```
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**RMSOSRCH                                    H 1
```

```
RRRRRRRR    MM      MM    SSSSSSSS    000000      SSSSSSSS  RRRRRRRR      CCCCCCCC  HH      HH
RRRRRRRR    MM      MM    SSSSSSSS    000000      SSSSSSSS  RRRRRRRR      CCCCCCCC  HH      HH
RR      RR  MMMM  MMMM  SS          00      00  SS          RR      RR  CC          HH      HH
RR      RR  MMMM  MMMM  SS          00      00  SS          RR      RR  CC          HH      HH
RR      RR  MM  MM  MM  SS          00    0000  SS          RR      RR  CC          HH      HH
RR      RR  MM  MM  MM  SS          00    0000  SS          RR      RR  CC          HH      HH
RRRRRRRR    MM      MM    SSSSSS    00  00  00    SSSSSS    RRRRRRRR    CC          HHHHHHHHHH
RRRRRRRR    MM      MM    SSSSSS    00  00  00    SSSSSS    RRRRRRRR    CC          HHHHHHHHHH
RR  RR      MM      MM          SS  0000    00          SS  RR  RR      CC          HH      HH
RR  RR      MM      MM          SS  0000    00          SS  RR  RR      CC          HH      HH
RR    RR    MM      MM          SS  00      00          SS  RR    RR    CC          HH      HH      ....
RR    RR    MM      MM          SS  00      00          SS  RR    RR    CC          HH      HH      ....
RR      RR  MM      MM  SSSSSSSS    000000    SSSSSSSS    RR      RR    CCCCCCCC  HH      HH      ....
RR      RR  MM      MM  SSSSSSSS    000000    SSSSSSSS    RR      RR    CCCCCCCC  HH      HH      ....

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
```

RM
VC

```
0000             $BEGIN  RMSOSRCH,000,RM$RMS,<SEARCH FOR NEXT WILDCARD FILE>
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: rms32
0000   ;
0000   ; Abstract:
0000   ;       this is the highest level routine to perform the
0000   ;       $remove and $search functions
0000   ;
0000   ; Environment:
0000   ;       vax/vms
0000   ;
0000   ; Author:
0000   ;       tim halvorsen   AUG-1979
0000   ;
0000   ; Modified By:
0000   ;
0000   ;       V03-014 JEJ0026         J E Johnson             11-Apr-1984
0000   ;               Tie off invalid network operations.
0000   ;
0000   ;       V03-013 DGB0022         Donald G. Blair         06-Mar-1984
0000   ;               Use full-length FIB to support access mode protected
0000   ;               files.  Also change RMS$RETDIRBDB to RETDIRBDB, a local
0000   ;               routine.
0000   ;
0000   ;       V03-012 RAS0219         Ron Schaefer             8-Dec-1983
0000   ;               Change references to FWA$T_SWB subfield to separate structure.
0000   ;
0000   ;       V03-011 RAS0201         Ron Schaefer            17-Oct-1983
0000   ;               Correct calls to RMS$PARSE_FILE to account for the fact
0000   ;               that it does NOT necessarily preserve R7.
0000   ;               Make sure we got a name, type and/or version from ESA/ESL.
0000   ;
0000   ;       V03-010 KBT0585         Keith B. Thompson       12-Aug-1983
0000   ;               Cleanup fwa constants
0000   ;
0000   ;       V03-009 KBT0559         Keith B. Thompson       20-Jul-1983
0000   ;               Convert DNF and FNF errors into NMF after a sucessful
0000   ;               search list operation
0000   ;
0000   ;       V03-008 KBT0533         Keith B. Thompson       1-Jun-1983
0000   ;               Turn on search list processing and remove ref. to
0000   ;               RMS$SKIP_SUBTREE (this was a JSB to a SSB!)
0000   ;
0000   ;       V03-007 RAS0122         Ron Schaefer             1-Feb-1983
0000   ;               Complete KBT0472 by correcting a problem that would
0000   ;               leave an IFAB marked busy if the saved NAM block IFI
0000   ;               was incorrect.
0000   ;
0000   ;       V03-006 KBT0472         Keith B. Thompson       24-Jan-1983
0000   ;               Fix some code i don't understand
0000   ;
0000   ;       V03-005 RAS0103         Ron Schaefer            19-Nov-1982
0000   ;               Correct saving of the caller's access mode so that
0000   ;               exits via RMS$EX_NOSTR have the caller's mode in R7;
0000   ;               and correct DMW4004 to correctly save the mode in the IFB.
0000   ;
0000   ;       V03-004 DMW4004         DMWalp                   2-Sep-1982
```

```
0000    ;               Added code so that RMS$FABCHK was not called twice;
0000    ;               it was called directly and then again in RM$FSETI
0000    ;
0000    ;       V03-003 KBT0194         Keith B. Thompson       23-Aug-1982
0000    ;               Reorganize psects
0000    ;
0000    ;       V03-002 KRM0058         K Malik                 10-Aug-1982
0000    ;               Changed FWA$B_UNDERLINE symbol to FWA$B_UNDER_DEV
0000    ;               (to distinguish it from new FWA$B_UNDER_NOD symbol).
0000    ;
0000    ;--
0000    ;
0000
```

```
0000            .SBTTL  DEFINITIONS
0000
0000    ;
0000    ; symbol definitions
0000    ;
0000
0000            $IODEF
0000            $RMSDEF
0000            $SSDEF
0000            $BDBDEF
0000            $DEVDEF
0000            $FABDEF
0000            $FIBDEF
0000            $FWADEF
0000            $IFBDEF
0000            $IMPDEF
0000            $NAMDEF
0000            $FSCBDEF
0000            $SWBDEF
```

```
                      0000          SBTTL  RMS$SEARCH, Search for next Filename in Sequence
                      0000
                      0000    ;++
                      0000    ;
                      0000    ; RMS$SEARCH
                      0000    ;
                      0000    ;       Search for next filename in sequence
                      0000    ;
                      0000    ; RMS$REMOVE
                      0000    ;
                      0000    ;       Remove a directory entry
                      0000    ;
                      0000    ; inputs:
                      0000    ;
                      0000    ;       ap = address of user argument list
                      0000    ;       wcc of nam block contains ifi of wildcard ifab
                      0000    ;
                      0000    ; outputs:
                      0000    ;
                      0000    ;       result name string is returned to user buffer
                      0000    ;       fid/did in nam block
                      0000    ;
                      0000    ;--
                      0000
                      0000          $ENTRY  RMS$REMOVE
      7E    35    9A  0000          MOVZBL  #IO$_DELETE,-(SP)       ; set acp function code = remove
            03    11  0003          BRB     COMMON
                      0005
                      0005          $ENTRY  RMS$SEARCH
      7E    32    9A  0005          MOVZBL  #IO$_ACCESS,-(SP)       ; set acp function code = search
                      0008                                          ; this cannot be popped until ret
                      0008                                          ; since rm$fset saves the sp for stall
                      0008    COMMON: $TSTPT  SEARCH
                      000E
                      000E    ;
                      000E    ; Get ifab and fwa addresses from ifi which resides in wcc
                      000E    ; of nam block.
                      000E    ;
                      000E
         FFEF'    30  000E          BSBW    RM$FABCHK               ; check fab validity returns only if ok
                      0011                                          ; r11 = impure area
                      0011                                          ; r8 = fab address
                      0011                                          ; r7 = caller's access mode
            07    13  0011          BEQL    10$                     ; check IFI
                      0013          RMSERR  IFI                     ; error if IFI non-zero
            24    11  0018          BRB     20$
                      001A
            57    DD  001A    10$:  PUSHL   R7                      ; save caller's mode
   57    28 A8    D0  001C          MOVL    FAB$L_NAM(R8),R7        ; get nam address
         FFDD'    30  0020          BSBW    RM$CHKNAM               ; check nam validity
      56    57    D0  0023          MOVL    R7,R6                   ; copy nam addr
      57    8E    D0  0026          MOVL    (SP)+,R7                ; restore caller's mode
         12 50    E9  0029          BLBC    R0,20$                  ; if error
                      002C                                          ;  take the 'nostruct' error exit
         32 A6    B3  002C          BITW    NAM$L_WCC+2(R6),-       ; check to see that no spurious bits
      3FFE 8F         002F                  #^C<<NAM$M_SVCTX!-      ; other than the IFI bit, the search
                      0032                      NAM$M_SRCHNMF-      ; NMF bit, or the save context bit are
```

```
                 0032                          a-16>!1>             ; set within the field NAM$L_WCC
         66  12  0032              BNEQ    ERRWCC                   ; error if illegal wcc value
         1E  E1  0034              BBC     #NAM$V_SRCHNMF,-         ; if NMF has been encountered,
   07 30 A6      0036                      NAM$L_WCC(R6),30$        ; then go immediately return
                 0039              RMSERR  NMF                      ; a status of NMF
         5F  11  003E      20$:    BRB     ENS
                 0040
 59   30 A6  3C  0040      30$:    MOVZWL  NAM$L_WCC(R6),R9         ; get ifi of previous ifab
         1E  13  0044              BEQL    50$                      ; branch if none
                 0046
         10  E1  0046              BBC     #NAM$V_IFI,-             ; if the IFI bit is not set then
   19 30 A6      0048                      NAM$L_WCC(R6),50$        ;  context has not been saved
 02 A8   59  B0  004B              MOVW    R9,FAB$W_IFI(R8)         ; set internal ifi into fab
      FFAE'  30  004F              BSBW    RM$FSET_ALT1             ; setup with ifi in fab
 14 AB   04  C0  0052              ADDL2   #4,IMP$L_SAVED_SP(R11)   ; adjust FSET saved sp for acp code
 06 69   39  E1  0056              BBC     #IFB$V_SEARCH,(R9),40$   ; branch if not our type of ifab
 5A   38 A9  D0  005A              MOVL    IFB$L_FWA_PTR(R9),R10    ; get fwa
         1B  12  005E              BNEQ    SRCH                     ; branch if have one
                 0060      40$:    CSB     #IFB$V_BUSY,(R9)         ; don't leave this IFAB marked busy
                 0064
                 0064      ;
                 0064      ; No previous context can be found, parse the expanded name
                 0064      ; string and proceed.
                 0064      ;
                 0064
      FF99'  30  0064      50$:    BSBW    RM$FSETI_ALT             ; allocate ifab/ifi
 14 AB   04  C0  0067              ADDL2   #4,IMP$L_SAVED_SP(R11)   ; adjust FSET saved sp for acp code
 57   28 A8  D0  006B              MOVL    FAB$L_NAM(R8),R7         ; get nam address
      FF8E'  30  006F              BSBW    RM$CHKNAM                ; check nam validity
      22 50  E9  0072              BLBC    R0,EXIT1                 ; quit on failure
      FF88'  30  0075              BSBW    RM$RECOVER_FWA           ; recover fwa context
      1C 50  E9  0078              BLBC    R0,EXIT1                 ; branch if error
                 007B
```

```
                        007B   210 ;
                        007B   211 ; Context has been recovered.  Check device characteristics. continue only
                        007B   212 ; for directory structured devices.
                        007B   213 ;
                        007B   214
 23 6A    19    E0      007B   215 SRCH:   BBS     #FWA$V_NODE,(R10),NTSRCH; branch if network operation
          03    E1      007F   216         BBC     #DEV$V_DIR,-            ; error if illegal device
       0F 69            0081   217                 IFB$L_PRIM_DEV(R9),ERRIOP
          06    E0      0083   218         BBS     #DEV$V_SPL,-            ; error if spooled device
 09 008C C9             0085   219                 IFB$L_AS_DEV(R9),ERRIOP
                        0089   220
                        0089   221 ;
                        0089   222 ; Get the next file in sequence
                        0089   223 ;
                        0089   224
    01FE CA    B5       0089   225         TSTW    FWA$T_FIBBUF+FIB$W_DID(R10) ; new directory needed?
          47    12      008D   226         BNEQ    READ_DIR                ; branch if not
        016F    31      008F   227         BRW     NEXT_DIR                ; and get next directory
                        0092   228
                        0092   229 ERRIOP: RMSERR  IOP                     ; illegal device type
        025B    31      0097   230 EXIT1:  BRW     EXIT                    ; exit cleaning up ifab
                        009A   231
                        009A   232 ERRWCC: RMSERR  WCC                     ; illegal wcc value
       FF5E'    31      009F   233 ENS:    BRW     RM$EX_NOSTR             ; exit without ifab with status
                        00A2   234
                        00A2   235 ;
                        00A2   236 ; Perform network search function.
                        00A2   237 ;
                        00A2   238
 EC 69    3F    E0      00A2   239 NTSRCH: BBS     #IFB$V_NSP,(R9),ERRIOP  ; search of node::"task=abc" is invalid
                        00A6   240
    6E    35    91      00A6   241         CMPB    #IO$_DELETE,(SP)        ; Is this a search or a remove op?
          05    12      00A9   242         BNEQ    5$                      ; Branch if a search operation
       FF52'    30      00AB   243         BSBW    NT$REMOVE               ; its a remove...
          E7    11      00AE   244         BRB     EXIT1                   ; branch aid to home
                        00B0   245
          50    D4      00B0   246 5$:     CLRL    R0                      ; clear first-time-thru flag
 06 69    25    E0      00B2   247         BBS     #IFB$V_ACCESSED,(R9),10$ ; branch if already connected to fal
       FF47'    30      00B6   248         BSBW    NT$ACCESS               ; establish logical link with fal
     0D 50      E9      00B9   249         BLBC    R0,20$                  ; branch on failure
                        00BC   250                                         ; note, first-time-thru flag is now set!
       FF41'    30      00BC   251 10$:    BSBW    NT$SEARCH               ; perform search at remote node
     11 50      E9      00BF   252         BLBC    R0,30$                  ; branch on failure
                        00C2   253         SSB     #IFB$V_FILEFOUND,(R9)   ; indicate at least one file found
        01C0    31      00C6   254         BRW     COPY_RESULT             ; branch aid
                        00C9   255 20$:    RMSERR  FND,R1                  ; set default error code
       FF2F'    30      00CE   256         BSBW    RM$MAPERR               ; map ss error to rms error if possible
          C4    11      00D1   257         BRB     EXIT1                   ; branch aid
        0147    31      00D3   258 30$:    BRW     ERROR                   ; branch aid
                        00D6   259
                        00D6   260 ;
                        00D6   261 ; If we are saving context (ifab/fwa) and we are searching a wildcard
                        00D6   262 ; specification and no directory file has been read yet, then read the
                        00D6   263 ; directory file into memory to optimize on obtaining file names.
                        00D6   264 ;
                        00D6   265
                        00D6   266 READ_DIR:
```

```
     58   24 A9   D0  00D6   267            MOVL    IFB$L_LAST_FAB(R9),R8    ; get fab address
        69   39   E1  00DA   268            BBC     #IFB$V_SEARCH,(R9),-     ; branch if not saving context
             16       00DD   269                    NEXT_FILE
        6A   18   E1  00DE   270            BBC     #FWA$V_WILDCARD,(R10),- ; branch if non-wild string
             12       00E1   271                    NEXT_FILE
          30 AA   D5  00E2   272            TSTL    FWA$L_DIRBDB(R10)        ; directory file read yet?
             0D   12  00E5   273            BNEQ    NEXT_FILE                ; if so, don't read again
   00000000'EF    16  00E7   274            JSB     RM$READDIR               ; read directory into memory
          04 50   E9  00ED   275            BLBC    R0,NEXT_FILE             ; branch if unable to read
     30 AA   57   D0  00F0   276            MOVL    R7,FWA$L_DIRBDB(R10)     ; save bdb address
                      00F4   277
                      00F4   278 ;
                      00F4   279 ; Get the file name pattern from the expanded name string
                      00F4   280 ;
                      00F4   281
                      00F4   282 NEXT_FILE:
     57   28 A8   D0  00F4   283            MOVL    FAB$L_NAM(R8),R7         ; and recover nam address again
          FF05'   30  00F8   284            BSBW    RM$CHKNAM                ; check nam validity
          6A 50   E9  00FB   285            BLBC    R0,EXIT2                 ; quit on failure
     56   0B A7   9A  00FE   286            MOVZBL  NAM$B_ESL(R7),R6         ; length of expanded string
             5F   13  0102   287            BEQL    ERRESL                   ; error if none
     57   0C A7   D0  0104   288            MOVL    NAM$L_ESA(R7),R7         ; address of expanded string
                      0108   289            IFNORD  R6,(R7),ERRESA           ; error if cannot read buffer
   52   0104 8F   3C  010E   290            MOVZWL  #FSCB$C_BLN,R2           ; get size of FSCB
          FEEA'   30  0113   291            BSBW    RM$GETSPC1               ; allocate it
          4F 50   E9  0116   292            BLBC    R0,EXIT2                 ; exit on error
             5B   DD  0119   293            PUSHL   R11                      ; save impure area
        5B   51   D0  011B   294            MOVL    R1,R11                   ; put FSCB in correct reg
   00000000'EF    16  011E   295            JSB     RM$SCAN_STRING           ; scan the string
     50   2C AB   7D  0124   296            MOVQ    FSCB$Q_NAME(R11),R0      ; get name
             0C   12  0128   297            BNEQ    10$                      ; got one
     50   34 AB   7D  012A   298            MOVQ    FSCB$Q_TYPE(R11),R0      ; how about type
             0A   12  012E   299            BNEQ    20$                      ; got one
     50   3C AB   7D  0130   300            MOVQ    FSCB$Q_VERSION(R11),R0   ; try version
             08   11  0134   301            BRB     30$                      ; exit
     50   34 AB   A0  0136   302 10$:       ADDW2   FSCB$Q_TYPE(R11),R0      ; add type
     50   3C AB   A0  013A   303 20$:       ADDW2   FSCB$Q_VERSION(R11),R0   ; add version
   0188 CA   50   B0  013E   304 30$.       MOVW    R0,FWA$Q_RNS(R10)        ; set string descriptor length (no flags)
   018C CA   51   D0  0143   305            MOVL    R1,FWA$Q_RNS+4(R10)      ;  and address
        54   5B   D0  0148   306            MOVL    R11,R4                   ; get ready to return
          5B 8ED0     014B   307            POPL    R11                      ; restore impure area
   52   0104 8F   3C  014E   308            MOVZWL  #FSCB$C_BLN,R2           ;
          FEAA'   30  0153   309            BSBW    RM$RETSPC1               ; return FSCB
        0188 CA   B5  0156   310            TSTW    FWA$Q_RNS(R10)           ; valid string for ACP?
             0F   12  015A   311            BNEQ    SETFIB
                      015C   312
                      015C   313 ERRESA:  RMSERR  ESA                      ; set esa error
             05   11  0161   314            BRB     EXIT2
                      0163   315
                      0163   316 ERRESL:  RMSERR  ESL                      ; set esl error
           018A   31  0168   317 EXIT2:   BRW     EXIT
                      016B   318
                      016B   319 .ENABL  LSB
                      016B   320
                      016B   321 ;
                      016B   322 ; Setup fib fields
                      016B   323 ;
```

```
                                016B
              51   01F4 CA   9E 016B   SETFIB: MOVAB   FWA$T_FIBBUF(R10),R1                 ; fib address
           14 A1   0100 8F   B0 0170           MOVW    #FIB$M_WILD,FIB$W_NMCTL(R1)          ; set wildcarding on
    10 AA   00000040 8F      D0 0176           MOVL    #FIB$C_LENGTH,FWA$Q_FIB(R10)         ; create fib descriptor
            14 AA    51      D0 017E           MOVL    R1,FWA$Q_FIB+4(R10)
                             3C 0182           MOVZWL  #FWA$S_NAMEBUF+-
                                0183                   FWA$S_TYPEBUF+FWA$S_VERBUF,-
      0170 CA    012E 8F        0183                   FWA$Q_NAME(R10)                      ; set length of result buffer
                                0189
                                0189   ;
                                0189   ; If remove and the nam fop bit is set, set fib bit to do
                                0189   ; find via fid rather than by name.
                                0189   ;
                                0189
                 35    6E    91 0189           CMPB    (SP),#IO$_DELETE                     ; remove function?
                       0F    12 018C           BNEQ    20$                                  ; branch if not
           0A 04 A8    18    E1 018E           BBC     #FAB$V_NAM,FAB$L_FOP(R8),20$         ; branch if nam bit not set
                 0A A1       D5 0193           TSTL    FIB$W_DID(R1)                        ; fid supplied?
                       05    13 0196           BEQL    20$                                  ; branch if not
                                0198           SSB     #FIB$V_FINDFID,FIB$W_NMCTL(R1)       ; find by fid
                                019D
                                019D   ;
                                019D   ; If the directory file has already been read into virtual
                                019D   ; memory, then skip the call to the acp and look in memory
                                019D   ; for the next file name in sequence.
                                019D   ;
                                019D
                 32    6E    91 019D   20$:    CMPB    (SP),#IO$_ACCESS                     ; access function?
                       13    12 01A0           BNEQ    22$                                  ; only on searches
              57    30 AA    D0 01A2           MOVL    FWA$L_DIRBDB(R10),R7                 ; is there a directory in memory?
                       0D    13 01A6           BEQL    22$                                  ; call acp if not
              52   0188 CA   7D 01A8           MOVQ    FWA$Q_RNS(R10),R2                    ; pass descriptor of file name
               00000000'EF   16 01AD           JSB     RM$DIRSCAN                           ; find the next find in sequence
                       13    11 01B3           BRB     24$                                  ; re-join after acp call
                                01B5
                                01B5   ;
                                01B5   ; Call acp for next file in this directory
                                01B5   ;
                                01B5
                 50    6E    D0 01B5   22$:    MOVL    (SP),R0                              ; get acp function code
                       7E    7C 01B8           CLRQ    -(SP)                                ; p5/p6 = 0
                  0170 CA    9F 01BA           PUSHAB  FWA$Q_NAME(R10)                      ; p4 = result descriptor
                                01BE                                                        ; also input to acp as previous
                                01BE                                                        ; position (file) in directory
                    6C A9    9F 01BE           PUSHAB  IFB$L_RNS_LEN(R9)                    ; p3 = longword to receive length
                                01C1                                                        ; also input to acp as previous
                                01C1                                                        ; position (file) in directory
                  0188 CA    9F 01C1           PUSHAB  FWA$Q_RNS(R10)                       ; p2 = name descriptor
                     FE38'   30 01C5           BSBW    RM$FCPFNC                            ; call acp and wait for reply
                    07 50    E9 01C8   24$:    BLBC    R0,ACPERR                            ; branch if error from acp
                                01CB           SSB     #IFB$V_FILEFOUND,(R9)                ; indicate at least one file found
                     00B7    31 01CF           BRW     COPY_RESULT                          ; and copy result string
                                01D2
            21 6A      1C    E1 01D2   ACPERR: BBC     #FWA$V_WILD_DIR,(R10),25$            ; if there are no wild directories
                                01D6                                                        ; report fnf if none were
           0910 8F     50    B1 01D6           CMPW    R0,#SS$_NOSUCHFILE                   ; no files in directory at all?
                       24    13 01DB           BEQL    NEXT_DIR                             ; if so, get next directory
```

```
   0930 8F    50   B1  01DD          CMPW    R0,#SS$_NOMOREFILES             ; no more files in directory?
              1D   13  01E2          BEQL    NEXT_DIR                        ; if so, get next directory
        01FE CA    B4  01E4          CLRW    FWA$T_FIBBUF+FIB$W_DID(R10)     ; mark fresh directory needed
     51    4C AA   D0  01E8          MOVL    FWA$L_SWB_PTR(R10),R1           ; get SWB ptr
                       01EC          SSB     #SWB$V_TRAVERSE,-               ; set to skip rest of subtree
                       01EC                  SWB$B_FLAGS(R1)
   0828 8F    50   B1  01F0          CMPW    R0,#SS$_BADIRECTORY             ; bad directory format?
              0A   13  01F5          BEQL    NEXT_DIR                        ; ignore bad directories on traversa
                       01F7  25$:    RMSERR  FND,R1                          ; set default error
           FE01'   30  01FC          BSBW    RM$MAPERR                       ; map error to rms error
              1C   11  01FF          BRB     ERROR                           ; process other type of error
                       0201
                       0201  ;
                       0201  ; If no more files in directory, skip to next directory
                       0201  ;
                       0201
                       0201  NEXT_DIR:
            0168   30  0201          BSBW    RETDIRBDB                       ; deallocate directory buffer
        69    04   E0  0204          BBS     #DEV$V_SDI,IFB$L_PRIM_DEV(R9),- ; nmf if sdi device
              10       0207                  ERRNMF
     00000000'EF   16  0208          JSB     RM$NEXTDIR                      ; get next directory
           0C 50   E9  020E          BLBC    R0,ERROR                        ; if error, copy result and exit
         0204 CA   D4  0211          CLRL    FWA$T_FIBBUF+FIB$L_WCC(R10)     ; start at 1st file in directory
            FEBE   31  0215          BRW     READ_DIR                        ;  and then get next file
                       0218
                       0218  ERRNMF: RMSERR  NMF                             ; no more files
                       021D
                       021D  ;
                       021D  ; If there is no wild card directory and the user did not specify NAM$V_SVCTX,
                       021D  ; then the ACP is maintaining context and we should just return the error
                       021D  ; it gave us.
                       021D  ;
                       021D  ; If we are maintaining context (wild directory or NAM$V_SVCTX), then we should
                       021D  ; convert NMF to FNF based on FILEFOUND bit.
                       021D  ;
                       021D
    4F 69     39   E1  021D  ERROR:  BBC     #IFB$V_SEARCH,(R9),35$          ; we are not keeping context
    4B 69     3C   E0  0221          BBS     #IFB$V_FILEFOUND,(R9),35$       ; skip if file found
  82CA 8F     50   B1  0225          CMPW    R0,#RMS$_NMF&^XFFFF             ; and error was NMF
              44   12  022A          BNEQ    35$
                       022C
                       022C  ;
                       022C  ; If there was a wild directory, move the expanded name string from
                       022C  ; the namblk to the resultant name string and return file not found
                       022C  ;
                       022C
    40 6A     1C   E1  022C          BBC     #FWA$V_WILD_DIR,(R10),35$       ; branch if dir not wild
    58    24 A9    D0  0230          MOVL    IFB$L_LAST_FAB(R9),R8           ; get the last fab's addr
    57    28 A8    D0  0234          MOVL    FAB$L_NAM(R8),R7                ; get the name block addr
           FDC5'   30  0238          BSBW    RM$CHKNAM                       ; check nam validity
           69 50   E9  023B          BLBC    R0,44$                          ; quit on failure
           03 A7   94  023E          CLRB    NAM$B_RSL(R7)                   ; assume can't set result string
    52    02 A7    9A  0241          MOVZBL  NAM$B_RSS(R7),R2                ; get length of resultant buffer
    53    04 B7    DE  0245          MOVAL   @NAM$L_RSA(R7),R3               ; get addr of resultant buffer
                       0249          IFNOWRT R2,(R3),50$                     ; probe the resultant string buff
                       024F                                                  ;  error if can't write it
    52    0A A7    9A  024F          MOVZBL  NAM$B_ESS(R7),R2                ; get the buffer size into longword
```

```
        51    0C B7   DE  0253   438          MOVAL   @NAM$L_ESA(R7),R1              ; get addr of expanded buffer
                          0257   439          IFNORD  R2,(R1),50$                    ; probe the expanded string buff
                          025D   440                                                 ;  error if can't read it
        52    0B A7   9A  025D   441          MOVZBL  NAM$B_ESL(R7),R2               ; get the string's actual length
        03 A7    52   90  0261   442          MOVB    R2,NAM$B_RSL(R7)               ; stuff the resultant length
     63   61     52   28  0265   443          MOVC3   R2,(R1),(R3)                   ; move the expanded string
                          0269   444                                                 ;  to the resultant string
                          0269   445          RMSERR  FNF                            ; restore the error
              39     11   026E   446          BRB     50$                            ;  and continue
                          0270   447
                          0270   448 ;
                          0270   449 ; Error has occurred - setup file name so that when result
                          0270   450 ; name string is copied, the file string sent to acp is returned.
                          0270   451 ;
                          0270   452
        15 6A    19   E0  0270   453 35$:     BBS     #FWA$V_NODE,(R10),COPY_RESULT  ; branch if network operation
              50     DD   0274   454          PUSHL   R0                             ; save status code
    6C A9   0188 CA   3C  0276   455          MOVZWL  FWA$Q_RNS(R10),IFB$L_RNS_LEN(R9); set length of string
  018C DA   0188 CA   28  027C   456          MOVC3   FWA$Q_RNS(R10),@FWA$Q_RNS+4(R10),-
            04B6 CA       0283   457                  FWA$T_NAMEBUF(R10)
              50  8ED0    0286   458          POPL    R0                             ; restore status
                          0289   459
                          0289   460 ;
                          0289   461 ; Copy result file name to user result buffer
                          0289   462 ; unless no file was found
                          0289   463 ;
                          0289   464
                          0289   465 COPY_RESULT:
              50     DD   0289   466          PUSHL   R0                             ; save status code
        58    24 A9   D0  028B   467          MOVL    IFB$L_LAST_FAB(R9),R8          ; get fab address
        57    28 A8   D0  028F   468          MOVL    FAB$L_NAM(R8),R7               ; set nam address
             FD6A'   30   0293   469          BSBW    RM$CHKNAM                      ; check nam validity
           0B 50     E9   0296   470          BLBC    R0,42$                         ; quit on failure
            0092     30   0299   471          BSBW    RM$COPY_RESULT                 ; copy result name string
           05 50     E9   029C   472          BLBC    R0,42$                         ; branc^ if error
              50  8ED0    029F   473          POPL    R0                             ; restore status code
              05     11   02A2   474          BRB     50$                            ;  and continue
          5E  04     C0   02A4   475 42$:     ADDL    #4,SP                          ; ignore saved status code
              4C     11   02A7   476 44$:     BRB     EXIT                           ;  and report one from copy_result
                          02A9   477
                          02A9   478 ;
                          02A9   479 ; If not remove, copy fid and did into nam block
                          02A9   480 ;
                          02A9   481
          35  6E     91   02A9   482 50$:     CMPB    (SP),#IO$_DELETE               ; remove function?
              10     13   02AC   483          BEQL    60$                            ; if so, skip this
        0C 6A    19   E0  02AE   484          BBS     #FWA$V_NODE,(R10),60$          ; skip also if network operation
                          02B2   485
                          02B2   486          ASSUME  FIB$W_DID       EQ      FIB$W_FID+6
                          02B2   487          ASSUME  NAM$W_DID       EQ      NAM$W_FID+6
                          02B2   488
    24 A7   01F8 CA   7D  02B2   489          MOVQ    FWA$T_FIBBUF+FIB$W_FID(R10),NAM$W_FID(R7)
    2C A7   0200 CA   D0  02B8   490          MOVL    FWA$T_FIBBUF+FIB$W_FID+8(R10),NAM$W_FID+8(R7)
                          02BE   491
                          02BE   492 ;
                          02BE   493 ; If this is a temporary ifab/fwa created for this call
                          02BE   494 ; only, then save the current acp position in the directory
```

```
                                  02BE    495 ; file and cleanup all internal structures.
                                  02BE    496 ;
                                  02BE    497
           0B 69   39     E0      02BE    498 60$:      BBS     #IFB$V_SEARCH,(R9),65$              ; branch if ifab to be saved
              26 50       E9      02C2    499           BLBC    R0,NMF                              ; go set NMF bit if any error
         0204 CA          3C      02C5    500           MOVZWL  FWA$T_FIBBUF+FIB$L_WCC(R10),-       ; save acp position
              30 A7               02C9    501                   NAM$L_WCC(R7)
                 28       11      02CB    502           BRB     EXIT
                                  02CD    503
                                  02CD    504 ;
                                  02CD    505 ; This is a permanent ifab/fwa (that is, it is kept around between
                                  02CD    506 ; calls in order to speed up things or keep extended context)
                                  02CD    507 ; If the status was successful or not enough privilege,
                                  02CD    508 ; then keep the wildcard sequence context around so that
                                  02CD    509 ; search can be called again.  else, cleanup everything.
                                  02CD    510 ;
                                  02CD    511
              17 50       E8      02CD    512 65$:      BLBS    R0,70$                              ; continue sequence if successful
      82CA 8F     50      B1      02D0    513           CMPW    R0,#RMS$_NMF&^XFFFF                 ; done with wildcard sequence?
                  21      13      02D5    514           BEQL    CHKLST                              ; if so, terminate sequence
      8292 8F     50      B1      02D7    515           CMPW    R0,#RMS$_FNF&^XFFFF                 ; file not found?
                  1A      13      02DC    516           BEQL    CHKLST                              ; if so, terminate sequence
         0B 6A    1C      E1      02DE    517           BBC     #FWA$V_WILD_DIR,(R10),NMF           ; if nonwild, cleanup
            0C A8         D5      02E2    518           TSTL    FAB$L_STV(R8)                       ; error from acp?
               06         13      02E5    519           BEQL    NMF                                 ; if not, terminate sequence
            02 A8         B4      02E7    520 70$:      CLRW    FAB$W_IFI(R8)                       ; mbz for subsequent operations on f
            FD13'         31      02EA    521           BRW     RMS$EXRMS                           ; exit without cleaning up
                                  02ED    522
30 A7   40000000 8F       D0      02ED    523 NMF:      MOVL    #NAM$M_SRCHNMF,NAM$L_WCC(R7)        ; indicate that another search isn't
                                  02F5    524                                                       ; to be done with this NAM
            FD08'         31      02F5    525 EXIT:     BRW     RMS$CLSCU                           ; cleanup ifab and buffers
                                  02F8    526
                                  02F8    527 .DSABL LSB
                                  02F8    528
                                  02F8    529 ;
                                  02F8    530 ; We are about to exit with No More Files or File Not Found, before we
                                  02F8    531 ; really do, check to see if there was a search list, if so try to
                                  02F8    532 ; parse a new string and if successful search for a new file
                                  02F8    533 ;
                                  02F8    534
         F1 6A    38      E1      02F8    535 CHKLST:   BBC     #FWA$V_SLPRESENT,(R10),NMF          ; are search list present?
                                  02FC    536           SSB     #FWA$V_SL_PASS,(R10)                ; indicate search list parse
            30 A7         DD      0300    537           PUSHL   NAM$L_WCC(R7)                       ; save wild card context
               57         DD      0303    538           PUSHL   R7                                  ; save NAM blk ptr
            FCF8'         30      0305    539           BSBW    RMS$PARSE_FILE                      ; parse a new string
               57       8ED0      0308    540           POPL    R7                                  ; restore NAM blk ptr
            30 A7       8ED0      030B    541           POPL    NAM$L_WCC(R7)                       ; restore wcc
            03 50         E9      030F    542           BLBC    R0,10$                              ; branch if error
             FD66         31      0312    543           BRW     SRCH                                ; go search new string
                                  0315    544
                                  0315    545 ;
                                  0315    546 ; If there was a file found on some previous search operation then
                                  0315    547 ; convert DNF and FNF errors into NMF
                                  0315    548 ;
                                  0315    549
         D4 69    3C      E1      0315    550 10$:      BBC     #IFB$V_FILEFOUND,(R9),NMF           ; no previous file found
      C04A 8F     50      B1      0319    551           CMPW    R0,#RMS$_DNF&^XFFFF                 ; directory not found
```

```
                   07   13  031E   552          BEQL    20$                                    ; yes, convert it
         8292 8F   50   B1  0320   553          CMPW    R0,#RMS$_FNF&^XFFFF                    ; file not found
                   C6   12  0325   554          BNEQ    NMF                                    ; no exit
                            0327   555 20$:     RMSERR  NMF                                    ; convert the error
                   BF   11  032C   556          BRB     NMF                                    ; exit
                            032E   557
                            032E          .SBTTL  RMS$COPY_RESULT, Return Result Name String
                            032E
                            032E  ;++
                            032E  ;
                            032E  ; RMS$COPY_RESULT
                            032E  ;
                            032E  ;       Construct the result name string and return to
                            032E  ;       the caller via the rsa and rss fields of the nam.
                            032E  ;
                            032E  ; inputs:
                            032E  ;
                            032E  ;       r7              = address of NAM
                            032E  ;       r9              = address of ifab
                            032E  ;       r10             = address of fwa
                            032E  ;       ifb$l_rns_len   = length of new file name
                            032E  ;       fwa$t_namebuf   = new file name string
                            032E  ;       fwa$q_device    = descriptor of device name
                            032E  ;       fwa$q_dir1-8    = descriptors of directory names
                            032E  ;       fwa$b_dirlen    = number of directory levels
                            032E  ;       fwa$b_dirterm   = directory specification terminator
                            032E  ;
                            032E  ; outputs:
                            032E  ;
                            032E  ;       result string buffer is output if requested.
                            032E  ;       NAM$L_FNB
                            032E  ;
                            032E  ;--
                            032E
                            032E  RMS$COPY_RESULT::
       23 6A   19           032E          BBS     #FWA$V_NODE,(R10),5$            ; branch if network operation
           6C A9            0332          MOVL    IFB$L_RNS_LEN(R9),-             ; set length of file name
         0170 CA                                  FWA$Q_NAME(R10)
       19 6A   0E           0335          BBC     #FWA$V_DIR,(R10),5$             ; skip if no directory in spec
       15 6A   1C      E1   033C          BBC     #FWA$V_WILD_DIR,(R10),5$        ;  or if there are no wild directori
 50    2E AA   01      83   0340          SUBB3   #1,FWA$B_DIRLEN(R10),R0         ; get number of subdirectory levels
          1D   50      F0   0345          INSV    R0,#FWA$V_DIR_LVLS,-            ; return current # of subdir.
          6A   03           0348                  #FWA$S_DIR_LVLS,(R10)           ;  levels in the FWA
          15   50      F0   034A          INSV    R0,#NAM$V_DIR_LVLS,-            ; return current # of subdir.
               03           034D                  #NAM$S_DIR_LVLS,-               ;  levels in the NAM
            34 A7           034E                  NAM$L_FNB(R7)
            05 AA      90   0350          MOVB    FWA$B_DIRWCFLGS(R10),-          ; if any ellipses were found,
            37 A7           0353                  NAM$L_FNB+3(R7)                 ;  set the appropriate wild
                            0355                                                  ;  flags in the nam blk
 5C       67'AF        9E   0355  5$:     MOVAB   B^10$,AP                        ; address of expstring arg list
            24 A7      DD   0359          PUSHL   NAM$W_FID(R7)                   ; save contents of nam fid
            24 A7      D4   035C          CLRL    NAM$W_FID(R7)                   ; clear fid so expstring will work
            FC9E'      30   035F          BSBW    RMS$EXPSTRING                   ; return result name string
            24 A7    8ED0   0362          POPL    NAM$W_FID(R7)                   ; restore contents of nam fid
                       05   0366          RSB
                            0367
                       04   0367  10$:    .BYTE   NAM$L_RSA                       ; offset to result buffer addr.
                            0368          RMSERR_WORD     RST                     ; error of bad buffer
                            036A          RMSERR_WORD     RSS                     ; error of buffer too short
```

```
                        036C            .SBTTL  RETDIRBDB, Deallocate Directory Buffer and BDB
                        036C
                        036C    ;++
                        036C    ;
                        036C    ; RETDIRBDB
                        036C    ;
                        036C    ;       This routine deallocates the directory buffer and the bdb
                        036C    ;       which is associated with it.
                        036C    ;
                        036C    ; inputs:
                        036C    ;
                        036C    ;       r10 = fwa address
                        036C    ;       r9 = ifab address
                        036C    ;       fwa$l_dirbdb = address of directory bdb
                        036C    ;
                        036C    ; outputs:
                        036C    ;
                        036C    ;       none
                        036C    ;--
                        036C    ;
                        036C
                        036C    RETDIRBDB:
54     30 AA    D0      036C            MOVL    FWA$L_DIRBDB(R10),R4    ; is there a directory in memory?
          0E    13      0370            BEQL    10$                     ; branch if not
          5A    DD      0372            PUSHL   R10                     ; save r10
     5A   59    D0      0374            MOVL    R9,R10                  ; rm$retbdb wants ifb address in r10
       FC86'    30      0377            BSBW    RM$RETBDB               ; deallocate it if there is
          5A  8ED0      037A            POPL    R10                     ; restore r10
       30 AA    D4      037D            CLRL    FWA$L_DIRBDB(R10)       ; and clear pointer
                05      0380    10$:    RSB
                        0381
                        0381            .END
```

```
                                                                   L  2
RMS0SRCH                          SEARCH FOR NEXT WILDCARD FILE              16-SEP-1984 01:32:07  VAX/VMS Macro V04-00        Page  16
Symbol table                                                                  5-SEP-1984 16:25:32  [RMS.SRC]RMS0SRCH.MAR;1            (7)

$$.PSECT_EP                   = 00000000                 FWA$V_WILD_DIR                  = 0000001C
$$RMSTEST                     = 0000001A                 IFB$L_AS_DEV                    = 0000008C
$$RMS_PBUGCHK                 = 00000010                 IFB$L_FWA_PTR                   = 00000038
$$RMS_TBUGCHK                 = 00000008                 IFB$L_LAST_FAB                  = 00000024
$$RMS_UMODE                   = 00000004                 IFB$L_PRIM_DEV                  = 00000000
ACPERR                          000001D2 R     01        IFB$L_RNS_LEN                   = 0000006C
CHKLST                          000002F8 R     01        IFB$V_ACCESSED                  = 00000025
COMMON                          00000008 R     01        IFB$V_BUSY                      = 00000020
COPY_RESULT                     00000289 R     01        IFB$V_FILEFOUND                 = 0000003C
DEV$V_DIR                     = 00000003                 IFB$V_NSP                       = 0000003F
DEV$V_SDI                     = 00000004                 IFB$V_SEARCH                    = 00000039
DEV$V_SPL                     = 00000006                 IMP$L_SAVED_SP                  = 00000014
ENS                             0000009F R     01        IO$_ACCESS                      = 00000032
ERRESA                          0000015C R     01        IO$_DELETE                      = 00000035
ERRESL                          00000163 R     01        NAM$B_ESL                       = 0000000B
ERRIOP                          00000092 R     01        NAM$B_ESS                       = 0000000A
ERRNMF                          00000218 R     01        NAM$B_RSL                       = 00000003
ERROR                           0000021D R     01        NAM$B_RSS                       = 00000002
ERRWCC                          0000009A R     01        NAM$L_ESA                       = 0000000C
EXIT                            000002F5 R     01        NAM$L_FNB                       = 00000034
EXIT1                           00000097 R     01        NAM$L_RSA                       = 00000004
EXIT2                           00000168 R     01        NAM$L_WCC                       = 00000030
FAB$L_FOP                     = 00000004                 NAM$M_SRCHNMF                   = 40000000
FAB$L_NAM                     = 00000028                 NAM$M_SVCTX                     = 80000000
FAB$L_STV                     = 0000000C                 NAM$S_DIR_LVLS                  = 00000003
FAB$V_NAM                     = 00000018                 NAM$V_DIR_LVLS                  = 00000015
FAB$W_IFI                     = 00000002                 NAM$V_IFI                       = 00000010
FIB$C_LENGTH                  = 00000040                 NAM$V_SRCHNMF                   = 0000001E
FIB$L_WCC                     = 00000010                 NAM$W_DID                       = 0000002A
FIB$M_WILD                    = 00000100                 NAM$W_FID                       = 00000024
FIB$V_FINDFID                 = 0000000B                 NEXT_DIR                          00000201 R       01
FIB$W_DID                     = 0000000A                 NEXT_FILE                         000000F4 R       01
FIB$W_FID                     = 00000004                 NMF                               000002ED R       01
FIB$W_NMCTL                   = 00000014                 NT$ACCESS                         ********     X   01
FSCB$C_BLN                    = 00000104                 NT$REMOVE                         ********     X   01
FSCB$Q_NAME                   = 0000002C                 NT$SEARCH                         ********     X   01
FSCB$Q_TYPE                   = 00000034                 NTSRCH                            000000A2 R       01
FSCB$Q_VERSION                = 0000003C                 PIO$A_TRACE                       ********     X   01
FWA$B_DIRLEN                  = 0000002E                 READ_DIR                          000000D6 R       01
FWA$B_DIRWCFLGS               = 00000005                 RETDIRBDB                         0000036C R       01
FWA$L_DIRBDB                  = 00000030                 RM$CHKNAM                         ********     X   01
FWA$L_SWB_PTR                 = 0000004C                 RM$CLSCU                          ********     X   01
FWA$Q_FIB                     = 00000010                 RM$COPY_RESULT                    0000032E RG      01
FWA$Q_NAME                    = 00000170                 RM$DIRSCAN                        ********     X   01
FWA$Q_RNS                     = 00000188                 RM$EXPSTRING                      ********     X   01
FWA$S_DIR_LVLS                = 00000003                 RM$EXRMS                          ********     X   01
FWA$S_NAMEBUF                 = 00000100                 RM$EX_NOSTR                       ********     X   01
FWA$S_TYPEBUF                 = 00000028                 RM$FABCHK                         ********     X   01
FWA$S_VERBUF                  = 00000006                 RM$FCPFNC                         ********     X   01
FWA$T_FIBBUF                  = 000001F4                 RM$FSETI_ALT                      ********     X   01
FWA$T_NAMEBUF                 = 000004B6                 RM$FSET_ALT1                      ********     X   01
FWA$V_DIR                     = 0000000E                 RM$GETSPC1                        ********     X   01
FWA$V_DIR_LVLS                = 0000001D                 RM$MAPERR                         ********     X   01
FWA$V_NODE                    = 00000019                 RM$NEXTDIR                        ********     X   01
FWA$V_SLPRESENT               = 00000038                 RM$PARSE_FILE                     ********     X   01
FWA$V_SL_PASS                 = 00000002                 RM$READDIR                        ********     X   01
FWA$V_WILDCARD                = 00000018                 RM$RECOVER_FWA                    ********     X   01
```

```
RMS$RETBDB                            ********   X   01
RMS$RETSPC1                           ********   X   01
RMS$SCAN_STRING                       ********   X   01
RMS$$REMOVE                         = FFFFFFFE RG    01
RMS$$SEARCH                         = 00000003 RG    01
RMS$_DNF                            = 0001C04A
RMS$_ESA                            = 000184FC
RMS$_ESL                            = 00018714
RMS$_FND                            = 0001C02A
RMS$_FNF                            = 00018292
RMS$_IFI                            = 00018564
RMS$_IOP                            = 00018574
RMS$_NMF                            = 000182CA
RMS$_RSS                            = 00018694
RMS$_RST                            = 0001869C
RMS$_WCC                            = 000182EA
SETFIB                                0000016B R     01
SRCH                                  0000007B R     01
SS$_BADIRECTORY                     = 00000828
SS$_NOMOREFILES                     = 00000930
SS$_NOSUCHFILE                      = 00000910
SWB$B_FLAGS                         = 00000000
SWB$V_TRAVERSE                      = 00000004
TPT$L_SEARCH                          ********   X   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
RMSRMS                               00000381  (  897.)  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     00:00:00.12    00:00:00.72
Command processing                   109     00:00:00.70    00:00:04.02
Pass 1                               522     00:00:21.26    00:00:53.49
Symbol table sort                      0     00:00:03.48    00:00:05.19
Pass 2                               130     00:00:03.92    00:00:09.10
Symbol table output                   17     00:00:00.16    00:00:00.56
Psect synopsis output                  1     00:00:00.02    00:00:00.09
Cross-reference output                 0     00:00:00.00    00:00:00.00
Assembler run totals                 810     00:00:29.66    00:01:13.18

The working set limit was 1800 pages.
119924 bytes (235 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2357 non-local and 34 local symbols.
644 source lines were read in Pass 1, producing 15 object records in Pass 2.
32 pages of virtual memory were used to define 31 macros.
```

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

Macro library name                      Macros defined
------------------                      --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                    16
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                     3
_$255$DUA28:[SYSLIB]STARLET.MLB;2                  8
TOTALS (all libraries)                            27

2488 GETS were required to define 27 macros.

There were no errors, warnings or information messages.

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

0331 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RMS0TRUNC
LIS

STAPRFLNM
LIS

RPGCVTPTO
LIS

RPGHANDLE
LIS

RPGMOVE1
LIS

RPGLIB
REQ

RMSGBL
LIS

RPGRTL

RPGDSPLY
LIS

RPGRTL
MAP

RPGPROLOG
REQ

RPGEXTIND
LIS

RPGLIB
LIS

RMS0SRCH
LIS

RMS0WAIT
LIS

RPGBTZ
LIS

RPGMOVE2
LIS

RPGDIVIDE
LIS

RPGIOEXCE
LIS

RPGDEF
REQ

RMS0UPDAT
LIS

RPGERROR
LIS