  2      IF .fsb_ptr [fsb$v_dot_seen] THEN fsb_ptr [fsb$v_dot_1] = 1;
;  834          1269  2
;  835          1270  2      IF .fsb_ptr [fsb$v_ppn_seen] THEN fsb_ptr [fsb$v_ppn_1] = 1;
;  836          1271  2
;  837          1272  2      IF .fsb_ptr [fsb$v_prot_seen] THEN fsb_ptr [fsb$v_prot_1] = 1;
;  838          1273  2
;  839          1274  2      IF .fsb_ptr [fsb$v_coln_seen] THEN fsb_ptr [fsb$v_coln_1] = 1;
;  840          1275  2
;  841          1276  2      IF .fsb_ptr [fsb$v_log_name] THEN fsb_ptr [fsb$v_log_1] = 1;
;  842          1277  2
;  843          1278  2      IF .fsb_ptr [fsb$v_status]                  ! If FSB indicates NFREP...
;  844          1279  2      THEN
;  845          1280  3          BEGIN
;  846          1281  3          fqb_ptr [fqb$b_nfrep] = fqb$k_nfrep;    ! ...set bit in FIRQB
;  847          1282  3          fsb_ptr [fsb$v_status] = 0;             ! and zero temp store
;  848          1283  2          END;
;  849          1284  2
;  850          1285  3      fsb_ptr [fsb$v_status] = (.fsb_ptr [fsb$v_name_wild]        !
;  851          1286  3      OR .fsb_ptr [fsb$v_name_ques]                               !
;  852          1287  3      OR .fsb_ptr [fsb$v_ext_wild]                                !
;  853          1288  3      OR .fsb_ptr [fsb$v_ext_ques]                                !
;  854          1289  3      OR .fsb_ptr [fsb$v_proj_wild]                               !
;  855          1290  3      OR .fsb_ptr [fsb$v_prog_wild]                               !
;  856          1291  2      OR .fsb_ptr [fsb$v_log_notr]);
;  857          1292  2  !+
;  858          1293  2  ! Load up the FIRQB
;  859          1294  2  !
;  860          1295  2  ! If filename or ext are wild, load rad50 of '??????' or '???'
;  861          1296  2  ! into filename or ext fields
;  862          1297  2  !-
;  863          1298  2
;  864          1299  2      IF .fsb_ptr [fsb$v_name_wild]
;  865          1300  2      THEN
;  866          1301  3          BEGIN
;  867          1302  3          fqb_ptr [fqb$w_fqnam1] = %O'134745';
;  868          1303  3          fqb_ptr [fqb$w_fqnam2] = %O'134745';
;  869          1304  2          END;
;  870          1305  2
;  871          1306  2      IF .fsb_ptr [fsb$v_ext_wild] THEN fqb_ptr [fqb$w_fqext] = %O'134745';
;  872          1307  2
;  873          1308  2  !+
;  874          1309  2  ! If numeric ppn and group or user fields are wild
;  875          1310  2  ! set the field to 255 (octal -1)
;  876          1311  2  !-
```

```
;  877          1312  2
;  878          1313  2      IF .fsb_ptr [fsb$v_proj_wild] THEN fqb_ptr [fqb$b_proj] = -1;
;  879          1314  2
;  880          1315  2      IF .fsb_ptr [fsb$v_prog_wild] THEN fqb_ptr [fqb$b_prog] = -1;
;  881          1316  2
;  882          1317  2  !+
;  883          1318  2  ! Load size, mode, clustersize & position.
;  884          1319  2  !-
;  885          1320  2      fqb_ptr [fqb$b_blkhi] = .fsb_ptr [fsb$b_blkhi];
;  886          1321  2      fqb_ptr [fqb$w_blklo] = .fsb_ptr [fsb$w_blklo];
;  887          1322  2      fqb_ptr [fqb$w_mode] = .fsb_ptr [fsb$w_mode];
;  888          1323  2      fqb_ptr [fqb$w_cluster] = .fsb_ptr [fsb$w_clust];
;  889          1324  2      fqb_ptr [fqb$w_position] = .fsb_ptr [fsb$w_position];
;  890          1325  2  !+
;  891          1326  2  ! Prot_code is loaded as follows:-
;  892          1327  2  !
;  893          1328  2  ! a)    the prot code specified in the filespec, or
;  894          1329  2  ! b)    the user defined default prot code, or
;  895          1330  2  ! c)    The basic plus default prot code (bpa$k_def_prot)
;  896          1331  2  !-
;  897          1332  2
;  898          1333  2      IF .fsb_ptr [fsb$v_prot_seen]
;  899          1334  2      THEN
;  900          1335  2          fqb_ptr [fqb$b_prot_code] = .fsb_ptr [fsb$b_prot]
;  901          1336  2      ELSE
;  902          1337  2
;  903          1338  2          IF .fsb_ptr [fsb$v_def_prot]
;  904          1339  2          THEN
;  905          1340  2              fqb_ptr [fqb$b_prot_code] = .bpa$gb_usr_prot
;  906          1341  2          ELSE
;  907          1342  2              fqb_ptr [fqb$b_prot_code] = bpa$k_def_prot;
;  908          1343  2
;  909          1344  2      fqb_ptr [fqb$b_prot_real] = -1;
;  910          1345  2  !+
;  911          1346  2  ! From now on we are interested in the contents of the
;  912          1347  2  ! parsed string (filename, devicename, ppn, etc), so
;  913          1348  2  ! we determine the offsets to the field delimeters
;  914          1349  2  ! ( ::, :, [ or <, ] or >, ., and ;). This allows us to
;  915          1350  2  ! find the specific parts of the string very easily.
;  916          1351  2  ! We also store the offset to the filename and the
;  917          1352  2  ! end of the ext.
;  918          1353  2  !-
;  919          1354  2      parsed_address = .fsb_ptr [fsb$a_fsa];
;  920          1355  2      parsed_length = .fsb_ptr [fsb$l_fsl];
;  921          1356  2      next_posn = 0;
;  922          1357  2  !
;  923          1358  2      node_ptr = CH$FIND_SUB (.parsed_length, .parsed_address, 2, UPLIT BYTE('::'));
;  924          1359  2
;  925          1360  2      IF .node_ptr NEQU 0
;  926          1361  2      THEN
;  927          1362  3          BEGIN
;  928          1363  3          node_ptr = .node_ptr - .parsed_address;
;  929          1364  3          next_posn = .node_ptr + 2;
;  930          1365  2          END;
;  931          1366  2
;  932          1367  2  !
;  933          1368  2      dev_ptr = CH$FIND_CH (.parsed_length - .next_posn, .parsed_address + .next_posn, %C':');
```

```
;  934          1369  2
;  935          1370  2      IF .dev_ptr NEQU 0
;  936          1371  2      THEN
;  937          1372  3          BEGIN
;  938          1373  3          dev_ptr = .dev_ptr - .parsed_address;
;  939          1374  3          next_posn = .dev_ptr + 1;
;  940          1375  2          END;
;  941          1376  2
;  942          1377  2  !
;  943          1378  2      ppn_ptr = CH$FIND_CH (.parsed_length - .next_posn, .parsed_address + .next_posn, %C'[');
;  944          1379  2
;  945          1380  2      IF .ppn_ptr NEQU 0
;  946          1381  2      THEN
;  947          1382  3          BEGIN
;  948          1383  3          ppn_ptr = .ppn_ptr - .parsed_address;
;  949          1384  3          next_posn = .ppn_ptr + 1;
;  950          1385  3          END
;  951          1386  2      ELSE
;  952          1387  3          BEGIN
;  953          1388  3          ppn_ptr = CH$FIND_CH (.parsed_length - .next_posn, .parsed_address + .next_posn, %C'<');
;  954          1389  3
;  955          1390  3          IF .ppn_ptr NEQU 0
;  956          1391  3          THEN
;  957          1392  4              BEGIN
;  958          1393  4              ppn_ptr = .ppn_ptr - .parsed_address;
;  959          1394  4              next_posn = .ppn_ptr + 1;
;  960          1395  3              END;
;  961          1396  3
;  962          1397  2          END;
;  963          1398  2
;  964          1399  2  !
;  965          1400  2      ppn_end_ptr = CH$FIND_CH (.parsed_length - .next_posn, .parsed_address + .next_posn, %C']');
;  966          1401  2
;  967          1402  2      IF .ppn_end_ptr NEQU 0
;  968          1403  2      THEN
;  969          1404  3          BEGIN
;  970          1405  3          ppn_end_ptr = .ppn_end_ptr - .parsed_address;
;  971          1406  3          next_posn = .ppn_end_ptr + 1;
;  972          1407  3          END
;  973          1408  2      ELSE
;  974          1409  3          BEGIN
;  975          1410  3          ppn_end_ptr = CH$FIND_CH (.parsed_length - .next_posn, .parsed_address + .next_posn, %C'>');
;  976          1411  3
;  977          1412  3          IF .ppn_end_ptr NEQU 0
;  978          1413  3          THEN
;  979          1414  4              BEGIN
;  980          1415  4              ppn_end_ptr = .ppn_end_ptr - .parsed_address;
;  981          1416  4              next_posn = .ppn_end_ptr + 1;
;  982          1417  3              END;
;  983          1418  3
;  984          1419  2          END;
;  985          1420  2
;  986          1421  2  !
;  987          1422  2      dot_ptr = CH$FIND_CH (.parsed_length - .next_posn, .parsed_address + .next_posn, %C'.');
;  988          1423  2
;  989          1424  2      IF .dot_ptr NEQU 0
;  990          1425  2      THEN
```

```
;  991          1426  3          BEGIN
;  992          1427  3          dot_ptr = .dot_ptr - .parsed_address;
;  993          1428  3          next_posn = .dot_ptr + 1;
;  994          1429  3          END;
;  995          1430  2
;  996          1431  2      !
;  997          1432  2      ver_ptr = CH$FIND_CH (.parsed_length - .next_posn, .parsed_address + .next_posn, %C';');
;  998          1433  2
;  999          1434  2      IF .ver_ptr NEQU 0
; 1000          1435  2      THEN
; 1001          1436  3          BEGIN
; 1002          1437  3          ver_ptr = .ver_ptr - .parsed_address;
; 1003          1438  3          parsed_end = .ver_ptr;
; 1004          1439  3          END
; 1005          1440  2      ELSE
; 1006          1441  3          parsed_end = .parsed_length;
; 1007          1442  2
; 1008          1443  2      !
; 1009          1444  2      fsb_ptr [fsb$b_file_off] = (MAXU (.node_ptr, .dev_ptr, .ppn_ptr, .ppn_end_ptr)) + 1;
; 1010          1445  2      !+
; 1011          1446  2      ! Convert the parsed filename & ext to rad-50 & store away
; 1012          1447  2      ! (if already loaded with 'wild' rad50, don't overwrite)
; 1013          1448  2      !
; 1014          1449  2      ! On entry here, dot_ptr is either an offset to the '.'
; 1015          1450  2      ! (if present), else the same value as parsed_end (if
; 1016          1451  2      ! no '.ext' is present). We can therefore calculate
; 1017          1452  2      ! the lengths of the filename and ext.
; 1018          1453  2      !-
; 1019          1454  2      file_size = .dot_ptr - .fsb_ptr [fsb$b_file_off];
; 1020          1455  2      ext_size = .parsed_end - .dot_ptr - 1;
; 1021          1456  2      !+
; 1022          1457  2      ! If .dot_ptr = .parsed_end, ext_size is now -1, so correct it
; 1023          1458  2      !-
; 1024          1459  2
; 1025          1460  2      IF .ext_size EQL -1 THEN ext_size = 0;
; 1026          1461  2
; 1027          1462  2      !+
; 1028          1463  2      ! Check for .file_size being LEQ 6. If FALSE we do not
; 1029          1464  2      ! load it to the Firqb as rad-50. Same if .ext_size
; 1030          1465  2      ! is GTR 3 (which will be thrown out by RMS later anyway!)
; 1031          1466  2      !-
; 1032          1467  2
; 1033          1468  3      IF (.file_size LEQU 6 AND .fqb_ptr [fqb$w_fqnam1] NEQ %O'134745')
; 1034          1469  2      THEN
; 1035          1470  2
; 1036          1471  2          IF .file_size LEQU 3
; 1037          1472  2          THEN
; 1038          1473  3              BEGIN
; 1039          1474  3              fqb_ptr [fqb$w_fqnam1] = rad_50 (.parsed_address + .fsb_ptr [fsb$b_file_off], .file_size);
; 1040          1475  3              fqb_ptr [fqb$w_fqnam2] = rad_50 (.parsed_address + .fsb_ptr [fsb$b_file_off] + 3, 0);
; 1041          1476  3              END
; 1042          1477  2          ELSE
; 1043          1478  3              BEGIN
; 1044          1479  3              fqb_ptr [fqb$w_fqnam1] = rad_50 (.parsed_address + .fsb_ptr [fsb$b_file_off], 3);
; 1045          1480  3              fqb_ptr [fqb$w_fqnam2] = rad_50 (.parsed_address + .fsb_ptr [fsb$b_file_off] + 3,   !
; 1046          1481  3                  .file_size - 3);
; 1047          1482  3              END
```

```
;  1048          1483  3
;  1049          1484  2      ELSE
;  1050          1485  2
;  1051          1486  2          IF .file_size GTRU 6 THEN fqb_ptr [fqb$b_nfrep] = fqb$k_nfrep;  ! Set NFREP
;  1052          1487  2
;  1053          1488  2      IF (.ext_size LEQU 3 AND .fqb_ptr [fqb$w_fqext] NEQ %O'134745')
;  1054          1489  2      THEN
;  1055          1490  2          fqb_ptr [fqb$w_fqext] = rad_50 (.parsed_address + .dot_ptr + 1, .ext_size);
;  1056          1491  2
;  1057          1492  2  !+
;  1058          1493  2  ! If a numeric ppn is present, load the decimal values into
;  1059          1494  2  ! the FIRQB
;  1060          1495  2  !-
;  1061          1496  2
;  1062          1497  2      IF .fsb_ptr [fsb$v_ppn_seen]
;  1063          1498  2      THEN
;  1064          1499  3          BEGIN
;  1065          1500  3
;  1066          1501  3          IF .fsb_ptr [fsb$v_proj_wild] EQLU 0
;  1067          1502  3          THEN
;  1068          1503  4              BEGIN
;  1069          1504  4              EXPAND = .parsed_address;
;  1070          1505  4              next_posn = 1;
;  1071          1506  4              ppn_val = 0;
;  1072          1507  4
;  1073          1508  5              WHILE (.EXPAND [.ppn_ptr + .next_posn] GEQU %C'0' AND .EXPAND [.ppn_ptr + .next_posn] LEQU %C'9'
;  1074          1509  4              DO
;  1075          1510  5                  BEGIN
;  1076          1511  5                  ppn_val = (.ppn_val*10) + ((.EXPAND [.ppn_ptr + .next_posn]) - %C'0');
;  1077          1512  5                  next_posn = .next_posn + 1;
;  1078          1513  4                  END;
;  1079          1514  4
;  1080          1515  4              fqb_ptr [fqb$b_proj] = .ppn_val;
;  1081          1516  4              END
;  1082          1517  3          ELSE
;  1083          1518  3              next_posn = 2;
;  1084          1519  3
;  1085          1520  3          IF .fsb_ptr [fsb$v_prog_wild] EQLU 0
;  1086          1521  3          THEN
;  1087          1522  4              BEGIN
;  1088          1523  4              EXPAND = .parsed_address;
;  1089          1524  4              next_posn = .next_posn + 1;
;  1090          1525  4              ppn_val = 0;
;  1091          1526  4
;  1092          1527  5              WHILE (.EXPAND [.ppn_ptr + .next_posn] GEQU %C'0' AND .EXPAND [.ppn_ptr + .next_posn] LEQU %C'9'
;  1093          1528  4              DO
;  1094          1529  5                  BEGIN
;  1095          1530  5                  ppn_val = (.ppn_val*10) + ((.EXPAND [.ppn_ptr + .next_posn]) - %C'0');
;  1096          1531  5                  next_posn = .next_posn + 1;
;  1097          1532  4                  END;
;  1098          1533  4
;  1099          1534  4              fqb_ptr [fqb$b_prog] = .ppn_val;
;  1100          1535  3              END;
;  1101          1536  3
;  1102          1537  2          END;
;  1103          1538  2
;  1104          1539  2  !+
```

```
;  1105          1540  2 ! Convert the VMS device to a RSTS-type device
;  1106          1541  2 ! and store the 2 device chars & unit # in the FIRQB
;  1107          1542  2 !-
;  1108          1543  2     EXPAND = .parsed_address;
;  1109          1544  2
;  1110          1545  2     IF .node_ptr NEQU 0 THEN next_posn = .node_ptr + 2 ELSE next_posn = 0;
;  1111          1546  2
;  1112          1547  2     device_size = .dev_ptr - .next_posn;
;  1113          1548  2 !+
;  1114          1549  2 ! 1st char  of device name is at offset .next_posn
;  1115          1550  2 ! End (:) is at offset .dev_ptr
;  1116          1551  2 ! .device_size is length of device name (less the colon)
;  1117          1552  2 !
;  1118          1553  2 ! If device is a proccess permenant logical name, it
;  1119          1554  2 ! will have a '$' in it (e.g. SYS$INPUT:) In this case
;  1120          1555  2 ! we put the original 2-char devicename in the FIRQB
;  1121          1556  2 ! (e.g. TI or KB) with 'unit_real' zeroed, else the VMS
;  1122          1557  2 ! physical name is loaded into the FIRQB.
;  1123          1558  2 ! Note that 'usr_dev_ptr' is the offset to the user
;  1124          1559  2 ! specified devicename.
;  1125          1560  2 !-
;  1126          1561  2     next_char = CH$FIND_CH (.device_size, .EXPAND + .next_posn, %C'$');
;  1127          1562  2
;  1128          1563  2     IF .next_char NEQU 0
;  1129          1564  2     THEN                                        ! retrieve user's 2-char devicename
;  1130          1565  3         BEGIN
;  1131          1566  3
;  1132          1567  3         BIND
;  1133          1568  3             user_dev = string [.usr_dev_ptr] : WORD;
;  1134          1569  3
;  1135          1570  3         fqb_ptr [fqb$w_devnam] = .user_dev;
;  1136          1571  3
;  1137          1572  3         IF (.user_dev EQLU %ASCII'KB' OR .user_dev EQLU %ASCII'TI') THEN fqb_ptr [fqb$b_nfrep] = 0;
;  1138          1573  3
;  1139          1574  3                                                 ! Cancel NFREP bit
;  1140          1575  3         END
;  1141          1576  2     ELSE                                        ! assume physical name
;  1142          1577  3         BEGIN
;  1143          1578  3         fsb_ptr [fsb$b_dev_name1] = .EXPAND [.next_posn];
;  1144          1579  3         fsb_ptr [fsb$b_dev_name2] = .EXPAND [.next_posn + 1];
;  1145          1580  3
;  1146          1581  4         IF (.EXPAND [.next_posn] EQLU %C'T' AND .EXPAND [.next_posn + 1] EQLU %C'U')
;  1147          1582  3         THEN
;  1148          1583  4             BEGIN
;  1149          1584  4             fsb_ptr [fsb$b_dev_name2] = %C'T';
;  1150          1585  4             unit_val = 128;
;  1151          1586  4             END
;  1152          1587  3         ELSE
;  1153          1588  3             unit_val = 0;
;  1154          1589  3
;  1155          1590  3         next_posn = .next_posn + 2;             ! point to controller letter
;  1156          1591  3
;  1157          1592  3         IF .EXPAND [.next_posn] NEQU %C':'
;  1158          1593  3         THEN
;  1159          1594  3
;  1160          1595  4             IF (.EXPAND [.next_posn] GEQU %C'A' AND .EXPAND [.next_posn] LEQU %C'Z')
;  1161          1596  3             THEN
```

```
;  1162          1597  4                 BEGIN
;  1163          1598  4                 ctlr_val = (.EXPAND [.next_posn] - %C'A')*16;
;  1164          1599  4                 next_posn = .next_posn + 1;
;  1165          1600  4                 unit_val = .unit_val + (.EXPAND [.next_posn] - %C'0');
;  1166          1601  4                 next_posn = .next_posn + 1;
;  1167          1602  4
;  1168          1603  4                 IF .EXPAND [.next_posn] NEQU %C':'
;  1169          1604  4                 THEN
;  1170          1605  4                     unit_val = (.unit_val*10) + (.EXPAND [.next_posn] - %C'0');
;  1171          1606  4
;  1172          1607  4                 fsb_ptr [fsb$b_dev_unit] = .ctlr_val + .unit_val;
;  1173          1608  4                 END
;  1174          1609  3             ELSE
;  1175          1610  4                 BEGIN
;  1176          1611  4
;  1177          1612  4                 WHILE (.EXPAND [.next_posn] GEQU %C'0' AND .EXPAND [.next_posn] LEQU %C'9') DO
;  1178          1613  5                     BEGIN
;  1179          1614  5                     unit_val = (.unit_val*10) + (.EXPAND [.next_posn] - %C'0');
;  1180          1615  5                     next_posn = .next_posn + 1;
;  1181          1616  4                     END;
;  1182          1617  4
;  1183          1618  4                 fsb_ptr [fsb$b_dev_unit] = .unit_val;
;  1184          1619  4                 END
;  1185          1620  4
;  1186          1621  3         ELSE
;  1187          1622  3             fsb_ptr [fsb$b_dev_unit] = 0;
;  1188          1623  3
;  1189          1624  3         fsb_ptr [fsb$b_unit_real] = -1;
;  1190          1625  3 !+
;  1191          1626  3 ! Load results into FIRQB
;  1192          1627  3 !-
;  1193          1628  3         (fqb_ptr [fqb$w_devnam])<0, 8> = .fsb_ptr [fsb$b_dev_name1];
;  1194          1629  3         (fqb_ptr [fqb$w_devnam])<8, 8> = .fsb_ptr [fsb$b_dev_name2];
;  1195          1630  3         fqb_ptr [fqb$b_devunit] = .fsb_ptr [fsb$b_dev_unit];
;  1196          1631  3         fqb_ptr [fqb$b_unit_real] = .fsb_ptr [fsb$b_unit_real];
;  1197          1632  2         END;
;  1198          1633  2
;  1199          1634  2     bpa$free_block (.string, .byte_count);
;  1200          1635  2     bpa$free_block (.copy, nam$c_maxrss);
;  1201          1636  2     RETURN ss$_normal;                          ! Return success to user
;  1202          1637  2 !
;  1203          1638  1     END;                                        ! End of routine BPA$FSS


                                .TITLE  BPA$FSS
                                .IDENT  \1-006\

                                .PSECT  _BPA$DATA,NOEXE,  PIC,2

                        00000 BPA$GB_USR_PROT::
                                .BLKB   1
                        00001           .BLKB   3
                        00004 BPA$GB_USR_REAL::
                                .BLKB   1
                        00005           .BLKB   3
                        00008 BPA$AL_USRPPN::
                                .BLKB   8
```

```
                                                                         .PSECT  _BPA$CODE,NOWRT,  SHR,  PIC,2

                                                   00# 00000 P.AAA:      .BYTE   0[33]                                           ;
2F  2E  2D  2C  2B  2A  5D  5B  27  26  25  24  23  22  21  00021             .BYTE   33, 34, 35, 36, 37, 38, 39, 91, 93, 42, -       ;
3E  3D  3C  3B  3A  39  38  37  36  35  34  33  32  31  30  00030                     43, 44, 45, 46, 47, 48, 49, 50, 51, 52, -       ;
4D  4C  4B  4A  49  48  47  46  45  44  43  42  41  40  3F  0003F                     53, 54, 55, 56, 57, 58, 59, 60, 61, 62, -       ;
5C  5B  5A  59  58  57  56  55  54  53  52  51  50  4F  4E  0004E                     63, 64, 65, 66, 67, 68, 69, 70, 71, 72, -       ;
4B  4A  49  48  47  46  45  44  43  42  41  60  5F  5E  5D  0005D                     73, 74, 75, 76, 77, 78, 79, 80, 81, 82, -       ;
5A  59  58  57  56  55  54  53  52  51  50  4F  4E  4D  4C  0006C                     83, 84, 85, 86, 87, 88, 89, 90, 91, 92, -       ;
                                        00  7E  7D  7C  7B  0007B                     93, 94, 95, 96, 65, 66, 67, 68, 69, 70, -       ;
                                                                                      71, 72, 73, 74, 75, 76, 77, 78, 79, 80, -       ;
                                                                                      81, 82, 83, 84, 85, 86, 87, 88, 89, 90, -       ;
                                                                                      123, 124, 125, 126, 0                           ;
                                                   00# 00080              .BYTE   0[33]                                           ;
2F  2E  2D  2C  2B  2A  5D  5B  27  26  25  24  23  22  21  000A1             .BYTE   33, 34, 35, 36, 37, 38, 39, 91, 93, 42, -       ;
3E  3D  3C  3B  3A  39  38  37  36  35  34  33  32  31  30  000B0                     43, 44, 45, 46, 47, 48, 49, 50, 51, 52, -       ;
4D  4C  4B  4A  49  48  47  46  45  44  43  42  41  40  3F  000BF                     53, 54, 55, 56, 57, 58, 59, 60, 61, 62, -       ;
5C  5B  5A  59  58  57  56  55  54  53  52  51  50  4F  4E  000CE                     63, 64, 65, 66, 67, 68, 69, 70, 71, 72, -       ;
4B  4A  49  48  47  46  45  44  43  42  41  60  5F  5E  5D  000DD                     73, 74, 75, 76, 77, 78, 79, 80, 81, 82, -       ;
5A  59  58  57  56  55  54  53  52  51  50  4F  4E  4D  4C  000EC                     83, 84, 85, 86, 87, 88, 89, 90, 91, 92, -       ;
                                        00  7E  7D  7C  7B  000FB                     93, 94, 95, 96, 65, 66, 67, 68, 69, 70, -       ;
                                                                                      71, 72, 73, 74, 75, 76, 77, 78, 79, 80, -       ;
                                                                                      81, 82, 83, 84, 85, 86, 87, 88, 89, 90, -       ;
                                                                                      123, 124, 125, 126, 0                           ;
                        3A  4D  45  54  53  59  53  24  53  59  53  00100 P.AAB:  .ASCII  \SYS$SYSTEM:\                                   ;
            3A  48  53  41  48  5F  52  49  44  24  41  50  42  0010B P.AAC:  .ASCII  \BPA$DIR_HASH:\                                  ;
3A  4D  41  4C  43  58  45  5F  52  49  44  24  41  50  42  00118 P.AAD:  .ASCII  \BPA$DIR_EXCLAM:\                                ;
    3A  52  45  50  4D  41  5F  52  49  44  24  41  50  42  00127 P.AAE:  .ASCII  \BPA$DIR_AMPER:\                                 ;
                                                            30  00135 P.AAF:  .ASCII  \0\                                             ;
                                                        3A  3A  00136 P.AAG:  .ASCII  \::\                                            ;
                                                        3A  3A  00138 P.AAH:  .ASCII  \::\                                            ;

                                                                 TABLE=                   P.AAA
                                                                 DOLLAR_STRING=           P.AAB
                                                                 HASH_STRING=             P.AAC
                                                                 EXCLAM_STRING=           P.AAD
                                                                 AMPER_STRING=            P.AAE
                                                                 END_STRING=              P.AAF
                                                                          .EXTRN  BPA$GET_BLOCK, BPA$FREE_BLOCK
                                                                          .EXTRN  BPA$$MOVTUC, BAS$_ILLNUM
                                                                          .EXTRN  BAS$_ILLSWIUSA, BAS$_ILLFILNAM

                                                 0FFC 00000               .ENTRY  BPA$FSS, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,- ; 0571
                                                                                  R11                                             ;
                                5E          A0  AE  9E 00002              MOVAB   -96(SP), SP                                     ;
                                            10  AC  D5 00006              TSTL    BYT_CNT                                         ; 0758
                                                0B  13 00009              BEQL    1$                                              ;
                          58    AE          10  AC  D0 0000B              MOVL    BYT_CNT, BYTE_COUNT                             ; 0761
                                57          0C  AC  D0 00010              MOVL    BUF_PTR, BUFFER_ADDRESS                         ; 0762
                                                14  11 00014              BRB     2$                                              ; 0758
                                56          08  AC  D0 00016 1$:          MOVL    FSB_PTR, R6                                     ; 0766
       24              00       6E              00  2C 0001A              MOVC5   #0, (SP), #0, #36, (R6)                         ;
                                                66     0001F                                                                      ;
                          04    A6          0C  AC  D0 00020              MOVL    BUF_PTR, 4(R6)                                  ; 0767
                                                66  D4 00025              CLRL    (R6)                                            ; 0768
                                              090A  31 00027              BRW     146$                                            ; 0769
```

```
                                3C    AE   9F 0002A 2$:     PUSHAB  STRING                                  : 0778
                                5C    AE   DD 0002D         PUSHL   BYTE_COUNT                              :
             00000000G    00          02   FB 00030         CALLS   #2, BPA$GET_BLOCK                       :
                   1C     AE          50   D0 00037         MOVL    R0, STS                                 :
                          03    1C    AE   E8 0003B         BLBS    STS, 3$                                 :
                              047F    31 0003F              BRW     72$                                     :
                          58    3C    AE   D0 00042 3$:     MOVL    STRING, R8                              : 0794
                          52          58   D0 00046         MOVL    R8, OP_PTR                              :
                   54     AE    FF    8F   9A 00049         MOVZBL  #255, DST_LENGTH                        : 0795
                                58    AE   D5 0004E 4$:     TSTL    BYTE_COUNT                              : 0797
                                      4A   13 00051         BEQL    6$                                      :
                                40    AE   9F 00053         PUSHAB  NEXT_DST_ADDR                           : 0807
                                48    AE   9F 00056         PUSHAB  BYTES_LEFT                              :
                                50    AE   9F 00059         PUSHAB  ESC_CHAR                                :
                                58    AE   9F 0005C         PUSHAB  BYTES_TO_MOVE                           :
                                60    AE   9F 0005F         PUSHAB  ESC_FLAG                                :
                                      52   DD 00062         PUSHL   OP_PTR                                  :
                                6C    AE   9F 00064         PUSHAB  DST_LENGTH                              :
                              FE5B    CF   9F 00067         PUSHAB  TABLE                                   :
                                58    AE   D4 0006B         CLRL    88(SP)                                  :
                                58    AE   9F 0006E         PUSHAB  88(SP)                                  :
                                      57   DD 00071         PUSHL   BUFFER_ADDRESS                          :
                                F8    AD   9F 00073         PUSHAB  BYTE_COUNT                              :
             00000000G    00          0B   FB 00076         CALLS   #11, BPA$$MOVTUC                        :
                          52    40    AE   D0 0007D         MOVL    NEXT_DST_ADDR, OP_PTR                   : 0809
                                50    AE   D5 00081         TSTL    ESC_FLAG                                : 0811
                                      12   13 00084         BEQL    5$                                      :
          57       48     AE          01   C1 00086         ADDL3   #1, ESC_CHAR, BUFFER_ADDRESS            : 0814
     58   AE       4C     AE          01   C3 0008B         SUBL3   #1, BYTES_TO_MOVE, BYTE_COUNT           : 0815
                   54     AE    44    AE   D0 00091         MOVL    BYTES_LEFT, DST_LENGTH                  : 0816
                                      B6   11 00096         BRB     4$                                      : 0811
                                58    AE   D4 00098 5$:     CLRL    BYTE_COUNT                              : 0819
                                      B1   11 0009B         BRB     4$                                      : 0797
     0C   AE              52          58   83 0009D 6$:     SUBB3   R8, OP_PTR, XLATE_LENGTH                : 0823
                          57    08    AC   D0 000A2         MOVL    FSB_PTR, R7                             : 0827
24        00              6E          00   2C 000A6         MOVC5   #0, (SP), #0, #36, (R7)                 :
                                      67      000AB                                                     :
                   04     A7          58   D0 000AC         MOVL    R8, 4(R7)                               : 0828
                          67    0C    AE   9A 000B0         MOVZBL  XLATE_LENGTH, (R7)                      : 0829
                          50    0C    AE   9A 000B4         MOVZBL  XLATE_LENGTH, R0                        : 0833
68        50     FF3F     CF          02   39 000B8         MATCHC  #2, P.AAG, R0, (R8)                     :
                                      03   13 000BF         BEQL    7$                                      :
                          53          02   D0 000C1         MOVL    #2, R3                                  :
                   28     AE          73   3E 000C4 7$:     MOVAW   -(R3), NODE_PTR                         :
                                      11   13 000C8         BEQL    8$                                      : 0835
                   28     AE          58   C2 000CA         SUBL2   R8, NODE_PTR                            : 0838
                   1C     A7    28    AE   90 000CE         MOVB    NODE_PTR, 28(R7)                        : 0839
     38   AE       28     AE          02   C1 000D3         ADDL3   #2, NODE_PTR, USR_DEV_PTR               : 0840
                                      06   11 000D9         BRB     9$                                      : 0835
                                1C    A7   94 000DB 8$:     CLRB    28(R7)                                  : 0844
                                38    AE   D4 000DE         CLRL    USR_DEV_PTR                             : 0845
                          50    1C    A7   9A 000E1 9$:     MOVZBL  28(R7), R0                              : 0856
                                      14   13 000E5         BEQL    10$                                     :
                          51    0C    AE   9A 000E7         MOVZBL  XLATE_LENGTH, R1                        : 0858
                          51          50   C2 000EB         SUBL2   R0, R1                                  :
                          51          02   C2 000EE         SUBL2   #2, R1                                  :
     02 A048              51          3A   3A 000F1         LOCC    #58, R1, 2(R0)[R8]                      :
```

```
                                   0C  13 000F7        BEQL    11$                                  :
                                   0C  11 000F9        BRB     12$                                  :
                      50      0C   AE  9A 000FB  10$:  MOVZBL  XLATE_LENGTH, R0                     : 0861
        68            50           3A  3A 000FF        LOCC    #58, R0, (R8)                        :
                                   02  12 00103        BNEQ    12$                                  :
                                   51  D4 00105  11$:  CLRL    R1                                   :
                34    AE           51  D0 00107  12$:  MOVL    R1, DEV_PTR                          :
                18    AE      1D   A7  9E 0010B        MOVAB   29(R7), 24(SP)                       : 0863
                              34   AE  D5 00110        TSTL    DEV_PTR                              :
                                   08  13 00113        BEQL    13$                                  :
  18    BE      34    AE           58  83 00115        SUBB3   R8, DEV_PTR, @24(SP)                 :
                                   03  11 0011B        BRB     14$                                  :
                              18   BE  94 0011D  13$:  CLRB    @24(SP)                              :
                      50      0C   AE  9A 00120  14$:  MOVZBL  XLATE_LENGTH, R0                     : 0868
        68            50           3C  3A 00124        LOCC    #60, R0, (R8)                        :
                                   02  12 00128        BNEQ    15$                                  :
                                   51  D4 0012A        CLRL    R1                                   :
                      5A           51  D0 0012C  15$:  MOVL    R1, PROT_PTR                         :
                                   54  13 0012F        BEQL    21$                                  : 0870
                      5A           58  C2 00131        SUBL2   R8, PROT_PTR                         : 0873
                      30      01 AA48  91 00134        CMPB    1(PROT_PTR)[R8], #48                 : 0875
                                   07  1F 00139        BLSSU   16$                                  :
                      39      01 AA48  91 0013B        CMPB    1(PROT_PTR)[R8], #57                 :
                                   29  1B 00140        BLEQU   18$                                  :
                      50      0C   AE  9A 00142  16$:  MOVZBL  XLATE_LENGTH, R0                     : 0883
                      50           5A  C2 00146        SUBL2   PROT_PTR, R0                         :
                                   50  D7 00149        DECL    R0                                   :
  01 AA48             50           3C  3A 0014B        LOCC    #60, R0, 1(PROT_PTR)[R8]             :
                                   02  12 00151        BNEQ    17$                                  :
                                   51  D4 00153        CLRL    R1                                   :
                      5A           51  D0 00155  17$:  MOVL    R1, PROT_PTR                         :
                                   2B  13 00158        BEQL    21$                                  : 0885
                      5A           58  C2 0015A        SUBL2   R8, PROT_PTR                         : 0888
                      30      01 AA48  91 0015D        CMPB    1(PROT_PTR)[R8], #48                 : 0890
                                   0D  1F 00162        BLSSU   19$                                  :
                      39      01 AA48  91 00164        CMPB    1(PROT_PTR)[R8], #57                 :
                                   06  1A 00169        BGTRU   19$                                  :
                1E    A7           5A  90 0016B  18$:  MOVB    PROT_PTR, 30(R7)                     : 0892
                                   14  11 0016F        BRB     21$                                  :
                              58   AE  DD 00171  19$:  PUSHL   BYTE_COUNT                           : 0895
                                   58  DD 00174        PUSHL   R8                                   :
           00000000G  00           02  FB 00176  20$:  CALLS   #2, BPA$FREE_BLOCK                   :
                        00000000G  8F  DD 0017D        PUSHL   #BAS$_ILLNUM                         : 0896
                                   52  11 00183        BRB     25$                                  :
                      50      0C   AE  9A 00185  21$:  MOVZBL  XLATE_LENGTH, R0                     : 0912
        68            50           2F  3A 00189        LOCC    #47, R0, (R8)                        :
                                   02  12 0018D        BNEQ    22$                                  :
                                   51  D4 0018F        CLRL    R1                                   :
                14    AE      1F   A7  9E 00191  22$:  MOVAB   31(R7), 20(SP)                       : 0917
                                   51  D5 00196        TSTL    SWIT_PTR                             : 0914
                                   11  13 00198        BEQL    23$                                  :
  14    BE            51           58  83 0019A        SUBB3   R8, SWIT_PTR, @20(SP)                : 0917
                14    BE      18   BE  91 0019F        CMPB    @24(SP), @20(SP)                     : 0919
                                   08  1B 001A4        BLEQU   24$                                  :
                              18   BE  94 001A6        CLRB    @24(SP)                              :
                                   03  11 001A9        BRB     24$                                  : 0914
                              14   BE  94 001AB  23$:  CLRB    @20(SP)                              : 0923
```

```
                        50       1E   A7  9A 001AE 24$:     MOVZBL  30(R7), R0                      ; 0930
                                      51  D4 001B2         CLRL    R1                              ;
                                      50  D5 001B4         TSTL    R0                              ;
                                      22  13 001B6         BEQL    26$                             ;
                                      51  D6 001B8         INCL    R1                              ;
                                  14  BE  95 001BA         TSTB    @20(SP)                         ;
                                  1B  13 001BD         BEQL    26$                             ;
                        50        14  BE  91 001BF         CMPB    @20(SP), R0                     ; 0933
                                  18  1A 001C3         BGTRU   27$                             ;
                                  58  AE  DD 001C5         PUSHL   BYTE_COUNT                      ; 0938
                                  58  DD 001C8         PUSHL   R8                              ;
00000000G                 00            02  FB 001CA         CALLS   #2, BPA$FREE_BLOCK              ;
                    00000000G         8F  DD 001D1         PUSHL   #BAS$_ILLSWIDSA                 ; 0939
                            02EA      31  DD 001D7 25$:     BRW     73$                             ;
                        05            51  E9 001DA 26$:     BLBC    R1, 28$                         ; 0944
                        56            50  D0 001DD 27$:     MOVL    R0, END_PART1                   ; 0946
                                      0F  11 001E0         BRB     30$                             ;
                                  14  BE  95 001E2 28$:     TSTB    @20(SP)                         ; 0949
                                  06  13 001E5         BEQL    29$                             ;
                        56        14  BE  9A 001E7         MOVZBL  @20(SP), END_PART1              ; 0951
                                      04  11 001EB         BRB     30$                             ;
                        56        0C  AE  9A 001ED 29$:     MOVZBL  XLATE_LENGTH, END_PART1         ; 0953
                    1C  A7        18  BE  91 001F1 30$:     CMPB    @24(SP), 28(R7)                 ; 0960
                                      06  1B 001F6         BLEQU   31$                             ;
                        59        18  BE  9A 001F8         MOVZBL  @24(SP), NEXT_POSN              ; 0962
                                      0F  11 001FC         BRB     33$                             ;
                                  1C  A7  95 001FE 31$:     TSTB    28(R7)                          ; 0965
                                  08  13 00201         BEQL    32$                             ;
                        59        1C  A7  9A 00203         MOVZBL  28(R7), NEXT_POSN               ; 0967
                                      59  D6 00207         INCL    NEXT_POSN                       ;
                                      02  11 00209         BRB     33$                             ;
                                      59  D4 0020B 32$:     CLRL    NEXT_POSN                       ; 0969
                            6E            59  D0 0020D 33$:     MOVL    NEXT_POSN, NEXT_CHAR            ; 0971
                                      02  13 00210         BEQL    34$                             ; 0973
                                      6E  D6 00212         INCL    NEXT_CHAR                       ;
                        50        00 BE48  9A 00214 34$:     MOVZBL  @NEXT_CHAR[R8], R0              ; 0980
                    5B  8F                50  91 00219         CMPB    R0, #91                         ; 0983
                                      19  12 0021D         BNEQ    36$                             ;
                        50        0C  AE  9A 0021F         MOVZBL  XLATE_LENGTH, R0                ; 0985
                        50            6E  C2 00223         SUBL2   NEXT_CHAR, R0                   ;
00 BE48                 50        5D  8F  3A 00226         LOCC    #93, R0, @NEXT_CHAR[R8]         ;
                                      02  12 0022D         BNEQ    35$                             ;
                                      51  D4 0022F         CLRL    R1                              ;
                        59            51  D0 00231 35$:     MOVL    R1, NEXT_POSN                   ;
                                      7A  13 00234         BEQL    44$                             ; 0987
                                      31  11 00236         BRB     38$                             ; 0990
                                  3C  50  91 00238 36$:     CMPB    R0, #60                         ; 1001
                                      33  12 0023B         BNEQ    39$                             ;
                        50        0C  AE  9A 0023D         MOVZBL  XLATE_LENGTH, R0                ; 1003
                        50            6E  C2 00241         SUBL2   NEXT_CHAR, R0                   ;
00 BE48                 50            3E  3A 00244         LOCC    #62, R0, @NEXT_CHAR[R8]         ;
                                      02  12 0024A         BNEQ    37$                             ;
                                      51  D4 0024C         CLRL    R1                              ;
                        59            51  D0 0024E 37$:     MOVL    R1, NEXT_POSN                   ;
                                      5D  13 00251         BEQL    44$                             ; 1005
                    50                6E  C1 00253         ADDL3   #1, NEXT_CHAR, R0               ; 1013
                            41    8F          6048  91 00257         CMPB    (R0)[R8], #65           ;
```

```
                          12   1F 0025C        BLSSU   39$                                    ;
      50      6E          01   C1 0025E        ADDL3   #1, NEXT_CHAR, R0                      ;
          5A  8F        6048   91 00262        CMPB    (R0)[R8], #90                          ;
                          07   1A 00267        BGTRU   39$                                    ;
              59          58   C2 00269 38$:   SUBL2   R8, NEXT_POSN                          ; 1016
              6E     01   A9   9E 0026C        MOVAB   1(R9), NEXT_CHAR                       ; 1017
          30  AE     09   A7   9E 00270 39$:   MOVAB   9(R7), 48(SP)                          ; 1029
          30  BE          6E   90 00275        MOVB    NEXT_CHAR, @48(SP)                     ;
              59          6E   D0 00279        MOVL    NEXT_CHAR, NEXT_POSN                   ; 1030
      52      56          59   C3 0027C        SUBL3   NEXT_POSN, END_PART1, R2               ; 1032
    6948      52          25   3A 00280        LOCC    #37, R2, (NEXT_POSN)[R8]               ;
                          02   12 00285        BNEQ    40$                                    ;
                          51   D4 00287        CLRL    R1                                     ;
                          51   D5 00289 40$:   TSTL    R1                                     ;
                          09   13 0028B        BEQL    41$                                    ;
              50     04   AC   D0 0028D        MOVL    FQB_PTR, R0                            ; 1034
          03  A0     80   8F   90 00291        MOVB    #-128, 3(R0)                           ;
    6948      52          3F   3A 00296 41$:   LOCC    #63, R2, (NEXT_POSN)[R8]               ; 1041
                          02   12 0029B        BNEQ    42$                                    ;
                          51   D4 0029D        CLRL    R1                                     ;
                          51   D5 0029F 42$:   TSTL    R1                                     ;
                          22   13 002A1        BEQL    47$                                    ;
    6948      52          25   3A 002A3        LOCC    #37, R2, (NEXT_POSN)[R8]               ; 1045
                          02   12 002A8        BNEQ    43$                                    ;
                          51   D4 002AA        CLRL    R1                                     ;
                          51   D5 002AC 43$:   TSTL    R1                                     ;
                          03   13 002AE        BEQL    45$                                    ;
                        00CE   31 002B0 44$:   BRW     60$                                    ;
    6948      52          3F   3A 002B3 45$:   LOCC    #63, R2, (NEXT_POSN)[R8]               ; 1057
                          02   12 002B8        BNEQ    46$                                    ;
                          51   D4 002BA        CLRL    R1                                     ;
                          51   D5 002BC 46$:   TSTL    PTR                                    ;
                          05   13 002BE        BEQL    47$                                    ;
              61          25   90 002C0        MOVB    #37, (PTR)                             ; 1058
                          EE   11 002C3        BRB     45$                                    ;
              50     00 BE48   9A 002C5 47$:   MOVZBL  @NEXT_CHAR[R8], R0                     ; 1070
              24          50   91 002CA        CMPB    R0, #36                                ; 1073
                          0B   12 002CD        BNEQ    48$                                    ;
          10  AE   FCF3   CF   9E 002CF        MOVAB   DOLLAR_STRING, LOG_DIR_ADDR            ; 1075
              52          0B   90 002D5        MOVB    #11, LOG_DIR_LEN                       ; 1076
                          2E   11 002D8        BRB     51$                                    ; 1077
              23          50   91 002DA 48$:   CMPB    R0, #35                                ; 1080
                          0B   12 002DD        BNEQ    49$                                    ;
          10  AE   FCEE   CF   9E 002DF        MOVAB   HASH_STRING, LOG_DIR_ADDR              ; 1082
              52          0D   90 002E5        MOVB    #13, LOG_DIR_LEN                       ; 1083
                          1E   11 002E8        BRB     51$                                    ; 1084
              21          50   91 002EA 49$:   CMPB    R0, #33                                ; 1087
                          0B   12 002ED        BNEQ    50$                                    ;
          10  AE   FCEB   CF   9E 002EF        MOVAB   EXCLAM_STRING, LOG_DIR_ADDR            ; 1089
              52          0F   90 002F5        MOVB    #15, LOG_DIR_LEN                       ; 1090
                          0E   11 002F8        BRB     51$                                    ; 1091
              26          50   91 002FA 50$:   CMPB    R0, #38                                ; 1094
                          0D   12 002FD        BNEQ    52$                                    ;
          10  AE   FCEA   CF   9E 002FF        MOVAB   AMPER_STRING, LOG_DIR_ADDR             ; 1096
              52          0E   90 00305        MOVB    #14, LOG_DIR_LEN                       ; 1097
                          59   D6 00308 51$:   INCL    NEXT_POSN                              ; 1098
                          24   11 0030A        BRB     54$                                    ; 1070
```

```
                  40   8F          50  91 0030C 52$:    CMPB    R0, #64                                    ; 1101
                                   1C  12 00310         BNEQ    53$                                        ;
                  10   AE 00000000'  EF  D0 00312         MOVL    BPA$AL_USRPPN+4, LOG_DIR_ADDR              ; 1103
                       52 00000000'  EF  90 0031A         MOVB    BPA$AL_USRPPN, LOG_DIR_LEN                 ; 1104
                                   59  D6 00321         INCL    NEXT_POSN                                  ; 1105
                            10     AE  D5 00323         TSTL    LOG_DIR_ADDR                               ; 1107
                                   08  12 00326         BNEQ    54$                                        ;
                  10   AE     FCCF CF  9E 00328         MOVAB   END_STRING, LOG_DIR_ADDR                   ; 1110
                                   52  94 0032E 53$:    CLRB    LOG_DIR_LEN                                ; 1117
        08   AE            56      59  C3 00330 54$:    SUBL3   NEXT_POSN, END_PART1, 8(SP)                ; 1124
        04   AE            58      59  C1 00335         ADDL3   NEXT_POSN, R8, 4(SP)                       ;
        04   BE       08   AE      24  3A 0033A         LOCC    #36, 8(SP), @4(SP)                         ;
                                   02  12 00340         BNEQ    55$                                        ;
                                   51  D4 00342         CLRL    R1                                         ;
                                   51  D5 00344 55$:    TSTL    R1                                         ;
                                   39  12 00346         BNEQ    60$                                        ;
        04   BE       08   AE      23  3A 00348         LOCC    #35, 8(SP), @4(SP)                         ; 1125
                                   02  12 0034E         BNEQ    56$                                        ;
                                   51  D4 00350         CLRL    R1                                         ;
                                   51  D5 00352 56$:    TSTL    R1                                         ;
                                   2B  12 00354         BNEQ    60$                                        ;
        04   BE       08   AE      21  3A 00356         LOCC    #33, 8(SP), @4(SP)                         ; 1126
                                   02  12 0035C         BNEQ    57$                                        ;
                                   51  D4 0035E         CLRL    R1                                         ;
                                   51  D5 00360 57$:    TSTL    R1                                         ;
                                   1D  12 00362         BNEQ    60$                                        ;
        04   BE       08   AE      26  3A 00364         LOCC    #38, 8(SP), @4(SP)                         ; 1127
                                   02  12 0036A         BNEQ    58$                                        ;
                                   51  D4 0036C         CLRL    R1                                         ;
                                   51  D5 0036E 58$:    TSTL    R1                                         ;
                                   0F  12 00370         BNEQ    60$                                        ;
        04   BE       08   AE  40  8F  3A 00372         LOCC    #64, 8(SP), @4(SP)                         ; 1128
                                   02  12 00379         BNEQ    59$                                        ;
                                   51  D4 0037B         CLRL    R1                                         ;
                                   51  D5 0037D 59$:    TSTL    R1                                         ;
                                   15  13 0037F         BEQL    61$                                        ;
                            58     AE  DD 00381 60$:    PUSHL   BYTE_COUNT                                 ; 1131
                                   58  DD 00384         PUSHL   R8                                         ;
                 00000000G  00     02  FB 00386         CALLS   #2, BPA$FREE_BLOCK                         ;
                       00000000G   8F  DD 0038D         PUSHL   #BAS$_ILLFILNAM                            ; 1132
                              012E     31 00393         BRW     73$                                        ;
                            5C     AE  9F 00396 61$:    PUSHAB  COPY                                       ; 1143
                       7E   FF     8F  9A 00399         MOVZBL  #255, -(SP)                                ;
                 00000000G  00     02  FB 0039D         CALLS   #2, BPA$GET_BLOCK                          ;
                       1C   AE     50  D0 003A4         MOVL    R0, STS                                    ;
                       08   1C     AE  E8 003A8         BLBS    STS, 62$                                   ;
                            58     AE  DD 003AC         PUSHL   BYTE_COUNT                                 ; 1146
                                   58  DD 003AF         PUSHL   R8                                         ;
                              0089     31 003B1         BRW     66$                                        ;
                       5B          52  9A 003B4 62$:    MOVZBL  LOG_DIR_LEN, R11                           ; 1150
                                   55  13 003B7         BEQL    64$                                        ;
                  2C   AE   FF     A9  9E 003B9         MOVAB   -1(R9), 44(SP)                             ; 1153
                  24   AE   FF     8F  9A 003BE         MOVZBL  #255, 36(SP)                               ;
                  20   AE   5C     AE  D0 003C3         MOVL    COPY, 32(SP)                               ; 1157
  24   AE       00         68      2C  AE 2C 003C8      MOVC5   44(SP), (R8), #0, 36(SP), @32(SP)          ;
                            20     BE     003CF                                                            ;
                                   27  18 003D1         BGEQ    63$                                        ;
```

```
                        20  AE      2C  AE  C0 003D3          ADDL2   44(SP), 32(SP)                                          ;
                        24  AE      2C  AE  C2 003D8          SUBL2   44(SP), 36(SP)                                          ;
     24  AE                 00      10  BE      5B  2C 003DD  MOVC5   R11, @LOG_DIR_ADDR, #0, 36(SP), @32(SP)                 ;
                                    20  BE      003E4                                                                         ;
                                        12  18  003E6          BGEQ    63$                                                     ;
                        20  AE          5B  C0  003E8          ADDL2   R11, 32(SP)                                             ;
                        24  AE          5B  C2  003EC          SUBL2   R11, 36(SP)                                             ;
     24  AE                 00      04  BE  08  AE  2C 003F0  MOVC5   8(SP), @4(SP), #0, 36(SP), @32(SP)                     ;
                                    20  BE      003F8                                                                         ;
                            50      0C  AE  9A  003FA 63$:    MOVZBL  XLATE_LENGTH, R0                                        ; 1158
                            51      FF AB40 9E  003FE         MOVAB   -1(R11)[R0], R1                                         ;
                        0C  AE          51  90  00403         MOVB    R1, XLATE_LENGTH                                        ;
                            56      FF AB46 9E  00407         MOVAB   -1(R11)[END_PART1], END_PART1                           ; 1159
                                        05  11  0040C         BRB     65$                                                     ; 1150
          5C  BE                    68      56  28  0040E 64$:  MOVC3   END_PART1, (R8), @COPY                               ; 1162
                        04  A7      5C  AE  D0  00413 65$:    MOVL    COPY, 4(R7)                                             ; 1167
                            67          56  D0  00418         MOVL    END_PART1, (R7)                                         ; 1168
                                        57  DD  0041B         PUSHL   R7                                                      ; 1170
                    0000V   CF          01  FB  0041D         CALLS   #1, BPA$FILE_SCAN                                       ;
                      1C    AE          50  D0  00422         MOVL    R0, STS                                                 ;
                            1C  1C  AE  E8  00426             BLBS    STS, 67$                                                ;
                            58  AE      DD  0042A             PUSHL   BYTE_COUNT                                              ; 1173
                                        58  DD  0042D         PUSHL   R8                                                      ;
              00000000G     00          02  FB  0042F         CALLS   #2, BPA$FREE_BLOCK                                      ;
                            7E  FF      8F  9A  00436         MOVZBL  #255, -(SP)                                             ; 1174
                            60  AE      DD  0043A             PUSHL   COPY                                                    ;
              00000000G     00          02  FB  0043D 66$:    CALLS   #2, BPA$FREE_BLOCK                                      ;
                                        7B  11  00444         BRB     72$                                                     ; 1175
                            1E  A7      95  00446 67$:        TSTB    30(R7)                                                  ; 1182
                                        54  13  00449         BEQL    71$                                                     ;
                            5A  1E      A7  9A  0044B         MOVZBL  30(R7), PROT_PTR                                        ; 1185
                                        5A  D6  0044F         INCL    PROT_PTR                                                ;
                                        52  B4  00451         CLRW    PROT_VAL                                                ; 1186
                            30          6A48 91 00453 68$:    CMPB    (PROT_PTR)[R8], #48                                     ; 1188
                                        19  1F  00457         BLSSU   69$                                                     ;
                            39          6A48 91 00459         CMPB    (PROT_PTR)[R8], #57                                     ;
                                        13  1A  0045D         BGTRU   69$                                                     ;
                            50          52  3C  0045F         MOVZWL  PROT_VAL, R0                                            ; 1190
                            50          0A  C4  00462         MULL2   #10, R0                                                 ;
                            51          8A48 9A 00465         MOVZBL  (PROT_PTR)+[R8], R1                                     ;
                            50          51  C0  00469         ADDL2   R1, R0                                                  ;
          52                50          30  A3  0046C         SUBW3   #48, R0, PROT_VAL                                       ;
                                        E1  11  00470         BRB     68$                                                     ; 1188
                            3E          6A48 91 00472 69$:    CMPB    (PROT_PTR)[R8], #62                                     ; 1194
                                        11  12  00476         BNEQ    70$                                                     ;
                      007F  8F          52  B1  00478         CMPW    PROT_VAL, #127                                          ; 1202
                                        0A  1A  0047D         BGTRU   70$                                                     ;
                        0C  A7          52  90  0047F         MOVB    PROT_VAL, 12(R7)                                        ; 1205
                        08  A7          01  8E  00483         MNEGB   #1, 8(R7)                                               ; 1206
                                        16  11  00487         BRB     71$                                                     ; 1202
                                    58  AE      DD  00489 70$:    PUSHL   BYTE_COUNT                                          ; 1210
                                        58  DD  0048C         PUSHL   R8                                                      ;
              00000000G     00          02  FB  0048E         CALLS   #2, BPA$FREE_BLOCK                                      ;
                            7E  FF      8F  9A  00495         MOVZBL  #255, -(SP)                                             ; 1211
                            60  AE      DD  00499             PUSHL   COPY                                                    ;
                                    FCD7    31  0049C         BRW     20$                                                     ;
                            50  14  BE      9A  0049F 71$:    MOVZBL  @20(SP), R0                                             ; 1221
```

```
                                   27  13 004A3           BEQL    74$                                  ;
                       51      0C  AE  9A 004A5           MOVZBL  XLATE_LENGTH, R1                     ; 1225
       7E              51          50  C3 004A9           SUBL3   R0, R1, -(SP)                        ;
                                 6048  9F 004AD           PUSHAB  (R0)[R8]                             ; 1224
                                   57  DD 004B0           PUSHL   R7                                   ;
               0000V   CF          03  FB 004B2           CALLS   #3, FIND_SWITCH                      ;
                  1C   AE          50  D0 004B7           MOVL    R0, STS                              ;
                       01      1C  AE  D1 004BB           CMPL    STS, #1                              ; 1225
                                   0B  13 004BF           BEQL    74$                                  ;
                               1C  AE  DD 004C1  72$:     PUSHL   STS                                  ; 1227
           00000000G   00          01  FB 004C4  73$:     CALLS   #1, LIB$SIGNAL                       ;
                                       04 004CB           RET                                          ;
                               1E  A7  95 004CC  74$:     TSTB    30(R7)                               ; 1234
                                   06  13 004CF           BEQL    75$                                  ;
                  21   A7          04  88 004D1           BISB2   #4, 33(R7)                           ; 1236
                                   0C  11 004D5           BRB     76$                                  ;
                         00000000'  EF 95 004D7  75$:     TSTB    BPA$GB_USR_REAL                      ; 1239
                                   04  13 004DD           BEQL    76$                                  ;
                  21   A7          08  88 004DF           BISB2   #8, 33(R7)                           ;
                               18  BE  95 004E3  76$:     TSTB    @24(SP)                              ; 1241
                                   04  13 004E6           BEQL    77$                                  ;
                  21   A7          10  88 004E8           BISB2   #16, 33(R7)                          ;
                       50      30  BE  9A 004EC  77$:     MOVZBL  @48(SP), R0                          ; 1243
       52              56          50  C3 004F0           SUBL3   R0, END_PART1, R2                    ;
     6048              52          2E  3A 004F4           LOCC    #46, R2, (R0)[R8]                    ;
                                   02  12 004F9           BNEQ    78$                                  ;
                                   51  D4 004FB           CLRL    R1                                   ;
                  2C   AE          51  D0 004FD  78$:     MOVL    R1, DOT_PTR                          ;
                                   0A  13 00501           BEQL    79$                                  ; 1247
                  2C   AE          58  C2 00503           SUBL2   R8, DOT_PTR                          ; 1250
                  20   A7          08  88 00507           BISB2   #8, 32(R7)                           ; 1251
                                   04  11 0050B           BRB     80$                                  ; 1247
                  2C   AE          56  D0 0050D  79$:     MOVL    END_PART1, DOT_PTR                   ; 1254
                       50      30  BE  9A 00511  80$:     MOVZBL  @48(SP), R0                          ; 1256
       52              56          50  C3 00515           SUBL3   R0, END_PART1, R2                    ;
     6048              52          25  3A 00519           LOCC    #37, R2, (R0)[R8]                    ;
                                   02  12 0051E           BNEQ    81$                                  ;
                                   51  D4 00520           CLRL    R1                                   ;
                       04          51  E9 00522  81$:     BLBC    R1, 82$                              ;
                  20   A7          04  88 00525           BISB2   #4, 32(R7)                           ; 1258
                       56      2C  AE  C2 00529  82$:     SUBL2   DOT_PTR, R6                          ; 1260
  2C BE48              56          25  3A 0052D           LOCC    #37, R6, @DOT_PTR[R8]                ;
                                   02  12 00533           BNEQ    83$                                  ;
                                   51  D4 00535           CLRL    R1                                   ;
                       05          51  E9 00537  83$:     BLBC    R1, 84$                              ;
                  20   A7      40  8F  88 0053A           BISB2   #64, 32(R7)                          ;
                       5B      20  A7  9E 0053F  84$:     MOVAB   32(R7), R11                          ; 1266
                       04          6B  E9 00543           BLBC    (R11), 85$                           ;
                  23   A7          01  88 00546           BISB2   #1, 35(R7)                           ;
       04              6B          03  E1 0054A  85$:     BBC     #3, (R11), 86$                       ; 1268
                  23   A7          02  88 0054E           BISB2   #2, 35(R7)                           ;
                                   6B  95 00552  86$:     TSTB    (R11)                                ; 1270
                                   04  18 00554           BGEQ    87$                                  ;
                  23   A7          04  88 00556           BISB2   #4, 35(R7)                           ;
       04              6B          0A  E1 0055A  87$:     BBC     #10, (R11), 88$                      ; 1272
                  23   A7          08  88 0055E           BISB2   #8, 35(R7)                           ;
       04              6B          0C  E1 00562  88$:     BBC     #12, (R11), 89$                      ; 1274
```

```
                                23  A7           10  88 00566          BISB2   #16, 35(R7)
                            04  6B           0E  E1 0056A 89$:       BBC     #14, (R11), 90$                 ;  1276
                                23  A7           20  88 0056E          BISB2   #32, 35(R7)
                                        23  A7   95 00572 90$:       TSTB    35(R7)                          ;  1278
                                            0E   18 00575            BGEQ    91$
                                50      04  AC   D0 00577            MOVL    FQB_PTR, R0                     ;  1281
                                03      A0  80   8F 90 0057B          MOVB    #-128, 3(R0)
                                23      A7  80   8F 8A 00580          BICB2   #128, 35(R7)                    ;  1282
                        50          6B      01   01 EF 00585 91$:     EXTZV   #1, #1, (R11), R0               ;  1286
                        51          6B      01   02 EF 0058A          EXTZV   #2, #1, (R11), R1
                                            50   51 C8 0058F          BISL2   R1, R0
                        52          6B      01   05 EF 00592          EXTZV   #5, #1, (R11), R2               ;  1287
                                            50   52 C8 00597          BISL2   R2, R0
                        51          6B      01   06 EF 0059A          EXTZV   #6, #1, (R11), R1               ;  1288
                                            50   51 C8 0059F          BISL2   R1, R0
                        52      01  AB      01   00 EF 005A2          EXTZV   #0, #1, 1(R11), R2              ;  1289
                                            50   52 C8 005A8          BISL2   R2, R0
                        51          6B      01   09 EF 005AB          EXTZV   #9, #1, (R11), R1               ;  1290
                                            50   51 C8 005B0          BISL2   R1, R0
                        52          6B      01   0F EF 005B3          EXTZV   #15, #1, (R11), R2              ;  1291
                                            52   50 88 005B8          BISB2   R0, R2
                23  A7              01      07   52 F0 005BB          INSV    R2, #7, #1, 35(R7)
                                    0C      6B   01 E1 005C1          BBC     #1, (R11), 92$                  ;  1299
                                            50      04  AC   D0 005C5  MOVL    FQB_PTR, R0                     ;  1302
                                        08  A0  B9E5B9E5   8F   D0 005C9  MOVL    #-1176127003, 8(R0)
                                    0A      6B           05   E1 005D1 92$:  BBC     #5, (R11), 93$          ;  1306
                                            50      04  AC   D0 005D5  MOVL    FQB_PTR, R0
                                        0C  A0      B9E5 8F   B0 005D9  MOVW    #-17947, 12(R0)
                                        08      01  AB   E9 005DF 93$:  BLBC    1(R11), 94$                 ;  1313
                                            50      04  AC   D0 005E3  MOVL    FQB_PTR, R0
                                        07  A0          01   8E 005E7  MNEGB   #1, 7(R0)
                                    08      6B          09   E1 005EB 94$:  BBC     #9, (R11), 95$          ;  1315
                                            50      04  AC   D0 005EF  MOVL    FQB_PTR, R0
                                        06  A0          01   8E 005F3  MNEGB   #1, 6(R0)
                                            55      04  AC   D0 005F7 95$:  MOVL    FQB_PTR, R5             ;  1320
                                        05  A5      0F  A7   90 005FB  MOVB    15(R7), 5(R5)
                                        0E  A5      0D  A7   B0 00600  MOVW    13(R7), 14(R5)              ;  1321
                                        12  A5      10  A7   B0 00605  MOVW    16(R7), 18(R5)              ;  1322
                                        1C  A5      12  A7   B0 0060A  MOVW    18(R7), 28(R5)              ;  1323
                                        1E  A5      0A  A7   B0 0060F  MOVW    10(R7), 30(R5)              ;  1324
                                07      6B          0A   E1 00614      BBC     #10, (R11), 96$             ;  1333
                                        17  A5      0C  A7   90 00618  MOVB    12(R7), 23(R5)              ;  1335
                                                    12   11 0061D      BRB     98$
                                    0A      6B          0B   E1 0061F 96$:  BBC     #11, (R11), 97$         ;  1338
                                        17  A5  00000000'   EF   90 00623  MOVB    BPA$GB_USR_PROT, 23(R5)     ;  1340
                                                    04   11 0062B      BRB     98$
                                        17  A5          3C   90 0062D 97$:  MOVB    #60, 23(R5)             ;  1342
                                        16  A5          01   8E 00631 98$:  MNEGB   #1, 22(R5)              ;  1344
                                            56      04  A7   D0 00635  MOVL    4(R7), PARSED_ADDRESS       ;  1354
                                            54          67   D0 00639  MOVL    (R7), PARSED_LENGTH         ;  1355
                                                        59   D4 0063C  CLRL    NEXT_POSN                   ;  1356
                    66          54      F9BB    CF      02   39 0063E  MATCHC  #2, P.AAH, PARSED_LENGTH, (PARSED_ADDRESS)  ;  1358
                                                        03   13 00645  BEQL    99$
                                            53          02   D0 00647  MOVL    #2, R3
                                        28  AE          73   3E 0064A 99$:  MOVAW   -(R3), NODE_PTR
                                                        09   13 0064E  BEQL    100$                        ;  1360
                                        28  AE          56   C2 00650  SUBL2   PARSED_ADDRESS, NODE_PTR      ;  1363
```

```
                59     28   AE          02  C1 00654          ADDL3   #2, NODE_PTR, NEXT_POSN                          ; 1364
                53          54          59  C3 00659 100$:    SUBL3   NEXT_POSN, PARSED_LENGTH, R3                     ; 1368
              6946          53          3A  3A 0065D          LOCC    #58, R3, (NEXT_POSN)[PARSED_ADDRESS]             ;
                                        02  12 00662          BNEQ    101$                                             ;
                                        51  D4 00664          CLRL    R1                                               ;
                      34    AE          51  D0 00666 101$:    MOVL    R1, DEV_PTR                                      ;
                                        09  13 0066A          BEQL    102$                                             ; 1370
                      34    AE          56  C2 0066C          SUBL2   PARSED_ADDRESS, DEV_PTR                          ; 1373
                59    34    AE          01  C1 00670          ADDL3   #1, DEV_PTR, NEXT_POSN                           ; 1374
                53          54          59  C3 00675 102$:    SUBL3   NEXT_POSN, PARSED_LENGTH, R3                     ; 1378
              6946          53    5B    8F  3A 00679          LOCC    #91, R3, (NEXT_POSN)[PARSED_ADDRESS]             ;
                                        02  12 0067F          BNEQ    103$                                             ;
                                        51  D4 00681          CLRL    R1                                               ;
                            5A          51  D0 00683 103$:    MOVL    R1, PPN_PTR                                      ;
                                        0E  12 00686          BNEQ    105$                                             ; 1380
              6946          53          3C  3A 00688          LOCC    #60, R3, (NEXT_POSN)[PARSED_ADDRESS]             ; 1388
                                        02  12 0068D          BNEQ    104$                                             ;
                                        51  D4 0068F          CLRL    R1                                               ;
                            5A          51  D0 00691 104$:    MOVL    R1, PPN_PTR                                      ;
                                        07  13 00694          BEQL    106$                                             ; 1390
                            5A          56  C2 00696 105$:    SUBL2   PARSED_ADDRESS, PPN_PTR                          ; 1393
                            59    01    AA  9E 00699          MOVAB   1(R10), NEXT_POSN                                ; 1394
                53          54          59  C3 0069D 106$:    SUBL3   NEXT_POSN, PARSED_LENGTH, R3                     ; 1400
              6946          53    5D    8F  3A 006A1          LOCC    #93, R3, (NEXT_POSN)[PARSED_ADDRESS]             ;
                                        02  12 006A7          BNEQ    107$                                             ;
                                        51  D4 006A9          CLRL    R1                                               ;
                            52          51  D0 006AB 107$:    MOVL    R1, PPN_END_PTR                                  ;
                                        0E  12 006AE          BNEQ    109$                                             ; 1402
              6946          53          3E  3A 006B0          LOCC    #62, R3, (NEXT_POSN)[PARSED_ADDRESS]             ; 1410
                                        02  12 006B5          BNEQ    108$                                             ;
                                        51  D4 006B7          CLRL    R1                                               ;
                            52          51  D0 006B9 108$:    MOVL    R1, PPN_END_PTR                                  ;
                                        07  13 006BC          BEQL    110$                                             ; 1412
                            52          56  C2 006BE 109$:    SUBL2   PARSED_ADDRESS, PPN_END_PTR                      ; 1415
                            59    01    A2  9E 006C1          MOVAB   1(R2), NEXT_POSN                                 ; 1416
                53          54          59  C3 006C5 110$:    SUBL3   NEXT_POSN, PARSED_LENGTH, R3                     ; 1422
              6946          53          2E  3A 006C9          LOCC    #46, R3, (NEXT_POSN)[PARSED_ADDRESS]             ;
                                        02  12 006CE          BNEQ    111$                                             ;
                                        51  D4 006D0          CLRL    R1                                               ;
                      2C    AE          51  D0 006D2 111$:    MOVL    R1, DOT_PTR                                      ;
                                        09  13 006D6          BEQL    112$                                             ; 1424
                      2C    AE          56  C2 006D8          SUBL2   PARSED_ADDRESS, DOT_PTR                          ; 1427
                59    2C    AE          01  C1 006DC          ADDL3   #1, DOT_PTR, NEXT_POSN                           ; 1428
                53          54          59  C3 006E1 112$:    SUBL3   NEXT_POSN, PARSED_LENGTH, R3                     ; 1432
              6946          53          3B  3A 006E5          LOCC    #59, R3, (NEXT_POSN)[PARSED_ADDRESS]             ;
                                        02  12 006EA          BNEQ    113$                                             ;
                                        51  D4 006EC          CLRL    R1                                               ;
                                        51  D5 006EE 113$:    TSTL    VER_PTR                                          ; 1434
                                        08  13 006F0          BEQL    114$                                             ;
                            51          56  C2 006F2          SUBL2   PARSED_ADDRESS, VER_PTR                          ; 1437
                            50          51  D0 006F5          MOVL    VER_PTR, PARSED_END                              ; 1438
                                        03  11 006F8          BRB     115$                                             ; 1434
                            50          54  D0 006FA 114$:    MOVL    PARSED_LENGTH, PARSED_END                        ; 1441
                            53    28    AE  D0 006FD 115$:    MOVL    NODE_PTR, R3                                     ; 1444
                      34    AE          53  D1 00701          CMPL    R3, DEV_PTR                                      ;
                                        04  1E 00705          BGEQU   116$                                             ;
                            53    34    AE  D0 00707          MOVL    DEV_PTR, R3                                      ;
```

```
                        5A              53   D1 0070B 116$:   CMPL    R3, PPN_PTR                                    ;
                                        03   1E 0070E         BGEQU   117$                                           ;
                        53              5A   D0 00710         MOVL    PPN_PTR, R3                                    ;
                        52              53   D1 00713 117$:   CMPL    R3, PPN_END_PTR                                ;
                                        03   1E 00716         BGEQU   118$                                           ;
                        53              52   D0 00718         MOVL    PPN_END_PTR, R3                                ;
                30  BE  53              01   81 0071B 118$:   ADDB3   #1, R3, @48(SP)                                 ;
                        53          30  BE   9A 00720         MOVZBL  @48(SP), R3                                    ; 1454
                52      2C      AE  53   83 00724         SUBB3   R3, DOT_PTR, FILE_SIZE                         ;
                        50      2C  AE   C2 00729         SUBL2   DOT_PTR, R0                                    ; 1455
                54              50   01   83 0072D         SUBB3   #1, R0, EXT_SIZE                               ;
                    FF      8F          54   91 00731         CMPB    EXT_SIZE, #-1                                  ; 1460
                                        02   12 00735         BNEQ    119$                                           ;
                                        54   94 00737         CLRB    EXT_SIZE                                       ;
                        06              52   91 00739 119$:   CMPB    FILE_SIZE, #6                                  ; 1468
                                        43   1A 0073C         BGTRU   122$                                           ;
                B9E5    8F      08  A5   B1 0073E         CMPW    8(R5), #47589                                  ;
                                        3B   13 00744         BEQL    122$                                           ;
                        03              52   91 00746         CMPB    FILE_SIZE, #3                                  ; 1471
                                        13   1A 00749         BGTRU   120$                                           ;
                        7E              52   9A 0074B         MOVZBL  FILE_SIZE, -(SP)                               ; 1474
                                    6346   9F 0074E         PUSHAB  (R3)[PARSED_ADDRESS]                           ;
                0000V   CF              02   FB 00751         CALLS   #2, RAD_50                                     ;
                  08    A5              50   B0 00756         MOVW    R0, 8(R5)                                      ;
                        7E              7E   D4 0075A         CLRL    -(SP)                                          ; 1475
                                        14   11 0075C         BRB     121$                                           ;
                        03              03   DD 0075E 120$:   PUSHL   #3                                            ; 1479
                                    6346   9F 00760         PUSHAB  (R3)[PARSED_ADDRESS]                           ;
                0000V   CF              02   FB 00763         CALLS   #2, RAD_50                                     ;
                  08    A5              50   B0 00768         MOVW    R0, 8(R5)                                      ;
                        7E              52   9A 0076C         MOVZBL  FILE_SIZE, -(SP)                               ; 1481
                        6E              03   C2 0076F         SUBL2   #3, (SP)                                       ;
                                03  A346   9F 00772 121$:   PUSHAB  3(R3)[PARSED_ADDRESS]                          ; 1480
                0000V   CF              02   FB 00776         CALLS   #2, RAD_50                                     ;
                  0A    A5              50   B0 0077B         MOVW    R0, 10(R5)                                     ;
                                        0A   11 0077F         BRB     123$                                           ; 1471
                        06              52   91 00781 122$:   CMPB    FILE_SIZE, #6                                  ; 1486
                                        05   1B 00784         BLEQU   123$                                           ;
                    03  A5          80  8F   90 00786         MOVB    #-128, 3(R5)                                   ;
                        03              54   91 0078B 123$:   CMPB    EXT_SIZE, #3                                   ; 1488
                                        1C   1A 0078E         BGTRU   124$                                           ;
                B9E5    8F          0C  A5   B1 00790         CMPW    12(R5), #47589                                 ;
                                        14   13 00796         BEQL    124$                                           ;
                        7E              54   98 00798         CVTBL   EXT_SIZE, -(SP)                                ; 1490
        50      30      AE              01   C1 0079B         ADDL3   #1, DOT_PTR, R0                                ;
                                    6046   9F 007A0         PUSHAB  (R0)[PARSED_ADDRESS]                           ;
                0000V   CF              02   FB 007A3         CALLS   #2, RAD_50                                     ;
                  0C    A5              50   B0 007A8         MOVW    R0, 12(R5)                                     ;
                                        6B   95 007AC 124$:   TSTB    (R11)                                          ; 1497
                                        6E   18 007AE         BGEQ    131$                                           ;
                        33          01  AB   E8 007B0         BLBS    1(R11), 127$                                   ; 1501
                        53              56   D0 007B4         MOVL    PARSED_ADDRESS, EXPAND                         ; 1504
                        59              01   D0 007B7         MOVL    #1, NEXT_POSN                                  ; 1505
                                        50   B4 007BA         CLRW    PPN_VAL                                        ; 1506
        51              5A              59   C1 007BC 125$:   ADDL3   NEXT_POSN, PPN_PTR, R1                       ; 1508
                        30          6143   91 007C0         CMPB    (R1)[EXPAND], #48                              ;
                                        1B   1F 007C4         BLSSU   126$                                           ;
```

```
                        39        6143  91 007C6          CMPB    (R1)[EXPAND], #57                       ;
                          15        1A 007CA            BGTRU   126$                                    ;
                        52          50 3C 007CC          MOVZWL  PPN_VAL, R2                             ; 1511
                        52          0A C4 007CF          MULL2   #10, R2                                 ;
                        54        6143 9A 007D2          MOVZBL  (R1)[EXPAND], R4                        ;
                        52          54 C0 007D6          ADDL2   R4, R2                                  ;
                50      52          30 A3 007D9          SUBW3   #48, R2, PPN_VAL                        ;
                                    59 D6 007DD          INCL    NEXT_POSN                               ; 1512
                                    DB 11 007DF          BRB     125$                                    ; 1508
                    07  A5          50 90 007E1 126$:    MOVB    PPN_VAL, 7(R5)                          ; 1515
                                    03 11 007E5          BRB     128$                                    ; 1501
                        59          02 D0 007E7 127$:    MOVL    #2, NEXT_POSN                           ; 1518
                30      6B          09 E0 007EA 128$:    BBS     #9, (R11), 131$                         ; 1520
                        53          56 D0 007EE          MOVL    PARSED_ADDRESS, EXPAND                  ; 1523
                                    59 D6 007F1          INCL    NEXT_POSN                               ; 1524
                                    50 B4 007F3          CLRW    PPN_VAL                                 ; 1525
                51      5A          59 C1 007F5 129$:    ADDL3   NEXT_POSN, PPN_PTR, R1                  ; 1527
                        30        6143 91 007F9          CMPB    (R1)[EXPAND], #48                       ;
                          1B        1F 007FD            BLSSU   130$                                    ;
                        39        6143 91 007FF          CMPB    (R1)[EXPAND], #57                       ;
                          15        1A 00803            BGTRU   130$                                    ;
                        54          50 3C 00805          MOVZWL  PPN_VAL, R4                             ; 1530
                        54          0A C4 00808          MULL2   #10, R4                                 ;
                        52        6143 9A 0080B          MOVZBL  (R1)[EXPAND], R2                        ;
                        54          52 C0 0080F          ADDL2   R2, R4                                  ;
                50      54          30 A3 00812          SUBW3   #48, R4, PPN_VAL                        ;
                                    59 D6 00816          INCL    NEXT_POSN                               ; 1531
                                    DB 11 00818          BRB     129$                                    ; 1527
                    06  A5          50 90 0081A 130$:    MOVB    PPN_VAL, 6(R5)                          ; 1534
                        53          56 D0 0081E 131$:    MOVL    PARSED_ADDRESS, EXPAND                  ; 1543
                                28  AE 05 00821          TSTL    NODE_PTR                                ; 1545
                                    07 13 00824          BEQL    132$                                    ;
                59      28  AE          02 C1 00826      ADDL3   #2, NODE_PTR, NEXT_POSN                 ;
                                    02 11 0082B          BRB     133$                                    ;
                                    59 D4 0082D 132$:    CLRL    NEXT_POSN                               ;
                50      34  AE          59 83 0082F 133$:    SUBB3   NEXT_POSN, DEV_PTR, DEVICE_SIZE    ; 1547
                        50          50 9A 00834          MOVZBL  DEVICE_SIZE, R0                         ; 1561
              6943              50    24 3A 00837          LOCC    #36, R0, (NEXT_POSN)[EXPAND]            ;
                                    02 12 0083C          BNEQ    134$                                    ;
                                    51 D4 0083E          CLRL    R1                                      ;
                                6E  51 D0 00840 134$:    MOVL    R1, NEXT_CHAR                           ;
                                    24 13 00843          BEQL    137$                                    ; 1563
                                38 BE48 9F 00845         PUSHAB  @USR_DEV_PTR[R8]                        ; 1570
                    18  A5          9E B0 00849          MOVW    @(SP)+, 24(R5)                          ;
                                38 BE48 9F 0084D         PUSHAB  @USR_DEV_PTR[R8]                        ; 1572
                  424B  8F          9E B1 00851          CMPW    @(SP)+, #16971                          ;
                                    0B 13 00856          BEQL    135$                                    ;
                                38 BE48 9F 00858         PUSHAB  @USR_DEV_PTR[R8]                        ;
                  4954  8F          9E B1 0085C          CMPW    @(SP)+, #18772                          ;
                                    03 12 00861          BNEQ    136$                                    ;
                                03  A5 94 00863 135$:    CLRB    3(R5)                                   ;
                                  00B1 31 00866 136$:    BRW     145$                                    ; 1563
                    16  A7        6943 90 00869 137$:    MOVB    (NEXT_POSN)[EXPAND], 22(R7)             ; 1578
                    17  A7      01 A943 90 0086E         MOVB    1(NEXT_POSN)[EXPAND], 23(R7)            ; 1579
                    54  8F        6943 91 00874          CMPB    (NEXT_POSN)[EXPAND], #84                ; 1581
                                    13 12 00879          BNEQ    138$                                    ;
                    55  8F      01 A943 91 0087B         CMPB    1(NEXT_POSN)[EXPAND], #85               ;
```

```
                              0B  12 00881          BNEQ    138$                             ;
               17   A7   54   8F  90 00883          MOVB    #84, 23(R7)                      ; 1584
                    50   80   8F  9B 00888          MOVZBW  #128, UNIT_VAL                   ; 1585
                              02  11 0088C          BRB     139$                             ; 1581
                              50  B4 0088E 138$:    CLRW    UNIT_VAL                         ; 1588
                    59        02  C0 00890 139$:    ADDL2   #2, NEXT_POSN                    ; 1590
                    3A      6943  91 00893          CMPB    (NEXT_POSN)[EXPAND], #58         ; 1592
                              6B  13 00897          BEQL    143$                             ;
               41   8F      6943  91 00899          CMPB    (NEXT_POSN)[EXPAND], #65         ; 1595
                              3F  1F 0089E          BLSSU   141$                             ;
               5A   8F      6943  91 008A0          CMPB    (NEXT_POSN)[EXPAND], #90         ;
                              38  1A 008A5          BGTRU   141$                             ;
                    51      8943  9A 008A7          MOVZBL  (NEXT_POSN)+[EXPAND], R1         ; 1598
                    51   BF   A1  9E 008AB          MOVAB   -65(R1), R1                      ;
      52            51        10  A5 008AF          MULW3   #16, R1, CTLR_VAL                ;
                    51        50  3C 008B3          MOVZWL  UNIT_VAL, R1                     ; 1600
                    54      8943  9A 008B6          MOVZBL  (NEXT_POSN)+[EXPAND], R4         ;
                    51        54  C0 008BA          ADDL2   R4, R1                           ;
      50            51        30  A3 008BD          SUBW3   #48, R1, UNIT_VAL                ;
                    3A      6943  91 008C1          CMPB    (NEXT_POSN)[EXPAND], #58         ; 1603
                              11  13 008C5          BEQL    140$                             ;
                    51        50  3C 008C7          MOVZWL  UNIT_VAL, R1                     ; 1605
                    51        0A  C4 008CA          MULL2   #10, R1                          ;
                    54      6943  9A 008CD          MOVZBL  (NEXT_POSN)[EXPAND], R4          ;
                    51        54  C0 008D1          ADDL2   R4, R1                           ;
      50            51        30  A3 008D4          SUBW3   #48, R1, UNIT_VAL                ;
15    A7            52        50  81 008D8 140$:    ADDB3   UNIT_VAL, CTLR_VAL, 21(R7)       ; 1607
                              28  11 008DD          BRB     144$                             ; 1595
                    30      6943  91 008DF 141$:    CMPB    (NEXT_POSN)[EXPAND], #48         ; 1612
                              19  1F 008E3          BLSSU   142$                             ;
                    39      6943  91 008E5          CMPB    (NEXT_POSN)[EXPAND], #57         ;
                              13  1A 008E9          BGTRU   142$                             ;
                    51        50  3C 008EB          MOVZWL  UNIT_VAL, R1                     ; 1614
                    51        0A  C4 008EE          MULL2   #10, R1                          ;
                    52      8943  9A 008F1          MOVZBL  (NEXT_POSN)+[EXPAND], R2         ;
                    51        52  C0 008F5          ADDL2   R2, R1                           ;
      50            51        30  A3 008F8          SUBW3   #48, R1, UNIT_VAL                ;
                              E1  11 008FC          BRB     141$                             ; 1612
               15   A7        50  90 008FE 142$:    MOVB    UNIT_VAL, 21(R7)                 ; 1618
                              03  11 00902          BRB     144$                             ; 1595
                         15   A7  94 00904 143$:    CLRB    21(R7)                           ; 1622
               14   A7        01  8E 00907 144$:    MNEGB   #1, 20(R7)                       ; 1624
               18   A5   16   A7  B0 0090B          MOVW    22(R7), 24(R5)                   ; 1628
               1A   A5   15   A7  90 00910          MOVB    21(R7), 26(R5)                   ; 1630
               1B   A5   14   A7  90 00915          MOVB    20(R7), 27(R5)                   ; 1631
                         58   AE  DD 0091A 145$:    PUSHL   BYTE_COUNT                       ; 1634
                              58  DD 0091D          PUSHL   R8                               ;
        00000000G   00        02  FB 0091F          CALLS   #2, BPA$FREE_BLOCK               ;
                    7E   FF   8F  9A 00926          MOVZBL  #255, -(SP)                      ; 1635
                         60   AE  DD 0092A          PUSHL   COPY                             ;
        00000000G   00        02  FB 0092D          CALLS   #2, BPA$FREE_BLOCK               ;
                    50        01  D0 00934 146$:    MOVL    #1, R0                           ; 1636
                                  04 00937          RET                                      ; 1638

; Routine Size:  2360 bytes,    Routine Base:  _BPA$CODE + 013A
```

; 1204 1639 1

```
;  1206     1640  1 ROUTINE bpa$file_scan (fsb_ptr) =
;  1207     1641  1
;  1208     1642  1 !++
;  1209     1643  1 ! FUNCTIONAL DESCRIPTION:
;  1210     1644  1 !
;  1211     1645  1 !       Routine parses the filespec described in the FSB
;  1212     1646  1 !       via $PARSE and loads the resultant data into
;  1213     1647  1 !       the FSB. It may also perform any required conversion
;  1214     1648  1 !       from a RSTS-type device name to the VMS-type name.
;  1215     1649  1 !
;  1216     1650  1 ! FORMAL PARAMETERS:
;  1217     1651  1 !
;  1218     1652  1 !       fsb_ptr  -  the longword address of the FSB
;  1219     1653  1 !
;  1220     1654  1 ! IMPLICIT INPUTS:
;  1221     1655  1 !
;  1222     1656  1 !       NONE
;  1223     1657  1 !
;  1224     1658  1 ! IMPLICIT OUTPUTS:
;  1225     1659  1 !
;  1226     1660  1 !       NONE
;  1227     1661  1 !
;  1228     1662  1 ! ROUTINE VALUE:
;  1229     1663  1 !
;  1230     1664  1 !       Returns error from RMS, else TRUE.
;  1231     1665  1 !
;  1232     1666  1 ! SIDE EFFECTS:
;  1233     1667  1 !
;  1234     1668  1 !       If parsed OK, FSB loaded with the address & length of the
;  1235     1669  1 !       expanded string, and status bits from the NAM block
;  1236     1670  1 !
;  1237     1671  1 !--
;  1238     1672  1
;  1239     1673  2     BEGIN
;  1240     1674  2
;  1241     1675  2     MAP
;  1242     1676  2         fsb_ptr : REF $fsb_def;                 ! address of FSB
;  1243     1677  2
;  1244     1678  2     LOCAL
;  1245     1679  2         fab : $fab_decl,                        ! File Access Block
;  1246     1680  2         nam : $nam_decl,                        ! Name block
;  1247     1681  2         esb : REF VECTOR [, BYTE],              ! Expanded string buffer
;  1248     1682  2         length,                                 ! Contains string length
;  1249     1683  2         sts;                                    ! Status from routines
;  1250     1684  2
;  1251     1685  2 !+
;  1252     1686  2 ! Get buffer from free core
;  1253     1687  2 !-
;  1254     1688  2
;  1255     1689  2     IF NOT (sts = bpa$get_block (nam$c_maxrss, esb)) THEN RETURN (.sts);
;  1256     1690  2
;  1257     1691  2 !+
;  1258     1692  2 ! Initialise fab and nam blocks
;  1259     1693  2 !-
;  1260     1694  2     $fab_init (fab = fab, fna = .fsb_ptr [fsb$a_fsa], fns = .fsb_ptr [fsb$l_fsl], nam = nam);
;  1261     1695  2 !
;  1262     1696  2     $nam_init (nam = nam, esa = .esb, ess = nam$c_maxrss);
```

```
;  1263     1697  2 !+
;  1264     1698  2 ! Call $parse and check return status
;  1265     1699  2 !-
;  1266     1700  2
;  1267     1701  2     IF .fsb_ptr [fsb$b_dev_off] NEQU 0          ! If a device is present...
;  1268     1702  2     THEN
;  1269     1703  2         fsb_ptr [fsb$v_status] = 1;
;  1270     1704  2
;  1271     1705  2     ! ...assume VMS device so set NFREP for now
;  1272     1706  2
;  1273     1707  3     IF NOT (sts = $parse (fab = fab))
;  1274     1708  2     THEN
;  1275     1709  2
;  1276     1710  2         SELECTONEU .sts OF
;  1277     1711  2             SET
;  1278     1712  2
;  1279     1713  2             [rms$_dev] :
;  1280     1714  3                 BEGIN
;  1281     1715  3                 length = .fsb_ptr [fsb$l_fsl];
;  1282     1716  3                 fsb_ptr [fsb$v_status] = 0;     ! Non-VMS device so zap NFREP
;  1283     1717  3
;  1284     1718  4                 IF NOT (translate_dev (.fsb_ptr [fsb$a_fsa], length))
;  1285     1719  3                 THEN
;  1286     1720  4                     BEGIN
;  1287     1721  4                     bpa$free_block (.esb, nam$c_maxrss);
;  1288     1722  4                     RETURN (bas$_illfilnam);
;  1289     1723  4                     END
;  1290     1724  3                 ELSE
;  1291     1725  4                     BEGIN
;  1292     1726  4                     fab [fab$l_fna] = .fsb_ptr [fsb$a_fsa];
;  1293     1727  4                     fab [fab$b_fns] = .length;
;  1294     1728  4
;  1295     1729  5                     IF NOT (sts = $parse (fab = fab))
;  1296     1730  4                     THEN
;  1297     1731  5                         BEGIN
;  1298     1732  5                         bpa$free_block (.esb, nam$c_maxrss);
;  1299     1733  5                         RETURN (bas$_illfilnam);
;  1300     1734  4                         END;
;  1301     1735  4
;  1302     1736  3                     END;
;  1303     1737  3
;  1304     1738  2                 END;
;  1305     1739  2
;  1306     1740  2             [OTHERWISE] :
;  1307     1741  3                 BEGIN
;  1308     1742  3                 bpa$free_block (.esb, nam$c_maxrss);
;  1309     1743  3                 RETURN (bas$_illfilnam);
;  1310     1744  2                 END;
;  1311     1745  2             TES;
;  1312     1746  2
;  1313     1747  2     fsb_ptr [fsb$a_fsa] = .esb;
;  1314     1748  2     fsb_ptr [fsb$l_fsl] = .nam [nam$b_esl];
;  1315     1749  2     fsb_ptr [fsb$l_stat_bits] = .nam [nam$l_fnb];
;  1316     1750  2
;  1317     1751  2     IF .nam [nam$v_exp_name] THEN fsb_ptr [fsb$v_name_seen] = 1;
;  1318     1752  2
;  1319     1753  2     IF .nam [nam$v_wild_name] THEN fsb_ptr [fsb$v_name_wild] = 1;
```

```
; 1320      1754  2
; 1321      1755  2      IF .nam [nam$v_exp_type] THEN fsb_ptr [fsb$v_ext_seen] = 1;
; 1322      1756  2
; 1323      1757  2      IF .nam [nam$v_wild_type] THEN fsb_ptr [fsb$v_ext_wild] = 1;
; 1324      1758  2
; 1325      1759  2      IF .nam [nam$v_grp_mbr] THEN fsb_ptr [fsb$v_ppn_seen] = 1;
; 1326      1760  2
; 1327      1761  2      IF .nam [nam$v_wild_grp] THEN fsb_ptr [fsb$v_proj_wild] = 1;
; 1328      1762  2
; 1329      1763  2      IF .nam [nam$v_wild_mbr] THEN fsb_ptr [fsb$v_prog_wild] = 1;
; 1330      1764  2
; 1331      1765  2      IF .nam [nam$v_exp_dev] THEN fsb_ptr [fsb$v_dev_seen] = 1;
; 1332      1766  2
; 1333      1767  2  !+
; 1334      1768  2  ! Set NFREP if node, non-ppn directory or version # present
; 1335      1769  2  !-
; 1336      1770  2
; 1337      1771  3      IF (.nam [nam$v_node] OR (.nam [nam$v_exp_dir] AND NOT .nam [nam$v_grp_mbr]) OR .nam [nam$v_exp_ver])
; 1338      1772  2      THEN
; 1339      1773  2          fsb_ptr [fsb$v_status] = 1;
; 1340      1774  2
; 1341      1775  2      RETURN ss$_normal;
; 1342      1776  1      END;                                        !End of routine BPA$FILE_SCAN


                                                          .EXTRN  SYS$PARSE

                                   01FC 00000 BPA$FILE_SCAN:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8              ; 1640
                     58 00000000G  00  9E 00002           MOVAB   SYS$PARSE, R8                          ;
                     5E       FF48 CE  9E 00009           MOVAB   -184(SP), SP                           ;
                                   5E  DD 0000E           PUSHL   SP                                     ; 1689
                     7E         FF 8F  9A 00010           MOVZBL  #255, -(SP)                            ;
              00000000G 00         02  FB 00014           CALLS   #2, BPA$GET_BLOCK                      ;
                     57            50  D0 0001B           MOVL    R0, STS                                ;
                     04            57  E8 0001E           BLBS    STS, 1$                                ;
                     50            57  D0 00021           MOVL    STS, R0                                ;
                                       04 00024           RET                                            ;
   0050   8F           00    6E    00  2C 00025 1$:       MOVC5   #0, (SP), #0, #80, $RMS_PTR            ; 1694
                              68   AE     0002C                                                          ;
                  68  AE    5003   8F  B0 0002E           MOVW    #20483, $RMS_PTR                       ;
                  7E  AE      02       90 00034           MOVB    #2, $RMS_PTR+22                        ;
                  CF  AD      02       90 00038           MOVB    #2, $RMS_PTR+31                        ;
                  D8  AD      08   AE  9E 0003C           MOVAB   NAM, $RMS_PTR+40                       ;
                  56          04   AC  D0 00041           MOVL    FSB_PTR, R6                            ;
                  DC  AD      04   A6  D0 00045           MOVL    4(R6), $RMS_PTR+44                     ;
                  E4  AD           66  90 0004A           MOVB    (R6), $RMS_PTR+52                      ;
   0060   8F           00    6E    00  2C 0004E           MOVC5   #0, (SP), #0, #96, $RMS_PTR            ; 1696
                              08   AE     00055                                                          ;
                  08  AE    6002   8F  B0 00057           MOVW    #24578, $RMS_PTR                       ;
                  12  AE           01  8E 0005D           MNEGB   #1, $RMS_PTR+10                        ;
                  52               6E  D0 00061           MOVL    ESB, R2                                ;
                  14  AE           52  D0 00064           MOVL    R2, $RMS_PTR+12                        ;
                              1D   A6  95 00068           TSTB    29(R6)                                 ; 1701
                                   05  13 0006B           BEQL    2$                                     ;
                  23  A6      80   8F  88 0006D           BISB2   #128, 35(R6)                           ; 1703
```

```
                                68   AE  9F 00072 2$:     PUSHAB  FAB                                     ; 1707
                            68       01  FB 00075         CALLS   #1, SYS$PARSE                           ;
                            57       50  D0 00078         MOVL    R0, STS                                 ;
                            4B       57  E8 0007B         BLBS    STS, 4$                                 ;
                000184C4    8F       57  D1 0007E         CMPL    STS, #99524                             ; 1713
                                     2D  12 00085         BNEQ    3$                                      ;
                        04  AE       66  D0 00087         MOVL    (R6), LENGTH                            ; 1715
                        23  A6   80   8F  8A 0008B         BICB2   #128, 35(R6)                            ; 1716
                            04   AE  9F 00090             PUSHAB  LENGTH                                  ; 1718
                            04   A6  DD 00093             PUSHL   4(R6)                                   ;
                    0000V   CF       02  FB 00096         CALLS   #2, TRANSLATE_DEV                       ;
                            16       50  E9 0009B         BLBC    R0, 3$                                  ;
                        DC  AD   04  A6  D0 0009E         MOVL    4(R6), FAB+44                           ; 1726
                        E4  AD   04  AE  90 000A3         MOVB    LENGTH, FAB+52                          ; 1727
                                68   AE  9F 000A8         PUSHAB  FAB                                     ; 1729
                            68       01  FB 000AB         CALLS   #1, SYS$PARSE                           ;
                            57       50  D0 000AE         MOVL    R0, STS                                 ;
                            15       57  E8 000B1         BLBS    STS, 4$                                 ;
                            7E   FF  8F  9A 000B4 3$:     MOVZBL  #255, -(SP)                             ; 1742
                                     52  DD 000B8         PUSHL   R2                                      ;
                00000000G   00       02  FB 000BA         CALLS   #2, BPA$FREE_BLOCK                      ;
                            50 00000000G 8F  D0 000C1     MOVL    #BAS$_ILLFILNAM, R0                     ; 1743
                                     04  00 000C8         RET                                             ;
                        04  A6       52  D0 000C9 4$:     MOVL    R2, 4(R6)                               ; 1747
                            66   13  AE  9A 000CD         MOVZBL  NAM+11, (R6)                            ; 1748
                        18  A6   3C  AE  D0 000D1         MOVL    NAM+52, 24(R6)                          ; 1749
                    04  3C  AE       02  E1 000D6         BBC     #2, NAM+52, 5$                          ; 1751
                        20  A6       01  88 000DB         BISB2   #1, 32(R6)                              ;
                    04  3C  AE       05  E1 000DF 5$:     BBC     #5, NAM+52, 6$                          ; 1753
                        20  A6       02  88 000E4         BISB2   #2, 32(R6)                              ;
                    04  3C  AE       01  E1 000E8 6$:     BBC     #1, NAM+52, 7$                          ; 1755
                        20  A6       10  88 000ED         BISB2   #16, 32(R6)                             ;
                    04  3C  AE       04  E1 000F1 7$:     BBC     #4, NAM+52, 8$                          ; 1757
                        20  A6       20  88 000F6         BISB2   #32, 32(R6)                             ;
                    05  3E  AE       03  E1 000FA 8$:     BBC     #3, NAM+54, 9$                          ; 1759
                        20  A6   80   8F  88 000FF         BISB2   #128, 32(R6)                            ;
                            04   3F  AE  E9 00104 9$:     BLBC    NAM+55, 10$                             ; 1761
                        21  A6       01  88 00108         BISB2   #1, 33(R6)                              ;
                    04  3F  AE       01  E1 0010C 10$:    BBC     #1, NAM+55, 11$                         ; 1763
                        21  A6       02  88 00111         BISB2   #2, 33(R6)                              ;
                                3C   AE  95 00115 11$:    TSTB    NAM+52                                  ; 1765
                                     04  18 00118         BGEQ    12$                                     ;
                        21  A6       20  88 0011A         BISB2   #32, 33(R6)                             ;
                    0E  3E  AE       01  E0 0011E 12$:    BBS     #1, NAM+54, 14$                         ; 1771
                    05  3C  AE       06  E1 00123         BBC     #6, NAM+52, 13$                         ;
                    04  3E  AE       03  E1 00128         BBC     #3, NAM+54, 14$                         ;
                            05   3C  AE  E9 0012D 13$:     BLBC    NAM+52, 15$                             ;
                        23  A6   80   8F  88 00131 14$:    BISB2   #128, 35(R6)                            ; 1773
                            50       01  D0 00136 15$:    MOVL    #1, R0                                  ; 1775
                                     04  00 00139         RET                                             ; 1776

; Routine Size:  314 bytes,    Routine Base:  _BPA$CODE + 0A72
```

```
;  1344          1777  1 ROUTINE rad_50 (addr, len) =
;  1345          1778  1
;  1346          1779  1 !++
;  1347          1780  1 ! FUNCTIONAL DESCRIPTION:
;  1348          1781  1 !
;  1349          1782  1 !       Routine converts an ascii string of upto 3 chars
;  1350          1783  1 !       into a word of rad50. The string will be space
;  1351          1784  1 !       padded upto 3 chars if passed length is less than 3.
;  1352          1785  1 !       A null string will result in zero being returned as
;  1353          1786  1 !       the value.
;  1354          1787  1 !
;  1355          1788  1 ! FORMAL PARAMETERS:
;  1356          1789  1 !
;  1357          1790  1 !       addr  -  longword containing the address of the
;  1358          1791  1 !                1st char to be converted
;  1359          1792  1 !       len   -  the length (0-3). If the length passed
;  1360          1793  1 !                is greater than 3, it will be truncated.
;  1361          1794  1 !
;  1362          1795  1 ! IMPLICIT INPUTS:
;  1363          1796  1 !
;  1364          1797  1 !       NONE
;  1365          1798  1 !
;  1366          1799  1 ! IMPLICIT OUTPUTS:
;  1367          1800  1 !
;  1368          1801  1 !       NONE
;  1369          1802  1 !
;  1370          1803  1 ! ROUTINE VALUE:
;  1371          1804  1 !
;  1372          1805  1 !       Value returned is a WORD of rad50
;  1373          1806  1 !
;  1374          1807  1 ! SIDE EFFECTS:
;  1375          1808  1 !
;  1376          1809  1 !       NONE
;  1377          1810  1 !
;  1378          1811  1 !--
;  1379          1812  1
;  1380          1813  2     BEGIN
;  1381          1814  2
;  1382          1815  2     LOCAL
;  1383          1816  2         chars : VECTOR [3, BYTE],
;  1384          1817  2         rad : WORD,
;  1385          1818  2         CODE : BYTE,
;  1386          1819  2         ptr : BYTE;
;  1387          1820  2
;  1388          1821  2     IF .len LEQU 3 THEN ptr = .len ELSE ptr = 3;
;  1389          1822  2
;  1390          1823  2     CH$MOVE (.ptr, .addr, chars);
;  1391          1824  2
;  1392          1825  2     WHILE .ptr LSSU 3 DO
;  1393          1826  3         BEGIN
;  1394          1827  3         chars [.ptr] = %O'40';
;  1395          1828  3         ptr = .ptr + 1;
;  1396          1829  2         END;
;  1397          1830  2
;  1398          1831  2     ptr = 0;
;  1399          1832  2     rad = 0;
;  1400          1833  2
```

```
; 1401      1834  2     INCR offset FROM 0 TO 2 DO
; 1402      1835  3         BEGIN
; 1403      1836  3         CODE = .chars [.offset];
; 1404      1837  3
; 1405      1838  3         SELECTONE .CODE OF
; 1406      1839  3             SET
; 1407      1840  3
; 1408      1841  3             [%C' '] :
; 1409      1842  3                 CODE = 0;
; 1410      1843  3
; 1411      1844  3             [%C'$'] :
; 1412      1845  3                 CODE = 33;
; 1413      1846  3
; 1414      1847  3             [%C'.'] :
; 1415      1848  3                 CODE = 34;
; 1416      1849  3
; 1417      1850  3             [%C'A' TO %C'Z'] :
; 1418      1851  3                 CODE = .CODE - %O'100';
; 1419      1852  3
; 1420      1853  3             [%C'0' TO %C'9'] :
; 1421      1854  3                 CODE = .CODE - %O'22';
; 1422      1855  3
; 1423      1856  3             [OTHERWISE] :
; 1424      1857  3                 CODE = 0;
; 1425      1858  3             TES;
; 1426      1859  3
; 1427      1860  3         rad = (.rad*%O'50') + .CODE;
; 1428      1861  2         END;
; 1429      1862  2
; 1430      1863  2     RETURN .rad;
; 1431      1864  1     END;                                ! End of routine RAD_50


                                 007C 00000 RAD_50: .WORD    Save R2,R3,R4,R5,R6            ; 1777
              5E              04   C2 00002         SUBL2    #4, SP                         ;
              03         08   AC   D1 00005         CMPL     LEN, #3                        ; 1821
                              06   1A 00009         BGTRU    1$                             ;
              56         08   AC   90 0000B         MOVB     LEN, PTR                       ;
                              03   11 0000F         BRB      2$                             ;
              56              03   90 00011 1$:     MOVB     #3, PTR                        ;
              50              56   9A 00014 2$:     MOVZBL   PTR, R0                        ; 1823
   6E      04 BC              50   28 00017         MOVC3    R0, @ADDR, CHARS               ;
              50              56   9A 0001C 3$:     MOVZBL   PTR, R0                        ; 1825
              03              50   91 0001F         CMPB     R0, #3                         ;
                              08   1E 00022         BGEQU    4$                             ;
            6E40              20   90 00024         MOVB     #32, CHARS[R0]                 ; 1827
                              56   96 00028         INCB     PTR                            ; 1828
                              F0   11 0002A         BRB      3$                             ; 1825
                              56   94 0002C 4$:     CLRB     PTR                            ; 1831
                              53   B4 0002E         CLRW     RAD                            ; 1832
                              51   D4 00030         CLRL     OFFSET                         ; 1834
              50            6E41   90 00032 5$:     MOVB     CHARS[OFFSET], CODE            ; 1836
              20              50   91 00036         CMPB     CODE, #32                      ; 1841
                              35   13 00039         BEQL     9$                             ;
```

```
                        24              50  91  0003B           CMPB    CODE, #36                       ; 1844
                                        05  12  0003E           BNEQ    6$
                        50              21  90  00040           MOVB    #33, CODE                       ; 1845
                                        2D  11  00043           BRB     10$
                        2E              50  91  00045  6$:      CMPB    CODE, #46                       ; 1847
                                        05  12  00048           BNEQ    7$
                        50              22  90  0004A           MOVB    #34, CODE                       ; 1848
                                        23  11  0004D           BRB     10$
                    41  8F              50  91  0004F  7$:      CMPB    CODE, #65                       ; 1850
                                        0C  1F  00053           BLSSU   8$
                    5A  8F              50  91  00055           CMPB    CODE, #90
                                        06  1A  00059           BGTRU   8$
                        50          C0  8F  80  0005B           ADDB2   #-64, CODE                      ; 1851
                                        11  11  0005F           BRB     10$
                        30              50  91  00061  8$:      CMPB    CODE, #48                       ; 1853
                                        0A  1F  00064           BLSSU   9$
                        39              50  91  00066           CMPB    CODE, #57
                                        05  1A  00069           BGTRU   9$
                        50              12  82  0006B           SUBB2   #18, CODE                       ; 1854
                                        02  11  0006E           BRB     10$
                                        50  94  00070  9$:      CLRB    CODE                            ; 1857
                        52              53  3C  00072  10$:     MOVZWL  RAD, R2                         ; 1860
                        52              28  C4  00075           MULL2   #40, R2
                        54              50  9A  00078           MOVZBL  CODE, R4
                53      52              54  A1  0007B           ADDW3   R4, R2, RAD
                AF      51              02  F3  0007F           AOBLEQ  #2, OFFSET, 5$                  ; 1834
                        50              53  3C  00083           MOVZWL  RAD, R0                         ; 1863
                                        04  00086           RET                                     ; 1864

; Routine Size:  135 bytes,    Routine Base:  _BPA$CODE + 0BAC
```

```
;  1433          1865  1 ROUTINE translate_dev (string, string_length) =
;  1434          1866  1
;  1435          1867  1 !++
;  1436          1868  1 ! FUNCTIONAL DESCRIPTION:
;  1437          1869  1 !
;  1438          1870  1 !       Routine scans the string given and translates it to a
;  1439          1871  1 !       RSTS device name, if possible.
;  1440          1872  1 !       The standard RSTS device name is of the form DDnnn,
;  1441          1873  1 !               where DD is the device (e.g. TT, MM, DB)
;  1442          1874  1 !               and nnn is a number from 0 to 255
;  1443          1875  1 !       The standard VAX device name is of the form DDCnn,
;  1444          1876  1 !               where DD is the device
;  1445          1877  1 !               where C is the controller id (A to ?)
;  1446          1878  1 !               and nn is a number from 0 to 15
;  1447          1879  1 !
;  1448          1880  1 !       Special cases:
;  1449          1881  1 !               TI: is translated to SYS$INPUT:
;  1450          1882  1 !               KB: is translated to SYS$INPUT:
;  1451          1883  1 !               SY: is translated to SYS$DISK:
;  1452          1884  1 !               SY0: is translated to SYS$DISK:
;  1453          1885  1 !               TT128 and greater are translated to
;  1454          1886  1 !                       TUA0 and greater
;  1455          1887  1 !
;  1456          1888  1 ! FORMAL PARAMETERS:
;  1457          1889  1 !       string          pointer to the string
;  1458          1890  1 !       string_length   pointer to the length of the string
;  1459          1891  1 !                       (it will be updated with the new length)
;  1460          1892  1 !
;  1461          1893  1 ! IMPLICIT INPUTS:
;  1462          1894  1 !
;  1463          1895  1 !       NONE
;  1464          1896  1 !
;  1465          1897  1 ! IMPLICIT OUTPUTS:
;  1466          1898  1 !
;  1467          1899  1 !       NONE
;  1468          1900  1 !
;  1469          1901  1 ! COMPLETION CODES:
;  1470          1902  1 !       1 if success (e.g. no error found)
;  1471          1903  1 !       0 otherwise
;  1472          1904  1 !
;  1473          1905  1 ! SIDE EFFECTS:
;  1474          1906  1 !
;  1475          1907  1 !       NONE
;  1476          1908  1 !
;  1477          1909  1 !--
;  1478          1910  1
;  1479          1911  2     BEGIN
;  1480          1912  2
;  1481          1913  2     MAP
;  1482          1914  2         string : REF VECTOR [, BYTE];
;  1483          1915  2
;  1484          1916  2     BIND
;  1485          1917  2         sysinput = UPLIT BYTE('SYS$INPUT'),
;  1486          1918  2         sysdisk = UPLIT BYTE('SYS$DISK'),
;  1487          1919  2         end_strg = UPLIT BYTE('0');
;  1488          1920  2
;  1489          1921  2     LITERAL
```

```
;  1490          1922  2          sysinput_l = sysdisk - sysinput,
;  1491          1923  2          sysdisk_l = end_strg - sysdisk;
;  1492          1924  2
;  1493          1925  2      LOCAL
;  1494          1926  2          string_len,                         ! the length of the string
;  1495          1927  2          dev_len,                            ! the length of the device name (keeps
;  1496          1928  2                                              ! track of how much is substituted)
;  1497          1929  2          dev_ptr,                            ! pointer to ':' after device name
;  1498          1930  2          start_posn,                         ! offset to start of device name
;  1499          1931  2          next_char,                          ! pointer to the next character
;  1500          1932  2          node_ptr,                           ! pointer to first ':' of node
;  1501          1933  2          temp_copy : REF VECTOR [, BYTE],    ! pointer to temp buffer
;  1502          1934  2          sts;                                ! status word
;  1503          1935  2
;  1504          1936  2      string_len = ..string_length;           ! set up length for easy reference
;  1505          1937  2  !
;  1506          1938  2  ! First, find the beginning and the end of the device name
;  1507          1939  2  !
;  1508          1940  2
;  1509          1941  2      IF (node_ptr = CH$FIND_SUB (.string_len, .string, 2, UPLIT BYTE('::'))) EQLU 0
;  1510          1942  2      THEN
;  1511          1943  3          BEGIN
;  1512          1944  3          start_posn = 0;
;  1513          1945  3          dev_ptr = CH$FIND_CH (.string_len, .string, %C':');
;  1514          1946  3          END
;  1515          1947  2      ELSE
;  1516          1948  3          BEGIN
;  1517          1949  3          start_posn = (.node_ptr + 2) - .string;
;  1518          1950  3          dev_ptr = CH$FIND_CH (.string_len - .start_posn, .node_ptr + 2, %C':');
;  1519          1951  2          END;
;  1520          1952  2
;  1521          1953  2  !
;  1522          1954  2  ! Check to see if a device has been found, if not, get out (error?)
;  1523          1955  2  !
;  1524          1956  2
;  1525          1957  2      IF .dev_ptr EQLU 0
;  1526          1958  2      THEN
;  1527          1959  2          RETURN 0
;  1528          1960  2      ELSE
;  1529          1961  2
;  1530          1962  2          IF .node_ptr EQLU 0 THEN dev_len = .dev_ptr - .string ELSE dev_len = .dev_ptr - .node_ptr - 2;
;  1531          1963  2
;  1532          1964  2  !
;  1533          1965  2  ! Check for underline, and jump over it
;  1534          1966  2  !
;  1535          1967  2
;  1536          1968  2      IF .string [.start_posn] EQLU %C'_' THEN start_posn = .start_posn + 1;
;  1537          1969  2
;  1538          1970  2  !
;  1539          1971  2  ! The next block of code does a SELECT on the first two characters
;  1540          1972  2  ! of the device name, and then processes each of the special cases.
;  1541          1973  2  ! At the end, it translates the RSTS type DDnnn to the VMS type
;  1542          1974  2  ! DDCnn, and handles all special cases as necessary.
;  1543          1975  2  !
;  1544          1976  3      BEGIN                                   ! beginning of translate block
;  1545          1977  3
;  1546          1978  3      LOCAL
```

```
;  1547          1979  3          device_id : VECTOR [3, BYTE],           ! buffer for translated device
;  1548          1980  3          substitute_addr,                        ! address and length of the
;  1549          1981  3          substitute_len;                         ! string to be substituted in
;  1550          1982  3
;  1551          1983  3  !
;  1552          1984  3  ! Grab the first two characters of the device
;  1553          1985  3  !
;  1554          1986  3
;  1555          1987  3      BIND
;  1556          1988  3          device_name = string [.start_posn] : WORD;
;  1557          1989  3
;  1558          1990  3      next_char = .start_posn + 2;                ! skip the first two characters
;  1559          1991  3  !
;  1560          1992  3  ! Select on the first two characters of the device name
;  1561          1993  3  !
;  1562          1994  3
;  1563          1995  3      SELECTU .device_name OF
;  1564          1996  3          SET
;  1565          1997  3
;  1566          1998  3          [%ASCII'SY'] :                          ! note that order is important, this must be first
;  1567          1999  4              BEGIN
;  1568          2000  4
;  1569          2001  4              IF .string [.next_char] EQLU %C':'  ! "SY:"
;  1570          2002  5                  OR (.string [.next_char] EQLU %C'0'     ! "SY0:"
;  1571          2003  5                  AND .string [.next_char + 1] EQLU %C':')
;  1572          2004  4              THEN
;  1573          2005  5                  BEGIN
;  1574          2006  5  !
;  1575          2007  5  ! substitute 'SYS$DISK' FOR 'SY' OR 'SY0'
;  1576          2008  5  !
;  1577          2009  5                  substitute_addr = sysdisk;
;  1578          2010  5                  substitute_len = sysdisk_l;
;  1579          2011  5                  END
;  1580          2012  4              ELSE
;  1581          2013  4                  RETURN 0;
;  1582          2014  4
;  1583          2015  3              END;
;  1584          2016  3
;  1585          2017  3          [%ASCII'KB'] :
;  1586          2018  4              BEGIN
;  1587          2019  4
;  1588          2020  4              IF .string [.next_char] EQLU %C':'
;  1589          2021  4              THEN
;  1590          2022  5                  BEGIN
;  1591          2023  5  !
;  1592          2024  5  ! substitute 'SYS$INPUT' for 'KB'
;  1593          2025  5  !
;  1594          2026  5                  substitute_addr = sysinput;
;  1595          2027  5                  substitute_len = sysinput_l;
;  1596          2028  5                  END
;  1597          2029  4              ELSE
;  1598          2030  5                  BEGIN
;  1599          2031  5
;  1600          2032  5                  LOCAL
;  1601          2033  5                      unit_number;                ! work field for conversion
;  1602          2034  5
;  1603          2035  5  !
```

```
;  1604          2036  5 ! This makes it easier to reference the fields of the
;  1605          2037  5 ! device_id
;  1606          2038  5 !
;  1607          2039  5
;  1608          2040  5                   BIND
;  1609          2041  5                       controller_id = device_id [0] : BYTE,
;  1610          2042  5                       unit_id = device_id [1] : VECTOR [2, BYTE];
;  1611          2043  5
;  1612          2044  5 !
;  1613          2045  5 ! substitute 'TT' for 'KB'
;  1614          2046  5 !
;  1615          2047  5                   device_name = %ASCII'TT';
;  1616          2048  5 !
;  1617          2049  5 ! and convert to VAX format
;  1618          2050  5 !
;  1619          2051  5 ! convert nnn to binary integer
;  1620          2052  5 !
;  1621          2053  5                   unit_number = 0;
;  1622          2054  5
;  1623          2055  5                   WHILE (.string [.next_char] GEQU %C'0' AND (.string [.next_char] LEQU %C'9')) DO
;  1624          2056  6                       BEGIN
;  1625          2057  6                       unit_number = (.unit_number*10) +  string [.next_char] - %C'0';
;  1626          2058  6                       next_char = .next_char + 1;
;  1627          2059  5                       END;
;  1628          2060  5
;  1629          2061  5 !
;  1630          2062  5 ! check it to see if it's ok (i.e., converted all of it)
;  1631          2063  5 !
;  1632          2064  5
;  1633          2065  5                   IF .string [.next_char] NEQU %C':' THEN RETURN 0;
;  1634          2066  5
;  1635          2067  5 !
;  1636          2068  5 ! if the unit_number is GTRU 255, error
;  1637          2069  5 !
;  1638          2070  5
;  1639          2071  5                   IF .unit_number GTRU 255 THEN RETURN 0;
;  1640          2072  5
;  1641          2073  5 !
;  1642          2074  5 ! check for nnn greater than 127, if it is then convert TT
;  1643          2075  5 ! to TU, and subtract 128 from the unit number.
;  1644          2076  5 ! Thus, TT128 is equivalent to TU0.
;  1645          2077  5 !
;  1646          2078  5
;  1647          2079  5                   IF .unit_number GTRU 127
;  1648          2080  5                   THEN
;  1649          2081  6                       BEGIN
;  1650          2082  6                       unit_number = .unit_number - 127;
;  1651          2083  6                       string [.start_posn + 1] = %C'U';
;  1652          2084  5                       END;
;  1653          2085  5
;  1654          2086  5 !
;  1655          2087  5 ! now convert nnn to Cnn.
;  1656          2088  5 ! Do this by shifting the unit_number right 4 (to drop
;  1657          2089  5 ! the unit_id) and adding %C'A' to get the ASCII value
;  1658          2090  5 ! of the controller_id, then, AND the unit_number with
;  1659          2091  5 ! 15 to mask out the controller and provide the unit_id
;  1660          2092  5 ! (needs to be converted to ASCII still)
```

```
;  1661          2093  5 !
;  1662          2094  5                       controller_id = .unit_number^(-4) + %C'A';
;  1663          2095  5                       unit_number = .unit_number AND 15;
;  1664          2096  5 !
;  1665          2097  5 ! and now, convert the unit_id back to ASCII and substitute
;  1666          2098  5 ! for the device unit_number (note that device_name stays
;  1667          2099  5 ! the same)
;  1668          2100  5 !
;  1669          2101  5
;  1670          2102  5                       IF .unit_number GTRU 9
;  1671          2103  5                       THEN
;  1672          2104  6                           BEGIN
;  1673          2105  6                           unit_id [0] = %C'1';
;  1674          2106  6                           unit_id [1] = .unit_number - 10 + %C'0';
;  1675          2107  6                           substitute_len = 3;          ! remember the length
;  1676          2108  6                           END
;  1677          2109  5                       ELSE
;  1678          2110  6                           BEGIN
;  1679          2111  6                           substitute_len = 2;          ! remember the length
;  1680          2112  6                           unit_id [0] = .unit_number + %C'0';
;  1681          2113  5                           END;
;  1682          2114  5
;  1683          2115  5                       dev_len = .dev_len - 2;          ! jump over the device name
;  1684          2116  5                       start_posn = .start_posn + 2;
;  1685          2117  5                       substitute_addr = device_id;     ! remember the address
;  1686          2118  4                       END;
;  1687          2119  4
;  1688          2120  3                   END;
;  1689          2121  3
;  1690          2122  3               [%ASCII'TI'] :
;  1691          2123  4                   BEGIN
;  1692          2124  4
;  1693          2125  4                   IF .string [.next_char] EQLU %C':'
;  1694          2126  4                   THEN
;  1695          2127  5                       BEGIN
;  1696          2128  5 !
;  1697          2129  5 ! substitute 'SYS$INPUT' for 'TI'
;  1698          2130  5 !
;  1699          2131  5                       substitute_addr = sysinput;
;  1700          2132  5                       substitute_len = sysinput_l;
;  1701          2133  5                       END
;  1702          2134  4                   ELSE
;  1703          2135  4                       RETURN 0;
;  1704          2136  4
;  1705          2137  3                   END;
;  1706          2138  3
;  1707          2139  3               [OTHERWISE] :
;  1708          2140  4                   BEGIN
;  1709          2141  4
;  1710          2142  4                   LOCAL
;  1711          2143  4                       unit_number;                     ! work field for conversion
;  1712          2144  4
;  1713          2145  4 !
;  1714          2146  4 ! This makes it easier to reference the fields of the
;  1715          2147  4 ! device_id
;  1716          2148  4 !
;  1717          2149  4
```

```
;  1718          2150  4              BIND
;  1719          2151  4                  controller_id = device_id [0] : BYTE,
;  1720          2152  4                  unit_id = device_id [1] : VECTOR [2, BYTE];
;  1721          2153  4
;  1722          2154  4 !
;  1723          2155  4 ! convert nnn to binary integer
;  1724          2156  4 !
;  1725          2157  4              unit_number = 0;
;  1726          2158  4
;  1727          2159  4              WHILE (.string [.next_char] GEQU %C'0' AND (.string [.next_char] LEQU %C'9')) DO
;  1728          2160  5                  BEGIN
;  1729          2161  5                  unit_number = (.unit_number*10) + .string [.next_char] - %C'0';
;  1730          2162  5                  next_char = .next_char + 1;
;  1731          2163  4                  END;
;  1732          2164  4
;  1733          2165  4 !
;  1734          2166  4 ! check it to see if it's ok (i.e., converted all of it)
;  1735          2167  4 !
;  1736          2168  4
;  1737          2169  4              IF .string [.next_char] NEQU %C':' THEN RETURN 0;
;  1738          2170  4
;  1739          2171  4 !
;  1740          2172  4 ! if the unit_number is GTRU 255, error
;  1741          2173  4 !
;  1742          2174  4
;  1743          2175  4              IF .unit_number GTRU 255 THEN RETURN 0;
;  1744          2176  4
;  1745          2177  4 !
;  1746          2178  4 ! TTnnn is a special case, we must now check for
;  1747          2179  4 ! nnn greater than 127, if it is then convert TT to TU,
;  1748          2180  4 ! and subtract 128 from the unit number
;  1749          2181  4 ! Thus, TT128 is equivalent to TU0.
;  1750          2182  4 !
;  1751          2183  4
;  1752          2184  4              IF .device_name EQLU %ASCII'TT' AND .unit_number GTRU 127
;  1753          2185  4              THEN
;  1754          2186  5                  BEGIN
;  1755          2187  5                  unit_number = .unit_number - 127;
;  1756          2188  5                  string [.start_posn + 1] = %C'U';
;  1757          2189  4                  END;
;  1758          2190  4
;  1759          2191  4 !
;  1760          2192  4 ! now convert nnn to Cnn (for all devices)
;  1761          2193  4 ! Do this by shifting the unit_number right 4 (to drop
;  1762          2194  4 ! the unit_id) and adding %C'A' to get the ASCII value
;  1763          2195  4 ! of the controller_id, then, AND the unit_number with
;  1764          2196  4 ! 15 to mask out the controller and provide the unit_id
;  1765          2197  4 ! (needs to be converted to ASCII still)
;  1766          2198  4 !
;  1767          2199  4              controller_id = .unit_number^(-4) + %C'A';
;  1768          2200  4              unit_number = .unit_number AND 15;
;  1769          2201  4 !
;  1770          2202  4 ! and now, convert the unit_id back to ASCII and substitute
;  1771          2203  4 ! for the device unit_number (note that device_name stays
;  1772          2204  4 ! the same)
;  1773          2205  4 !
;  1774          2206  4
```

```
;  1775          2207  4              IF .unit_number GTRU 9
;  1776          2208  4              THEN
;  1777          2209  5                  BEGIN
;  1778          2210  5                  unit_id [0] = %C'1';
;  1779          2211  5                  unit_id [1] = .unit_number - 10 + %C'0';
;  1780          2212  5                  substitute_len = 3;                 ! remember the length
;  1781          2213  5                  END
;  1782          2214  4              ELSE
;  1783          2215  5                  BEGIN
;  1784          2216  5                  substitute_len = 2;                 ! remember the length
;  1785          2217  5                  unit_id [0] = .unit_number + %C'0';
;  1786          2218  4                  END;
;  1787          2219  4
;  1788          2220  4              dev_len = .dev_len - 2;                 ! jump over the device name
;  1789          2221  4              start_posn = .start_posn + 2;
;  1790          2222  4              substitute_addr = device_id;            ! remember the address
;  1791          2223  3              END;
;  1792          2224  3          TES;
;  1793          2225  3
;  1794          2226  3      !
;  1795          2227  3      ! Get a temporary buffer from free core
;  1796          2228  3      !
;  1797          2229  3
;  1798          2230  3          IF NOT (sts = bpa$get_block (nam$c_maxrss, temp_copy)) THEN RETURN 0;
;  1799          2231  3
;  1800          2232  3      !
;  1801          2233  3      ! Now, do the substitution
;  1802          2234  3      ! 1) get all the bits up to the beginning of the device (start_posn)
;  1803          2235  3      ! 2) get the string to be substituted
;  1804          2236  3      ! 3) get the rest of the string from the ':' to the very end
;  1805          2237  3      ! 4) fill is 0
;  1806          2238  3      ! 5) copy into the temporary string
;  1807          2239  3      !
;  1808          2240  3          CH$COPY (.start_posn, .string, .substitute_len, .substitute_addr, .string_len - (.dev_ptr - .string),
;  1809          2241  3              .dev_ptr, 0, nam$c_maxrss, .temp_copy);
;  1810          2242  3      !
;  1811          2243  3      ! Now, copy from the temporary buffer into STRING
;  1812          2244  3      ! and update the length
;  1813          2245  3      !
;  1814          2246  3          .string_length = .string_len - .dev_len + .substitute_len;
;  1815          2247  3          CH$MOVE (.string_length, .temp_copy, .string);
;  1816          2248  3      !
;  1817          2249  3      ! Return the temporary buffer
;  1818          2250  3      !
;  1819          2251  3          bpa$free_block (.temp_copy, nam$c_maxrss);
;  1820          2252  2          END;                                        ! end of translate block
;  1821          2253  2          RETURN 1;
;  1822          2254  1          END;                                        ! End of routine TRANSLATE_DEV


                    54  55  50  4E  49  24  53  59  53  00C33 P.AAI:  .ASCII  \SYS$INPUT\                    ;
                        4B  53  49  44  24  53  59  53  00C3C P.AAJ:  .ASCII  \SYS$DISK\                     ;
                                                    30  00C44 P.AAK:  .ASCII  \0\                           ;
                                                3A  3A  00C45 P.AAL:  .ASCII  \::\                          ;

                                                        SYSINPUT=           P.AAI
```

```
                                        SYSDISK=               P.AAJ
                                        END_STRG=              P.AAK

                                  0FFC 00000 TRANSLATE_DEV:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11      ; 1865
                    5E          18  C2 00002            SUBL2   #24, SP                                   ;
                    56      08  BC  D0 00005            MOVL    @STRING_LENGTH, STRING_LEN                ; 1936
                    57      04  AC  D0 00009            MOVL    STRING, R7                                ; 1941
 67        56   ED  AF          02  39 0000D            MATCHC  #2, P.AAL, STRING_LEN, (R7)               ;
                                03  13 00013            BEQL    1$                                        ;
                    53          02  D0 00015            MOVL    #2, R3                                    ;
                    52          73  3E 00018 1$:        MOVAW   -(R3), NODE_PTR                           ;
                                0A  12 0001B            BNEQ    2$                                        ;
                                5B  D4 0001D            CLRL    START_POSN                                ; 1944
           67       56          3A  3A 0001F            LOCC    #58, STRING_LEN, (R7)                     ; 1945
                                18  13 00023            BEQL    3$                                        ;
                                18  11 00025            BRB     4$                                        ;
           53       52          57  C3 00027 2$:        SUBL3   R7, NODE_PTR, R3                          ; 1949
                    5B      02  A3  9E 0002B            MOVAB   2(R3), START_POSN                         ;
           50       56          5B  C3 0002F            SUBL3   START_POSN, STRING_LEN, R0                ; 1950
                    53          52  D0 00033            MOVL    NODE_PTR, R3                              ;
      02   A3       50          3A  3A 00036            LOCC    #58, R0, 2(R3)                            ;
                                02  12 0003B            BNEQ    4$                                        ;
                                51  D4 0003D 3$:        CLRL    R1                                        ;
           6E                   51  D0 0003F 4$:        MOVL    R1, DEV_PTR                               ;
                                03  12 00042            BNEQ    6$                                        ;
                              01EE  31 00044 5$:        BRW     33$                                       ; 1957
                                52  D5 00047 6$:        TSTL    NODE_PTR                                  ; 1962
                                07  12 00049            BNEQ    7$                                        ;
      0C   AE       6E          57  C3 0004B            SUBL3   R7, DEV_PTR, DEV_LEN                      ;
                                09  11 00050            BRB     8$                                        ;
           51       6E          52  C3 00052 7$:        SUBL3   NODE_PTR, DEV_PTR, R1                     ;
                0C  AE      FE  A1  9E 00056            MOVAB   -2(R1), DEV_LEN                           ;
                5F  8F        6B47  91 0005B 8$:        CMPB    (START_POSN)[R7], #95                     ; 1968
                                02  12 00060            BNEQ    9$                                        ;
                                5B  D6 00062            INCL    START_POSN                                ;
           58       57          5B  C1 00064 9$:        ADDL3   START_POSN, R7, R8                        ; 1988
                    53      02  AB  9E 00068            MOVAB   2(R11), NEXT_CHAR                         ; 1990
                    55          68  3C 0006C            MOVZWL  (R8), R5                                  ; 1995
                    54          01  D0 0006F            MOVL    #1, R4                                    ;
              5953  8F          55  B1 00072            CMPW    R5, #22867                                ; 1998
                                1E  12 00077            BNEQ    11$                                       ;
                                54  D4 00079            CLRL    R4                                        ;
                    3A        6347  91 0007B            CMPB    (NEXT_CHAR)[R7], #58                      ; 2001
                                0D  13 0007F            BEQL    10$                                       ;
                    30        6347  91 00081            CMPB    (NEXT_CHAR)[R7], #48                      ; 2002
                                BD  12 00085            BNEQ    5$                                        ;
                    3A     01 A347  91 00087            CMPB    1(NEXT_CHAR)[R7], #58                     ; 2003
                                B6  12 0008C            BNEQ    5$                                        ;
                08  AE    FF63  CF  9E 0008E 10$:       MOVAB   SYSDISK, SUBSTITUTE_ADDR                  ; 2009
                    5A          08  D0 00094            MOVL    #8, SUBSTITUTE_LEN                        ; 2010
              424B  8F          55  B1 00097 11$:       CMPW    R5, #16971                                ; 2017
                                11  12 0009C            BNEQ    12$                                       ;
                                54  D4 0009E            CLRL    R4                                        ;
                    3A        6347  91 000A0            CMPB    (NEXT_CHAR)[R7], #58                      ; 2020
                                0B  12 000A4            BNEQ    13$                                       ;
```

```
                        08    AE      FF42    CF  9E 000A6          MOVAB   SYSINPUT, SUBSTITUTE_ADDR                ; 2026
                              5A              09  D0 000AC          MOVL    #9, SUBSTITUTE_LEN                       ; 2027
                                              7B  11 000AF 12$:     BRB     19$                                      ; 2020
                              68      5454    8F  B0 000B1 13$:     MOVW    #21588, (R8)                             ; 2047
                                              50  D4 000B6          CLRL    UNIT_NUMBER                              ; 2053
                              52          6347    9A 000B8 14$:     MOVZBL  (NEXT_CHAR)[R7], R2                      ; 2055
                              30              52  91 000BC          CMPB    R2, #48                                  ;
                                              12  1F 000BF          BLSSU   15$                                      ;
                              39              52  91 000C1          CMPB    R2, #57                                  ;
                                              0D  1A 000C4          BGTRU   15$                                      ;
                    51                        0A  C5 000C6          MULL3   #10, UNIT_NUMBER, R1                     ; 2057
                              50        D0 A241   9E 000CA          MOVAB   -48(R2)[R1], UNIT_NUMBER                 ;
                                              53  D6 000CF          INCL    NEXT_CHAR                                ; 2058
                                              E5  11 000D1          BRB     14$                                      ; 2055
                              3A          6347    91 000D3 15$:     CMPB    (NEXT_CHAR)[R7], #58                     ; 2065
                                              60  12 000D7          BNEQ    20$                                      ;
                  000000FF    8F                50  D1 000D9          CMPL    UNIT_NUMBER, #255                        ; 2071
                                                57  1A 000E0          BGTRU   20$                                      ;
                  0000007F    8F                50  D1 000E2          CMPL    UNIT_NUMBER, #127                        ; 2079
                                                0A  1B 000E9          BLEQU   16$                                      ;
                              50          81    A0  9E 000EB          MOVAB   -127(R0), UNIT_NUMBER                    ; 2082
                        01 AB47           55    8F  90 000EF          MOVB    #85, 1(START_POSN)[R7]                   ; 2083
                    51                  50    FC  8F  78 000F5 16$:     ASHL    #-4, UNIT_NUMBER, R1                     ; 2094
                10  AE                  51    41  8F  81 000FA          ADDB3   #65, R1, CONTROLLER_ID                   ;
          50        50                        04  00  EF 00100          EXTZV   #0, #4, UNIT_NUMBER, UNIT_NUMBER         ; 2095
                                              09  50  D1 00105          CMPL    UNIT_NUMBER, #9                          ; 2102
                                                  0E  1B 00108          BLEQU   17$                                      ;
                        11  AE                        31  90 0010A          MOVB    #49, UNIT_ID                             ; 2105
                12  AE                  50              26  81 0010E          ADDB3   #38, UNIT_NUMBER, UNIT_ID+1              ; 2106
                                        5A              03  D0 00113          MOVL    #3, SUBSTITUTE_LEN                       ; 2107
                                                        08  11 00116          BRB     18$                                      ; 2102
                                        5A              02  D0 00118 17$:     MOVL    #2, SUBSTITUTE_LEN                       ; 2111
                11  AE                  50              30  81 0011B          ADDB3   #48, UNIT_NUMBER, UNIT_ID                ; 2112
                              0C        AE              02  C2 00120 18$:     SUBL2   #2, DEV_LEN                              ; 2115
                                        5B              02  C0 00124          ADDL2   #2, START_POSN                           ; 2116
                              08        AE        10    AE  9E 00127          MOVAB   DEVICE_ID, SUBSTITUTE_ADDR               ; 2117
                            4954        8F              55  B1 0012C 19$:     CMPW    R5, #18772                               ; 2122
                                                        11  12 00131          BNEQ    21$                                      ;
                                                        54  D4 00133          CLRL    R4                                       ;
                                        3A          6347    91 00135          CMPB    (NEXT_CHAR)[R7], #58                     ; 2125
                                                        30  12 00139 20$:     BNEQ    25$                                      ;
                              08        AE      FEAD    CF  9E 0013B          MOVAB   SYSINPUT, SUBSTITUTE_ADDR                ; 2131
                                        5A              09  D0 00141          MOVL    #9, SUBSTITUTE_LEN                       ; 2132
                                        03              54  E8 00144 21$:     BLBS    R4, 22$                                  ; 2139
                                                      0080  31 00147          BRW     31$                                      ;
                                                        50  D4 0014A 22$:     CLRL    UNIT_NUMBER                              ; 2157
                                        52          6347    9A 0014C 23$:     MOVZBL  (NEXT_CHAR)[R7], R2                      ; 2159
                                        30              52  91 00150          CMPB    R2, #48                                  ;
                                                        12  1F 00153          BLSSU   24$                                      ;
                                        39              52  91 00155          CMPB    R2, #57                                  ;
                                                        0D  1A 00158          BGTRU   24$                                      ;
                    51                  50              0A  C5 0015A          MULL3   #10, UNIT_NUMBER, R1                     ; 2161
                                        50        D0 A241   9E 0015E          MOVAB   -48(R2)[R1], UNIT_NUMBER                 ;
                                                        53  D6 00163          INCL    NEXT_CHAR                                ; 2162
                                                        E5  11 00165          BRB     23$                                      ; 2159
                                        3A          6347    91 00167 24$:     CMPB    (NEXT_CHAR)[R7], #58                     ; 2169
                                                        03  13 0016B 25$:     BEQL    27$                                      ;
```

```
                                      00C5  31 0016D 26$:   BRW     33$                                        ;
               000000FF     8F              50 D1 00170 27$:   CMPL    UNIT_NUMBER, #255                          ; 2175
                                            F4 1A 00177        BGTRU   26$                                        ;
                   5454     8F              68 B1 00179        CMPW    (R8), #21588                               ; 2184
                                            13 12 0017E        BNEQ    28$                                        ;
               0000007F     8F              50 D1 00180        CMPL    UNIT_NUMBER, #127                          ;
                                            0A 1B 00187        BLEQU   28$                                        ;
                      50           81       A0 9E 00189        MOVAB   -127(R0), UNIT_NUMBER                      ; 2187
                01 AB47            55       8F 90 0018D        MOVB    #85, 1(START_POSN)[R7]                     ; 2188
            51        50           FC       8F 78 00193 28$:   ASHL    #-4, UNIT_NUMBER, R1                       ; 2199
       10   AE        51           41       8F 81 00198        ADDB3   #65, R1, CONTROLLER_ID                     ;
  50        50        04                    00 EF 0019E        EXTZV   #0, #4, UNIT_NUMBER, UNIT_NUMBER           ; 2200
                      09                    50 D1 001A3        CMPL    UNIT_NUMBER, #9                            ; 2207
                                            0E 1B 001A6        BLEQU   29$                                        ;
                 11   AE                    31 90 001A8        MOVB    #49, UNIT_ID                               ; 2210
       12   AE        50                    26 81 001AC        ADDB3   #38, UNIT_NUMBER, UNIT_ID+1                ; 2211
                      5A                    03 D0 001B1        MOVL    #3, SUBSTITUTE_LEN                         ; 2212
                                            08 11 001B4        BRB     30$                                        ; 2207
                      5A                    02 D0 001B6 29$:   MOVL    #2, SUBSTITUTE_LEN                         ; 2216
       11   AE        50                    30 81 001B9        ADDB3   #48, UNIT_NUMBER, UNIT_ID                  ; 2217
                 0C   AE                    02 C2 001BE 30$:   SUBL2   #2, DEV_LEN                                ; 2220
                      5B                    02 C0 001C2        ADDL2   #2, START_POSN                             ; 2221
                 08   AE           10       AE 9E 001C5        MOVAB   DEVICE_ID, SUBSTITUTE_ADDR                 ; 2222
                                   14       AE 9F 001CA 31$:   PUSHAB  TEMP_COPY                                  ; 2230
                      7E           FF       8F 9A 001CD        MOVZBL  #255, -(SP)                                ;
              00000000G   00                02 FB 001D1        CALLS   #2, BPA$GET_BLOCK                          ;
                      5A                    50 E9 001D8        BLBC    STS, 33$                                   ;
            58        57                    6E C3 001DB        SUBL3   DEV_PTR, R7, R8                            ; 2240
                      58                    56 C0 001DF        ADDL2   STRING_LEN, R8                             ;
                 04   AE           FF       8F 9A 001E2        MOVZBL  #255, 4(SP)                                ;
                      59           14       AE D0 001E7        MOVL    TEMP_COPY, R9                              ; 2241
 04   AE        00    67                    5B 2C 001EB        MOVC5   START_POSN, (R7), #0, 4(SP), (R9)          ;
                                            69    001F1                                                           ;
                                            20 18 001F2        BGEQ    32$                                        ;
                      59                    5B C0 001F4        ADDL2   START_POSN, R9                             ;
                 04   AE                    5B C2 001F7        SUBL2   START_POSN, 4(SP)                          ;
 04   AE        00    08   BE               5A 2C 001FB        MOVC5   SUBSTITUTE_LEN, @SUBSTITUTE_ADDR, #0, -    ;
                                            69    00202                4(SP), (R9)                                ;
                                            0F 18 00203        BGEQ    32$                                        ;
                      59                    5A C0 00205        ADDL2   SUBSTITUTE_LEN, R9                         ;
                 04   AE                    5A C2 00208        SUBL2   SUBSTITUTE_LEN, 4(SP)                      ;
 04   AE        00    00   BE               58 2C 0020C        MOVC5   R8, @DEV_PTR, #0, 4(SP), (R9)              ;
                                            69    00213                                                           ;
                      56           0C       AE C2 00214 32$:   SUBL2   DEV_LEN, R6                                ; 2246
       08   BC        56                    5A C1 00218        ADDL3   SUBSTITUTE_LEN, R6, @STRING_LENGTH         ;
            67   14   BE           08       BC 28 0021D        MOVC3   @STRING_LENGTH, @TEMP_COPY, (R7)           ; 2247
                      7E           FF       8F 9A 00223        MOVZBL  #255, -(SP)                                ; 2251
                                   18       AE DD 00227        PUSHL   TEMP_COPY                                  ;
              00000000G   00                02 FB 0022A        CALLS   #2, BPA$FREE_BLOCK                         ;
                          50                01 D0 00231        MOVL    #1, R0                                     ; 2253
                                            04    00234        RET                                                ;
                                            50 D4 00235 33$:   CLRL    R0                                         ; 2254
                                            04    00237        RET                                                ;

; Routine Size:  568 bytes,    Routine Base:  _BPA$CODE + 0C47
```

```
;  1824          2255  1 ROUTINE find_switch (fsb, sw_string_addr, sw_string_size) =
;  1825          2256  1
;  1826          2257  1 !++
;  1827          2258  1 ! FUNCTIONAL DESCRIPTION:
;  1828          2259  1 !
;  1829          2260  1 !       Routine scans the switch string to locate the
;  1830          2261  1 !       following switches:-
;  1831          2262  1 !
;  1832          2263  1 !       /FILESIZE:n
;  1833          2264  1 !       /SIZE:n
;  1834          2265  1 !       /MODE:n
;  1835          2266  1 !       /CLUSTERSIZE:n
;  1836          2267  1 !       /POSITION:n
;  1837          2268  1 !       /RONLY
;  1838          2269  1 !
;  1839          2270  1 !       If a match is found, the value of 'n' is determined if
;  1840          2271  1 !       if relevant and checked for legality. If within limits
;  1841          2272  1 !       the value is loaded into the required field in the FSB.
;  1842          2273  1 !
;  1843          2274  1 ! FORMAL PARAMETERS:
;  1844          2275  1 !
;  1845          2276  1 !       fsb.ra.v                Pointer to fsb
;  1846          2277  1 !       sw_string_addr.ra.v     Pointer to switch string
;  1847          2278  1 !       sw_string_size.rl.v     Length of switch string
;  1848          2279  1 !
;  1849          2280  1 ! IMPLICIT INPUTS:
;  1850          2281  1 !
;  1851          2282  1 !       NONE
;  1852          2283  1 !
;  1853          2284  1 ! IMPLICIT OUTPUTS:
;  1854          2285  1 !
;  1855          2286  1 !       The appropriate switch value field in the FSB
;  1856          2287  1 !       is loaded if a good match is found.
;  1857          2288  1 !
;  1858          2289  1 ! COMPLETION CODES:
;  1859          2290  1 !
;  1860          2291  1 !       ss$_normal if success
;  1861          2292  1 !       bas$_illswiusa if syntax error
;  1862          2293  1 !       bas$_illnum if number error
;  1863          2294  1 !
;  1864          2295  1 ! SIDE EFFECTS:
;  1865          2296  1 !
;  1866          2297  1 !       NONE
;  1867          2298  1 !
;  1868          2299  1 !--
;  1869          2300  1
;  1870          2301  2     BEGIN
;  1871          2302  2
;  1872          2303  2     MAP
;  1873          2304  2         fsb : REF $fsb_def,                         ! Pointer to FSB
;  1874          2305  2         sw_string_addr : REF VECTOR [, BYTE];       ! Pointer to sw string
;  1875          2306  2
;  1876          2307  2     LOCAL
;  1877          2308  2         curr_switch : REF VECTOR [, BYTE],          ! Pointer to curr switch
;  1878          2309  2         next_switch,                                ! Pointer to next switch
;  1879          2310  2         next_value,                                 ! Pointer to next value
;  1880          2311  2         radix,                                      ! Radix of switch value
```

```
;  1881          2312  2          scan_posn,                          ! Current scan position
;  1882          2313  2          sign_flag,                          ! Negative flag
;  1883          2314  2          string_length,                      ! Length of string left
;  1884          2315  2          sts,                                ! Returned status value
;  1885          2316  2          switch_length,                      ! # of chars in switch
;  1886          2317  2          switch_value : SIGNED;              ! Value of switch
;  1887          2318  2
;  1888          2319  2      !
;  1889          2320  2      ! Make copies of parameters to allow modification
;  1890          2321  2      !
;  1891          2322  2          string_length = .sw_string_size;
;  1892          2323  2          curr_switch = .sw_string_addr;
;  1893          2324  2      !
;  1894          2325  2      ! Loop until string is exhausted
;  1895          2326  2      !
;  1896          2327  2
;  1897          2328  2          WHILE .string_length GTRU 0 DO
;  1898          2329  3              BEGIN
;  1899          2330  3              scan_posn = 0;
;  1900          2331  3      !
;  1901          2332  3      ! Check we are really pointing to the first switch, then set up pointers
;  1902          2333  3      ! to the next switch and to the next colon (value separator)
;  1903          2334  3      !
;  1904          2335  3
;  1905          2336  3              IF .curr_switch [.scan_posn] NEQU %C'/' THEN RETURN bas$_illswiusa;
;  1906          2337  3
;  1907          2338  3              next_switch = CH$FIND_CH (.string_length - 1, curr_switch [.scan_posn + 1], %C'/');
;  1908          2339  3              next_value = CH$FIND_CH (.string_length, curr_switch [.scan_posn], %C':');
;  1909          2340  3
;  1910          2341  3              IF .next_value EQLU 0
;  1911          2342  3              THEN
;  1912          2343  4                  BEGIN
;  1913          2344  4                  next_value = .string_length;
;  1914          2345  4
;  1915          2346  4                  IF .next_switch EQLU 0
;  1916          2347  4                  THEN
;  1917          2348  5                      BEGIN
;  1918          2349  5                      switch_length = .string_length;
;  1919          2350  5                      next_switch = .string_length;
;  1920          2351  5                      END
;  1921          2352  4                  ELSE
;  1922          2353  5                      BEGIN
;  1923          2354  5                      next_switch = .next_switch - curr_switch [.scan_posn];
;  1924          2355  5                      switch_length = .next_switch;
;  1925          2356  5                      END
;  1926          2357  5
;  1927          2358  4                  END
;  1928          2359  3              ELSE
;  1929          2360  4                  BEGIN
;  1930          2361  4                  next_value = .next_value - curr_switch [.scan_posn];
;  1931          2362  4
;  1932          2363  4                  IF .next_switch EQLU 0
;  1933          2364  4                  THEN
;  1934          2365  5                      BEGIN
;  1935          2366  5                      switch_length = .next_value;
;  1936          2367  5                      next_switch = .string_length;
;  1937          2368  5                      END
```

```
;  1938     2369  4              ELSE
;  1939     2370  5                  BEGIN
;  1940     2371  5                  next_switch = .next_switch - curr_switch [.scan_posn];
;  1941     2372  5                  switch_length = MIN0 (.next_switch, .next_value);
;  1942     2373  5                  END
;  1943     2374  5
;  1944     2375  3              END;
;  1945     2376  3
;  1946     2377  3          !
;  1947     2378  3          ! Point to the first character of the keyword and use the first two
;  1948     2379  3          ! characters to index to the required switch processing routine.
;  1949     2380  3          ! If there is no match then an error is generated.
;  1950     2381  3          !
;  1951     2382  3              scan_posn = 1;
;  1952     2383  4              BEGIN
;  1953     2384  4
;  1954     2385  4              BIND
;  1955     2386  4                  switch_name = curr_switch [.scan_posn] : WORD;      ! First two chars of switch
;  1956     2387  4
;  1957     2388  4              SELECTONEU .switch_name OF
;  1958     2389  4                  SET
;  1959     2390  4
;  1960     2391  4                  [%ASCII'CL'] :
;  1961     2392  5                      BEGIN
;  1962     2393  5
;  1963     2394  5                      IF .fsb [fsb$v_cl_seen]          ! Already had /CLUS...?
;  1964     2395  5                      THEN
;  1965     2396  5                          RETURN bas$_illswiusa;
;  1966     2397  5
;  1967     2398  5                      IF CH$FAIL (CH$FIND_SUB (11,     ! Check keyword
;  1968     2399  5                                  UPLIT BYTE(%ASCII'CLUSTERSIZE'), .switch_length - 1, curr_switch [.scan_posn]))
;  1969     2400  5                      THEN
;  1970     2401  5                          RETURN bas$_illswiusa;
;  1971     2402  5
;  1972     2403  5                      IF .switch_length GTRU .next_value      ! Check for value
;  1973     2404  5                      THEN
;  1974     2405  5                          RETURN bas$_illnum;
;  1975     2406  5
;  1976     2407  5                      scan_posn = .next_value + 1;
;  1977     2408  5
;  1978     2409  5                      IF .curr_switch [.scan_posn] EQLU %C'-'
;  1979     2410  5                      THEN
;  1980     2411  6                          BEGIN                        ! Set up for -ve values
;  1981     2412  6                          scan_posn = .scan_posn + 1;
;  1982     2413  6                          sign_flag = 1;
;  1983     2414  6                          END
;  1984     2415  5                      ELSE
;  1985     2416  5                          sign_flag = 0;
;  1986     2417  5
;  1987     2418  5                      IF .curr_switch [.scan_posn] EQLU %C'#'
;  1988     2419  5                      THEN
;  1989     2420  6                          BEGIN                        ! Setup for octal/decimal radix
;  1990     2421  6                          scan_posn = .scan_posn + 1;
;  1991     2422  6                          radix = 8;
;  1992     2423  6                          END
;  1993     2424  5                      ELSE
;  1994     2425  5                          radix = 10;
```

```
;  1995          2426  5                    switch_value = 0;
;  1996          2427  5
;  1997          2428  5                    WHILE (.curr_switch [.scan_posn] GEQU %C'0')    !
;  1998          2429  6                        AND (.curr_switch [.scan_posn] LSSU (.radix + %C'0'))       !
;  1999          2430  6                        AND (.scan_posn LSSU .next_switch) DO
;  2000          2431  5                        BEGIN                           ! Convert switch value
;  2001          2432  6                        switch_value = (.switch_value*.radix) + .curr_switch [.scan_posn] - %C'0';
;  2002          2433  6                        scan_posn = .scan_posn + 1;
;  2003          2434  6                        END;
;  2004          2435  5
;  2005          2436  5                    IF .sign_flag THEN switch_value = -.switch_value;       ! Negate if -ve specified
;  2006          2437  5
;  2007          2438  5                    IF (.switch_value GTR 32767)    ! Check value
;  2008          2439  6                        OR (.switch_value LSS -32768)       ! For legality
;  2009          2440  6                    THEN
;  2010          2441  5                        RETURN bas$_illnum;
;  2011          2442  5
;  2012          2443  5                    IF ((.curr_switch [.scan_posn] EQLU %C'.') AND (.radix EQLU 8)) THEN RETURN bas$_illnum;
;  2013          2444  5
;  2014          2445  5                                                    ! If decimal and octal specified
;  2015          2446  5                    fsb [fsb$w_clust] = .switch_value;      ! Store away value
;  2016          2447  5                    fsb [fsb$v_cl_seen] = 1;        ! and set flag
;  2017          2448  5                    string_length = .string_length - .scan_posn;
;  2018          2449  5                    curr_switch = curr_switch [.scan_posn];
;  2019          2450  5
;  2020          2451  5                    IF .scan_posn NEQU .next_switch THEN RETURN bas$_illnum;         ! Set up for next and check
;  2021          2452  5
;  2022          2453  5                                                    ! that in correct place
;  2023          2454  5                    END;
;  2024          2455  4
;  2025          2456  4                [%ASCII'FI', %ASCII'SI'] :
;  2026          2457  4                    BEGIN
;  2027          2458  5
;  2028          2459  5                    IF .fsb [fsb$v_fisi_seen] THEN RETURN bas$_illswiusa;
;  2029          2460  5
;  2030          2461  5                    BEGIN
;  2031          2462  6
;  2032          2463  6                    LOCAL
;  2033          2464  6                        sw_name_skip;               ! # chars to skip
;  2034          2465  6
;  2035          2466  6                    IF .switch_name EQLU %ASCII'FI' ! Find if /FILESIZE
;  2036          2467  6                    THEN                            ! of /SIZE to set offset
;  2037          2468  6                        sw_name_skip = 0
;  2038          2469  6                    ELSE
;  2039          2470  6                        sw_name_skip = 4;
;  2040          2471  6
;  2041          2472  6                    IF CH$FAIL (CH$FIND_SUB ((8 - .sw_name_skip),
;  2042          2473  6                                (UPLIT BYTE(%ASCII'FILESIZE')
;  2043          2474  7                                                    +.sw_name_skip), .switch_length - 1, curr_switch [.scan_posn
;  2044          2475  6                    THEN
;  2045          2476  6                        RETURN bas$_illswiusa;
;  2046          2477  6
;  2047          2478  6                    END;
;  2048          2479  5
;  2049          2480  5                    IF .switch_length GTRU .next_value      ! Check for value
;  2050          2481  5                    THEN
;  2051          2482  5
```

```
;  2052          2483  5                RETURN bas$_illnum;
;  2053          2484  5
;  2054          2485  5            scan_posn = .next_value + 1;
;  2055          2486  5
;  2056          2487  5            IF .curr_switch [.scan_posn] EQLU %C'#'
;  2057          2488  5            THEN
;  2058          2489  6                BEGIN
;  2059          2490  6                scan_posn = .scan_posn + 1;
;  2060          2491  6                radix = 8;
;  2061          2492  6                END
;  2062          2493  5            ELSE
;  2063          2494  5                radix = 10;
;  2064          2495  5
;  2065          2496  5            switch_value = 0;
;  2066          2497  5
;  2067          2498  6            WHILE (.curr_switch [.scan_posn] GEQU %C'0')     !
;  2068          2499  6                AND (.curr_switch [.scan_posn] LSSU (.radix + %C'0'))        !
;  2069          2500  5                AND (.scan_posn LSSU .next_switch) DO
;  2070          2501  6                BEGIN
;  2071          2502  6                switch_value = (.switch_value*.radix) + .curr_switch [.scan_posn] - %C'0';
;  2072          2503  6                scan_posn = .scan_posn + 1;
;  2073          2504  5                END;
;  2074          2505  5
;  2075          2506  5            IF (.switch_value GTRU ((1^23) - 1)) THEN RETURN bas$_illnum;
;  2076          2507  5
;  2077          2508  5            IF ((.curr_switch [.scan_posn] EQLU %C'.') AND (.radix EQLU 8)) THEN RETURN bas$_illnum;
;  2078          2509  5
;  2079          2510  5            fsb [fsb$v_size] = .switch_value;
;  2080          2511  5            fsb [fsb$v_fisi_seen] = 1;
;  2081          2512  5            string_length = .string_length - .scan_posn;
;  2082          2513  5            curr_switch = curr_switch [.scan_posn];
;  2083          2514  5
;  2084          2515  5            IF .scan_posn NEQU .next_switch THEN RETURN bas$_illnum;
;  2085          2516  5
;  2086          2517  4            END;
;  2087          2518  4
;  2088          2519  4        [%ASCII'MO'] :
;  2089          2520  5            BEGIN
;  2090          2521  5
;  2091          2522  5            IF .fsb [fsb$v_moro_seen] THEN RETURN bas$_illswiusa;
;  2092          2523  5
;  2093          2524  5            IF CH$FAIL (CH$FIND_SUB (4, UPLIT BYTE(%ASCII'MODE'), .switch_length - 1,
;  2094          2525  5                        curr_switch [.scan_posn]))
;  2095          2526  5            THEN
;  2096          2527  5                RETURN bas$_illswiusa;
;  2097          2528  5
;  2098          2529  5            IF .switch_length NEQU .next_value      ! Check for value
;  2099          2530  5            THEN
;  2100          2531  5                RETURN bas$_illnum;
;  2101          2532  5
;  2102          2533  5            scan_posn = .next_value + 1;
;  2103          2534  5
;  2104          2535  5            IF .curr_switch [.scan_posn] EQLU %C'#'
;  2105          2536  5            THEN
;  2106          2537  6                BEGIN
;  2107          2538  6                scan_posn = .scan_posn + 1;
;  2108          2539  6                radix = 8;
```

```
;  2109          2540  6                    END
;  2110          2541  5                ELSE
;  2111          2542  5                    radix = 10;
;  2112          2543  5
;  2113          2544  5                switch_value = 0;
;  2114          2545  5
;  2115          2546  6                WHILE (.curr_switch [.scan_posn] GEQU %C'0')     !
;  2116          2547  6                    AND (.curr_switch [.scan_posn] LSSU (.radix + %C'0'))       !
;  2117          2548  5                    AND (.scan_posn LSSU .next_switch) DO
;  2118          2549  6                    BEGIN
;  2119          2550  6                    switch_value = (.switch_value*.radix) + .curr_switch [.scan_posn] - %C'0';
;  2120          2551  6                    scan_posn = .scan_posn + 1;
;  2121          2552  5                    END;
;  2122          2553  5
;  2123          2554  5                IF (.switch_value GTRU 32767) THEN RETURN bas$_illnum;
;  2124          2555  5
;  2125          2556  5                IF ((.curr_switch [.scan_posn] EQLU %C'.') AND (.radix EQLU 8)) THEN RETURN bas$_illnum;
;  2126          2557  5
;  2127          2558  5                fsb [fsb$w_mode] = .switch_value + (1^15);
;  2128          2559  5                fsb [fsb$v_moro_seen] = 1;
;  2129          2560  5                string_length = .string_length - .scan_posn;
;  2130          2561  5                curr_switch = curr_switch [.scan_posn];
;  2131          2562  5
;  2132          2563  5                IF .scan_posn NEQU .next_switch THEN RETURN bas$_illnum;
;  2133          2564  5
;  2134          2565  4                END;
;  2135          2566  4
;  2136          2567  4            [%ASCII'PO'] :
;  2137          2568  5                BEGIN
;  2138          2569  5
;  2139          2570  5                IF .fsb [fsb$v_pos_seen] THEN RETURN bas$_illswiusa;
;  2140          2571  5
;  2141          2572  5                IF CH$FAIL (CH$FIND_SUB (8, UPLIT BYTE(%ASCII'POSITION'), .switch_length - 1,
;  2142          2573  5                        curr_switch [.scan_posn]))
;  2143          2574  5                THEN
;  2144          2575  5                    RETURN bas$_illswiusa;
;  2145          2576  5
;  2146          2577  5                IF .switch_length GTRU .next_value      ! Check for value
;  2147          2578  5                THEN
;  2148          2579  5                    RETURN bas$_illnum;
;  2149          2580  5
;  2150          2581  5                scan_posn = .next_value + 1;
;  2151          2582  5                switch_value = 0;
;  2152          2583  5
;  2153          2584  6                WHILE (.curr_switch [.scan_posn] GEQU %C'0')     !
;  2154          2585  6                    AND (.curr_switch [.scan_posn] LSSU %C'9')   !
;  2155          2586  5                    AND (.scan_posn LSSU .next_switch) DO
;  2156          2587  6                    BEGIN
;  2157          2588  6                    switch_value = (.switch_value*10) + .curr_switch [.scan_posn] - %C'0';
;  2158          2589  6                    scan_posn = .scan_posn + 1;
;  2159          2590  5                    END;
;  2160          2591  5
;  2161          2592  5                IF .switch_value GTRU 65535 THEN RETURN bas$_illnum;
;  2162          2593  5
;  2163          2594  5                fsb [fsb$w_position] = .switch_value;
;  2164          2595  5                fsb [fsb$v_pos_seen] = 1;
;  2165          2596  5                string_length = .string_length - .scan_posn;
```

```
;  2166          2597  5                 curr_switch = curr_switch [.scan_posn];
;  2167          2598  5
;  2168          2599  5                 IF .scan_posn NEQU .next_switch THEN RETURN bas$_illnum;
;  2169          2600  5
;  2170          2601  4                 END;
;  2171          2602  4
;  2172          2603  4             [%ASCII'RO'] :
;  2173          2604  5                 BEGIN
;  2174          2605  5
;  2175          2606  6                 IF .fsb [fsb$v_moro_seen] OR (.switch_length NEQU .next_switch) ! If has a value
;  2176          2607  5                 THEN
;  2177          2608  5                     RETURN bas$_illswiusa;
;  2178          2609  5
;  2179          2610  5                 IF CH$FAIL (CH$FIND_SUB (5, UPLIT BYTE(%ASCII'RONLY'), .switch_length - 1,
;  2180          2611  5                           curr_switch [.scan_posn]))
;  2181          2612  5                 THEN
;  2182          2613  5                     RETURN bas$_illswiusa;
;  2183          2614  5
;  2184          2615  5                 fsb [fsb$w_mode] = (1^15) + 8192;       ! Set up ronly value
;  2185          2616  5                 fsb [fsb$v_moro_seen] = 1;
;  2186          2617  5                 scan_posn = .switch_length;
;  2187          2618  5                 string_length = .string_length - .scan_posn;
;  2188          2619  5                 curr_switch = curr_switch [.scan_posn];
;  2189          2620  5
;  2190          2621  5                 IF .scan_posn NEQU .next_switch THEN RETURN bas$_illnum;
;  2191          2622  5
;  2192          2623  4                 END;
;  2193          2624  4
;  2194          2625  4             [OTHERWISE] :
;  2195          2626  4                 RETURN bas$_illswiusa;          ! Switch did not match
;  2196          2627  4                 TES;
;  2197          2628  4
;  2198          2629  3             END;
;  2199          2630  2             END;
;  2200          2631  2
;  2201          2632  2         RETURN ss$_normal;
;  2202          2633  1         END;                                        ! End of routine FIND_SWITCH


                45 5A 49 53 52 45 54 53 55 4C 43  00E7F P.AAM:  .ASCII  \CLUSTERSIZE\                    ;
                         45 5A 49 53 45 4C 49 46  00E8A P.AAN:  .ASCII  \FILESIZE\                       ;
                                  45 44 4F 4D     00E92 P.AAO:  .ASCII  \MODE\                           ;
                      4E 4F 49 54 49 53 4F 50     00E96 P.AAP:  .ASCII  \POSITION\                       ;
                               59 4C 4E 4F 52     00E9E P.AAQ:  .ASCII  \RONLY\                          ;


                                                  0FFC 00000 FIND_SWITCH:
                                                             .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11  ; 2255
                                                5E    10  C2 00002         SUBL2   #16, SP                ;
                                                57 0C  AC  D0 00005         MOVL    SW_STRING_SIZE, STRING_LENGTH  ; 2322
                                                58 08  AC  D0 00009         MOVL    SW_STRING_ADDR, CURR_SWITCH    ; 2323
                                                      57  D5 0000D 1$:      TSTL    STRING_LENGTH                  ; 2328
                                                      03  12 0000F         BNEQ    2$                             ;
                                                   032E   31 00011         BRW     64$                            ;
                                                      5B  D4 00014 2$:      CLRL    SCAN_POSN                      ; 2330
```

```
                52              58          5B  C1 00016          ADDL3   SCAN_POSN, CURR_SWITCH, R2                  ; 2336
                                2F          62  91 0001A          CMPB    (R2), #47                                   ;
                                            03  13 0001D          BEQL    4$                                          ;
                                        0318    31 0001F 3$:       BRW     63$                                         ;
                                50      FF  A7  9E 00022 4$:       MOVAB   -1(R7), R0                                  ; 2338
                01 AB48         50          2F  3A 00026          LOCC    #47, F0, 1(SCAN_POSN)[CURR_SWITCH]          ;
                                            02  12 0002C          BNEQ    5$                                          ;
                                            51  D4 0002E          CLRL    R1                                          ;
                            08  AE          51  D0 00030 5$:       MOVL    R1, NEXT_SWITCH                             ;
                            62              57  3A 00034          LOCC    #58, STRING_LENGTH, (R2)                    ; 2339
                                            02  12 00038          BNEQ    6$                                          ;
                                            51  D4 0003A          CLRL    R1                                          ;
                            04  AE          51  D0 0003C 6$:       MOVL    R1, NEXT_VALUE                              ;
                                            18  12 00040          BNEQ    8$                                          ; 2341
                            04  AE          57  D0 00042          MOVL    STRING_LENGTH, NEXT_VALUE                   ; 2344
                                08  AE      D5  00046          TSTL    NEXT_SWITCH                                 ; 2346
                                            05  12 00049          BNEQ    7$                                          ;
                                55          57  D0 0004B          MOVL    STRING_LENGTH, SWITCH_LENGTH                ; 2349
                                            17  11 0004E          BRB     9$                                          ; 2350
                            08  AE          52  C2 00050 7$:       SUBL2   R2, NEXT_SWITCH                             ; 2354
                                55      08  AE  D0 00054          MOVL    NEXT_SWITCH, SWITCH_LENGTH                  ; 2355
                                            28  11 00058          BRB     12$                                         ; 2346
                            04  AE          52  C2 0005A 8$:       SUBL2   R2, NEXT_VALUE                              ; 2361
                                08  AE      D5  0005E          TSTL    NEXT_SWITCH                                 ; 2363
                                            0A  12 00061          BNEQ    10$                                         ;
                                55      04  AE  D0 00063          MOVL    NEXT_VALUE, SWITCH_LENGTH                   ; 2366
                            08  AE          57  D0 00067 9$:       MOVL    STRING_LENGTH, NEXT_SWITCH                  ; 2367
                                            15  11 0006B          BRB     12$                                         ; 2363
                            08  AE          52  C2 0006D 10$:      SUBL2   R2, NEXT_SWITCH                             ; 2371
                                51      08  AE  D0 00071          MOVL    NEXT_SWITCH, R1                             ; 2372
                            04  AE          51  D1 00075          CMPL    R1, NEXT_VALUE                              ;
                                            04  1B 00079          BLEQU   11$                                         ;
                                51      04  AE  D0 0007B          MOVL    NEXT_VALUE, R1                              ;
                                55          51  D0 0007F 11$:      MOVL    R1, SWITCH_LENGTH                           ;
                                5B          01  D0 00082 12$:      MOVL    #1, SCAN_POSN                               ; 2382
                5A              58          5B  C1 00085          ADDL3   SCAN_POSN, CURR_SWITCH, R10                  ; 2386
                        4C43    8F          6A  B1 00089          CMPW    (R10), #19523                               ; 2391
                                            03  13 0008E          BEQL    13$                                         ;
                                        00A6    31 00090          BRW     26$                                         ;
                                59      04  AC  D0 00093 13$:      MOVL    FSB, R9                                     ; 2394
                                84      22  A9  E8 00097          BLBS    34(R9), 3$                                  ;
                            0C  AE      FF  A5  9E 0009B          MOVAB   -1(R5), 12(SP)                              ; 2399
FF34    CF          0B  A           6A      0C  AE  39 000A0          MATCHC  12(SP), (R10), #11, P.AAM                   ;
                                            04  13 000A8          BEQL    14$                                         ;
                                53      0C  AE  D0 000AA          MOVL    12(SP), R3                                  ;
                                53      0C  AE  C2 000AE 14$:      SUBL2   12(SP), R3                                  ;
                                            03  12 000B2          BNEQ    15$                                         ;
                                        0283    31 000B4          BRW     63$                                         ;
                            04  AE          55  D1 000B7 15$:      CMPL    SWITCH_LENGTH, NEXT_VALUE                   ; 2403
                                            03  1B 000BB          BLEQU   17$                                         ;
                                        0272    31 000BD 16$:      BRW     62$                                         ;
                5B          04  AE          01  C1 000C0 17$:      ADDL3   #1, NEXT_VALUE, SCAN_POSN                   ; 2407
                                2D          6B48    91 000C5          CMPB    (SCAN_POSN)[CURR_SWITCH], #45               ; 2409
                                            07  12 000C9          BNEQ    18$                                         ;
                                            5B  D6 000CB          INCL    SCAN_POSN                                   ; 2412
                                6E          01  D0 000CD          MOVL    #1, SIGN_FLAG                               ; 2413
                                            02  11 000D0          BRB     19$                                         ; 2409
```

```
                                    6E  D4 000D2 18$:     CLRL    SIGN_FLAG                                       ; 2416
                                23  6B48  91 000D4 19$:     CMPB    (SCAN_POSN)[CURR_SWITCH], #35                   ; 2418
                                    07  12 000D8           BNEQ    20$                                             ;
                                    5B  D6 000DA           INCL    SCAN_POSN                                       ; 2421
                                56  08  D0 000DC           MOVL    #8, RADIX                                       ; 2422
                                    03  11 000DF           BRB     21$                                             ; 2418
                                56  0A  D0 000E1 20$:     MOVL    #10, RADIX                                      ; 2425
                                    54  D4 000E4 21$:     CLRL    SWITCH_VALUE                                    ; 2427
                                51  6B48  9A 000E6 22$:     MOVZBL  (SCAN_POSN)[CURR_SWITCH], R1                    ; 2429
                                30  51  91 000EA           CMPB    R1, #48                                         ;
                                    1C  1F 000ED           BLSSU   23$                                             ;
                                50  30  A6  9E 000EF       MOVAB   48(R6), R0                                      ; 2430
                                50  51  D1 000F3           CMPL    R1, R0                                          ;
                                    13  1E 000F6           BGEQU   23$                                             ;
                            08  AE  5B  D1 000F8           CMPL    SCAN_POSN, NEXT_SWITCH                          ; 2431
                                    0D  1E 000FC           BGEQU   23$                                             ;
                        50  54  56  C5 000FE               MULL3   RADIX, SWITCH_VALUE, R0                         ; 2433
                                54  D0 A140  9E 00102      MOVAB   -48(R1)[R0], SWITCH_VALUE                       ;
                                    5B  D6 00107           INCL    SCAN_POSN                                       ; 2434
                                    DB  11 00109           BRB     22$                                             ; 2429
                                03  6E  E9 0010B 23$:     BLBC    SIGN_FLAG, 24$                                  ; 2437
                                54  54  CE 0010E           MNEGL   SWITCH_VALUE, SWITCH_VALUE                      ;
                        00007FFF  8F  54  D1 00111 24$:     CMPL    SWITCH_VALUE, #32767                            ; 2439
                                    A3  14 00118           BGTR    16$                                             ;
                        FFFF8000  8F  54  D1 0011A         CMPL    SWITCH_VALUE, #-32768                           ; 2440
                                    9A  19 00121           BLSS    16$                                             ;
                                2E  6B48  91 00123         CMPB    (SCAN_POSN)[CURR_SWITCH], #46                   ; 2444
                                    05  12 00127           BNEQ    25$                                             ;
                                08  56  D1 00129           CMPL    RADIX, #8                                       ;
                                    8F  13 0012C           BEQL    16$                                             ;
                            12  A9  54  B0 0012E 25$:     MOVW    SWITCH_VALUE, 18(R9)                            ; 2447
                            22  A9  01  88 00132           BISB2   #1, 34(R9)                                      ; 2448
                                01EA  31 00136             BRW     61$                                             ; 2449
                        4946  8F  6A  B1 00139 26$:     CMPW    (R10), #18758                                   ; 2457
                                    0A  13 0013E           BEQL    27$                                             ;
                        4953  8F  6A  B1 00140             CMPW    (R10), #18771                                   ;
                                    03  13 00145           BEQL    27$                                             ;
                                C09B  31 00147             BRW     39$                                             ;
                                59  04  AC  D0 0014A 27$:     MOVL    FSB, R9                                     ; 2460
                        03      22  A9  02  E1 0014E       BBC     #2, 34(R9), 29$                                 ;
                                    01E4  31 00153 28$:     BRW     63$                                             ;
                        4946  8F  6A  B1 00156 29$:     CMPW    (R10), #18758                                   ; 2467
                                    04  12 0015B           BNEQ    30$                                             ;
                                    50  D4 0015D           CLRL    SW_NAME_SKIP                                    ; 2469
                                    03  11 0015F           BRB     31$                                             ;
                                50  04  D0 00161 30$:     MOVL    #4, SW_NAME_SKIP                                ; 2471
                        51      08  50  C3 00164 31$:     SUBL3   SW_NAME_SKIP, #8, R1                            ; 2473
                            0C  AE  FF  A5  9E 00168       MOVAB   -1(R5), 12(SP)                                  ; 2475
FE71 CF40               51  6A  0C  AE  39 0016D           MATCHC  12(SP), (R10), R1, P.AAN[SW_NAME_SKIP]          ;
                                    04  13 00176           BEQL    32$                                             ;
                                53  0C  AE  D0 00178       MOVL    12(SP), R3                                      ;
                                53  0C  AE  C2 0017C 32$:     SUBL2   12(SP), R3                                  ;
                                    D1  13 00180           BEQL    28$                                             ;
                            04  AE  55  D1 00182           CMPL    SWITCH_LENGTH, NEXT_VALUE                       ; 2481
                                    43  1A 00186           BGTRU   37$                                             ;
                        5B  04  AE  01  C1 00188           ADDL3   #1, NEXT_VALUE, SCAN_POSN                       ; 2485
                                23  6B48  91 0018D         CMPB    (SCAN_POSN)[CURR_SWITCH], #35                   ; 2487
```

```
                              07  12 00191           BNEQ    33$                                      ;
                              5B  D6 00193           INCL    SCAN_POSN                                ; 2490
                     56       08  D0 00195           MOVL    #8, RADIX                                ; 2491
                              03  11 00198           BRB     34$                                      ; 2487
                     56       0A  D0 0019A 33$:      MOVL    #10, RADIX                               ; 2494
                              54  D4 0019D 34$:      CLRL    SWITCH_VALUE                             ; 2496
                     51     6B48  9A 0019F 35$:      MOVZBL  (SCAN_POSN)[CURR_SWITCH], R1             ; 2498
                     30       51  91 001A3           CMPB    R1, #48                                  ;
                              1C  1F 001A6           BLSSU   36$                                      ;
                     50   30  A6  9E 001A8           MOVAB   48(R6), R0                               ; 2499
                     50       51  D1 001AC           CMPL    R1, R0                                   ;
                              13  1E 001AF           BGEQU   36$                                      ;
                08   AE       5B  D1 001B1           CMPL    SCAN_POSN, NEXT_SWITCH                   ; 2500
                              0D  1E 001B5           BGEQU   36$                                      ;
         50          54       56  C5 001B7           MULL3   RADIX, SWITCH_VALUE, R0                  ; 2502
                     54  D0 A140  9E 001BB           MOVAB   -48(R1)[R0], SWITCH_VALUE                ;
                              5B  D6 001C0           INCL    SCAN_POSN                                ; 2503
                              DB  11 001C2           BRB     35$                                      ; 2498
           007FFFFF  8F       54  D1 001C4 36$:      CMPL    SWITCH_VALUE, #8388607                   ; 2506
                              4D  1A 001CB 37$:      BGTRU   44$                                      ;
                     2E     6B48  91 001CD           CMPB    (SCAN_POSN)[CURR_SWITCH], #46            ; 2508
                              05  12 001D1           BNEQ    38$                                      ;
                     08       56  D1 001D3           CMPL    RADIX, #8                                ;
                              42  13 001D6           BEQL    44$                                      ;
 0D   A9     18               00                     
                              54  F0 001D8 38$:      INSV    SWITCH_VALUE, #0, #24, 13(R9)            ; 2510
                22   A9       04  88 001DE           BISB2   #4, 34(R9)                               ; 2511
                            013E  31 001E2           BRW     61$                                      ; 2512
              4F4D   8F       6A  B1 001E5 39$:      CMPW    (R10), #20301                            ; 2519
                              03  13 001EA           BEQL    40$                                      ;
                            008B  31 001EC           BRW     51$                                      ;
                     59   04  AC  D0 001EF 40$:      MOVL    FSB, R9                                  ; 2522
         03     22   A9       01  E1 001F3           BBC     #1, 34(R9), 42$                          ;
                            013F  31 001F8 41$:      BRW     63$                                      ;
                0C   AE   FF  A5  9E 001FB 42$:      MOVAB   -1(R5), 12(SP)                           ; 2524
FDE7  CF     04      6A   0C  AE  39 00200           MATCHC  12(SP), (R10), #4, P.AAO                 ; 2525
                              04  13 00208           BEQL    43$                                      ;
                     53   0C  AE  D0 0020A           MOVL    12(SP), R3                               ;
                     53   0C  AE  C2 0020E 43$:      SUBL2   12(SP), R3                               ;
                              E4  13 00212           BEQL    41$                                      ;
                04   AE       55  D1 00214           CMPL    SWITCH_LENGTH, NEXT_VALUE                ; 2529
                              03  13 00218           BEQL    45$                                      ;
                            0115  31 0021A 44$:      BRW     62$                                      ;
         5B     04   AE       01  C1 0021D 45$:      ADDL3   #1, NEXT_VALUE, SCAN_POSN                ; 2533
                     23     6B48  91 00222           CMPB    (SCAN_POSN)[CURR_SWITCH], #35            ; 2535
                              07  12 00226           BNEQ    46$                                      ;
                              5B  D6 00228           INCL    SCAN_POSN                                ; 2538
                     56       08  D0 0022A           MOVL    #8, RADIX                                ; 2539
                              03  11 0022D           BRB     47$                                      ; 2535
                     56       0A  D0 0022F 46$:      MOVL    #10, RADIX                               ; 2542
                              54  D4 00232 47$:      CLRL    SWITCH_VALUE                             ; 2544
                     51     6B48  9A 00234 48$:      MOVZBL  (SCAN_POSN)[CURR_SWITCH], R1             ; 2546
                     30       51  91 00238           CMPB    R1, #48                                  ;
                              1C  1F 0023B           BLSSU   49$                                      ;
                     50   30  A6  9E 0023D           MOVAB   48(R6), R0                               ; 2547
                     50       51  D1 00241           CMPL    R1, R0                                   ;
                              13  1E 00244           BGEQU   49$                                      ;
                08   AE       5B  D1 00246           CMPL    SCAN_POSN, NEXT_SWITCH                   ; 2548
```

```
                                          0D  1E 0024A        BGEQU   49$                                        ;
               50         54              56  C5 0024C        MULL3   RADIX, SWITCH_VALUE, R0                    ; 2550
                          54      D0 A140     9E 00250        MOVAB   -48(R1)[R0], SWITCH_VALUE                  ;
                                          5B  D6 00255        INCL    SCAN_POSN                                  ; 2551
                                          DB  11 00257        BRB     48$                                        ; 2546
                00007FFF  8F              54  D1 00259 49$:   CMPL    SWITCH_VALUE, #32767                       ; 2554
                                          B8  1A 00260        BGTRU   44$                                        ;
                          2E         6B48     91 00262        CMPB    (SCAN_POSN)[CURR_SWITCH], #46              ; 2556
                                          05  12 00266        BNEQ    50$                                        ;
                          08              56  D1 00268        CMPL    RADIX, #8                                  ;
                                          AD  13 0026B        BEQL    44$                                        ;
          10   A9         54     8000     8F  A1 0026D 50$:   ADDW3   #32768, SWITCH_VALUE, 16(R9)               ; 2558
                     22   A9              02  88 00274        BISB2   #2, 34(R9)                                 ; 2559
                                          6B  11 00278        BRB     58$                                        ; 2560
                   4F50   8F              6A  B1 0027A 51$:   CMPW    (R10), #20304                              ; 2567
                                          66  12 0027F        BNEQ    59$                                        ;
                          59       04     AC  D0 00281        MOVL    FSB, R9                                    ; 2570
               03    22   A9              03  E1 00285        BBC     #3, 34(R9), 53$                            ;
                                        00AD  31 0028A 52$:   BRW     63$                                        ;
                     0C   AE       FF     A5  9E 0028D 53$:   MOVAB   -1(R5), 12(SP)                             ; 2572
     FD59   CF   08       6A       0C     AE  39 00292        MATCHC  12(SP), (R10), #8, P.AAP                   ; 2573
                                          04  13 0029A        BEQL    54$                                        ;
                          53       0C     AE  D0 0029C        MOVL    12(SP), R3                                 ;
                          53       0C     AE  C2 002A0 54$:   SUBL2   12(SP), R3                                 ;
                                          E4  13 002A4        BEQL    52$                                        ;
                     04   AE              55  D1 002A6        CMPL    SWITCH_LENGTH, NEXT_VALUE                  ; 2577
                                          2F  1A 002AA        BGTRU   57$                                        ;
               5B    04   AE              01  C1 002AC        ADDL3   #1, NEXT_VALUE, SCAN_POSN                  ; 2581
                                          54  D4 002B1        CLRL    SWITCH_VALUE                               ; 2582
                          51         6B48     9A 002B3 55$:   MOVZBL  (SCAN_POSN)[CURR_SWITCH], R1               ; 2584
                          30              51  91 002B7        CMPB    R1, #48                                    ;
                                          18  1F 002BA        BLSSU   56$                                        ;
                          39              51  91 002BC        CMPB    R1, #57                                    ; 2585
                                          13  1E 002BF        BGEQU   56$                                        ;
                     08   AE              5B  D1 002C1        CMPL    SCAN_POSN, NEXT_SWITCH                     ; 2586
                                          0D  1E 002C5        BGEQU   56$                                        ;
               50         54              0A  C5 002C7        MULL3   #10, SWITCH_VALUE, R0                      ; 2588
                          54      D0 A140     9E 002CB        MOVAB   -48(R1)[R0], SWITCH_VALUE                  ;
                                          5B  D6 002D0        INCL    SCAN_POSN                                  ; 2589
                                          DF  11 002D2        BRB     55$                                        ; 2584
                0000FFFF  8F              54  D1 002D4 56$:   CMPL    SWITCH_VALUE, #65535                       ; 2592
                                          55  1A 002DB 57$:   BGTRU   62$                                        ;
                     0A   A9              54  B0 002DD        MOVW    SWITCH_VALUE, 10(R9)                       ; 2594
                     22   A9              08  88 002E1        BISB2   #8, 34(R9)                                 ; 2595
                                          3C  11 002E5 58$:   BRB     61$                                        ; 2596
                   4F52   8F              6A  B1 002E7 59$:   CMPW    (R10), #20306                              ; 2603
                                          4C  12 002EC        BNEQ    63$                                        ;
                          59       04     AC  D0 002EE        MOVL    FSB, R9                                    ; 2606
               43    22   A9              01  E0 002F2        BBS     #1, 34(R9), 63$                            ;
                     08   AE              55  D1 002F7        CMPL    SWITCH_LENGTH, NEXT_SWITCH                 ;
                                          3D  12 002FB        BNEQ    63$                                        ;
                     0C   AE       FF     A5  9E 002FD        MOVAB   -1(R5), 12(SP)                             ; 2610
     FCF1   CF   05       6A       0C     AE  39 00302        MATCHC  12(SP), (R10), #5, P.AAQ                   ; 2611
                                          04  13 0030A        BEQL    60$                                        ;
                          53       0C     AE  D0 0030C        MOVL    12(SP), R3                                 ;
                          53       0C     AE  C2 00310 60$:   SUBL2   12(SP), R3                                 ;
                                          24  13 00314        BEQL    63$                                        ;
```

```
                                10   A9      A000    8F   B0 00316          MOVW    #-24576, 16(R9)                          ; 2615
                                22   A9              02   88 0031C          BISB2   #2, 34(R9)                               ; 2616
                                     5B              55   D0 00320          MOVL    SWITCH_LENGTH, SCAN_POSN                 ; 2617
                                     57              5B   C2 00323 61$:     SUBL2   SCAN_POSN, STRING_LENGTH                 ; 2618
                                     58              5B   C0 00326          ADDL2   SCAN_POSN, CURR_SWITCH                   ; 2619
                                08   AE              5B   D1 00329          CMPL    SCAN_POSN, NEXT_SWITCH                   ; 2621
                                                     03   12 0032D          BNEQ    62$                                      ;
                                                 FCDB    31 0032F          BRW     1$                                       ;
                                     50 00000000G    8F   D0 00332 62$:     MOVL    #BAS$_ILLNUM, R0                         ;
                                                         04 00339          RET                                              ;
                                     50 00000000G    8F   D0 0033A 63$:     MOVL    #BAS$_ILLSWIUSA, R0                      ; 2626
                                                         04 00341          RET                                              ;
                                     50              01   D0 00342 64$:     MOVL    #1, R0                                   ; 2632
                                                         04 00345          RET                                              ; 2633

; Routine Size:  838 bytes,    Routine Base:  _BPA$CODE + 0EA3

; 2203          2634  1 END                                          ! End of module BPA$FSS
; 2204          2635  1
; 2205          2636  0 ELUDOM



                                                                      .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
;  _BPA$DATA                              16  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _BPA$CODE                            4585  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                             Library Statistics
;                                                 -------- Symbols --------      Pages       Processing
;       File                                      Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1               9776        71         0       581          00:01.0



;                                       COMMAND QUALIFIERS

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

; Size:          4215 code + 386 data bytes
; Run Time:        01:32.1
; Elapsed Time:    03:46.6
```

```
; Lines/CPU Min:    1716
; Lexemes/CPU-Min: 15630
; Memory Used:  774 pages
; Compilation Complete
```

0034 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASXLATE
LIS

BPAMESAG
LIS

BPAFSS
LIS

BPAGETBLK
LIS

BASZIRET
LIS