C 13

```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
**FILE**ID**RMOSCAN                                D 16




RRRRRRRR    MM      MM    000000      SSSSSSSS    CCCCCCCC    AAAAAA    NN      NN
RRRRRRRR    MM      MM    000000      SSSSSSSS    CCCCCCCC    AAAAAA    NN      NN
RR      RR  MMMM  MMMM  00      00  SS          CC          AA      AA  NN      NN
RR      RR  MMMM  MMMM  00      00  SS          CC          AA      AA  NN      NN
RR      RR  MM  MM  MM  00    0000  SS          CC          AA      AA  NNNN    NN
RR      RR  MM  MM  MM  00    0000  SS          CC          AA      AA  NNNN    NN
RRRRRRRR    MM      MM  00  00  00    SSSSSS    CC          AA      AA  NN  NN  NN
RRRRRRRR    MM      MM  00  00  00    SSSSSS    CC          AA      AA  NN  NN  NN
RR  RR      MM      MM  0000    00          SS  CC          AAAAAAAAAA  NN    NNNN
RR  RR      MM      MM  0000    00          SS  CC          AAAAAAAAAA  NN    NNNN
RR    RR    MM      MM  00      00          SS  CC          AA      AA  NN      NN      ....
RR    RR    MM      MM  00      00          SS  CC          AA      AA  NN      NN      ....
RR      RR  MM      MM    000000    SSSSSSSS      CCCCCCCC  AA      AA  NN      NN      ....
RR      RR  MM      MM    000000    SSSSSSSS      CCCCCCCC  AA      AA  NN      NN      ....

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

Table of contents

```
0000          $BEGIN  RMOSCAN,000,RM$RMSFILENAME,<SCAN FILENAME STRING>
0000
0000   ;
0000   ;****************************************************************************
0000   ;*                                                                          *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000   ;*  ALL RIGHTS RESERVED.                                                    *
0000   ;*                                                                          *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000   ;*  TRANSFERRED.                                                            *
0000   ;*                                                                          *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000   ;*  CORPORATION.                                                            *
0000   ;*                                                                          *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000   ;*                                                                          *
0000   ;*                                                                          *
0000   ;****************************************************************************
0000   ;
```

```
0000    ;++
0000    ;
0000    ; Facility:
0000    ;
0000    ;       RMS
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This routine will parse a file specification into it's
0000    ;       component parts.  This routine must not call other RMS
0000    ;       routines or assume any normal RMS conventions.  It can
0000    ;       be called as a user mode system  service  outside  the
0000    ;       context of RMS.
0000    ;
0000    ; Environment:
0000    ;
0000    ;       VAX/VMS, executive mode (RMS) system serivce
0000    ;
0000    ; Author:
0000    ;
0000    ;       Keith B. Thompson               Creation Date:  3-Mar-1983
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-011 RAS0313         Ron Schaefer            18-Jun-1984
0000    ;               Accept a leading "_" as a valid filename character.
0000    ;
0000    ;       V03-010 RAS0275         Ron Schaefer            20-Mar-1984
0000    ;               Fix RAS0259 to recognize "[ooo,*]" and "[*,*]" as
0000    ;               valid UIC-format directories.  Also disallow "(" and ")".
0000    ;
0000    ;       V03-009 RAS0259         Ron Schaefer            28-Feb-1984
0000    ;               Convert to being table-driven.  Remove most of the little
0000    ;               subroutines and use attribute/tables instead.
0000    ;
0000    ;       V03-008 RAS0228         Ron Schaefer            4-Jan-1984
0000    ;               Recognize "[nnn,]" as a syntax error; namely FSCB$V_NULL
0000    ;               and FSCB$V_GRPMBR is not valid.
0000    ;
0000    ;       V03-007 RAS0223         Ron Schaefer            16-Dec-1983
0000    ;               Change $SCBDEF and SCB$xxx to $FSCBDEF and FSCB$xxx.
0000    ;
0000    ;       V03-006 RAS0199         Ron Schaefer            6-Oct-1983
0000    ;               Recognize "_$n$ddcm:" as a valid device name.
0000    ;               Eliminate "__x:" as a valid device name.
0000    ;               Completely recognize "$" as a valid initial character.
0000    ;
0000    ;       V03-005 RAS0190         Ron Schaefer            11-Sep-1983
0000    ;               Correct bugcheck caused by specifying more than 8 directory
0000    ;               levels.  In that case FOUND_DIR left R0 not-set.
0000    ;
0000    ;       V03-004 KBT0562         Keith B. Thompson       13-Jul-1983
0000    ;               Allow root directory between device and real directory
0000    ;
0000    ;       V03-003 KBT0534         Keith B. Thompson       25-May-1983
0000    ;               Don't allow real nodes after null nodes and allow
0000    ;               minus signs in the middle of directory strings (yec!)
```

```
                  0000     85 ;
                  0000     86 ;        V03-002 KBT0527         Keith B. Thompson       24-May-1983
                  0000     87 ;                Set SCB$V_PWD correctly and remove global symbols
                  0000     88 ;
                  0000     89 ;        V03-001 KBT0508         Keith B. Thompson       4-May-1983
                  0000     90 ;                Change SCB$V_ACCS to SCB$V_ACS and add concealed device
                  0000     91 ;                detection
                  0000     92 ;
                  0000     93 ;--
```

```
                                    0000     95          .SBTTL  DECLARATIONS
                                    0000     96
                                    0000     97 ;
                                    0000     98 ; Include Files:
                                    0000     99 ;
                                    0000    100
                                    0000    101          $FSCBDEF                          ; Scan Control Block definitions
                                    0000    102
```

```
                                   0000  ;
                                   0000  ; Macros:
                                   0000  ;
                                   0000
                         00000000  0000  C_LAMBDA        =  0                       ; EOS or invalid character
                         00000001  0000  C_ALPHA         =  1                       ; Upper, lower or multinational letter
                         00000002  0000  C_OCTAL         =  2                       ; Digits 0,1,2,3,4,5,6,7
                         00000003  0000  C_DECIMAL       =  3                       ; Digits 8,9
                         00000004  0000  C_DOT           =  4                       ; "."
                         00000005  0000  C_COLON         =  5                       ; ":"
                         00000006  0000  C_SEMI          =  6                       ; ";"
                         00000007  0000  C_MINUS         =  7                       ; "-"
                         00000008  0000  C_WILD          =  8                       ; "*","%"
                         00000009  0000  C_OPAREN        =  9                       ; "<","["
                         0000000A  0000  C_CPAREN        = 10                       ; ">","]"
                         0000000B  0000  C_QUOTE         = 11                       ; """"
                         0000000C  0000  C_COMMA         = 12                       ; ","
                         0000000D  0000  C_UNDER         = 13                       ; "_"
                         0000000E  0000  C_DOLLAR        = 14                       ; "$"
                         0000000E  0000  C_MAX_CLASS     = 14                       ; largest class code
                                   0000
                                   0000          ASSUME  C_MAX_CLASS     LE      15
                                   0000  ;
                                   0000  ; Local Data
                                   0000  ;
                                   0000
                                   0000  ;
                                   0000  ; Table of character class codes, indexed by ASCII character value
                                   0000  ; This tables reduces the 256 possible characters in 1 of 15 classes.
                                   0000  ;
                                   0000  CHAR_CLASS:
                                   0000
                                   0000  ;
                                   0000  ;        7-bit ASCII
                                   0000  ;
                                   0000
 00 00 00 00 00 00 00 00           0000          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; NUL - BEL
 00 00 00 00 00 00 00 00           0008          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; BS - SI
 00 00 00 00 00 00 00 00           0010          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; DLE - ETB
 00 00 00 00 00 00 00 00           0018          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; CAN - US
 00 00 08 0E 00 0B 00 00           0020          .BYTE   0,  0, 11,  0, 14,  8,  0,  0   ; SP ! " # $ % & '
 00 04 07 0C 00 08 00 00           0028          .BYTE   0,  0,  8,  0, 12,  7,  4,  0   ; ( ) * + , - . /
 02 02 02 02 02 02 02 02           0030          .BYTE   2,  2,  2,  2,  2,  2,  2,  2   ; 0 1 2 3 4 5 6 7
 00 0A 00 09 06 05 03 03           0038          .BYTE   3,  3,  5,  6,  9,  0, 10,  0   ; 8 9 : ; < = > ?
 01 01 01 01 01 01 01 00           0040          .BYTE   0,  1,  1,  1,  1,  1,  1,  1   ; @ A B C D E F G
 01 01 01 01 01 01 01 01           0048          .BYTE   1,  1,  1,  1,  1,  1,  1,  1   ; H I J K L M N O
 01 01 C1 01 01 01 01 01           0050          .BYTE   1,  1,  1,  1,  1,  1,  1,  1   ; P Q R S T U V W
 0D 00 0A 00 09 01 01 01           0058          .BYTE   1,  1,  1,  9,  0, 10,  0, 13   ; X Y Z [ \ ] ^ _
 01 01 01 01 01 01 01 00           0060          .BYTE   0,  1,  1,  1,  1,  1,  1,  1   ; ` a b c d e f g
 01 01 01 01 01 01 01 01           0068          .BYTE   1,  1,  1,  1,  1,  1,  1,  1   ; h i j k l m n o
 01 01 01 01 01 01 01 01           0070          .BYTE   1,  1,  1,  1,  1,  1,  1,  1   ; p q r s t u v w
 00 00 00 00 00 01 01 01           0078          .BYTE   1,  1,  1,  0,  0,  0,  0,  0   ; x y z { | } ~ DEL
                                   0080
                                   0080  ;
                                   0080  ;        8-bit DEC Multinational
                                   0080  ;
                                   0080
```

```
00 00 00 00 00 00 00 00   0080          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; 8-bit controls
00 00 00 00 00 00 00 00   0088          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; 8-bit controls
00 00 00 00 00 00 00 00   0090          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; 8-bit controls
00 00 00 00 00 00 00 00   0098          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; 8-bit controls
00 00 00 00 00 00 00 00   00A0          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; 8-bit graphics
00 00 00 00 00 00 00 00   00A8          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; 8-bit graphics
00 00 00 00 00 00 00 00   00B0          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; 8-bit graphics
00 00 00 00 00 00 00 00   00B8          .BYTE   0,  0,  0,  0,  0,  0,  0,  0   ; 8-bit graphics
01 01 01 01 01 01 01 01   00C0          .BYTE   1,  1,  1,  1,  1,  1,  1,  1   ; 8-bit upcase alphas
01 01 01 01 01 01 01 01   00C8          .BYTE   1,  1,  1,  1,  1,  1,  1,  1   ; 8-bit upcase alphas
01 01 01 01 01 01 01 00   00D0          .BYTE   0,  1,  1,  1,  1,  1,  1,  1   ; 8-bit upcase alphas
01 00 01 01 01 01 01 01   00D8          .BYTE   1,  1,  1,  1,  1,  1,  0,  1   ; 8-bit upcase alphas
01 01 01 01 01 01 01 01   00E0          .BYTE   1,  1,  1,  1,  1,  1,  1,  1   ; 8-bit lowcase alphas
01 01 01 01 01 01 01 01   00E8          .BYTE   1,  1,  1,  1,  1,  1,  1,  1   ; 8-bit lowcase alphas
01 01 01 01 01 01 01 00   00F0          .BYTE   0,  1,  1,  1,  1,  1,  1,  1   ; 8-bit lowcase alphas
00 00 01 01 01 01 01 01   00F8          .BYTE   1,  1,  1,  1,  1,  1,  0,  0   ; 8-bit lowcase alphas
                          0100
```

```
0100          .SBTTL  RM$SCAN_STRING, Main Parsing Routine
0100
0100    ;++
0100    ;
0100    ; Functional Description:
0100    ;
0100    ;   Working registers:
0100    ;
0100    ;       R0      - Return status
0100    ;       R1      - Current character
0100    ;       R2      - Local work register
0100    ;       R3      - Flags
0100    ;       R4      - Size of token
0100    ;       R5      - Address of token
0100    ;       R6      - Length of string
0100    ;       R7      - Address of current character
0100    ;       R11     - Address of control block
0100    ;
0100    ;
0100    ;   State Format:
0100    ;
0100    ;       label   - Description
0100    ;
0100    ;       token   -> label        ; Transitions
0100    ;
0100    ;
0100    ;       Tokens are parsed in this order:
0100    ;
0100    ;       eos                     ; end of string
0100    ;       numeric                 ; 0-9
0100    ;       alpha_numeric           ; ASCII alpha numeric characters
0100    ;       eightbit                ; 8 Bit character set
0100    ;       .                       ; period
0100    ;       ;                       ; semi-colon
0100    ;       :                       ; colon
0100    ;                               ; underscore
0100    ;       [ (])                   ; braket
0100    ;       < (>)                   ; angle braket
0100    ;       ,                       ; comma
0100    ;       "                       ; quote
0100    ;       special                 ; _ $ -
0100    ;       wild                    ; * %
0100    ;       lambda                  ; anything (including eos)
0100    ;
0100    ;
0100    ; Calling Sequence:
0100    ;
0100    ;       BSBW    RM$SCAN_STRING
0100    ;
0100    ; Input Parameters:
0100    ;
0100    ;       R6,R7   - input string descriptor
0100    ;       R11     - scan control block
0100    ;
0100    ; Implicit Inputs:
0100    ;       none
0100    ;
```

```
                                     0100    236 ; Outputs:
                                     0100    237 ;
                                     0100    238 ;       Control block filled in
                                     0100    239 ;
                                     0100    240 ; Implicit Outputs:
                                     0100    241 ;
                                     0100    242 ;       R1-R7   Destroved
                                     0100    243 ;
                                     0100    244 ; Completion Codes:
                                     0100    245 ;
                                     0100    246 ;       SS$_NORMAL
                                     0100    247 ;       SCN$_SPECNF
                                     0100    248 ;
                                     0100    249 ; Side Effects:
                                     0100    250 ;
                                     0100    251 ;       None
                                     0100    252 ;
                                     0100    253 ;--
                                     0100    254
                                     0100    255 RMSSCAN_STRING::
   6B   0104 8F    00   6E   00   2C 0100    256         MOVC5   #0,(SP),#0,#FSCB$C_BLN,(R11)    ; zero the control block
                        55   57   D0 0108    257         MOVL    R7,R5                           ; pointer to first token
                     08 AB   55   D0 010B    258         MOVL    R5,FSCB$Q_FILESPEC+4(R11)       ; save start address
                             53   7C 010F    259         CLRQ    R3                              ; clear flags and size
                             04   10 0111    260         BSBB    S0                              ; parse the string
                        50   01   9A 0113    261         MOVZBL  #1,R0                           ; set success
                                  05 0116    262         RSB                                     ; exit
                                  0117     ;
                                  0117     ;          S0       - Start state
                                  0117     ;
                                  0117     ;          eos               -> EOS
                                  0117     ;          alpha_numeric     -> S1
                                  0117     ;          eightbit          -> S1
                                  0117     ;          .                 -> NAME2
                                  0117     ;          ;                 -> NAME2
                                  0117     ;          :                 -> S2
                                  0117     ;          -                 -> S6
                                  0117     ;          _                 -> S4
                                  0117     ;          [                 -> S5
                                  0117     ;          <                 -> S5
                                  0117     ;          $                 -> S1
                                  0117     ;          wild              -> S3
                                  0117     ;          lambda            -> S6
                                  0117     ;
                                  0117
                  0334      30    0117     S0:        BSBW      GET_CHAR                    ; get next character
                    35      13    011A                BEQL      EOS_A                       ; quit if invalid or no more
             0D     51      91    011C                CMPB      R1,#C_UNDER
                    03      12    011F                BNEQ      10$
                  010D      31    0121                BRW       S4
             07     51      91    0124     10$:       CMPB      R1,#C_MINUS
                    25      13    0127                BEQL      70$
   26 0461'CF       51      E0    0129     20$:       BBS       R1,W^ALPHA_NUM,S1
             04     51      91    012F                CMPB      R1,#C_DOT
                    1E      13    0132                BEQL      N2_A
             06     51      91    0134                CMPB      R1,#C_SEMI
                    19      13    0137                BEQL      N2_A
             05     51      91    0139     40$:       CMPB      R1,#C_COLON
                    03      12    013C                BNEQ      50$
                  00BD      31    013E                BRW       S2
             09     51      91    0141     50$:       CMPB      R1,#C_OPAREN
                    03      12    0144                BNEQ      60$
                  0162      31    0146                BRW       S5
             08     51      91    0149     60$:       CMPB      R1,#C_WILD
                    29      13    014C                BEQL      S3_A
                  01D5      31    014E     70$:       BRW       S6
                                  0151
                            05    0151     EOS_A:     RSB
                                  0152
                  025B      31    0152     N2_A:      BRW       NAME2
```

```
                         0155  ;
                         0155  ;        S1      - character was alpha numeric
                         0155  ;
                         0155  ;        eos             -> NAME1
                         0155  ;        alpha_numeric   -> S1
                         0155  ;        .               -> NAME2
                         0155  ;        ;               -> NAME3
                         0155  ;        :               -> S4_4
                         0155  ;        special         -> S1
                         0155  ;        wild            -> S3
                         0155  ;        lambda          -> ACS
                         0155  ;
                         0155
            02F6     30  0155  S1:      BSBW    GET_CHAR                ; get next character
              6A     13  0158           BEQL    NO_A                    ; quit if invalid or no more
F5 0461'CF    51     E0  015A           BBS     R1,W^ALPHA_NUM,S1
        04    51     91  0160           CMPB    R1,#C_DOT
              ED     13  0163           BEQL    N2_A
        06    51     91  0165           CMPB    R1,#C_SEMI
              E8     13  0168           BEQL    N2_A
        05    51     91  016A  30$:     CMPB    R1,#C_COLON
              03     12  016D           BNEQ    40$
            00EC     31  016F           BRW     S4_4
        08    51     91  0172  40$:     CMPB    R1,#C_WILD
              03     12  0175           BNEQ    ACS
            0096     31  0177  S3_A:    BRW     S3_W
                         017A
                         017A  ;
                         017A  ;        ACS     - looking for access control string of the form:
                         017A  ;
                         017A  ;                        "abc...xyz"::
                         017A  ;
                         017A  ;        if not found it transfers to NAME0
                         017A  ;
                         017A  ;        This routine is branched to from above and S4_1
                         017A  ;
                         017A
        7E    56     7D  017A  ACS:     MOVQ    R6,-(SP)                        ; save place in string
        0B    51     91  017D           CMPB    R1,#C_QUOTE                     ; starting quote?
              3F     12  0180           BNEQ    90$
                         0182
              56     D7  0182  10$:     DECL    R6                              ; loop until we have a
              3B     19  0184           BLSS    90$                             ;  terminating quote
        22    87     91  0186           CMPB    (R7)+,#^A/"/                    ;  then check for '::'
              F7     12  0189           BNEQ    10$
                         018B
        56    02     C2  018B           SUBL2   #2,R6                           ; must have '::'
              31     19  018E           BLSS    90$                             ; if not exit
   3A3A 8F    87     B1  0190           CMPW    (R7)+,#^A/::/
              2A     12  0195           BNEQ    90$                             ; not quite
                         0197           SSB     #FSCB$V_ACS,R3                  ; flag that a acs was seen
        5E    08     C0  019B           ADDL2   #8,SP                           ; restore stack
     54 57    55     C3  019E           SUBL3   R5,R7,R4                        ; get length of string
        0B    54     D1  01A2           CMPL    R4,#11                          ; are there enough to have a pw?
              24     19  01A5           BLSS    NODE1                           ; no, skip test
73736170 8F  F5 A7   D1  01A7           CMPL    -11(R7),#^A/pass/               ; check the first half
              1A     12  01AF           BNEQ    NODE1                           ; no
```

```
          64726F77 8F    F9 A7   D1  01B1   365          CMPL    -7(R7),#^A/word/        ; check the second half
                            10   12  01B9   366          BNEQ    NODE1                   ; no
                                     01BB   367          SSB     #FSCB$V_PWD,R3          ; got one
                            0A   11  01BF   368          BRB     NODE1                   ; success means parse a node
                                     01C1   369
                      56    8E   7D  01C1   370 90$:     MOVQ    (SP)+,R6                ; restore pointers
                          01CE   31  01C4   371 NO_A:    BRW     NAME0                   ; failure means we only have
                                     01C7   372                                          ;  a name
```

```
                         01C7  ;
                         01C7  ;              NODE    - Node was found
                         01C7  ;
                         01C7  ;              lambda  -> S0
                         01C7  ;
                         01C7
                         01C7                 ASSUME  FSCB$B_FLDFLAGS EQ      0
                         01C7
54   57      55    C3    01C7  NODE:          SUBL3   R5,R7,R4                          ; get the length
 0B 6B       00    E3    01CB  NODE1:         BBCS    #FSCB$V_NODE,(R11),10$            ; is this the first node
  44 AB      14    E0    01CF                 BBS     #FSCB$V_NULL,FSCB$Q_NODE1(R11),-         ; there better not be a null
             3B          01D3                         EOS_B                             ;  node present, if so exit
  CC AB      54    A0    01D4                 ADDW2   R4,FSCB$Q_NODE(R11)               ; no, simply add length
             04    11    01D8                 BRB     20$                               ;
  0C AB      54    7D    01DA  10$:           MOVQ    R4,FSCB$Q_NODE(R11)               ; yes, copy whole descriptor
 50    01 AB       9A    01DE  20$:           MOVZBL  FSCB$B_NODES(R11),R0              ; get number of nodes
       08    50    B1    01E2                 CMPW    R0,#FSCB$C_MAXNODE                ; if gtr then 8 don't copy it
             08    14    01E5                 BGTR    30$                               ;
       54    53    C0    01E7                 ADDL2   R3,R4                             ; set local flags
44 AB40      54    7D    01EA                 MOVQ    R4,FSCB$Q_NODE1(R11)[R0]          ; copy into proper descriptor
       01 AB       96    01EF  30$:           INCB    FSCB$B_NODES(R11)                 ; up count of nodes
  04 AB      54    A0    01F2                 ADDW2   R4,FSCB$Q_FILESPEC(R11)           ; keep running sum
       55    57    D0    01F6                 MOVL    R7,R5                             ; update pointer
             53    D4    01F9                 CLRL    R3                                ; clear flags for next time
           FF19    31    01FB                 BRW     S0                                ; continue processing
                         01FE
```

```
                    01FE   401 ;
                    01FE   402 ;        S2      - Initial character was ':' can only be null node name
                    01FE   403 ;
                    01FE   404 ;        eos             -> EOS
                    01FE   405 ;        :               -> NODE
                    01FE   406 ;        lambda          -> EOS
                    01FE   407 ;
                    01FE   408
                    01FE   409 ;
                    01FE   410 ; NOTE: if a node has already been seen then this is a bad syntax
                    01FE   411 ;
                    01FE   412
                    01FE   413          ASSUME  FSCB$B_FLDFLAGS EQ      0
                    01FE   414          ASSUME  FSCB$V_NODE     EQ      0
                    01FE   415
    0E 6B      E8   01FE   416 S2:      BLBS    (R11),EOS_B                     ; bad!
                    0201   417          SSB     #FSCB$V_NULL,R3                 ; set null field flag
      0246     30   0205   418          BSBW    GET_CHAR                ; get next character
        05     13   0208   419          BEQL    EOS_B                   ; quit if invalid or no more
  05    51     91   020A   420          CMPB    R1,#C_COLON
        B8     13   020D   421          BEQL    NODE
               05   020F   422 EOS_B:   RSB
                    0210   423
```

```
                          0210  ;
                          0210  ;         S3       - A character was detected which can only apear in name or type
                          0210  ;
                          0210  ;         eos              -> NAME1
                          0210  ;         alpha_numeric    -> S3
                          0210  ;         eightbit         -> S3
                          0210  ;         .                -> NAME2
                          0210  ;         ;                -> NAME2
                          0210  ;         special          -> S3
                          0210  ;         wild             -> S3
                          0210  ;         lambda           -> NAME0
                          0210  ;
                          0210  S3_W:     SSB      #FSCB$V_WILD,R3          ; set wildcard
            0237     30   0214  S3:       BSBW     GET_CHAR                 ; get next character
              AB     13   0217            BEQL     NO_A                     ; quit if invalid or no more
F5 0461'CF    51     E0   0219            BBS      R1,W^ALPHA_NUM,S3
        08    51     91   021F            CMPB     R1,#C_WILD
              EC     13   0222            BEQL     S3_W
        04    51     91   0224            CMPB     R1,#C_DOT
              05     13   0227            BEQL     N2_B
        06    51     91   0229            CMPB     R1,#C_SEMI
              96     12   022C            BNEQ     NO_A
            017F     31   022E  N2_B:     BRW      NAME2
```

```
                        0231     ;
                        0231     ;       S4      - Initial character was '_'
                        0231     ;
                        0231     ;       eos                     -> NAME0
                        0231     ;       alpha_numeric           -> S4
                        0231     ;       wild                    -> S3
                        0231     ;       :                       -> S4_4
                        0231     ;       .                       -> NAME2
                        0231     ;       ;                       -> NAME2
                        0231     ;       "                       -> ACS
                        0231     ;       lambda                  -> NAME0
                        0231     ;
                        0231
           021A   30    0231     S4:     BSBW    GET_CHAR                ; get next character
             1F   13    0234             BEQL    10$                     ; quit if invalid or no more
F5 0461'CF   51   E0    0236             BBS     R1,W^ALPHA_NUM,S4
        08   51   91    023C             CMPB    R1,#C_WILD
             CF   13    023F             BEQL    S3_W
        05   51   91    0241             CMPB    R1,#C_COLON
             18   13    0244             BEQL    S4_4
        04   51   91    0246             CMPB    R1,#C_DOT
             E3   13    0249             BEQL    N2_B
        06   51   91    024B             CMPB    R1,#C_SEMI
             DE   13    024E             BEQL    N2_B
        0B   51   91    0250             CMPB    R1,#C_QUOTE
             03   13    0253             BEQL    20$
           013D   31    0255     10$:    BRW     NAME0
                        0258
           FF1F   31    0258     20$:    BRW     ACS
                        025B
           FF69   31    025B     S4_3:   BRW     NODE
                        025E
                        025E     ;
                        025E     ;       S4_4    - We have found a ':' which could be a device or node
                        025E     ;
                        025E     ;       eos                     -> DEVICE
                        025E     ;       :                       -> NODE
                        025E     ;       lambda                  -> DEVICE
                        025E     ;
                        025E
           01ED   30    025E     S4_4:   BSBW    GET_CHAR                ; get next character
             07   13    0261             BEQL    DEVICE0                 ; quit if invalid or no more
        05   51   91    0263             CMPB    R1,#C_COLON
             F3   13    0266             BEQL    S4_3                    ; we have a node
             56   D6    0268             INCL    R6                      ; if this is not a node
                        026A     ;       DECL    R7                      ;  back up one character
                        026A     ;       BRB     DEVICE                  ;  and get device name
                        026A
```

```
                         026A  ;
                         026A  ;        DEVICE  - A device name was found
                         026A  ;
                         026A  ;        eos             -> EOS
                         026A  ;        alpha_numeric   -> S7
                         026A  ;        .               -> NAME2
                         026A  ;        ;               -> NAME2
                         026A  ;        [               -> S5
                         026A  ;        <               -> S5
                         026A  ;        wild            -> S7
                         026A  ;        lambda          -> S6
                         026A  ;
                         026A
                         026A  DEVICE0:
              57    D7   026A          DECL    R7
                         026C  DEVICE:
                         026C
                         026C          ASSUME  FSCB$B_FLDFLAGS EQ      0
                         026C
          6B  02    88   026C          BISB2   #FSCB$M_DEVICE,(R11)
    54    57  55    C3   026F          SUBL3   R5,R7,R4
          54  53    C0   0273          ADDL2   R3,R4
       14 AB  54    7D   0276          MOVQ    R4,FSCB$Q_DEVICE(R11)
       04 AB  54    A0   027A          ADDW2   R4,FSCB$Q_FILESPEC(R11)          ; keep running sum
          55  57    D0   027E          MOVL    R7,R5
              53    D4   0281          CLRL    R3
            01C8    30   0283          BSBW    GET_CHAR                 ; get next character
              63    13   0286          BEQL    EOS_G                    ; quit if invalid or no more
          04  51    91   0288          CMPB    R1,#C_DOT
              1B    13   028B          BEQL    N2_C
          06  51    91   028D          CMPB    R1,#C_SEMI
              16    13   0290          BEQL    N2_C
 79 0461'CF   51    E0   0292          BBS     R1,W^ALPHA_NUM,DIR1
          09  51    91   0298          CMPB    R1,#C_OPAREN
              0E    13   029B          BEQL    S5
          08  51    91   029D          CMPB    R1,#C_WILD
              03    13   02A0          BEQL    40$
            0081    31   02A2          BRW     S6
            00CA    31   02A5  40$:    BRW     S7_W
                         02A8
            0105    31   02A8  N2_C:   BRW     NAME2
```

```
                              02AB   540 ;
                              02AB   541 ;         S5      - '[' or '<' was detected
                              02AB   542 ;
                              02AB   543 ;         Check for rooted directory, if so copy the descriptors into the correct
                              02AB   544 ;         set and check for a normal directory
                              02AB   545 ;
                              02AB   546 ;
                              02AB   547 ;         If rooted directory
                              02AB   548 ;
                              02AB   549 ;         eos              -> EOS
                              02AB   550 ;         [                -> S5
                              02AB   551 ;         <                -> S5
                              02AB   552 ;         alpha_numeric    -> S7
                              02AB   553 ;         .                -> NAME2
                              02AB   554 ;         ;                -> NAME2
                              02AB   555 ;         wild             -> S7
                              02AB   556 ;         lambda           -> S6
                              02AB   557 ;
                              02AB   558 ;
                              02AB   559 ;         If normal directory
                              02AB   560 ;
                              02AB   561 ;         eos              -> EOS
                              02AB   562 ;         alpha_numeric    -> S7
                              02AB   563 ;         .                -> NAME2
                              02AB   564 ;         ;                -> NAME2
                              02AB   565 ;         wild             -> S7
                              02AB   566 ;         lambda           -> S6
                              02AB   567 ;
                              02AB   568
        01B5       30         02AB   569 S5:     BSBW    SCAN_DIRECTORY
       3A 50       E9         02AE   570         BLBC    R0,EOS_G
    37 53   1A     E1         02B1   571         BBC     #FSCB$V_ROOTED,R3,DIRECTORY     ; is this root directory
    32 6B   02     E2         02B5   572         BBSS    #FSCB$V_ROOT,(R11),EOS_G        ; can only be one root
 54    57   55     C3         02B9   573         SUBL3   R5,R7,R4
       54   53     C0         02BD   574 10$:    ADDL2   R3,R4                           ; set any flags
    1C AB   54     7D         02C0   575         MOVQ    R4,FSCB$Q_ROOT(R11)             ; copy descriptor
    04 AB   54     A0         02C4   576         ADDW2   R4,FSCB$Q_FILESPEC(R11)         ; keep running sum
                              02C8   577
                              02C8   578 ;
                              02C8   579 ; Copy the normal directory descriptors into the rooted directory ones
                              02C8   580 ;
                              02C8   581
     0040 8F       28         02C8   582         MOVC3   #<FSCB$C_MAXROOT*8>,-           ; copy all the descriptors
     00C4 CB                  02CC   583                 FSCB$Q_DIRECTORY1(R11),-
     0084 CB                  02CF   584                 FSCB$Q_ROOT1(R11)
02 AB   03 AB      90         02D2   585         MOVB    FSCB$B_DIRS(R11),FSCB$B_ROOTS(R11); copy count of roots
        03 AB      94         02D7   586         CLRB    FSCB$B_DIRS(R11)                        ; zero count
     55    57      D0         02DA   587         MOVL    R7,R5
           53      D4         02DD   588         CLRL    R3
         016C      30         02DF   589         BSBW    GET_CHAR                ; get next character
           07      13         02E2   590         BEQL    EOS_G                   ; quit if invalid or no more
     09    51      91         02E4   591         CMPB    R1,#C_OPAREN
           C2      13         02E7   592         BEQL    S5
           26      11         02E9   593         BRB     DIR1
                   05         02EB   594 EOS_G:  RSB
```

```
                              02EC      DIRECTORY:
                              02EC
                              02EC                ASSUME  FSCB$B_FLDFLAGS EQ      0
                              02EC
           6B     08     88   02EC                BISB2   #FSCB$M_DIRECTORY,(R11)          ; set dir flag
     54    57     55     C3   02EF                SUBL3   R5,R7,R4                         ; get size of total dir spec
           54     02     B1   02F3                CMPW    #2,R4                            ; was this a null directory?
                  04     12   02F6                BNEQ    10$                              ;  [] or <>
                              02F8                SSB     #FSCB$V_NULL,R3                  ; if so set over all bit
           54     53     C0   02FC      10$:      ADDL2   R3,R4                            ; set any flags
        24 AB     54     7D   02FF                MOVQ    R4,FSCB$Q_DIRECTORY(R11)         ; copy descriptor
        04 AB     54     A0   0303                ADDW2   R4,FSCB$Q_FILESPEC(R11)          ; keep running sum
           55     57     D0   0307                MOVL    R7,R5
                  53     D4   030A                CLRL    R3
                013F     30   030C                BSBW    GET_CHAR                      ; get next character
                  DA     13   030F                BEQL    EOS_G                         ; quit if invalid or no more
                              0311
  5F 0461'CF      51     E0   0311      DIR1:     BBS     R1,W^ALPHA_NUM,S7
           04     51     91   0317                CMPB    R1,#C_DOT
                  8C     13   031A                BEQL    N2_C
           06     51     91   031C                CMPB    R1,#C_SEMI
                  87     13   031F                BEQL    N2_C
           08     51     91   0321                CMPB    R1,#C_WILD
                  4C     13   0324                BEQL    S7_W
                              0326      ;         BRB     S6
                              0326
```

```
                        0326      ;
                        0326      ;       S6      - Search for a quoted string, could be foreign file spec or
                        0326      ;                 ANSI magtape name:
                        0326      ;
                        0326      ;                         node::"abc...xyz"
                        0326      ;
                        0326      ;                                 - or -
                        0326      ;
                        0326      ;                 "abc...xyz"    "abc...xyz";n    "abc...xyz".;n
                        0326      ;
                        0326      ;
                        0326
        0B   51   91    0326      S6:     CMPB    R1,#C_QUOTE                 ; do we have first quote
             C0   12    0329              BNEQ    EOS_G
             56   D7    032B      10$:    DECL    R6                          ; loop to find terminating quote
             BC   19    032D              BLSS    EOS_G                       ; no more? then exit
        22   87   91    032F              CMPB    (R7)+,#^A/"/                ; is this it?
             F7   12    0332              BNEQ    10$                         ; no, continue
             56   D7    0334              DECL    R6                          ; is there a next character?
             09   19    0336              BLSS    30$                         ; no, we have a string
        22   87   91    0338              CMPB    (R7)+,#^A/"/                ; is it a double quote?
             EE   13    033B              BEQL    10$                         ; yes, continue with string
             56   D6    033D              INCL    R6                          ; no, back up one character
             57   D7    033F              DECL    R7                          ;  and process quoted name
                        0341
                        0341              ASSUME  FSCB$B_FLDFLAGS EQ      0
                        0341
                        0341      30$:    SSB     #FSCB$V_QUOTED,R3           ; we have a quoted string which is
        6B   10   88    0345              BISB2   #FSCB$M_NAME,(R11)          ;  a name of some type
   54   57   55   C3    0348              SUBL3   R5,R7,R4
        54   53   C0    034C              ADDL2   R3,R4                       ; set flags if any
             53   D4    034F              CLRL    R3
     2C AB   54   7D    0351              MOVQ    R4,FSCB$Q_NAME(R11)
     04 AB   54   A0    0355              ADDW2   R4,FSCB$Q_FILESPEC(R11) ; keep running sum
                        0359
                        0359      ;
                        0359      ;       if this was a node::"quoted" name then don't check for version number
                        0359      ;
                        0359
                        0359              ASSUME  FSCB$V_NODE     EQ      0
                        0359
        53 6B     E8    0359              BLBS    (R11),EOS_E                 ; if node present then exit
          00EF    30    035C              BSBW    GET_CHAR                    ; get next character
            4E    13    035F              BEQL    EOS_E                       ; quit if invalid or no more
   55   57   01   C3    0361              SUBL3   #1,R7,R5                    ; count the leading terminator
        04   51   91    0365              CMPB    R1,#C_DOT                   ; is there a null type
             5F   13    0368              BEQL    S10                         ;  check for type then version #
        06   51   91    036A              CMPB    R1,#C_SEMI                  ; is there a version number?
             40   12    036D              BNEQ    EOS_E                       ; no, exit
          0098    31    036F              BRW     S11                         ;  check for version #
```

```
                           0372  ;
                           0372  ;        S7       - Finished device or directory and ready for name, type version
                           0372  ;
                           0372  ;        eos               -> NAME1
                           0372  ;        alpha_numeric     -> S7
                           0372  ;        .                 -> NAME2
                           0372  ;        ;                 -> NAME2
                           0372  ;        "                 -> S6
                           0372  ;        special           -> S7
                           0372  ;        wild              -> S7
                           0372  ;        lambda            -> NAME0
                           0372  ;
                           0372  S7_W:    SSB     #FSCB$V_WILD,R3           ; set wildcard
                           0376
               00D5   30   0376  S7:      BSBW    GET_CHAR                  ; get next character
                 1A   13   0379           BEQL    NAME0                     ; quit if invalid or no more
      F5 0461'CF 51   E0   037B           BBS     R1,W^ALPHA_NUM,S7
              04 51   91   0381           CMPB    R1,#C_DOT
                 2A   13   0384           BEQL    NAME2
              06 51   91   0386           CMPB    R1,#C_SEMI
                 25   13   0389           BEQL    NAME2
              0B 51   91   038B           CMPB    R1,#C_QUOTE
                 96   13   038E           BEQL    S6
              08 51   91   0390           CMPB    R1,#C_WILD
                 DD   13   0393           BEQL    S7_W
                           0395
```

```
                    0395      ;
                    0395      ;       NAME1,NAME2     - a file name has been found
                    0395      ;
                    0395
          57   D7   0395 NAME0:   DECL    R7
                    0397
                    0397          ASSUME  FSCB$B_FLDFLAGS EQ      0
                    0397
    6B    10   88   0397 NAME1:   BISB2   #FSCB$M_NAME,(R11)
 54 57    55   C3   039A          SUBL3   R5,R7,R4
          04   12   039E          BNEQ    10$                             ; was it a null name field?
                    03A0          SSB     #FSCB$V_NULL,R3
    54    53   C0   03A4 10$:     ADDL2   R3,R4
 2C AB    54   7D   03A7          MOVQ    R4,FSCB$Q_NAME(R11)
 04 AB    54   A0   03AB          ADDW2   R4,FSCB$Q_FILESPEC(R11)         ; keep running sum
               05   03AF EOS_E:   RSB
                    03B0
                    03B0          ASSUME  FSCB$B_FLDFLAGS EQ      0
                    03B0
          E3   10   03B0 NAME2:   BSBB    NAME0                   ; process name field
          53   D4   03B2          CLRL    R3
    55    57   D0   03B4          MOVL    R7,R5                   ; reset start addr
          57   D6   03B7          INCL    R7                      ; count the leading terminator
    06    51   91   03B9          CMPB    R1,#C_SEMI              ; what was the terminator?
          4C   13   03BC          BEQL    S11                     ; ';' the version number
    04    51   91   03BE          CMPB    R1,#C_DOT               ; '.' then type
          EC   12   03C1          BNEQ    EOS_E
          04   11   03C3          BRB     S10
                    03C5
```

```
                          03C5      ;
                          03C5      ;       S10     - A '.' was found so check for file type
                          03C5      ;
                          03C5      ;       eos             -> TYPE1
                          03C5      ;       alpha_numeric   -> S10
                          03C5      ;       .               -> TYPE2
                          03C5      ;       ;               -> TYPE2
                          03C5      ;       special         -> S10
                          03C5      ;       wild            -> S10
                          03C5      ;       lambda          -> TYPE0
                          03C5      ;
                          03C5
                          03C5      S10_W:  SSB     #FSCB$V_WILD,R3         ; set wildcard
                          03C9
             0082    30   03C9      S10:    BSBW    GET_CHAR                ; get next character
               15    13   03CC              BEQL    TYPE0                   ; quit if invalid or no more
   F5 0461'CF  51    E0   03CE              BBS     R1,W^ALPHA_NUM,S10
           04  51    91   03D4              CMPB    R1,#C_DOT
               28    13   03D7              BEQL    TYPE2
           06  51    91   03D9              CMPB    R1,#C_SEMI
               23    13   03DC              BEQL    TYPE2
           08  51    91   03DE              CMPB    R1,#C_WILD
               E2    13   03E1              BEQL    S10_W
                          03E3
```

```
RMO
Sym

S6
S7
S7
SCA
TYP
TYP
TYP
VER
VER

PSE
---
.
RMS
$AB

Pha
---
Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
274
The
115
12

Mac
---
_$2
_$2
_$2
TOT

192

The
```

```
                            03E3  ;
                            03E3  ;        TYPE1,TYPE2     - a file type has been found
                            03E3  ;
                            03E3
                 57    D7   03E3  TYPE0:  DECL    R7
                            03E5
                            03E5          ASSUME  FSCB$B_FLDFLAGS EQ      0
                            03E5
            6B   20    88   03E5  TYPE1:  BISB2   #FSCB$M_TYPE,(R11)
       54   57   55    C3   03E8          SUBL3   R5,R7,R4
            54   01    B1   03EC          CMPW    #1,R4                           ; was it a null type field?
                 04    12   03EF          BNEQ    10$
                            03F1          SSB     #FSCB$V_NULL,R3
            54   53    C0   03F5  10$:    ADDL2   R3,R4
         34 AB   54    7D   03F8          MOVQ    R4,FSCB$Q_TYPE(R11)
         04 AB   54    A0   03FC          ADDW2   R4,FSCB$Q_FILESPEC(R11)         ; keep running sum
                       05   0400  EOS_F:  RSB
                            0401
                            0401          ASSUME  FSCB$B_FLDFLAGS EQ      0
                            0401
                 E0    10   0401  TYPE2:  BSBB    TYPE0                   ; process type field
                 53    D4   0403          CLRL    R3
            55   57    D0   0405          MOVL    R7,R5                   ; reset start addr
                 57    D6   0408          INCL    R7                      ; count the leading terminator
                            040A
```

```
                          040A  ;
                          040A  ;       S11     - a ';' or '.' has been found so check for version
                          040A  ;
                          040A  ;       eos             -> VERSION1
                          040A  ;       numeric         -> S11_1
                          040A  ;       -               -> S11_1
                          040A  ;       *               -> VERSION1
                          040A  ;       lambda          -> VERSION0
                          040A  ;
                          040A
                 42   10  040A  S11:    BSBB    GET_CHAR                ; get next character
                 1F   13  040C          BEQL    VERSION0                ; quit if invalid or no more
      11 4C'AF   51   E0  040E          BBS     R1,B^NUMERIC,S11_1
            07   51   91  0413          CMPB    R1,#C_MINUS
                 0C   13  0416          BEQL    S11_1
        2A   FF  A7   91  0418          CMPB    -1(R7),#^A/*/
                 0F   12  041C          BNEQ    VERSION0
                          041E          SSB     #FSCB$V_WILD,R3
                 0B   11  0422          BRB     VERSION1
                          0424
                 28   10  0424  S11_1:  BSBB    GET_CHAR                ; get next character
                 05   13  0426          BEQL    VERSION0                ; quit if invalid or no more
      F7 4C'AF   51   E0  0428          BBS     R1,B^NUMERIC,S11_1
                          042D
                          042D  ;
                          042D  ;       VERSION0,VERSION1 - a file type has been found
                          042D  ;
                          042D
                          042D  VERSION0:
                 57   D7  042D          DECL    R7
                          042F
                          042F          ASSUME  FSCB$B_FLDFLAGS EQ      0
                          042F
                          042F  VERSION1:
       6B    40  8F   88  042F          BISB2   #FSCB$M_VERSION,(R11)
      54    57   55   C3  0433          SUBL3   R5,R7,R4
            54   01   B1  0437          CMPW    #1,R4                           ; was it a null type field?
                 04   12  043A          BNEQ    10$
                          043C          SSB     #FSCB$V_NULL,R3
            54   53   C0  0440  10$:    ADDL2   R3,R4
         3C AB   54   7D  0443          MOVQ    R4,FSCB$Q_VERSION(R11)
         04 AB   54   A0  0447          ADDW2   R4,FSCB$Q_FILESPEC(R11)         ; keep running sum
                      05  044B          RSB
                          044C
                    000C  044C  NUMERIC:.WORD   <1@C_OCTAL>!<1@C_DECIMAL>
```

```
                                044E          .SBTTL  GET_CHAR, Get Next Character and Classify it
                                044E
                                044E     ;++
                                044E     ;
                                044E     ; Functional Description:
                                044E     ;
                                044E     ;        This routine gets the current character and classifies it into a
                                044E     ;        particular set
                                044E     ;
                                044E     ; Calling Sequence:
                                044E     ;
                                044E     ;        BSBW    GET_CHAR
                                044E     ;
                                044E     ; Input Parameters:
                                044E     ;
                                044E     ;        R6/R7   - Descriptor of string remaining
                                044E     ;
                                044E     ; Implicit Inputs:
                                044E     ;        none
                                044E     ;
                                044E     ; Outputs:
                                044E     ;
                                044E     ;        Z-bit   - true if valid character, false if invalid or string exhausted
                                044E     ;        R1      - Character class to check
                                044E     ;
                                044E     ; Implicit Outputs:
                                044E     ;        R6/R7   - updated to next character
                                044E     ;
                                044E     ; Side Effects:
                                044E     ;        none
                                044E     ;
                                044E     ;--
                                044E
                                044E              ASSUME  C_LAMBDA        EQ 0
                                044E
                                044E     GET_CHAR:
                    56   D7     044E              DECL    R6              , any more chars?
                    0A   19     0450              BLSS    10$             ; nope
               51   87   9A     0452              MOVZBL  (R7)+,R1        ; get char
    51   FBA6 CF41       9A     0455              MOVZBL  W^CHAR_CLASS[R1],R1 ; classify it
                         05     045B              RSB
                                045C
                    57   D6     045C     10$:     INCL    R7              ; advance ptr
                    51   D4     045E              CLRL    R1              ; set z-bit and invalid class
                         05     0460              RSB
                                0461
                                0461     ;
                                0461     ;  Alpha-numeric character classes.
                                0461     ;  Note that "-" is considered alphabetic, so non-alphabetic uses
                                0461     ;  must be checked before checking for ALPHA_NUM
                                0461     ;
                       608E     0461     ALPHA_NUM:       .WORD   <1@C_ALPHA>!    -                ; letters
                                0463                              <1@C_OCTAL>!    -                ; digits
                                0463                              <1@C_DECIMAL>!  -                ; digits
                                0463                              <1@C_MINUS>!    -                ; -
                                0463                              <1@C_UNDER>!    -                ;
                                0463                              <1@C_DOLLAR>                     ; $
                         0463         .SBTTL  SCAN_DIRECTORY, Parse a Directory String
                         0463
                         0463    ;++
                         0463    ;
                         0463    ; Functional Description:
                         0463    ;
                         0463    ;       This routine will parse a directory string
                         0463    ;
                         0463    ; Calling Sequence:
                         0463    ;
                         0463    ;       BSBW    SCAN_DIRECTORY
                         0463    ;
                         0463    ; Input Parameters:
                         0463    ;
                         0463    ;       R1      - Begining directory delemiter '[' or '<'
                         0463    ;       R6      - Length of string
                         0463    ;       R7      - Address of next character
                         0463    ;       R11     - Address of control block
                         0463    ;
                         0463    ; Implicit Inputs:
                         0463    ;       none
                         0463    ;
                         0463    ; Outputs:
                         0463    ;
                         0463    ;       R0      - true or false
                         0463    ;       R1,R2   - terminator character (R1 on input)
                         0463    ;       R6      - Length of string
                         0463    ;       R7      - Address of next character
                         0463    ;
                         0463    ; Implicit Outputs:
                         0463    ;       none
                         0463    ;
                         0463    ; Side Effects:
                         0463    ;       none
                         0463    ;
                         0463    ;--
                         0463
                         0463    SCAN_DIRECTORY:
               55    DD  0463         PUSHL   R5                ; save start address of string
               55    D6  0465         INCL    R5                ; skip leading terminator
  52   FF A7   02    81  0467         ADDB3   #2,-1(R7),R2      ; save terminator character to look for
                         046C                                   ;  NOTE:  '[' + 2 = ']' and
                         046C                                   ;          '<' + 2 = '>'
```

```
                        046C     ;
                        046C     ;       D0      - Start state
                        046C     ;
                        046C     ;       eos             -> ERROR
                        046C     ;       octal           -> D1
                        046C     ;       alpha_numeric   -> D2
                        046C     ;       .               -> D3
                        046C     ;       -               -> D4
                        046C     ;       ]               -> DIR_DONE
                        046C     ;       >               -> DIR_DONE
                        046C     ;       wild            -> D1
                        046C     ;       lambda          -> ERROR
                        046C     ;
                        046C
              E0    10  046C     D0:     BSBB    GET_CHAR                  ; get next character
              28    13  046E             BEQL    DERR1                     ; quit if invalid or no more
         02   51    91  0470             CMPB    R1,#C_OCTAL
              21    13  0473             BEQL    D1
         07   51    91  0475             CMPB    R1,#C_MINUS
              03    12  0478             BNEQ    10$
            00CC    31  047A             BRW     D4
   5D E0 A7   51    E0  047D     10$:    BBS     R1,B^ALPHA_NUM,D2
         04   51    91  0482             CMPB    R1,#C_DOT
              76    13  0485             BEQL    D3
     52   FF  A7    91  0487             CMPB    -1(R7),R2
              45    13  048B             BEQL    DIR_DONE1
         08   51    91  048D             CMPB    R1,#C_WILD
              68    12  0490             BNEQ    DERR2
                        0492     ;       BRB     D1_W
                        0492
```

```
                  0492  ;
                  0492  ;           D1      - a possible [group,member] directory is being scaned
                  0492  ;
                  0492  ;           eos             -> ERROR
                  0492  ;           octal           -> D1
                  0492  ;           ,               -> D1_1
                  0492  ;           wild            -> D1
                  0492  ;           lambda          -> D2 (backing up)
                  0492  ;
                  0492  D1_W:       SSB     #FSCB$V_WILD,R3             ; set wildcard
                  0496
           B6  10 0496  D1:         BSBB    GET_CHAR                    ; get next character
           60  13 0498  DERR1:      BEQL    DERR2                       ; quit if invalid or no more
     02    51  91 049A              CMPB    R1,#C_OCTAL
           F7  13 049D              BEQL    D1
     0C    51  91 049F              CMPB    R1,#C_COMMA
           0B  13 04A2              BEQL    D1_1
     08    51  91 04A4              CMPB    R1,#C_WILD
           E9  13 04A7              BEQL    D1_W
           56  D6 04A9              INCL    R6                          ; we did not find a match so backup
           57  D7 04AB              DECL    R7                          ;  and act like we are parsing a normal
           30  11 04AD              BRB     D2                          ;  type directory
                  04AF
                  04AF  ;
                  04AF  ; we have a group member directory
                  C4AF  ;
                  04AF
                  04AF  D1_1:       SSB     #FSCB$V_GRPMBR,R3           ; mark the first descriptor
         00DA  30 04B3              BSBW    FOUND_DIR                   ;  and stuff it
  40 60    14  E0 04B6              BBS     #FSCB$V_NULL,(R0),DERR2 ; can't be UIC-format and null
                  04BA
           92  10 04BA  D1_2:       BSBB    GET_CHAR                    ; get next character
           3C  13 04BC              BEQL    DERR2                       ; quit if invalid or no more
     02    51  91 04BE              CMPB    R1,#C_OCTAL
           F7  13 04C1              BEQL    D1_2
     08    51  91 04C3              CMPB    R1,#C_WILD
           0D  13 04C6              BEQL    D1_0
  52    FF A7  91 04C8              CMPB    -1(R7),R2                   ; if nothing matches then we have
           2C  12 04CC              BNEQ    DERR2                       ;  a problem
                  04CE              SSB     #FSCB$V_GRPMBR,R3           ; mark the second descriptor
                  04D2
                  04D2  DIR_DONE1:
         008D  31 04D2              BRW     DIR_DONE
                  04D5
                  04D5  D1_0:       SSB     #FSCB$V_WILD,R3             ; mark wild
           DF  11 04D9              BRB     D1_2                        ; rejoin processing
```

```
                          04DB  ;
                          04DB  ;      D2       - a normal directory is being parsed
                          04DB  ;
                          04DB  ;      eos             -> ERROR
                          04DB  ;      alpha_numeric   -> D2
                          04DB  ;      .               -> D3
                          04DB  ;      ]               -> DIR_DONE
                          04DB  ;      >               -> DIR_DONE
                          04DB  ;      special         -> D2
                          04DB  ;      wild            -> D2
                          04DB  ;      lambda          -> ERROR
                          04DB  ;
                          04DB
                          04DB  D2_W:  SSB     #FSCB$V_WILD,R3         ; set wildcard
                          04DF
            FF6C     30   04DF  D2:    BSBW    GET_CHAR                ; get next character
              16     13   04E2         BEQL    DERR2                   ; quit if invalid or no more
         04   51     91   04E4         CMPB    R1,#C_DOT
              14     13   04E7         BEQL    D3
F0 FF73 CF    51     E0   04E9  D2_1:  BBS     R1,W^ALPHA_NUM,D2
      52   FF A7     91   04EF         CMPB    -1(R7),R2
              DD     13   04F3         BEQL    DIR_DONE1
         08   51     91   04F5         CMPB    R1,#C_WILD
              E1     13   04F8         BEQL    D2_W
            008D     31   04FA  DERR2: BRW     DIR_ERROR
                          04FD
```

```
                         04FD  ;
                         04FD  ;          D3      - a normal directory has been found
                         04FD  ;
                         04FD  ;          eos              -> ERROR
                         04FD  ;          alpha_numeric    -> D2
                         04FD  ;          .                -> D3_1
                         04FD  ;          -                -> D4
                         04FD  ;          ]                -> DIR_DONE
                         04FD  ;          >                -> DIR_DONE
                         04FD  ;          wild             -> D2
                         04FD  ;          lambda           -> ERROR
                         04FD  ;
                         04FD
              0090  30   04FD  D3:    BSBW    FOUND_DIR
              FF4B  30   0500         BSBW    GET_CHAR                 ; get next character
                F5  13   0503         BEQL    DERR2                    ; quit if invalid or no more
          07    51  91   0505         CMPB    R1,#C_MINUS
                3F  13   0508         BEQL    D4
          04    51  91   050A         CMPB    R1,#C_DOT
                DA  12   050D         BNEQ    D2_1
              FF3C  30   050F         BSBW    GET_CHAR                 ; get next character
                E6  13   0512         BEQL    DERR2                    ; quit if invalid or no more
          04    51  91   0514         CMPB    R1,#C_DOT
                E1  12   0517         BNEQ    DERR2
                         0519
```

```
                          0519  ;
                          0519  ;       D3_2    - We have an ellipsis
                          0519  ;
                          0519  ;       eos             -> ERROR
                          0519  ;       alpha_numeric   -> D2
                          0519  ;       ]               -> DIR_DONEX
                          0519  ;       >               -> DIR_DONEX
                          0519  ;       wild            -> D2
                          0519  ;       lambda          -> ERROR
                          0519  ;
                          0519  ;
                          0519  ;       NOTE: The elips flags must be set in the previous
                          0519  ;       directory descriptor
                          0519  ;
                          0519
        50   03 AB    9A  0519  D3_2:   MOVZBL  FSCB$B_DIRS(R11),R0                     ; get number of current directory
                50    D7  051D          DECL    R0                                      ; back up one
           08   50    B1  051F          CMPW    R0,#FSCB$C_MAXDIR                       ; is there a descriptor for it?
                0A    1A  0522          BGTRU   10$                                     ; no
     50    00C4 CB40  7E  0524          MOVAQ   FSCB$Q_DIRECTORY1(R11)[R0],R0           ; get address of descriptor
                          052A          SSB     #FSCB$V_ELIPS,(R0)                      ; set flag
           55   57    D0  052E  10$:    MOVL    R7,R5                                   ; start next directory after elips
              FF1A    30  0531          BSBW    GET_CHAR                                ; get next character
                54    13  0534          BEQL    DIR_ERROR                               ; quit if invalid or no more
   A3 FF26 CF   51    E0  0536          BBS     R1,@^ALPHA_NUM,D2
        52   FF A7    91  053C          CMPB    -1(R7),R2
                3D    13  0540          BEQL    DIR_DONEX
           08   51    91  0542          CMPB    R1,#C_WILD
                43    12  0545          BNEQ    DIR_ERROR
                92    11  0547          BRB     D2_@
```

```
                          0549       ;
                          0549       ;            D4       - scanning a series of minus signs
                          0549       ;
                          0549       ;            eos               -> ERROR
                          0549       ;            .                 -> D3
                          0549       ;            -                 -> D4
                          0549       ;            ]                 -> DIR_DONE
                          0549       ;            >                 -> DIR_DONE
                          0549       ;            lambda            -> ERROR
                          0549       ;
                          0549
                          0549        D4:         SSB      #FSCB$V_MINUS,R3                  ; we have a minus sign(s)
             FEFE   30    054D                    BSBW     GET_CHAR                          ; get next character
               38   13    0550                    BEQL     DIR_ERROR                         ; quit if invalid or no more
         04    51   91    0552                    CMPB     R1,#C_DOT
               A6   13    0555                    BEQL     D3
         07    51   91    0557                    CMPB     R1,#C_MINUS
               ED   13    055A                    BEQL     D4
      52    FF A7   91    055C                    CMPB     -1(R7),R2
               28   12    0560                    BNEQ     DIR_ERROR
```

```
                          0562 DIR_DONE:
               2C    10   0562         BSBB    FOUND_DIR                        ; we found a directory
      01    03 AB    91   0564         CMPB    FSCB$B_DIRS(R11),#1              ; is there more then one dir?
               15    13   0568         BEQL    DIR_DONEX                        ; no, ignore
      08    03 AB    91   056A         CMPB    FSCB$B_DIRS(R11),#FSCB$C_MAXDIR  ; more then the descriptors?
               0F    1A   056E         BGTRU   DIR_DONEX                        ; yes
      12 60    16    E0   0570         BBS     #FSCB$V_GRPMBR,(R0),DIR_DONEUIC  ; is this group member type?
      07 60    14    E1   0574         BBC     #FSCB$V_NULL,(R0),DIR_DONEX      ; branch if not null
            03 AB    97   0578         DECB    FSCB$B_DIRS(R11)                 ; remove this directory
                          057B         SSB     #FSCB$V_ROOTED,R3                ; set the concealed flag
                          057F
                          057F DIR_DONEX:                               ; exit from ellips
               55  8ED0   057F         POPL    R5                       ; restore orginal start
         50    01    9A   0582         MOVZBL  #1,R0                    ; signal success
                     05   0585         RSB                              ; exit
                          0586
                          0586 DIR_DONEUIC:
      F5 60    14    E1   0586         BBC     #FSCB$V_NULL,(R0),DIR_DONEX ; can't be UIC-format and null
                          058A
                          058A DIR_ERROR:
               55  8ED0   058A         POPL    R5                       ; restore stack
               50    D4   058D         CLRL    R0                       ; signal error
                     05   058F         RSB                              ; return
                          0590
                          0590 ;
                          0590 ; Process a directory
                          0590 ;
                          0590 FOUND_DIR:
      50    03 AB    9A   0590         MOVZBL  FSCB$B_DIRS(R11),R0              ; get number of directories
         08    50    91   0594         CMPB    R0,#FSCB$C_MAXDIR                ; more then the descriptors?
               18    14   0597         BGTR    20$                              ; yes
   54    57    55    C3   0599         SUBL3   R5,R7,R4                         ; find size
               54    D7   059D         DECL    R4                               ; remove trailing terminator
               04    12   059F         BNEQ    10$                              ; if it is null
                          05A1         SSB     #FSCB$V_NULL,R3                  ;  set flag
         54    53    C0   05A5 10$:    ADDL2   R3,R4                            ; copy flags into descriptor
 50   00C4 CB40    7E     05A8         MOVAQ   FSCB$Q_DIRECTORY1(R11)[R0],R0    ; get addr of descriptor
         60    54    7D   05AE         MOVQ    R4,(R0)                          ; copy descriptor
            03 AB    96   05B1 20$:    INCB    FSCB$B_DIRS(R11)                 ; count it
               53    D4   05B4         CLRL    R3                               ; clear flag
         55    57    D0   05B6         MOVL    R7,R5                            ; update start address
                     05   05B9         RSB                                      ; NOTE: leave r0 = addr of desc
                          05BA
                          05BA         .END                             ; End of module
```

```
                                                                  N  2
RMOSCAN                          SCAN FILENAME STRING                          16-SEP-1984 00:35:34  VAX/VMS Macro V04-00        Page  34     RM
Symbol table                                                                    5-SEP-1984 16:22:26  [RMS.SRC]RMOSCAN.MAR;1          (29)     VO

$$.PSECT_EP                    = 00000000               FSCB$C_BLN                    = 00000104
$$RMSTEST                      = 0000001A               FSCB$C_MAXDIR                 = 00000008
$$RMS_PBUGCHK                  = 00000010               FSCB$C_MAXNODE                = 00000008
$$RMS_TBUGCHK                  = 00000008               FSCB$C_MAXROOT                = 00000008
$$RMS_UMODE                    = 00000004               FSCB$M_DEVICE                 = 00000002
ACS                              0000017A R     01      FSCB$M_DIRECTORY              = 00000008
ALPHA_NUM                        00000461 R     01      FSCB$M_NAME                   = 00000010
CHAR_CLASS                       00000000 R     01      FSCB$M_TYPE                   = 00000020
C_ALPHA                        = 00000001               FSCB$M_VERSION                = 00000040
C_COLON                        = 00000005               FSCB$Q_DEVICE                 = 00000014
C_COMMA                        = 0000000C               FSCB$Q_DIRECTORY              = 00000024
C_CPAREN                       = 0000000A               FSCB$Q_DIRECTORY1             = 000000C4
C_DECIMAL                      = 00000003               FSCB$Q_FILESPEC               = 00000004
C_DOLLAR                       = 0000000E               FSCB$Q_NAME                   = 0000002C
C_DOT                          = 00000004               FSCB$Q_NODE                   = 0000000C
C_LAMBDA                       = 00000000               FSCB$Q_NODE1                  = 00000044
C_MAX_CLASS                    = 0000000E               FSCB$Q_ROOT                   = 0000001C
C_MINUS                        = 00000007               FSCB$Q_ROOT1                  = 00000084
C_OCTAL                        = 00000002               FSCB$Q_TYPE                   = 00000034
C_OPAREN                       = 00000009               FSCB$Q_VERSION                = 0000003C
C_QUOTE                        = 0000000B               FSCB$V_ACS                    = 00000012
C_SEMI                         = 00000006               FSCB$V_ELIPS                  = 00000010
C_UNDER                        = 0000000D               FSCB$V_GRPMBR                 = 00000016
C_WILD                         = 00000008               FSCB$V_MINUS                  = 00000017
D0                               0000046C R     01      FSCB$V_NODE                   = 00000000
D1                               00000496 R     01      FSCB$V_NULL                   = 00000014
D1_0                             000004D5 R     01      FSCB$V_PWD                    = 00000015
D1_1                             000004AF R     01      FSCB$V_QUOTED                 = 00000013
D1_2                             000004BA R     01      FSCB$V_ROOT                   = 00000002
D1_W                             00000492 R     01      FSCB$V_ROOTED                 = 0000001A
D2                               000004DF R     01      FSCB$V_WILD                   = 00000011
D2_1                             000004E9 R     01      GET_CHAR                        0000044E R     01
D2_W                             000004DB R     01      N0_A                            000001C4 R     01
D3                               000004FD R     01      N2_A                            00000152 R     01
D3_2                             00000519 R     01      N2_B                            0000022E R     01
D4                               00000549 R     01      N2_C                            000002A8 R     01
DERR1                            00000498 R     01      NAME0                           00000395 R     01
DERR2                            000004FA R     01      NAME1                           00000397 R     01
DEVICE                           0000026C R     01      NAME2                           000003B0 R     01
DEVICE0                          0000026A R     01      NODE                            000001C7 R     01
DIR1                             00000311 R     01      NODE1                           000001CB R     01
DIRECTORY                        000002EC R     01      NUMERIC                         0000044C R     01
DIR_DONE                         00000562 R     01      RM$SCAN_STRING                  00000100 RG    01
DIR_DONE1                        000004D2 R     01      S0                              00000117 R     01
DIR_DONEUIC                      00000586 R     01      S1                              00000155 R     01
DIR_DONEX                        0000057F R     01      S10                             000003C9 R     01
DIR_ERROR                        0000058A R     01      S10_W                           000003C5 R     01
EOS_A                            00000151 R     01      S11                             0000040A R     01
EOS_B                            0000020F R     01      S11_1                           00000424 R     01
EOS_E                            000003AF R     01      S2                              000001FE R     01
EOS_F                            00000400 R     01      S3                              00000214 R     01
EOS_G                            000002EB R     01      S3_A                            00000177 R     01
FOUND_DIR                        00000590 R     01      S3_W                            00000210 R     01
FSCB$B_DIRS                    = 00000003               S4                              00000231 R     01
FSCB$B_FLDFLAGS                = 00000000               S4_3                            0000025B R     01
FSCB$B_NODES                   = 00000001               S4_4                            0000025E R     01
FSCB$B_ROOTS                   = 00000002               S5                              000002AB R     01
```

```
S6                               00000326 R     01
S7                               00000376 R     01
S7_W                             00000372 R     01
SCAN_DIRECTORY                   00000463 R     01
TYPE0                            000003E3 R     01
TYPE1                            000003E5 R     01
TYPE2                            00000401 R     01
VERSION0                         0000042D R     01
VERSION1                         0000042F R     01

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

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
RMS$RMSFILENAME                  000005BA  (  1466.)  01 (  1.)    PIC    USR    CON    REL    GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                            00000000  (     0.)  02 (  2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE

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

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                    29     00:00:00.0?    00:00:01.19
Command processing               108     00:00:00.68    00:00:03.60
Pass 1                           200     00:00:04.68    00:00:14.77
Symbol table sort                  0     00:00:00.30    00:00:00.53
Pass 2                           188     00:00:02.24    00:00:06.93
Symbol table output               15     00:00:00.09    00:00:00.16
Psect synopsis output              1     00:00:00.01    00:00:00.20
Cross-reference output             0     00:00:00.00    00:00:00.00
Assembler run totals             543     00:00:08.11    00:00:27.71

The working set limit was 1200 pages.
27494 bytes (54 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 197 non-local and 49 local symbols.
1156 source lines were read in Pass 1, producing 16 object records in Pass 2.
12 pages of virtual memory were used to define 11 macros.

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

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                       3
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       0
_$255$DUA28:[SYSLIB]STARLET.MLB;2                    4
TOTALS (all libraries)                               7

192 GETS were required to define 7 macros.

There were no errors, warnings or information messages.
```

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

0319 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM0RECLCK
LIS

RM0RSET
LIS

RM0SCAN
LIS

RM0PRFLNM
LIS

RM0RCLCK2
LIS

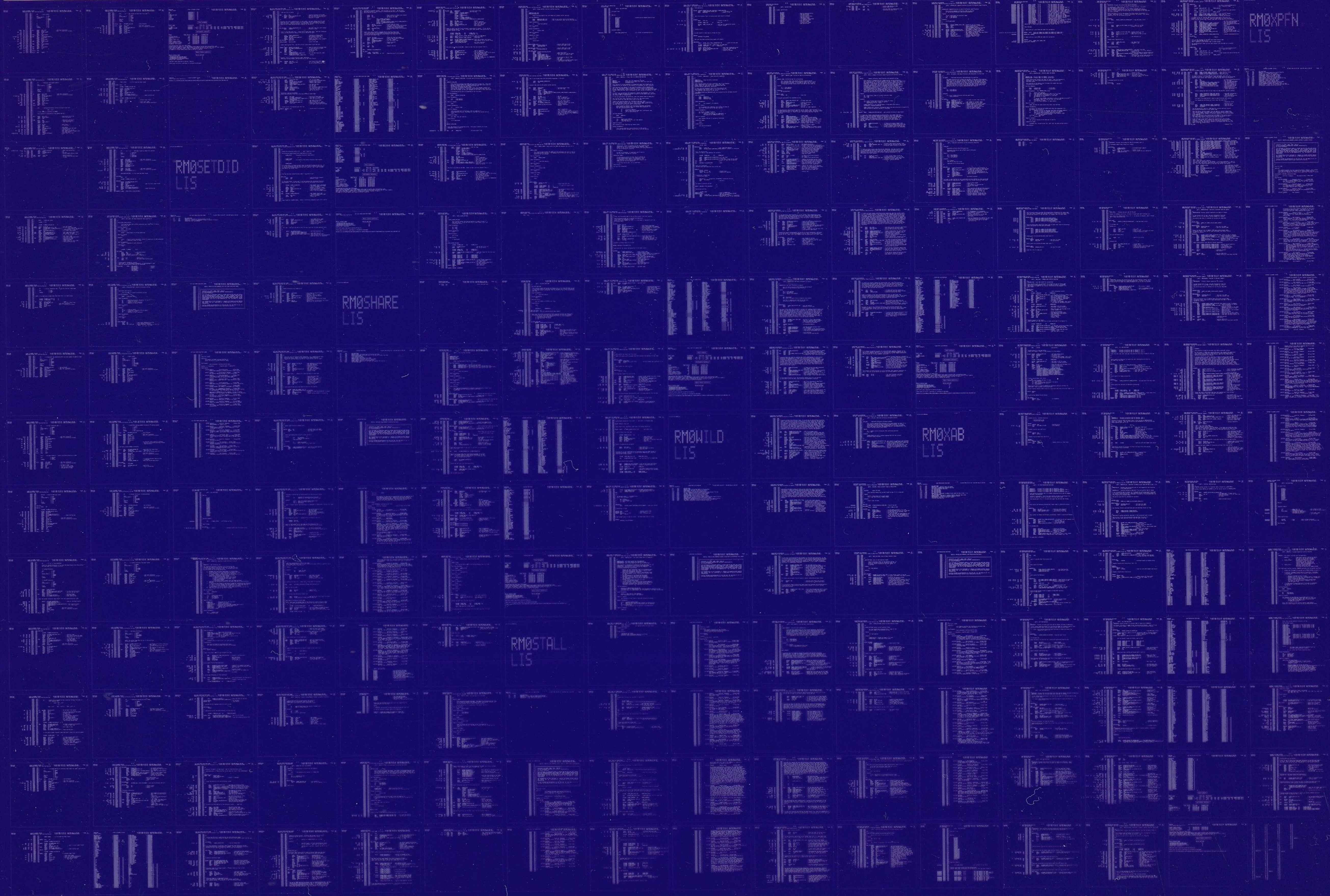
RM0RABCHK
LIS

RM0RELEAS
LIS

RM0NAMSTR
LIS

0320 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



RM0XPFN
LIS

RM0SETDID
LIS

RM0SHARE
LIS

RM0WILD
LIS

RM0XAB
LIS

RM0STALL
LIS