```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      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  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM      MM    000000    NN      NN    AAAAAA    MM      MM    SSSSSSSS  TTTTTTTTTT  RRRRRRRR
RRRRRRRR    MM      MM    000000    NN      NN    AAAAAA    MM      MM    SSSSSSSS  TTTTTTTTTT  RRRRRRRR
RR      RR  MMMM  MMMM  00      00  NN      NN  AA      AA  MMMM  MMMM  SS            TT      RR      RR
RR      RR  MMMM  MMMM  00      00  NN      NN  AA      AA  MMMM  MMMM  SS            TT      RR      RR
RR      RR  MM  MM  MM  00      00  NNNN    NN  AA      AA  MM  MM  MM  SS            TT      RR      RR
RR      RR  MM  MM  MM  00      00  NNNN    NN  AA      AA  MM  MM  MM  SS            TT      RR      RR
RRRRRRRR    MM      MM  00  00  00  NN  NN  NN  AA      AA  MM      MM    SSSSSS      TT      RRRRRRRR
RRRRRRRR    MM      MM  00  00  00  NN  NN  NN  AA      AA  MM      MM    SSSSSS      TT      RRRRRRRR
RR  RR      MM      MM  0000    00  NN    NNNN  AAAAAAAAAA  MM      MM          SS    TT      RR  RR
RR  RR      MM      MM  0000    00  NN    NNNN  AAAAAAAAAA  MM      MM          SS    TT      RR  RR
RR    RR    MM      MM  00      00  NN      NN  AA      AA  MM      MM          SS    TT      RR    RR
RR    RR    MM      MM  00      00  NN      NN  AA      AA  MM      MM          SS    TT      RR    RR
RR      RR  MM      MM    000000    NN      NN  AA      AA  MM      MM  SSSSSSSS      TT      RR      RR
RR      RR  MM      MM    000000    NN      NN  AA      AA  MM      MM  SSSSSSSS      TT      RR      RR

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     1          $BEGIN  RMONAMSTR,000,RM$RMS0,<RETURN FILENAME STRINGS>
0000     2
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
```

```
0000     28 ;++
0000     29 ;
0000     30 ; Facility: RMS
0000     31 ;
0000     32 ; Abstract:
0000     33 ;
0000     34 ;       This module builds either an expanded name string or a resultant
0000     35 ;       name string (as requested on entry) and returns it to the user via
0000     36 ;       the Name Block.
0000     37 ;
0000     38 ; Environment: VAX/VMS, executive mode
0000     39 ;
0000     40 ; Author: Leo F. Laverdure,     Creation Date:  01-MAR-1977
0000     41 ;
0000     42 ; Modified By:
0000     43 ;
0000     44 ;       V03-019 JWT0192         Jim Teague              12-Aug-1984
0000     45 ;               Fix a terrible bug in filename-length checking:
0000     46 ;               RMS was doing unsigned, one-byte compares against
0000     47 ;               the users's ESS or RSS, and taking an error path
0000     48 ;               if the accumulated size was greater than that.
0000     49 ;               Fine, usually.  However, if the user specified 255
0000     50 ;               for ESS or RSS, how many one-byte values can YOU
0000     51 ;               think of that are greater than 'FF'?
0000     52 ;
0000     53 ;       V03-018 RAS0296         Ron Schaefer            18-Apr-1984
0000     54 ;               Return secondary device name in NAM$T_DVI if the
0000     55 ;               device is spooled.
0000     56 ;
0000     57 ;       V03-017 JEJ0029         J E Johnson             19-Apr-1984
0000     58 ;               Remove wildcard status storage for quoted network
0000     59 ;               files as that has been moved to NTOACCESS, also
0000     60 ;               replace reference to NAM$V_ROD with NAM$V_PWD as
0000     61 ;               someone actually used the function it provided.
0000     62 ;
0000     63 ;       V03-016 JEJ0027         J E Johnson             11-Apr-1984
0000     64 ;               Eliminate unused reference to NAM$V_ROD field.
0000     65 ;
0000     66 ;       V03-015 JEJ0006         J E Johnson             08-Mar-1984
0000     67 ;               Store the wildcard status on qucted network files.
0000     68 ;               Also slight code cleanup done.
0000     69 ;
0000     70 ;       V03-014 RAS0254         Ron Schaefer            16-Feb-1984
0000     71 ;               Return as many fields as possible in expanded/resultant
0000     72 ;               strings for PPFs.   Add $DEVDEF and $RMSDEF macros.
0000     73 ;               Cleanup use of ESA/RSA device name when doing open
0000     74 ;               by device-id/file-id as well as network FAL returns.
0000     75 ;
0000     76 ;       V03-013 RAS0223         Ron Schaefer            16-Dec-1983
0000     77 ;               Change $SCBDEF and SCB$xxx to $FSCBDEF and FSCB$xxx.
0000     78 ;
0000     79 ;       V03-012 RAS0203         Ron Schaefer            18-Oct-1983
0000     80 ;               Fix network node processing to only mask out the
0000     81 ;               password for the first node in the list.
0000     82 ;
0000     83 ;       V03-011 KRM0115         Karl Malik              27-Sep-1983
0000     84 ;               Add UPDATE_FNB routine to set appropriate FNB bits
```

```
0000     85 ;                  during a network $search (currently not returned
0000     86 ;                  by FAL).
0000     87 ;
0000     88 ;          V03-010 KBT0563         Keith B. Thompson       21-Jul-1983
0000     89 ;                  Change the syntax of a non-concealed name to:
0000     90 ;                  DEV:[ROOT.][DIRECTORY]
0000     91 ;
0000     92 ;          V03-009 KPL0001         Peter Lieberwirth       23-Jun-1983
0000     93 ;                  Modify RM$GETFILNAM to return the unconcealed (real)
0000     94 ;                  device name as well as the rest of the file spec.
0000     95 ;
0000     96 ;          V03-008 KBT0516         Keith B. Thompson       23-May-1983
0000     97 ;                  Remove the RM$CHKNAMBLK hack
0000     98 ;
0000     99 ;          V03-007 KBT0506         Keith B. Thompson       3-May-1983
0000    100 ;                  Change SCB$M_ACCS to SCB$M_ACS and add some concealed
0000    101 ;                  device/directory features?
0000    102 ;
0000    103 ;          V03-006 KBT0503         Keith B. Thompson       18-Apr-1983
0000    104 ;                  Use SCB flags to check to access control string in
0000    105 ;                  node descriptor
0000    106 ;
0000    107 ;          V03-005 LJA0064         Laurie J. Anderson      22-Feb-1983
0000    108 ;                  Move some routines related to the NAM block to this
0000    109 ;                  module, for consistancy sake and ease of locating them.
0000    110 ;                  They include RM$FILLNAM, RM$WRITE_DVI, RM$CHKNAMBLK
0000    111 ;                  and RM$CHKNAM.  Note, any audit trails corresponding to
0000    112 ;                  mods to these routines did NOT move with these modules.
0000    113 ;
0000    114 ;          V03-004 JWH0188         Jeffrey W. Horn         15-Feb-1983
0000    115 ;                  Add an additional entry point RM$GETFILNAM to return
0000    116 ;                  the resultant name string to a buffer for journaling.
0000    117 ;
0000    118 ;          V03-003 KBT0211         Keith B. Thompson       23-Aug-1982
0000    119 ;                  Reorganize psects
0000    120 ;
0000    121 ;          V03-002 KBT0099         Keith B. Thompson       13-Jul-1982
0000    122 ;                  Clean up psects
0000    123 ;
0000    124 ;          V03-001 JAK0073         J A Krycka              12-Apr-1982
0000    125 ;                  Eliminate the practice of prefixing a <del> character to an
0000    126 ;                  expanded or resultant name string to indicate that the password
0000    127 ;                  has been masked out of the nodespec.
0000    128 ;
0000    129 ;          V02-090 KEK018          K. E. Kinnear           17-Jan-1982
0000    130 ;                  Add modifications for ANSI-"a" magtape filespecs.
0000    131 ;
0000    132 ;          V02-089 KEK0019         K. E. Kinnear           17-Jan-1982
0000    133 ;                  Remove references to NAM$B_QUOTED and NAM$L_QUOTED and
0000    134 ;                  replace with references to NAM$x_NAME.
0000    135 ;
0000    136 ;          V02-088 JWH0001         Jeffrey W. Horn         16-Dec-1981
0000    137 ;                  If FWA$V_DIR bit is clear, return [] as directory
0000    138 ;                  specification.
0000    139 ;
0000    140 ;          V02-087 RAS0039         Ron Schaefer            25-Sep-1981
0000    141 ;                  Return the ESA device name string if the DVI device name
```

```
0000   142 ;               was used.  This hides the hidden device name for
0000   143 ;               users doing explicit $PARSE before $OPEN or $CREATE.
0000   144 ;
0000   145 ;       V02-086 JAK0062         J A Krycka              14-AUG-1981
0000   146 ;               Return remote file system type in NAM$B_RFS.
0000   147 ;
0000   148 ;       V02-085 JAK0059         J A Krycka              11-JUN-1981
0000   149 ;               Multiplex the QUOTED descriptor in the NAM block with the
0000   150 ;               NAME descriptor instead of the DEV descriptor.
0000   151 ;
0000   152 ;       V02-084 JAK0058         J A Krycka      04-JUN-1981
0000   153 ;               Continuation of V02-084.
0000   154 ;
0000   155 ;       V02-083 KRM0015         K R Malik       26-May-1981
0000   156 ;               Fix bug in RMSEXPSTRING which returned incorrect values
0000   157 ;               for the extended NAM block TYPE and VER fields.
0000   158 ;
0000   159 ;       V02-082 JAK0058         J A Krycka      22-MAY-1981
0000   160 ;               This module was created from RMSEXPSTRING code previously
0000   161 ;               residing in RMOXPFN.
0000   162 ;
0000   163 ;       The following edit history entries were copied from RMOXPFN:
0000   164 ;
0000   165 ;       V02-081 KRM0014         K R Malik       11-MAY-1981
0000   166 ;               Fill in the extended NAM block fields with the resultant
0000   167 ;               name string filespec elements for a resultant name string
0000   168 ;               that was received from the remote FAL.
0000   169 ;
0000   170 ;       V02-078 KRM0013         K R Malik       22-APR-1981
0000   171 ;               Make RMSEXPSTRING fill in the extended NAM block fields with
0000   172 ;               the addresses & lengths of the various filespec elements
0000   173 ;               of the expanded or resultant name string (as appropriate for
0000   174 ;               the operation).
0000   175 ;
0000   176 ;--
```

```
                     0000   178          .SBTTL  DECLARATIONS
                     0000   179
                     0000   180 ;
                     0000   181 ; Include Files:
                     0000   182 ;
                     0000   183
                     0000   184          $DEVDEF                         ; Define Device characteristics
                     0000   185          $FABDEF                         ; Define File Access Block symbols
                     0000   186          $FIBDEF                         ; Define File Information Block
                     0000   187          $FSCBDEF                        ; Define FSCB symbols
                     0000   188          $FWADEF                         ; Define File Work Area symbols
                     0000   189          $IFBDEF                         ; Define IFAB symbols
                     0000   190          $LOGDEF                         ; Define Logical Name Table symbols
                     0000   191          $NAMDEF                         ; Define Name Block symbols
                     0000   192          $NWADEF                         ; Define Network Work Area symbols
                     0000   193          $PSLDEF                         ; Define Process Status Longword symbols
                     0000   194          $RMSDEF                         ; Define RMS error codes
                     0000   195
                     0000   196 ;
                     0000   197 ; Macros:
                     0000   198 ;
                     0000   199 ;          None
                     0000   200 ;
                     0000   201 ; Equated symbols:
                     0000   202 ;
                     0000   203 ;          The following symbol definitions were copied from RMOXFPN:
                     0000   204 ;
                     0000   205
00000020             0000   206          FOP=FAB$L_FOP*8                 ; Bit offset to FOP
                     0000   207
00000020             0000   208          SPACE           = ^X20          ; ASCII value for space
00000009             0000   209          HOR_TAB         = ^X09          ; ASCII value for horizontal tab
```

```
0000    211          .SBTTL  RM$EXPSTRING, Build Expanded or Resultant Name String
0000    212
0000    213 ;++
0000    214 ;
0000    215 ; RM$EXPSTRING - examines the user's NAM block for an expanded name string or
0000    216 ;       resultant name string buffer, and if found builds the string in the
0000    217 ;       buffer utilizing the separately parsed elements stored in the FWA and
0000    218 ;       NWA control blocks.
0000    219 ;
0000    220 ; Calling Sequence:
0000    221 ;
0000    222 ;       BSBW    RM$EXPSTRING
0000    223 ;
0000    224 ; Input Parameters:
0000    225 ;
0000    226 ;       R7      NAM block address
0000    227 ;       R8      FAB address
0000    228 ;       R9      IFAB address
0000    229 ;       R10     FWA address
0000    230 ;       AP      Address of 5-byte argument list of the format:
0000    231 ;               .byte   offset to either NAM$L_ESA or NAM$L_RSA
0000    232 ;               RMSERR_WORD     NAM$_ESA (or _RST)
0000    233 ;               RMSERR_WORD     NAM$_ESS (or _RSS)
0000    234 ;
0000    235 ; Implicit Inputs:
0000    236 ;
0000    237 ;       The current contents of the FWA.
0000    238 ;       FAB$L_NAM, NAM$L_ESA (or _RSA), NAM$B_ESS (or _RSS)
0000    239 ;       NAM$B_BID, NAM$B_BLN
0000    240 ;
0000    241 ; Outputs:
0000    242 ;
0000    243 ;       R0      Status code
0000    244 ;       R1-R7   Destroyed
0000    245 ;       AP      Destroyed
0000    246 ;
0000    247 ; Implicit Outputs:
0000    248 ;
0000    249 ;       If the specified buffer exists, the buffer is filled in with the
0000    250 ;       expanded or resultant name string, and its length is stored in the
0000    251 ;       NAM$B_ESL or NAM$B_RSL field, as appropriate.
0000    252 ;
0000    253 ;       If the NAM$B_BLN field indicates that this is an extended NAM block
0000    254 ;       then the following filespec element fields are also filled in:
0000    255 ;
0000    256 ;               NAM$B_NODE      NAM$L_NODE
0000    257 ;               NAM$B_DEV       NAM$L_DEV
0000    258 ;               NAM$B_DIR       NAM$L_DIR
0000    259 ;               NAM$B_NAME      NAM$L_NAME
0000    260 ;               NAM$B_TYPE      NAM$L_TYPE
0000    261 ;               NAM$B_VER       NAM$L_VER
0000    262 ;
0000    263 ; Completion Codes:
0000    264 ;
0000    265 ;       Standard RMS completion codes, including SUC, ESA, ESS, RSA, RSS, NAM.
0000    266 ;
0000    267 ; Side Effects:
```

```
                                0000   268 ;
                                0000   269 ;       If a NAM block exists it will have been probed.
                                0000   270 ;
                                0000   271 ;--
                                0000   272
                                0000   273 ;
                                0000   274 ; Return the expanded or resultant name string to the user buffer and, if this
                                0000   275 ; is an 'extended' NAM block, fill in the various filespec element fields.
                                0000   276 ;
                                0000   277
                                0000   278 RMS$EXPSTRING::
                         0594   30 0000   279          BSBW    RMS$CHKNAM                      ; validate NAM block
                       01 50    E8 0003   280          BLBS    R0,5$                           ; if ok continue
                                05 0006   281          RSB                                     ; else exit now
                                   0007   282
                                   0007   283          ASSUME  NAM$L_ESA       EQ      NAM$B_ESL+1
                                   0007   284          ASSUME  NAM$B_ESL       EQ      NAM$B_ESS+1
                                   0007   285          ASSUME  NAM$L_RSA       EQ      NAM$B_RSL+1
                                   0007   286          ASSUME  NAM$B_RSL       EQ      NAM$B_RSS+1
                                   0007   287
                     0900 8F    BB 0007   288 5$:      PUSHR   #^M<R8,R11>                     ; save registers
                          53    8C 000B   289          MOVZBL  (AP)+,R3                        ; get offset to ESA or RSA field
                  58      57    53 000E   290          ADDL3   R3,R7,R8                        ;  and use to compute address
                          53    68 0012   291          MOVL    (R8),R3                         ; Get address of buffer
                                10 0015   292          BEQL    20$                             ; Branch if non-zero
                            54   FE A8 0017   293          MOVZBL  -2(R8),R4                       ; Get buffer size
                                0A 001B   294          BEQL    20$                             ; Branch if non-zero
                08 00 BE        38 001D   295          BBC     #FAB$V_NAM+FOP,@0(SP),30$       ; Branch if not open by Name Block
                                24 A7 0022   296          TSTL    NAM$W_FID(R7)                   ; File ID input?
                                03 0025   297          BEQL    30$                             ; Branch if not
                                01D0   31 0027   298 20$:     BRW     EXIT_SUC                        ; Branch, omitting directory
                                   002A   299                                                     ;  and file name
                                   002A   300 30$:     IFWRT   R4,(R3),40$,IFB$B_MODE(R9)
                                01CE   31 0031   301          BRW     ERRSA                           ; Branch if no write access
                                   78   94 0034   302 40$:     CLRB    -(R8)                           ; Clear ESL (or RSL)
                                   0036   303
                                   0036   304
                                   0036   305 ;
                                   0036   306 ; If this is an 'extended' NAM block, initialize the filespec element fields.
                                   0036   307 ;
                                   0036   308
                                   0036   309          ASSUME  NAM$B_NODE+1    EQ      NAM$B_DEV
                                   0036   310          ASSUME  NAM$B_DEV+1     EQ      NAM$B_DIR
                                   0036   311          ASSUME  NAM$B_DIR+1     EQ      NAM$B_NAME
                                   0036   312          ASSUME  NAM$B_NAME+1    EQ      NAM$B_TYPE
                                   0036   313          ASSUME  NAM$B_TYPE+1    EQ      NAM$B_VER
                                   0036   314
            60 8F    01 A7    91 0036   315 45$:     CMPB    NAM$B_BLN(R7),#NAM$C_BLN        ; Is this an extended NAM block?
                                54   1F 003B   316          BLSSU   50$                             ; Branch if not
                             50    39 A7 003D   317          MOVZBL  NAM$B_DEV(R7),R0                ; Get ESA device name len
                             51    44 A7 0041   318          MOVL    NAM$L_DEV(R7),R1                ; and ptr
                                   0045   319
                                   38 A7   7C 0045   320          CLRQ    NAM$B_NODE(R7)                  ; Initialize all filespec element
                                   0048   321                                                     ;  length fields with a count of zero
                          40 A7     53   D0 0048   322          MOVL    R3,NAM$L_NODE(R7)               ; Initialize all filespec element
                          44 A7     53   D0 004C   323          MOVL    R3,NAM$L_DEV(R7)                ;  address fields with the address
                          48 A7     53   D0 0050   324          MOVL    R3,NAM$L_DIR(R7)                ;  of the start of the user buffer
```

```
                4C A7    53    D0  0054   325          MOVL    R3,NAM$L_NAME(R7)       ; ''
                50 A7    53    D0  0058   326          MOVL    R3,NAM$L_TYPE(R7)       ; ''
                54 A7    53    D0  005C   327          MOVL    R3,NAM$L_VER(R7)        ; ''
                                   0060   328
                                   0060   329 ;
                                   0060   330 ; If the DVI device name was used, and an 'extended NAM block' is present,
                                   0060   331 ; and there is a non-zero DEV field; then use the DEV field as the device
                                   0060   332 ; name and not the DVI name.   This hides the expansion of hidden devices.
                                   0060   333 ; Copy the device string to the unused concealed device buffer and set
                                   0060   334 ; it from there.  If saved field is zero or inaccessible, fall back on DVI.
                                   0060   335 ; Can't be a network filespec if the DVI was used.
                                   0060   336 ;
                                   0060   337
                   2D 6A    02    E0  0060   338          BBS     #FWA$V_SL_PASS,(R10),50$; branch if search list pass
              28 00 BE      38    E1  0064   339          BBC     #FAB$V_NAM+FOP,@0(SP),50$; Branch if not open by Name Block
                   24 6A    05    E1  0069   340          BBC     #FWA$V_NAM_DVI,(R10),50$; branch if not using dvi
                        14 A7    95  006D   341          TSTB    NAM$T_DVI(R7)           ; Did we have a DVI?
                           1F    13  0070   342          BEQL    50$                     ; Branch if not
                           50    D7  0072   343          DECL    R0                      ; eliminate ":"
                           1B    1B  0074   344          BLEQU   50$                     ; Ignore null or bogus field
                                   0076   345          IFNORD  R0,(R1),50$,IFB$B_MODE(R9) ; Can we read it?
                56    00E8 CA    9E  007D   346          MOVAB   FWA$Q_CONCEAL_DEV(R10),R6 ; Get device descriptor
                           66    50  0082   347          MOVL    R0,(R6)                 ; Set length
                           53    DD  0085   348          PUSHL   R3                      ; Save output ptr
              04 B6        61    50  28  0087   349          MOVC3   R0,(R1),@4(R6)          ; Copy string
                           53 8ED0  008C   350          POPL    R3                      ; restore ptr
                           1C    11  008F   351          BRB     COPY_DVI                ; join device copy code
                                   0091   352
                                   0091   353 ;
                                   0091   354 ; Check for the occurrence of a node name (and possilby a quoted string).
                                   0091   355 ;
                                   0091   356
                      6A   19    E1  0091   357 50$:     BBC     #FWA$V_NODE,(R10),-     ; Check for node name
                           03          0094   358                  COPY_DEVICE             ;
                         019A  31  0095   359          BRW     COPY_NODE               ; Branch if node name found
                                   0098   360
                                   0098   361 ;
                                   0098   362 ; Move the device name string and, if this is an 'extended' NAM block, fill in
                                   0098   363 ; the DEV element fields.
                                   0098   364 ;
                                   0098   365
                                   0098   366 COPY_DEVICE:
                   2A 6A    0F    E1  0098   367          BBC     #FWA$V_DEVICE,(R10),COPY_DIR    ; branch if no device seen
                56    00E0 CA    7E  009C   368          MOVAQ   FWA$Q_DEVICE(R10),R6            ; get device descriptor address
                      6A   39    E1  00A1   369          BBC     #FWA$V_CONCEAL_DEV,(R10),-      ; is there a concealed device?
                           08          00A4   370                  COPY_DVI
                           04    E1  00A5   371          BBC     #NAM$V_NOCONCEAL,-              ; do we display it?
                        03 08 A7       00A7   372                  NAM$B_NOP(R7),COPY_DVI
                                   00AA   373
                                   00AA   374          ASSUME  FWA$Q_CONCEAL_DEV       EQ      FWA$Q_DEVICE+8
                                   00AA   375
                           56    08    C0  00AA   376          ADDL2   #8,R6                           ; yes
                                   00AD   377 COPY_DVI:
                 60 8F       01 A7    91  00AD   378          CMPB    NAM$B_BLN(R7),#NAM$C_BLN        ; is this an extended NAM block?
                           09    1F  00B2   379          BLSSU   10$                             ; branch if not
                        44 A7    53    D0  00B4   380          MOVL    R3,NAM$L_DEV(R7)                ; fill in DEV address field
              39 A7        01    66    81  00B8   381          ADDB3   (R6),#1,NAM$B_DEV(R7)           ; fill in DEV length field
```

```
                     00BD    382                                            ;  (+ colon)
          0147   30 00BD    383 10$:    BSBW    MOVE_NEXT                  ; copy device name
     50     3A   90 00C0    384         MOVB    #^A/:/,R0                  ; append a colon
          0154   30 00C3    385         BSBW    MOVE_CHAR                  ;
                 00C6    386
                 00C6    387 ;
                 00C6    388 ; Build up full directory spec from the parts and, if this is an 'extended'
                 00C6    389 ; NAM block, fill in the DIR element fields.
                 00C6    390 ;
                 00C6    391
                 00C6    392 COPY_DIR:
           0A AA   95 00C6    393         TSTB    FWA$B_DIRTERM(R10)         ; was there a right bracket?
                46   13 00C9    394         BEQL    60$                        ; no, skip copy?
 60 8F     01 A7   91 00CB    395 10$:    CMPB    NAM$B_BLN(R7),#NAM$C_BLN   ; is this an extended NAM block?
                04   1F 00D0    396         BLSSU   20$                        ; branch if not
     48 A7     53   D0 00D2    397         MOVL    R3,NAM$L_DIR(R7)           ; fill in DIR address field
     3A 6A     3A   E1 00D6    398 20$:    BBC     #FWA$V_ROOT_DIR,(R10),80$  ; was there a root directory?
     05 6A     3C   E0 00DA    399         BBS     #FWA$V_EXP_ROOT,(R10),30$  ; was it explicit?
                04   E1 00DE    400         BBC     #NAM$V_NOCONCEAL,-         ; should we display it?
        31 08 A7         00E0    401                 NAM$B_NOP(R7),80$
                         00E3    402
                         00E3    403 ;
                         00E3    404 ; We have rooted directories and need to display them
                         00E3    405 ;
                         00E3    406
50     0B AA     02   83 00E3    407 30$:    SUBB3   #2,FWA$B_ROOTERM(R10),R0  ; copy open left bracket
              012F   30 00E8    408         BSBW    MOVE_CHAR                  ;
   56     00F0 CA   7F 00EB    409         MOVAQ   FWA$Q_CDIR1(R10),R6        ; get concealed descriptor list
              0114   30 00F0    410 40$:    BSBW    MOVE_NEXT                  ; copy next directory name
           50   2E   90 00F3    411         MOVB    #^A/./,R0                  ;
              0121   30 00F6    412         BSBW    MOVE_CHAR                  ; and append delimiter
                66   D5 00F9    413         TSTL    (R6)                       ; any left?
                09   13 00FB    414         BEQL    50$                        ; no
           0128 CA   7F 00FD    415         PUSHAQ  FWA$Q_CDIR8(R10)           ; check end of list
           8E   56   D1 0101    416         CMPL    R6,(SP)+                   ;
                EA   1B 0104    417         BLEQU   40$                        ; not yet
                         0106    418
                         0106    419 ;
                         0106    420 ; We have displayed the rooted directories, check for normal ones
                         0106    421 ;
                         0106    422
        50     0B AA   90 0106    423 50$:    MOVB    FWA$B_ROOTERM(R10),R0      ; copy close right bracket
                 010D   30 010A    424         BSBW    MOVE_CHAR                  ;
        03 6A     0E   E0 010D    425         BBS     #FWA$V_DIR,(R10),80$       ;
                 008A   31 0111    426 60$:    BRW     COPY_NAME                  ; branch if no directory wanted
                         0114    427
                         0114    428 ;
                         0114    429 ; Display the normal directory levels
                         0114    430 ;
                         0114    431
50     0A AA     02   83 0114    432 80$:    SUBB3   #2,FWA$B_DIRTERM(R10),R0   ; copy left bracket
                 00FE   30 0119    433         BSBW    MOVE_CHAR                  ;  to user buffer
        67 6A     0E   E1 011C    434         BBC     #FWA$V_DIR,(R10),170$      ; branch if no directo.y wanted
     56     0130 CA   7E 0120    435         MOVAQ   FWA$Q_DIR1(R10),R6         ; get directory descriptors
        27 6A     1B   E0 0125    436         BBS     #FWA$V_GRPMBR,(R10),110$   ; branch if [group,member] format
                  1D   EF 0129    437         EXTZV   #FWA$V_DIR_LVLS,-          ; get number of subdirectories
     5B   6A     03         012B    438                 #FWA$S_DIR_LVLS,(R10),R11
```

```
                00D6   30   012E    439 90$:    BSBW    MOVE_NEXT                       ; copy next directory name
             50   2E   90   0131    440         MOVB    #^A/./,R0                       ;
                00E3   30   0134    441         BSBW    MOVE_CHAR                       ; and append delimiter
          06 F8 A6   10   E1   0137    442         BBC     #FSCB$V_ELIPS,-8(R6),100$       ; elipsis following this name?
                00DB   30   013C    443         BSBW    MOVE_CHAR                       ; if so, append '..'
                00D8   30   013F    444         BSBW    MOVE_CHAR                       ;
              E9 5B   F4   0142    445 100$:   SOBGEQ  R11,90$                         ; loop until done
                           0145    446
                           0145    447 ;
                           0145    448 ; Done with string, check for trailing elipsis, if so keep the 3 dots
                           0145    449 ;
                           0145    450
          3D F8 A6   10   E0   0145    451         BBS     #FSCB$V_ELIPS,-8(R6),170$       ; branch if trailing elipsis?
                   53   D7   014A    452         DECL    R3                              ; remove trailing '.'
                   68   97   014C    453         DECB    (R8)                            ; decrement string size
                   37   11   014E    454         BRB     170$                            ; add right bracket
                           0150    455
                           0150    456 ;
                           0150    457 ; Copy group member direcotries
                           0150    458 ;
                           0150    459
                   04   E0   0150    460 110$:   BBS     #NAM$V_NOCONCEAL,-              ; convert if noconceal
             0E 08 A7           0152    461                 NAM$B_NOP(R7),120$              ;  to non-uic format
                00AF   30   0155    462         BSBW    MOVE_NEXT                       ; copy group portion
                50   2C   90   0158    463         MOVB    #^A/,/,R0                       ; copy UIC format separator
                00BC   30   015B    464         BSBW    MOVE_CHAR                       ;  to user buffer
                00A6   30   015E    465         BSBW    MOVE_NEXT                       ; copy member portion
                   24   11   0161    466         BRB     170$
                           0163    467
                50   30   90   0163    468 120$:   MOVB    #^A/0/,R0                       ; get fill character
                51   66   9A   0166    469         MOVZBL  (R6),R1                         ; get size of group
                   03   11   0169    470         BRB     140$
                00AC   30   016B    471 130$:   BSBW    MOVE_CHAR                       ; pad one
             F9 51   03   F3   016E    472 140$:   AOBLEQ  #3,R1,130$
                0092   30   0172    473         BSBW    MOVE_NEXT                       ; move group
                50   30   90   0175    474         MOVB    #^A/0/,R0                       ; get fill character
                51   66   9A   0178    475         MOVZBL  (R6),R1                         ; get size of member
                   03   11   017B    476         BRB     160$
                009A   30   017D    477 150$:   BSBW    MOVE_CHAR                       ; pad one
             F9 51   03   F3   0180    478 160$:   AOBLEQ  #3,R1,150$
                0080   30   0184    479         BSBW    MOVE_NEXT                       ; move member
                           0187    480
                           0187    481 ;
                           0187    482 ; Finish off the directory string
                           0187    483 ;
                           0187    484
                50   0A AA   90   0187    485 170$:   MOVB    FWA$B_DIRTERM(R10),R0           ; Copy right bracket
                   008C   30   018B    486         BSBW    MOVE_CHAR                       ;  (']' or '>') to user buffer
             60 8F   01 A7   91   018E    487         CMPB    NAM$B_BLN(R7),#NAM$C_BLN        ; Is this an extended NAM block?
                   09   1F   0193    488         BLSSU   COPY_NAME                       ; Branch if not
          51   53   48 A7   C3   0195    489         SUBL3   NAM$C_DIR(R7),R3,R1             ; Compute DIR length
                3A A7   51   90   019A    490         MOVB    R1,NAM$B_DIR(R7)                ; Fill in DIR length field
                           019E    491
                           019E    492 ;
                           019E    493 ; Now move the file name, type, and version (all stored in one string) and,
                           019E    494 ; if this is an 'extended' NAM block, fill in the individual filespec
                           019E    495 ; element fields.
```

```
                     019E   496 ;
                     019E   497 ; Note: This concatenated name string is guaranteed to have the "." and ";"
                     019E   498 ;       delimiters present, and it may simply be ".;" if all elements are null.
                     019E   499 ;
                     019E   500
                     019E   501 COPY_NAME:                                  ;
60 8F    01 A7   91  019E   502          CMPB    NAM$B_BLN(R7),#NAM$C_BLN; Is this an extended NAM block?
            4E   1F  01A3   503          BLSSU   10$                       ; Branch if not
 54  0170 CA   7D  01A5   504          MOVQ    FWA$Q_NAME(R10),R4        ; Get descriptor of name-type-ver string
                   01AA   505                                            ;  that will be copied to user buffer
      65   22   91  01AA   506          CMPB    #^A/"/,(R5)               ; is the 1st character a quote?
           16   12  01AD   507          BNEQ    5$                        ; if neq no, continue
                   01AF   508
                   01AF   509 ;
                   01AF   510 ; Since the first character of the string is a quote we cannot simply scan
                   01AF   511 ; for a . to find the end of the name.  We must instead look for the last
                   01AF   512 ; quote in the string, and assume that that ends the name field.
                   01AF   513 ;
                   01AF   514
      50   54   7D  01AF   515          MOVQ    R4,R0                     ; copy descriptor of string
      51   50   C0  01B2   516          ADDL2   R0,R1                     ; point past last char of string
                   01B5   517
      71   22   91  01B5   518 3$:      CMPB    #^A/"/,-(R1)              ; pick of next char from end
           FB   12  01B8   519          BNEQ    3$                        ; if neq its not a quote, yet
                51  D6  01BA   520          INCL    R1                        ; readjust address for later code
 52   51   55   C3  01BC   521          SUBL3   R5,R1,R2                  ; calc length passed over in R2
      50   52   C2  01C0   522          SUBL2   R2,R0                     ; calc lenght left
           08   11  01C3   523          BRB     6$                        ; continue like we found .
                   01C5   524
 65   54   2E   3A  01C5   525 5$:      LOCC    #^A/./,R4,(R5)            ; Find dot delimiter
 52   51   55   C3  01C9   526          SUBL3   R5,R1,R2                  ; Compute NAME length
   4C A7   53   D0  01CD   527 6$:      MOVL    R3,NAM$L_NAME(R7)         ; Fill in NAME address field
   3B A7   52   90  01D1   528          MOVB    R2,NAM$B_NAME(R7)         ; Fill in NAME length field
        54   50   7D  01D5   529          MOVQ    R0,R4                     ; Get descriptor of type-ver string
 65   54   3B   3A  01D8   530          LOCC    #^A/;/,R4,(R5)            ; Find semi-colon delimiter
50 A7   52   53   C1  01DC   531          ADDL3   R3,R2,NAM$L_TYPE(R7)      ; Compute and fill in TYPE address field
 52   51   55   C3  01E1   532          SUBL3   R5,R1,R2                  ; Compute TYPE length
   3C A7   52   90  01E5   533          MOVB    R2,NAM$B_TYPE(R7)         ; Fill in TYPE length field
54 A7   52   50 A7   C1  01E9   534          ADDL3   NAM$L_TYPE(R7),R2,-       ; Compute and fill in VER address field
                   01EF   535                  NAM$L_VER(R7)             ;
   3D A7   50   90  01EF   536          MOVB    R0,NAM$B_VER(R7)          ; Fill in VER length field
 56   0170 CA   7E  01F3   537 10$:     MOVAQ   FWA$Q_NAME(R10),R6        ; Get address of descriptor of name
           0D   10  01F8   538          BSBB    MOVE_NEXT                 ;  string and append it to user buffer
                   01FA   539
                   01FA   540 EXIT_SUC:
                   01FA   541          RMSSUC                            ; Declare success
                   01FD   542
                   01FD   543 EXIT_ERR:
      0900 8F   BA  01FD   544          POPR    #^M<R8,R11>               ; Restore registers
           05   0201   545          RSB                               ; Exit to caller
                0202   546
                0202   547 ;
                0202   548 ; User buffer is not writeable.
                0202   549 ;
                0202   550
      50   6C   3C  0202   551 ERRSA:  MOVZWL  (AP),R0                   ; Get error code
           F6   11  0205   552          BRB     EXIT_ERR                  ; And take exit path
```

```
                     0207   554 ;++
                     0207   555 ;
                     0207   556 ; This routine moves a field to the expanded or resultant string buffer.
                     0207   557 ;
                     0207   558 ; Update ESL (or RSL) count while checking for overflow, and if so, exit
                     0207   559 ; with ESS (or RSS) error.
                     0207   560 ;
                     0207   561 ; Inputs:
                     0207   562 ;
                     0207   563 ;       R3      Address of output buffer
                     0207   564 ;       R6      Address of descriptor of input string
                     0207   565 ;       R8      ESL (or RSL) address in NAM block
                     0207   566 ;       AP      Address of expanded or resultant string argument list
                     0207   567 ;
                     0207   568 ; Outputs:
                     0207   569 ;
                     0207   570 ;       R0-R2   Destroyed
                     0207   571 ;       R3      Address following string in the output buffer
                     0207   572 ;       R4-R5   Destroyed
                     0207   573 ;       R6      R6 on input + 8
                     0207   574 ;       R8,AP   Unchanged
                     0207   575 ;
                     0207   576 ;       *** NOTE ***
                     0207   577 ;
                     0207   578 ;       If the user hasn't specified a name block, there is no
                     0207   579 ;       enforcement of the maximum for filename length.  In that
                     0207   580 ;       case, the file name consists of its individual pieces in
                     0207   581 ;       the FWA.  This problem should be addressed someday.
                     0207   582 ;--
                     0207   583
                     0207   584 MOVE_NEXT:
     68    66    80    0207   585         ADDB2   (R6),(R8)               ; Count the string
           1C    1F    020A   586         BCS     POPPC                   ; greater than 255?
   FF  A8    68    91    020C   587         CMPB    (R8),-1(R8)             ; Does it fit?
           16    1A    0210   588         BGTRU   POPPC                   ; Branch if not
     50    86    D0    0212   589         MOVL    (R6)+,R0                ; Get length of string
63   96    50    28    0215   590         MOVC3   R0,@(R6)+,(R3)          ; Move field
                 05    0219   591         RSB                             ;
                       021A   592
                       021A   593 ;++
                       021A   594 ;
                       021A   595 ; This routine moves the character in R0 to the expanded or resultant string
                       021A   596 ; while checking for overflow.
                       021A   597 ;
                       021A   598 ;--
                       021A   599
                       021A   600 MOVE_CHAR:                              ;
           68    96    021A   601         INCB    (R8)                    ; Count character
           0A    1F    021C   602         BCS     POPPC                   ; Greater than 255?
   FF  A8    68    91    021E   603         CMPB    (R8),-1(R8)             ; Does it fit?
           04    1A    0222   604         BGTRU   POPPC                   ; Branch if not
     83    50    90    0224   605         MOVB    R0,(R3)+                ; Move in the byte
                 05    0227   606         RSB                             ;
                       0228   607
                       0228   608 ;
                       0228   609 ; Field will overflow user (expanded or resultant string) buffer.
                       0228   610 ; Return ESS or RSS error.
```

```
                     0228   611 ;
                     0228   612
          01   BA    0228   613 POPPC:  POPR    #^M<R0>                 ; Pop return PC
68   FF A8   90    022A   614          MOVB    -1(R8),(R8)             ; Make string length = buffer length
                     022E   615
                     022E   616 ;
                     022E   617 ; User buffer is too small.
                     022E   618 ;
                     022E   619
          8C   B5    022E   620 ERRSS:  TSTW    (AP)+                   ; Move to ESS or RSS error code
          D0   11    0230   621          BRB     ERRSA                   ; Branch aid
```

```
                     0232   623 ;++
                     0232   624 ;
                     0232   625 ; Copy node spec list to the user buffer and optionally mask out the
                     0232   626 ; password string in each node spec (if present). Also return the remote
                     0232   627 ; file system type to the user.
                     0232   628 ;
                     0232   629 ;       Inputs:
                     0232   630 ;
                     0232   631 ;       R3      Next user buffer address
                     0232   632 ;       R7      NAM block address
                     0232   633 ;       R9      IFAB address
                     0232   634 ;       R10     FWA address
                     0232   635 ;
                     0232   636 ;       Outputs:
                     0232   637 ;
                     0232   638 ;       R0-R2   Destroyed
                     0232   639 ;       R3      Next user buffer address (updated)
                     0232   640 ;       R4-R6   Destroyed
                     0232   641 ;       R11     Destroyed
                     0232   642 ;
                     0232   643 ;--
                     0232   644
                     0232   645 COPY_NODE:
                     0232   646
                     0232   647 ;
                     0232   648 ; First return the remote file system type in the NAM block.
                     0232   649 ;
                     0232   650 ; Note: The information is available only after FAL has been accessed. Thus
                     0232   651 ;       the information is valid only when returning a resultant name string.
                     0232   652 ;
                     0232   653
 52  3C A9   D0      0232   654         MOVL    IFB$L_NWA_PTR(R9),R2    ; Get address of NWA
00C5 C2     90      0236   655         MOVB    NWA$B_FILESYS(R2),-     ; Copy remote file system type stored
   09 A7              0239   656                 NAM$B_RFS(R7)           ;  in NWA to NAM block
                     023C   657
                     023C   658 ;
                     023C   659 ; Now copy node spec list to the user buffer.
                     023C   660 ;
                     023C   661
        5B   D4      023C   662         CLRL    R11                     ; zero loop counter
56  01B4 CA4B   7E   023E   663 10$:     MOVAQ   FWA$Q_NODE1(R10)[R11],R6; get address of next node spec desc
  09 66    12   E1   0244   664         BBC     #FSCB$V_ACS,(R6),20$    ; branch if this node spec has
                     0248   665                                         ;  no access control string, (flags
                     0248   666                                         ;  are stored in bytes 2-3 of descriptor)
        5B   D5      0248   667         TSTL    R11                     ; Do not mask the password for
        05   12      024A   668         BNEQ    20$                     ;  secondary node names
        00   E1      024C   669         BBC     #NAM$V_PWD,-            ; branch if password is to be
   05 08 A7           024E   670                 NAM$B_NOP(R7),30$       ;  masked out of access control string
        B4   10      0251   671 20$:     BSBB    MOVE_NEXT               ; Move this node spec string
      00B0   31      0253   672         BRW     80$                     ;  to user buffer (unaltered)
                     0256   673
                     0256   674 ;
                     0256   675 ; The node spec string contains an embedded access control string, i.e.,
                     0256   676 ; the string is guaranteed to be of the form:
                     0256   677 ;
                     0256   678 ;       nodename"access_control_string"::
                     0256   679 ;
```

```
                                0256   680 ; Search the node spec string for a password substring and replace it with a
                                0256   681 ; dummy substring. Then copy the modified node spec string to the user buffer.
                                0256   682 ;
                                0256   683 ; Find delimiter before password (either space or tab).
                                0256   684 ;
                                0256   685
04 B6   66   22   3A            0256   686 30$:     LOCC    #^A/"/,(R6),@4(R6)      ; Locate start of ACS
                F4   13         025B   687          BEQL    20$                     ; Something is wrong!!
                50   D7         025D   688 40$:     DECL    R0                      ; Make <R0,R1> point to next
                51   D6         025F   689          INCL    R1                      ;  character in string
           20   61   91         0261   690          CMPB    (R1),#SPACE             ; Look for space delimiter
                0C   13         0264   691          BEQL    50$                     ; Branch if found
           09   61   91         0266   692          CMPB    (R1),#HOR_TAB           ; Look for horizontal tab delimiter
                07   13         0269   693          BEQL    50$                     ; Branch if found
           22   61   91         026B   694          CMPB    (R1),#^A/"/             ; Look for end of ACS
                E1   13         026E   695          BEQL    20$                     ; Branch if ACS is of the form:
                                0270   696                                          ;  "" or "username"
                EB   11         0270   697          BRB     40$                     ; End-of-username not yet reached
                                0272   698
                                0272   699 ;
                                0272   700 ; Delimiter before password has been found.
                                0272   701 ;
                                0272   702
                50   D7         0272   703 50$:     DECL    R0                      ; Make <R0,R1> point to possible
                51   D6         0274   704          INCL    R1                      ;  start of password string
           20   61   91         0276   705          CMPB    (R1),#SPACE             ; Skip over multiple spaces
                F7   13         0279   706          BEQL    50$                     ;  ""
           09   61   91         027B   707          CMPB    (R1),#HOR_TAB           ; Skip over multiple tabs
                F2   13         027E   708          BEQL    50$                     ;  ""
                                0280   709
                                0280   710 ;
                                0280   711 ; Set-up three descriptors that will describe the modified node spec string.
                                0280   712 ;
                                0280   713
          52   3C   A9   D0     0280   714          MOVL    IFB$L_NWA_PTR(R9),R2    ; Get address of NWA
     52   0120   C2   9E         0284   715          MOVAB   NWA$T_TEMP(R2),R2       ; Get address of scratch buffer
          62   66   7D           0289   716          MOVQ    (R6),(R2)               ; 1st descriptor describes node name
          62   50   C2           028C   717          SUBL2   R0,(R2)                 ;  string up to password
        08 A2   08   9A           028F   718          MOVZBL  #8,8(R2)                ; 2nd descriptor describes dummy
     0C A2   18 A2   9E           0293   719          MOVAB   24(R2),12(R2)           ;  string that replaces password
18 A2   64726F77 73736170 8F   7D 0298   720          MOVQ    #^A/password/,24(R2)    ; Store dummy string
                                02A4   721
                                02A4   722 ;
                                02A4   723 ; Find delimiter after password (either quote, space, or tab).
                                02A4   724 ;
                                02A4   725
           22   61   91         02A4   726 60$:     CMPB    (R1),#^A/"/             ; Branch if ACS is of the form:
                0E   13         02A7   727          BEQL    70$                     ;  "username password"
                50   D7         02A9   728          DECL    R0                      ; Make <R0,R1> point to next
                51   D6         02AB   729          INCL    R1                      ;  character in string
           20   61   91         02AD   730          CMPB    (R1),#SPACE             ; Branch if ACS is of the form:
                05   13         02B0   731          BEQL    70$                     ;  "username password account"
           09   61   91         02B2   732          CMPB    (R1),#HOR_TAB           ; Branch if end-of-password not
                ED   12         02B5   733          BNEQ    60$                     ;  yet reached
     10 A2   50   7D            02B7   734 70$:     MOVQ    R0,16(R2)               ; 3rd descriptor describes node
                                02BB   735                                          ;  name string after password
                                02BB   736
```

```
                     02BB    737 ;
                     02BB    738 ; Copy node spec string with password masked out.
                     02BB    739 ;
                     02BB    740
      56    52     D0  02BB    741          MOVL    R2,R6                   ; Copy address of 1st descriptor in set
   52    3C A9     D0  02BE    742          MOVL    IFB$L_NWA_PTR(R9),R2    ; Get address of NWA
 0164 C2    53     D0  02C2    743          MOVL    R3,NWA$Q_[NODE+4(R2)    ; Save address of returned string
         FF3D     30  02C7    744          BSBW    MOVE_NEXT               ; Use 1st descriptor
         FF3A     30  02CA    745          BSBW    MOVE_NEXT               ; Use 2nd descriptor
         FF37     30  02CD    746          BSBW    MOVE_NEXT               ; Use 3rd descriptor
                     02D0    747
                     02D0    748 ;
                     02D0    749 ; Create special logical name in user mode entered in the process logical name
                     02D0    750 ; table for the node-spec-string-with-password-masked-out with an equivalence
                     02D0    751 ; string that is the merged-unaltered-node-spec string. This is done so that
                     02D0    752 ; the expanded and resultant strings have password masked-out, yet are
                     02D0    753 ; still valid when used as a related file input string.
                     02D0    754 ;
                     02D0    755
   52    3C A9     D0  02D0    756          MOVL    IFB$L_NWA_PTR(R9),R2    ; Get address of NWA
55    0164 C2     D0  02D4    757          MOVL    NWA$Q_LNODE+4(R2),R5    ; Get address of returned string
54    53    55    C3  02D9    758          SUBL3   R5,R3,R4                ; Build descriptor of node spec string
      54    02    C2  02DD    759          SUBL2   #2,R4                   ;  in <R4,R5>
   5F 8F    65    91  02E0    760          CMPB    (R5),#^A/_/             ; Remove leading underscore (if present)
            04    12  02E4    761          BNEQ    75$                     ;  before creating special logical name
            54    D7  02E6    762          DECL    R4                      ;  of node spec (less double colon)
            55    D6  02E8    763          INCL    R5                      ;  with password masked out
0160 C2     54    7D  02EA    764 75$:     MOVQ    R4,NWA$Q_LNODE(R2)      ; Store desc of special logical name
                     02EF    765          $CRELOG_S-                      ; Create the special logical name
                     02EF    766                  TBLFLG=#LOG$C_PROCESS-  ;  Process logical name table
                     02EF    767                  LOGNAM=NWA$Q_[NODE(R2)- ;  String desc with dummy password
                     02EF    768                  EQLNAM=FWA$Q_NODE1(R10)[R11]- ;  String desc with real password
                     02EF    769                  ACMODE=#PSL$C_USER      ;  Put in process table in user mode
         1D 50     E9  0303    770          BLBC    R0,ERRDME               ; Branch on failure
   60 8F    01 A7    91  0306    771 80$:     CMPB    NAM$B_BLN(R7),#NAM$C_BLN; Is this an extended NAM block?
            09    1F  030B    772          BLSSU   90$                     ; Branch if not
51    53    40 A7     C3  030D    773          SUBL3   NAM$L_NODE(R7),R3,R1   ; Compute NODE length
      38 A7    51    90  0312    774          MOVB    R1,NAM$B_NODE(R7)       ; Fill in NODE length field
         6A    35    E0  0316    775 90$:     BBS     #FWA$V_REMRESULT,(R10),-; Branch if remote FAL has
               75        0319    776                  COPY_REMRESULT          ;  returned a resultant filespec
            2F AA    9D  031A    777          ACBB    FWA$B_SUBNODCNT(R10),-  ; Branch if there is another secondary
   FF1D 5B    01        031D    778                  #1,R11,10$              ;  node spec to process
            08    11  0321    779          BRB     COPY_QUOTED             ; Check for quoted string
                     0323    780
                     0323    781 ERRDME:  RMSERR  DME                     ; Declare error
         FED2     31  0328    782          BRW     EXIT_ERR                ; Branch aid
```

```
                        032B   784 ;++
                        032B   785 ;
                        032B   786 ; Copy the quoted string to the user buffer and optionally mask out
                        032B   787 ; /netacp_data (if present).  Also store the wildcard context of the
                        032B   788 ; quoted name.
                        032B   789 ;
                        032B   790 ;--
                        032B   791
                        032B   792 COPY_QUOTED:                                 ;
   03 6A   1A   E0      032B   793         BBS     #FWA$V_QUOTED,(R10),10$ ; Branch if quoted string follows
                        032F   794                                         ;  node name string
         FD66   31      032F   795         BRW     COPY_DEVICE             ; Rejoin mainline
60 8F    01 A7   91      0332   796 10$:    CMPB    NAM$B_BLN(R7),#NAM$C_BLN; Is this an extended NAM block
              04   1F      0337   797         BLSSU   20$                     ; Branch if not
   4C A7     53   D0      0339   798         MOVL    R3,NAM$L_NAME(R7)       ; Fill in NAME address field
 56    0170 CA   7E      033D   799 20$:    MOVAQ   FWA$Q_QUOTED(R10),R6    ; Get quoted string descriptor address
   34 6A     32   E1      0342   800         BBC     #FWA$V_NETSTR,(R10),30$ ; Branch if /netacp_data
                        0346   801                                         ;  was not present in quoted string
              00   E0      0346   802         BBS     #NAM$V_PWD,-            ; Branch if /netacp_data is not to
      2F 08 A7          0348   803                 NAM$B_NOP(R7),30$       ; be masked out from quoted string
                        034B   804 ;
                        034B   805 ; The quoted string contains an embedded /netacp_data. Replace it with a
                        034B   806 ; dummy string and copy the modified quoted string to the user buffer.
                        034B   807 ; Set-up two descriptors that will describe the modified quoted string.
                        034B   808 ;
                        034B   809
   50   3C A9   D0      034B   810         MOVL    IFB$L_NWA_PTR(R9),R0    ; Get address of NWA
51    016F C0   9A      034F   811         MOVZBL  NWA$B_NETSTRSIZ(R0),R1  ; Get length of /netacp_data''
52    0120 C0   9E      0354   812         MOVAB   NWA$T_TEMP(R0),R2       ; Get address of scratch buffer
      62   66   7D      0359   813         MOVQ    (R6),(R2)               ; 1st descriptor describes quoted
      62   51   C2      035C   814         SUBL2   R1,(R2)                 ;  string up to slash
         08 A2   08   9A      035F   815         MOVZBL  #8,8(R2)                ; 2nd descriptor describes dummy
      0C A2   10 A2   9E      0363   816         MOVAB   16(R2),12(R2)           ;  string that replaces /netacp_data''
10 A2   22617461 64706F2F 8F   7D      0368   817         MOVQ    #^A\/opdata"\,16(R2)    ; Store dummy string
                        0374   818
                        0374   819 ;
                        0374   820 ; Copy quoted string with /netacp_data masked out.
                        0374   821 ;
                        0374   822
         56   52   D0      0374   823         MOVL    R2,R6                   ; Copy descriptor set address
            FE8D   30      0377   824         BSBW    MOVE_NEXT               ; Use 1st descriptor
                        037A   825                                         ; Finish-up with 2nd descriptor
            FE8A   30      037A   826 30$:    BSBW    MOVE_NEXT               ; Copy quoted string
   60 8F    01 A7   91      037D   827         CMPB    NAM$B_BLN(R7),#NAM$C_BLN; Is this an extended NAM block?
              09   1F      0382   828         BLSSU   40$                     ; Branch if not
51    53   4C A7   C3      0384   829         SUBL3   NAM$L_NAME(R7),R3,R1    ; Compute NAME length
      3B A7   51   90      0389   830         MOVB    R1,NAM$B_NAME(R7)       ; Fill in NAME length field
                        038D   831
              29   11      038D   832 40$:    BRB     EX_SUC                  ; Rejoin mainline
```

```
                     038F    834 ;++
                     038F    835 ;
                     038F    836 ; Copy resultant string returned by remote FAL (which does not contain the
                     038F    837 ; leading node spec) to the user buffer.
                     038F    838 ;
                     038F    839 ; Note: This resultant string is described by the quoted string descriptor.
                     038F    840 ;
                     038F    841 ; Note: If the user quoted the filespec and entered only a primary node spec,
                     038F    842 ;       then the resultant string returned by FAL is quoted and copied to the
                     038F    843 ;       user buffer, because the quotes were removed by NT$GET_FILESPEC when
                     038F    844 ;       building the filespec to send to FAL, thus causing FAL to return a
                     038F    845 ;       resultant name string without quotes!
                     038F    846 ;
                     038F    847 ;--
                     038F    848
                     038F    849 COPY_REMRESULT:
 56   0170 CA   7E   038F    850         MOVAQ   FWA$Q_QUOTED(R10),R6    ; Get result string descriptor address
   16 6A   1A   E1   0394    851         BBC     #FWA$V_QUOTED,(R10),10$ ; Branch if user did not enclose
                     0398    852                                         ;  filespec in quotes
     2F AA   95   0398    853         TSTB    FWA$B_SUBNODCNT(R10)    ; Branch if there is more than one
        11   12   039B    854         BNEQ    10$                     ;  node spec in node spec list
   50   22   90   039D    855         MOVB    #^A/"/,R0               ; Add leading quote to resultant
      FE77   30   03A0    856         BSBW    MOVE_CHAR               ;  string
      FE61   30   03A3    857         BSBW    MOVE_NEXT               ; Copy the returned resultant string
   50   22   90   03A6    858         MOVB    #^A/"/,R0               ; Add trailing quote to resultant
      FE6E   30   03A9    859         BSBW    MOVE_CHAR               ;  string
        03   11   03AC    860         BRB     20$                     ;
      FE56   30   03AE    861 10$:    BSBW    MOVE_NEXT               ; Copy the returned resultant string
60 8F   01 A7   91   03B1    862 20$:    CMPB    NAM$B_BLN(R7),#NAM$C_BLN; Is this an extended NAM block?
        05   1E   03B6    863         BGEQU   PARSE_REMRESULT         ; Go parse the resultant name string
      FE3F   31   03B8    864 EX_SUC: BRW     EXIT_SUC                ; Rejoin mainline
```

```
                        03BB    866 ;++
                        03BB    867 ;
                        03BB    868 ; The resultant name string returned by FAL has not been parsed by RMS, so
                        03BB    869 ; there will not be FWA descriptors of the various filespec elements setup.
                        03BB    870 ; Therefore, hand-parse the string and fill in the extended NAM block fields.
                        03BB    871 ;
                        03BB    872 ; Note: FWA$Q_QUOTED is used to describe the resultant string returned by FAL.
                        03BB    873 ;       This does NOT imply that FAL has returned a quoted string.
                        03BB    874 ;
                        03BB    875 ; Note also that this code will not work if remote nodes ever return
                        03BB    876 ; a quoted file name string, e.g. an ANSI-"a" filespec.  Since magtape
                        03BB    877 ; access is not allowed over the network this is not currently a problem.
                        03BB    878 ;
                        03BB    879 ;--
                        03BB    880 TWO_COLONS:
                     3A 3A  03BB    881         .ASCII  ^::^                    ; Text for MATCHC instruction
                        03BD    882
                        03BD    883 PARSE_REMRESULT:
                  55   40 A7   D0  03BD    884         MOVL    NAM$L_NODE(R7),R5       ; Get address of resultant name string
                        03C1    885                                                 ;  already copied to user buffer
                     56   68   9A  03C1    886         MOVZBL  (R8),R6                 ; Get length of resultant name string
                        03C4    887                                                 ;  in user buffer (size of first node
                        03C4    888                                                 ;  name + size of FAL's returned string)
                        03C4    889         IFNORD  R6,(R5),EX_SUC,IFB$B_MODE(R9) ; Can we read it?
                        03CB    890
                        03CB    891 ;
                        03CB    892 ; Reparse the nodespec list, as the resultant string returned by FAL may have
                        03CB    893 ; additional node names to contribute to the nodespec list.
                        03CB    894 ;
                        03CB    895
                        03CB    896 REM_NODE:
65   56   EC A5   02   39  03CB    897 10$:    MATCHC  #2,B^TWO_COLONS,R6,(R5) ; Now find the next "::"
                  0B   12  03D1    898         BNEQ    20$                     ; Branch if no more nodes
                  55   53   D0  03D3    899         MOVL    R3,R5                   ; Save the address
                  56   52   D0  03D6    900         MOVL    R2,R6                   ; Save the # of remaining bytes
                  65   22   91  03D9    901         CMPB    #^A/"/,(R5)             ; Do we have a quoted string?
                       ED   12  03DC    902         BNEQ    10$                     ; Branch if not; look for another node
               54   55   40 A7   C3  03DE    903 20$:    SUBL3   NAM$L_NODE(R7),R5,R4    ; Compute NODE length
                  36 A7   54   90  03E3    904         MOVB    R4,NAM$B_NODE(R7)       ; Fill in NODE length field
                        03E7    905
                        03E7    906 REM_QUOTED:                                 ;
                  65   22   91  03E7    907         CMPB    #^A/"/,(R5)             ; Do we have a quoted string?
                       0A   12  03EA    908         BNEQ    REM_DEV                 ; Branch if not
                  4C A7   55   D0  03EC    909         MOVL    R5,NAM$L_NAME(R7)       ; Fill in NAME address field
                  3B A7   56   90  03F0    910         MOVB    R6,NAM$B_NAME(R7)       ; Fill in NAME length field
                       7A   11  03F4    911         BRB     SUC                     ; All done
                        03F6    912
                        03F6    913 REM_DEV:                                    ;
               65   56   3A   3A  03F6    914         LOCC    #^A/:/,R6,(R5)          ; Find single colon device delimiter
                       16   13  03FA    915         BEQL    REM_DIR                 ; Branch if no device seen
                  44 A7   55   D0  03FC    916         MOVL    R5,NAM$L_DEV(R7)        ; Fill in DEV address field
                       51   D6  0400    917         INCL    R1                      ; Step past the colon
               54   51   55   C3  0402    918         SUBL3   R5,R1,R4                ; Compute DEV length
                  39 A7   54   90  0406    919         MOVB    R4,NAM$B_DEV(R7)        ; Fill in DEV length field
                  55   51   DC  040A    920         MOVL    R1,R5                   ; Move pointer to next field
                  56   54   C2  040D    921         SUBL2   R4,R6                   ; Compute remaining length
                       5E   13  0410    922         BEQL    SUC                     ; Branch if end of string
```

```
                     0412   923
                     0412   924 REM_DIR:                                    ;
65   56   5D 8F  3A 0412   925          LOCC    #^A/]/,R6,(R5)              ; Look for closing square bracket
                06  12 0417   926          BNEQ    10$                         ; Branch if directory found
     65   56    3E  3A 0419   927          LOCC    #^A/>/,R6,(R5)              ; Look for closing angle bracket
                16  13 041D   928          BEQL    REM_NAME                    ; Branch if no directory found
       48 A7    55  D0 041F   929 10$:     MOVL    R5,NAM$L_DIR(R7)            ; Fill in DIR address field
                51  D6 0423   930          INCL    R1                          ; Step past the bracket delimiter
     54   51    55  C3 0425   931          SUBL3   R5,R1,R4                    ; Compute DIR length
       3A A7    54  90 0429   932          MOVB    R4,NAM$B_DIR(R7)            ; Fill in DIR length field
            55   51  D0 042D   933          MOVL    R1,R5                       ; Bump pointer to next field
            56   54  C2 0430   934          SUBL2   R4,R6                       ; Compute remaining length
                 3B  13 0433   935          BEQL    SUC                         ; Branch if end of string
                     0435   936
                     0435   937 REM_NAME:                                    ;
     65   56   2E  3A 0435   938          LOCC    #^A/./,R6,(R5)              ; Find dot delimiter
       4C A7    55  D0 0439   939          MOVL    R5,NAM$L_NAME(R7)           ; Fill in NAME address field
     54   51    55  C3 043D   940          SUBL3   R5,R1,R4                    ; Compute NAME length
       3B A7    54  90 0441   941          MOVB    R4,NAM$B_NAME(R7)           ; Fill in NAME length field
            55   51  D0 0445   942          MOVL    R1,R5                       ; Bump pointer to next field
            56   54  C2 0448   943          SUBL2   R4,R6                       ; Compute remaining length
                 23  13 044B   944          BEQL    SUC                         ; Branch if end of string
                     044D   945
                     044D   946 REM_TYPE:                                    ;
     65   56   3B  3A 044D   947          LOCC    #^A/;/,R6,(R5)              ; Look for semi-colon version delimiter
                07  12 0451   948          BNEQ    10$                         ; Branch if found
                56  D7 0453   949          DECL    R6                          ; Skip past leading dot!
01 A5    56   2E  3A 0455   950          LOCC    #^A/./,R6,1(R5)             ; Look for dot version delimiter
     50 A7    55  D0 045A   951 10$:     MOVL    R5,NAM$L_TYPE(R7)           ; Fill in TYPE address field
     54   51    55  C3 045E   952          SUBL3   R5,R1,R4                    ; Compute TYPE length
       3C A7    54  90 0462   953          MOVB    R4,NAM$B_TYPE(R7)           ; Fill in TYPE length field
                     0466   954
                     0466   955 REM_VER:                                    ;
3D A7    50   90 0466   956          MOVB    R0,NAM$B_VER(R7)            ; Fill in VER length field
                04  13 046A   957          BEQL    SUC                         ; Branch if end of string
54 A7    51   D0 046C   958          MOVL    R1,NAM$L_VER(R7)            ; Fill in VER address field
                03  10 0470   959 SUC:     BSBB    UPDATE_FNB                  ; Set appropriate FNB bits
            FD85  31 0472   960          BRW     EXIT_SUC                    ; Rejoin mainline
                     0475   961
```

```
                     0475   963 ;++
                     0475   964 ;
                     0475   965 ; This routine sets the wildcard directory FNB bits for a network $search.
                     0475   966 ; Currently FAL does not return these bits.
                     0475   967 ; These bits are input to an OFP if the input
                     0475   968 ; file NAM block is used as the RLF.
                     0475   969 ; Since we don't actually know how many are wild, we set as many wild bits
                     0475   970 ; as there are sub-directories; the UFD is never set wild.
                     0475   971 ; This is appropriate for most common cases:
                     0475   972 ;
                     0475   973 ;  works: COPY node::[A...]*.TXT        [Z...]*
                     0475   974 ;         COPY node::[A.*]*.TXT         [Z.*]*
                     0475   975 ;
                     0475   976 ;  fails: COPY node::[A.B...]*.TXT      [Z...]* ; gets B.DIR and shouldn't
                     0475   977 ;         COPY node::[...]*.TXT         [Z...]* ; doesn't get top level
                     0475   978 ;
                     0475   979 ; Inputs:
                     0475   980 ;
                     0475   981 ;       R7      NAM block address
                     0475   982 ;
                     0475   983 ; Outputs:
                     0475   984 ;
                     0475   985 ;       R0-R3   destroyed
                     0475   986 ;       FNB bits set
                     0475   987 ;
                     0475   988 ;--
                     0475   989
                     0475   990 UPDATE_FNB:
          53   D4   0475   991         CLRL    R3                              ; Zero dir_lvl counter
     50   3A A7   9A  0477   992         MOVZBL  NAM$B_DIR(R7),R0                ; Get dir length
               30   13  047B   993         BEQL    100$                            ; 0 length = no sub dirs
     51   48 A7   D0  047D   994         MOVL    NAM$L_DIR(R7),R1                ; Get adr of dir string
                     0481   995         IFNORD  R0,(R1),100$,IFB$B_MODE(R9)     ; Can we read it?
61   50   2E   3A  0488   996 10$:     LOCC    #^A/./,R0,(R1)                  ; Locate a dot
               08   13  048C   997         BEQL    20$                             ; No more sub dirs
               53   D6  048E   998         INCL    R3                              ; Found one/ update count
               51   D6  0490   999         INCL    R1                              ; Step passed found dot
               50   D7  0492  1000         DECL    R0                              ; Decr the length
               F2   11  0494  1001         BRB     10$                             ; See if there are any more
               53   D5  0496  1002 20$:     TSTL    R3                              ; How many did we find?
               13   13  0498  1003         BEQL    100$                            ; Branch if none
          15   53   F0  049A  1004         INSV    R3,#NAM$V_DIR_LVLS,-            ; Set the DIR_LVLS field
     34 A7   03       049D  1005                 #NAM$S_DIR_LVLS,NAM$L_FNB(R7)   ;  (a 3 bit field)
     09 6A   1C   E1  04A0  1006         BBC     #FWA$V_WILD_DIR,(R10),100$      ; Don't set any if none wild
          50   01   CE  04A4  1007         MNEGL   #1,R0                           ; Set all the bits
34 A7   53   19   50  F0  04A7  1008         INSV    R0,#NAM$V_WILD_SFD1,R3,-        ; Set as many wild bits as there
                         04AD  1009                 NAM$L_FNB(R7)                   ;  are dir_lvls (SFD1 -> SFD7)
                    05   04AD  1010 100$:    RSB                             ; All done.
                         04AE  1011
```

```
                        04AE  1013          .SBTTL  RMSGETFILNAM, Build Resultant File Name for Journaling
                        04AE  1014
                        04AE  1015 ;++
                        04AE  1016 ;
                        04AE  1017 ; RMSGETFILNAM - This routine uses the name block routines above to
                        04AE  1018 ; return  device:[directory...]filename.ext;version into a buffer
                        04AE  1019 ; for journaling.
                        04AE  1020 ;
                        04AE  1021 ; Calling Sequence:
                        04AE  1022 ;
                        04AE  1023 ;       BSBW    RMSGETFILNAM
                        04AE  1024 ;
                        04AE  1025 ; Input Parameters:
                        04AE  1026 ;
                        04AE  1027 ;       R3      Address of Buffer to return file name string
                        04AE  1028 ;       R4      Size of Buffer
                        04AE  1029 ;       R9      IFB address
                        04AE  1030 ;       R10     FWA address
                        04AE  1031 ;       R11     impure area address
                        04AE  1032 ;
                        04AE  1033 ; Implicit Inputs:
                        04AE  1034 ;
                        04AE  1035 ;       The current contents of the FWA.
                        04AE  1036 ;
                        04AE  1037 ; Outputs:
                        04AE  1038 ;
                        04AE  1039 ;       R0      Status code
                        04AE  1040 ;       R1-R3   Destroyed
                        04AE  1041 ;       R4      Return String Length
                        04AE  1042 ;
                        04AE  1043 ; Implicit Outputs:
                        04AE  1044 ;
                        04AE  1045 ;       None.
                        04AE  1046 ;
                        04AE  1047 ; Completion Codes:
                        04AE  1048 ;
                        04AE  1049 ;       Standard RMS completion codes, including SUC, ESA, ESS, RSA, RSS, NAM.
                        04AE  1050 ;
                        04AE  1051 ; Side Effects:
                        04AE  1052 ;
                        04AE  1053 ;       None.
                        04AE  1054 ;
                        04AE  1055 ;--
                        04AE  1056
                        04AE  1057 RMSGETFILNAM::
    01F8 8F     BB      04AE  1058          PUSHR   #^M<R3,R4,R5,R6,R7,R8>  ; Save registers
52    60 8F     9A      04B2  1059          MOVZBL  #NAM$C_BLN,R2           ; Allocate a fake nam block
    51   59     D0      04B6  1060          MOVL    R9,R1
       FB44'    30      04B9  1061          BSBW    RMSGETSPC
     05 50      E8      04BC  1062          BLBS    R0,10$                  ; Continue on error
    01F8 8F     BA      04BF  1063          POPR    #^M<R3,R4,R5,R6,R7,R8>  ; Restore registers
                05      04C3  1064          RSB
                        04C4  1065
    57   51     D0      04C4  1066 10$:     MOVL    R1,R7                   ; Save address of block
    53   8E     7D      04C7  1067          MOVQ    (SP)+,R3                ; restore R3,R4
01 A7    60 8F     90   04CA  1068          MOVB    #NAM$C_BLN,NAM$B_BLN(R7); Fill in block length
                        04CF  1069          SSB     #NAM$V_NOCONCEAL,NAM$B_NOP(R7) ; we need "real" physical device
```

```
                58   02 A7    9E  04D4  1070          MOVAB   NAM$B_RSS(R7),R8         ; Get RSS address
                   88   54     90  04D8  1071          MOVB    R4,(R8)+                ; Fill in RSS, point R8 at RSL
                        68     94  04DB  1072          CLRB    (R8)                    ; Initialize RSL
             56   0130 CA     7E  04DD  1073          MOVAQ   FWA$Q_DIR1(R10),R6      ; Copy from start of directory info
               7E   0A AA     9A  04E2  1074          MOVZBL  FWA$B_DIRTERM(R10),-(SP); Save directory terminator
          0A AA   5D 8F       90  04E6  1075          MOVB    #^A/]/,FWA$B_DIRTERM(R10), Force square brackets
                  F5'AF       9F  04EB  1076          PUSHAB  B^20$                   ; Set return address for COPY_DEVICE
                  0900 8F     BB  04EE  1077          PUSHR   #^M<R8,R11>             ; Push registers COPY_DEVICE pops
                     FBA3     31  04F2  1078          BRW     COPY_DEVICE             ; Go off and get name
                              04F5  1079
                              04F5  1080 ;
                              04F5  1081 ;       COPY_DIR will return here
                              04F5  1082 ;
                              04F5  1083
          0A AA      6E       90  04F5  1084 20$:     MOVB    (SP),FWA$B_DIRTERM(R10) ; Restore directory terminator
                     8E       D5  04F9  1085          TSTL    (SP)+                   ; Pop off stack
                58   03 A7     9A  04FB  1086          MOVZBL  NAM$B_RSL(R7),R8        ; Save return length
                   54   57     D0  04FF  1087          MOVL    R7,R4                   ; Return fake nam block
                   53   59     D0  0502  1088          MOVL    R9,R3
                52   60 8F     9A  0505  1089          MOVZBL  #NAM$C_BLN,R2
                     FAF4'     30  0509  1090          BSBW    RM$RETSPC
                   54   58     D0  050C  1091          MOVL    R8,R4                   ; Return file name length
                   01E0 8F     BA  050F  1092          POPR    #^M<R5,R6,R7,R8>        ; Restore registers
                               05  0513  1093          RSB
                              0514  1094
```

```
                        0514  1096          .SBTTL  RM$FILLNAM, Output Resultant File Name and other NAM Fields
                        0514  1097
                        0514  1098 ;+
                        0514  1099 ;
                        0514  1100 ; RM$FILLNAM --  subroutine to output resultant name string and other
                        0514  1101 ;                 nam fields.
                        0514  1102 ;
                        0514  1103 ;  Inputs:
                        0514  1104 ;       R11     impure area address
                        0514  1105 ;       R10     fwa address
                        0514  1106 ;       R9      ifab address
                        0514  1107 ;       R8      fab address
                        0514  1108 ;
                        0514  1109 ;  Outputs:
                        0514  1110 ;       R7      nam block address
                        0514  1111 ;       R0      status code
                        0514  1112 ;       R1-R6   destroyed
                        0514  1113 ;
                        0514  1114 ;-
                        0514  1115
                        0514  1116 EXPARGL:
                 04     0514  1117          .BYTE   NAM$L_RSA                ; arg list for rm$expstring
                        0515  1118          RMSERR_WORD     RST
                        0517  1119          RMSERR_WORD     RSS
                        0519  1120
                        0519  1121 RM$FILLNAM::
             57  28 A8  D0    051A  1122          MOVL    FAB$L_NAM(R8),R7         ; get name block addr
                    3B  13    051D  1123          BEQL    30$                      ; branch if no nam block
                        051F  1124
                    18  E0    051F  1125          BBS     #DEV$V_FOR,-             ; if the device is mounted foreign then
                 04 69          0521  1126                  IFB$L_PRIM_DEV(R9),5$    ; the directory is to be of null length
                 04 69  1C  E0  0523  1127          BBS     #DEV$V_RND,IFB$L_PRIM_DEV(R9),10$ ; branch if full directory dev
                        0527  1128
                        0527  1129 5$:      CSB     #FWA$V_DIR,(R10)                 ; clear DIR bit if not
                 5C  E6 AF  9E  052B  1130 10$:      MOVAB   EXPARGL,AP               ; arg list for rm$expstring
                      FACE  30  052F  1131          BSBW    RM$EXPSTRING             ; fill in resultant name string
                    28 50   E9  0532  1132          BLBC    R0,40$                   ; quit on error
                 25 6A  19  E0  0535  1133          BBS     #FWA$V_NODE,(R10),50$    ; branch if nodename was found
                      002B  30  0539  1134          BSBW    RM$WRITE_DVI             ; Write DVI field in NAM block
           24 A7  01F8 CA  D0  053C  1135          MOVL    FWA$T_FIBBUF+FIB$W_FID(R10),NAM$W_FID(R7); copy fid
           28 A7  01FC CA  B0  0542  1136          MOVW    FWA$T_FIBBUF+FIB$W_FID+4(R10),NAM$W_FID+4(R7); copy fid
                  01FE CA  B5  0548  1137          TSTW    FWA$T_FIBBUF+FIB$W_DID(R10); any did?
                       0C  13  054C  1138          BEQL    30$                      ; branch if yes
           2A A7  01FE CA  D0  054E  1139          MOVL    FWA$T_FIBBUF+FIB$W_DID(R10),NAM$W_DID(R7); copy did
           2E A7  0202 CA  B0  0554  1140          MOVW    FWA$T_FIBBUF+FIB$W_DID+4(R10),NAM$W_DID+4(R7); copy did
                        055A  1141 30$:     RMSSUC
                    05  05  055D  1142 40$:     RSB
14 A7  1C   00  6E  00  2C  055E  1143 50$:     MOVC5   #0,(SP),#0,#<16+6+6>,NAM$T_DVI(R7)
                    F3  11  0565  1144          BRB     30$                      ; zero dvi, fid, and did fields
                        0567  1145
```

```
                     0567  1147          .SBTTL  RM$WRITE_DVI, Write DVI field of NAM block
                     0567  1148 ;++
                     0567  1149 ;
                     0567  1150 ;       This routine writes the DVI field of the NAM block from
                     0567  1151 ;       the information stored in the FWA.
                     0567  1152 ;       The DVI field is the PPF device string if this is a PPF;
                     0567  1153 ;       else it is a canoical device name of the form:
                     0567  1154 ;               _ddcuuuu
                     0567  1155 ;
                     0567  1156 ; Inputs:
                     0567  1157 ;
                     0567  1158 ;       R10 = Address of FWA
                     0567  1159 ;       R7 = Address of user's NAM block
                     0567  1160 ;
                     0567  1161 ; Outputs:
                     0567  1162 ;
                     0567  1163 ;       R0-R5 destroyed.
                     0567  1164 ;       The DVI field is written.
                     0567  1165 ;
                     0567  1166 ;--
                     0567  1167
                     0567  1168 RM$WRITE_DVI::
  53   14 A7   9E    0567  1169          MOVAB   NAM$T_DVI(R7),R3        ; ptr to destination
       0C AA   95    056B  1170          TSTB    FWA$B_ESCFLG(R10)       ; is this a PPF file?
          07   13    056E  1171          BEQL    10$                     ; branch if not
                     0570  1172
                     0570  1173 ;
                     0570  1174 ; If PPF, put special name in DVI
                     0570  1175 ;
                     0570  1176
  50   00E0 CA   7D  0570  1177          MOVQ    FWA$Q_DEVICE(R10),R0    ; length/addr of name
          10   11    0575  1178          BRB     20$
                     0577  1179
  50   0190 CA   7D  0577  1180 10$:     MOVQ    FWA$Q_SHRFIL(R10),R0    ; length/addr of string
05 008C C9   06   E1  057C  1181          BBC     #DEV$V_SPL,IFB$L_AS_DEV(R9),20$ ; ok if not spooled
  50   01A0 CA   7D  0582  1182          MOVQ    FWA$Q_AS_SHRFIL(R10),R0 ; use secondary name if spooled
                     0587  1183
       0F   50   B1  0587  1184 20$:     CMPW    R0,#15                  ; room for count field and name?
            08   1A  058A  1185          BGTRU   30$                     ; nope
       83   50   90  058C  1186          MOVB    R0,(R3)+                ; stuff count (for counted string)
  63   61   50   28  058F  1187          MOVC3   R0,(R1),(R3)            ; copy name past count field
                 05  0593  1188          RSB
                     0594  1189
            63   94  0594  1190 30$:     CLRB    (R3)                    ; say no DVI field
                 05  0596  1191          RSB
```

```
                     0597  1193          .SBTTL  RMS$CHKNAM, Check NAM Block Validity
                     0597  1194
                     0597  1195 ;++
                     0597  1196 ;
                     0597  1197 ; Subroutine to verify that R7 really points to an accessible Name Block.
                     0597  1198 ;
                     0597  1199 ; Inputs:
                     0597  1200 ;
                     0597  1201 ;       R7      NAM block address
                     0597  1202 ;
                     0597  1203 ; Outputs:
                     0597  1204 ;
                     0597  1205 ;       If an error occurs, R0 is set to the error code.
                     0597  1206 ;
                     0597  1207 ;--
                     0597  1208
                     0597  1209
                     0597  1210          ASSUME  NAM$B_BLN       EQ      NAM$B_BID+1
                     0597  1211
                     0597  1212 RMS$CHKNAM::
                     0597  1213          IFNORD  #NAM$B_BLN+1,(R7),ERRNAM; Branch if BID and BLN not readable
          02   67   91  059D  1214          CMPB    NAM$B_BID(R7),#NAM$C_BID; Right ID?
               13   12  05A0  1215          BNEQ    ERRNAM                  ; Branch if not
     50   01  A7   9A  05A2  1216          MOVZBL  NAM$B_BLN(R7),R0        ; Get user specified size
          38   50   91  05A6  1217          CMPB    R0,#NAM$C_BLN_V2        ; Long enough (version 2 size)?
               0A   1F  05A9  1218          BLSSU   ERRNAM                  ; Branch if not long enough
                    05AB  1219          IFNOWRT R0,(R7),ERRNAM          ; Branch if not writeable
                    05B1  1220          RMSSUC                          ; Success
          05        05B4  1221          RSB                             ;
                    05B5  1222
                    05B5  1223 ERRNAM:
                    05B5  1224          RMSERR  NAM                     ; Show problem
          05        05BA  1225          RSB                             ;
                    05BB  1226
                    05BB  1227          .END
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
$$T1                            = 00000000
COPY_DEVICE                       00000098 R      01
COPY_DIR                          000000C6 R      01
COPY_DVI                          000000AD R      01
COPY_NAME                         0000019E R      01
COPY_NODE                         00000232 R      01
COPY_QUOTED                       0000032B R      01
COPY_REMRESULT                    0000038F R      01
DEV$V_FOR                       = 00000018
DEV$V_RND                       = 0000001C
DEV$V_SPL                       = 00000006
ERRDME                            00000323 R      01
ERRNAM                            000005B5 R      01
ERRSA                             00000202 R      01
ERRSS                             0000022E R      01
EXIT_ERR                          000001FD R      01
EXIT_SUC                          000001FA R      01
EXPARGL                           00000514 R      01
EX_SUC                            000003B8 R      01
FAB$L_FOP                       = 00000004
FAB$L_NAM                       = 00000028
FAB$V_NAM                       = 00000018
FIB$W_DID                       = 0000000A
FIB$W_FID                       = 00000004
FOP                             = 00000020
FSCB$V_ACS                      = 00000012
FSCB$V_ELIPS                    = 00000010
FWA$B_DIRTERM                   = 0000000A
FWA$B_ESCFLG                    = 0000000C
FWA$B_ROOTERM                   = 0000000B
FWA$B_SUBNODCNT                 = 0000002F
FWA$Q_AS_SHRFIL                 = 0C0001A0
FWA$Q_CDIR1                     = 000000F0
FWA$Q_CDIR8                     = 00000128
FWA$Q_CONCEAL_DEV               = 000000E8
FWA$Q_DEVICE                    = 000000E0
FWA$Q_DIR1                      = 00000130
FWA$Q_NAME                      = 00000170
FWA$Q_NODE1                     = 000001B4
FWA$Q_QUOTED                    = 00000170
FWA$Q_SHRFIL                    = 00000190
FWA$S_DIR_LVLS                  = 00000003
FWA$T_FIBBUF                    = 000001F4
FWA$V_CONCEAL_DEV               = 00000039
FWA$V_DEVICE                    = 0000000F
FWA$V_DIR                       = 0C0000E
FWA$V_DIR_LVLS                  = 0000001D
FWA$V_EXP_ROOT                  = 0000003C
FWA$V_GRPABR                    = 0000001B
FWA$V_NAM_DVI                   = 00000005
FWA$V_NETSTR                    = 00000032
FWA$V_NODE                      = 00000019
FWA$V_QUOTED                    = 0000001A
FWA$V_REMRESULT                 = 00000035
FWA$V_ROOT_DIR                  = 0000003A
FWA$V_SL_PASS                   = 00000002
FWA$V_WILD_DIR                  = 0000001C
HOR_TAB                         = 00000009
IFB$B_MODE                      = 0000000A
IFB$L_AS_DEV                    = 0000008C
IFB$L_NWA_PTR                   = 0000003C
IFB$L_PRIM_DEV                  = 00000000
LOG$C_PROCESS                   = 00000002
MOVE_CHAR                         0000021A R      01
MOVE_NEXT                         00000207 R      01
NAM$B_BID                       = 00000000
NAM$B_BLN                       = 00000001
NAM$B_DEV                       = 00000039
NAM$B_DIR                       = 0000003A
NAM$B_ESL                       = 0000000B
NAM$B_ESS                       = 0000000A
NAM$B_NAME                      = 0000003B
NAM$B_NODE                      = 00000038
NAM$B_NOP                       = 00000008
NAM$B_RFS                       = 00000009
NAM$B_RSL                       = 00000003
NAM$B_RSS                       = 00000002
NAM$B_TYPE                      = 0000003C
NAM$B_VER                       = 0000003D
NAM$C_BID                       = 00000002
NAM$C_BLN                       = 00000060
NAM$C_BLN_V2                    = 00000038
NAM$L_DEV                       = 00000044
NAM$L_DIR                       = 00000048
NAM$L_ESA                       = 0000000C
NAM$L_FNB                       = 00000034
NAM$L_NAME                      = 0000004C
NAM$L_NODE                      = 00000040
NAM$L_RSA                       = 00000004
NAM$L_TYPE                      = 00000050
NAM$L_VER                       = 00000054
NAM$S_DIR_LVLS                  = 00000003
NAM$T_DVI                       = 00000014
NAM$V_DIR_LVLS                  = 00000015
NAM$V_NOCONCEAL                 = 00000004
NAM$V_PWD                       = 00000000
NAM$V_WILD_SFD1                 = 00000019
NAM$W_DID                       = 0000002A
NAM$W_FID                       = 00000024
NWA$B_ALLXABCNT                   0000011C
NWA$B_DAP_RAC                     000000C9
NWA$B_FILESYS                     000000C5
NWA$B_KEYXABCNT                   0000011D
NWA$B_NETSTRSIZ                   0000016F
NWA$B_NODBUFSIZ                   00000168
NWA$B_ORG                         000000C6
NWA$B_OSTYPE                      000000C4
NWA$B_RFM                         000000C7
NWA$B_RMS_RAC                     000000C8
```

```
NWA$C_BLN                       00000800
NWA$K_BLN                       00000800
NWA$L_ALLXABADR                 00000100
NWA$L_DATXABADR                 00000104
NWA$L_DEV                       000000C0
NWA$L_FHCXABADR                 00000108
NWA$L_KEYXABADR                 0000010C
NWA$L_MSG_MASK                  000000D4
NWA$L_PROXABADR                 00000110
NWA$L_RDTXABADR                 00000114
NWA$L_SAVE_FLGS                 00000128
NWA$L_SUMXABADR                 00000118
NWA$L_THREAD                    000000FC
NWA$L_XLTATTR                   00000238
NWA$L_XLTBUFFLG                 0000022C
NWA$L_XLTCNT                    00000228
NWA$L_XLTMAXINDX                00000234
NWA$L_XLTSIZ                    00000230
NWA$Q_ACS                       00000244
NWA$Q_BIGBUF                    00000170
NWA$Q_BLD                       000000F0
NWA$Q_FLG                       00000000
NWA$Q_INODE                     0000025C
NWA$Q_IOSB                      000000D8
NWA$Q_LNODE                     00000160
NWA$Q_LOGNAME                   0000023C
NWA$Q_NCB                       00000264
NWA$Q_RCV                       000000E0
NWA$Q_SAVE_DESC                 00000120
NWA$Q_XLTBUF1                   0000024C
NWA$Q_XLTBUF2                   00000254
NWA$Q_XMT                       000000E8
NWA$T_ACSBUF                    0000026C
NWA$T_AUXBUF                    000005E0
NWA$T_DAP                       00000000
NWA$T_INODEBUF                  000004AC
NWA$T_ITM_ATTR                  00000200
NWA$T_ITM_END                   00000224
NWA$T_ITM_LST                   00000200
NWA$T_ITM_MAXINDX               00000218
NWA$T_ITM_STRING                0000020C
NWA$T_NCBBUF                    0000052C
NWA$T_NODEBUF                   00000169
NWA$T_RCVBUF                    000001A0
NWA$T_SCAN                      00000100
NWA$T_TEMP                      00000120
NWA$T_XLTBUF1                   000002AC
NWA$T_XLTBUF2                   000003AC
NWA$T_XMTBUF                    000003C0
NWA$W_BUILD                     000000D2
NWA$W_DAPBUFSIZ                 000000CA
NWA$W_DIR_OFF                   000000CC
NWA$W_DISPLAY                   000000D0
NWA$W_FIL_OFF                   000000CE
NWA$W_JNLXABJOP                 0000011E
PARSE_REMRESULT                 000003BD R      01
POPPC                           00000228 R      01
PSL$C_USER                    = 00000003
REM_DEV                         000003F6 R      01
REM_DIR                         00000412 R      01
REM_NAME                        00000435 R      01
REM_NODE                        000003CB R      01
REM_QUOTED                      000003E7 R      01
REM_TYPE                        0000044D R      01
REM_VER                         00000466 R      01
RMS$CHKNAM                      00000597 RG     01
RMS$EXPSTRING                   00000000 RG     01
RMS$FILLNAM                     00000519 RG     01
RMS$GETFILNAM                   000004AE RG     01
RMS$GETSPC                      ********    X   01
RMS$RETSPC                      ********    X   01
RMS$WRITE_DVI                   00000567 RG     01
RMS$_DME                      = 000184D4
RMS$_NAM                      = 000185DC
RMS$_RSS                      = 00018694
RMS$_RST                      = 0001869C
SPACE                         = 00000020
SUC                             00000470 R      01
SYS$CRELOG                      ********  GX    01
TWO_COLONS                      000003BB R      01
UPDATE_FNB                      00000475 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
RMSRMS0	000005BB (1467.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000800 (2048.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.11	00:00:00.56
Command processing	132	00:00:00.84	00:00:06.36
Pass 1	435	00:00:17.28	00:00:50.73
Symbol table sort	0	00:00:02.30	00:00:04.91
Pass 2	216	00:00:04.11	00:00:14.99
Symbol table output	24	00:00:00.15	00:00:00.36
Psect synopsis output	2	00:00:00.02	00:00:00.09
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	848	00:00:24.81	00:01:18.01

The working set limit was 1950 pages.
97445 bytes (191 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1632 non-local and 62 local symbols.
1227 source lines were read in Pass 1, producing 16 object records in Pass 2.
33 pages of virtual memory were used to define 32 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	14
_255DUA28:[SYS.OBJ]LIB.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	28

1802 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

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

RM0RECLCK
LIS

RM0SET
LIS

RM0SCAN
LIS

RM0PRFLNM
LIS

RM0RCLCK2
LIS

RM0RABCHK
LIS

RM0RELEAS
LIS

RM0NAMSTR
LIS