```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMMMMM    MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM   SSS
RRR         RRR  MMM   MMM   MMM   SSS
RRR         RRR  MMM   MMM   MMM   SSS
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  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

FILEID**RMSOPARSE

```
RRRRRRRR    MM      MM    SSSSSSSS    000000    PPPPPPPP      AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
RRRRRRRR    MM      MM    SSSSSSSS    000000    PPPPPPPP      AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
RR      RR  MMMM  MMMM  SS          00      00  PP      PP  AA      AA  RR      RR  SS            EE
RR      RR  MMMM  MMMM  SS          00      00  PP      PP  AA      AA  RR      RR  SS            EE
RR      RR  MM  MM  MM  SS          00    0000  PP      PP  AA      AA  RR      RR  SS            EE
RR      RR  MM  MM  MM  SS          00    000C  PP      PP  AA      AA  RR      RR  SS            EE
RRRRRRRR    MM      MM    SSSSSS    00  00  00  PPPPPPPP    AA      AA  RRRRRRRR      SSSSSS      EEEEEEEE
RRRRRRRR    MM      MM    SSSSSS    00  00  00  PPPPPPPP    AA      AA  RRRRRRRR      SSSSSS      EEEEEEEE
RR  RR      MM      MM          SS  0000    00  PP          AAAAAAAAAA  RR  RR              SS  EE
RR  RR      MM      MM          SS  0000    00  PP          AAAAAAAAAA  RR  RR              SS  EE
RR    RR    MM      MM          SS  00      00  PP          AA      AA  RR    RR            SS  EE                ....
RR    RR    MM      MM          SS  00      00  PP          AA      AA  RR    RR            SS  EE                ....
RR      RR  MM      MM  SSSSSSSS      000000    PP          AA      AA  RR      RR  SSSSSSSS    EEEEEEEEEE        ....
RR      RR  MM      MM  SSSSSSSS      000000    PP          AA      AA  RR      RR  SSSSSSSS    EEEEEEEEEE        ....

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          $BEGIN  RMSOPARSE,000,RM$RMS,<PARSE FILE SPECIFICATION>
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 ; Facility: rms32
0000     30 ;
0000     31 ; Abstract:
0000     32 ;                    This is the highest level routine to perform the $parse function
0000     33 ;
0000     34 ; Environment:
0000     35 ;                    VAX/VMS
0000     36 ;
0000     37 ; Author:
0000     38 ;                    Tim Halvorsen   AUG-1979
0000     39 ;
0000     40 ; Modified By:
0000     41 ;
0000     42 ;           V03-019 RAS0287         Ron Schaefer            3-Apr-1984
0000     43 ;                   Minor changes to searchlist and wildcard processing:
0000     44 ;                   1.  Prevent searchlist continuation if the error
0000     45 ;                       is not a continuable one;
0000     46 ;                   2.  Don't force context to be saved for ANY wildcard,
0000     47 ;                       only for searchlists, network or wildcard directories.
0000     48 ;
0000     49 ;           V03-018 RAS0251         Ron Schaefer            9-Feb-1984
0000     50 ;                   Add support for special parsing flag NAM$V_SLPARSE
0000     51 ;                   for BACKUP.   This flag causes multiple $PARSEs of
0000     52 ;                   a searchlist filespec to return successive searchlist
0000     53 ;                   elements.  Reorganize the cleanup of old saved context.
0000     54 ;
0000     55 ;           V03-017 RAS0241,RAS0242 Ron Schaefer            23-Jan-1984
0000     56 ;                   Fix bugcheck in saved-context searches caused by
0000     57 ;                   a bogus null directory spec.
0000     58 ;                   Fix bugchecks caused by lack of a fwa (valid R10)
0000     59 ;                   on calls to RM$PARSE_FILE.
0000     60 ;                   Return DNF error if no directory is found by RM$NEXTDIR.
0000     61 ;
0000     62 ;           V03-016 RAS0231         Ron Schaefer            9-Jan-1984
0000     63 ;                   Support NAM$V_SYNCHK by not doing the $ASSIGN nor
0000     64 ;                   the directory lookup.  Use FWA$V_SYNTAX_CHK flag.
0000     65 ;                   Don't save context in this case either so subsequent
0000     66 ;                   $SEARCHs will not work.
0000     67 ;
0000     68 ;           V03-015 RAS0219         Ron Schaefer            8-Dec-1983
0000     69 ;                   Process errors from RM$INIT_SWB.
0000     70 ;
0000     71 ;           V03-014 RAS0218         Ron Schaefer            5-Dec-1983
0000     72 ;                   Make node names work as search list elements.
0000     73 ;
0000     74 ;           V03-013 RAS0209         Ron Schaefer            4-Nov-1983
0000     75 ;                   Clean-up returned device characteristics by calling
0000     76 ;                   a central routine  RM$RET_DEV_CHAR.
0000     77 ;
0000     78 ;           V03-012 RAS0201         Ron Schaefer            17-Oct-1983
0000     79 ;                   Correct calls to RM$PARSE_FILE to account for the fact
0000     80 ;                   that it does NOT necessarily preserve R7.
0000     81 ;
0000     82 ;           V03-011 KBT0565         Keith B. Thompson       26-Jul-1983
0000     83 ;                   Save context if any wild field present
0000     84 ;
```

```
0000     85 ;       V03-010 KBT0530         Keith B. Thompson       31-May-1983
0000     86 ;               Add search list support
0000     87 ;
0000     88 ;       V03-009 KBT0519         Keith B. Thompson       23-May-1983
0000     89 ;               RM$XPFN moved so change ref to JSB
0000     90 ;
0000     91 ;       V03-008 LJA0062         Laurie J. Anderson      23-Feb-1983
0000     92 ;               Move RM$WRITE_DVI to RMONAMSTR.MAR which is where it belongs.
0000     93 ;
0000     94 ;       V03-007 KBT0430         Keith B. Thompson       3-Dec-1982
0000     95 ;               Change the way the device name is returned by write_dvi
0000     96 ;
0000     97 ;       V03-006 KBT0427         Keith B. Thompson       2-Dec-1982
0000     98 ;               Fix rm$write_dvi to use new type of name in shrfilbuf
0000     99 ;
0000    100 ;       V03-005 KBT0405         Keith B. Thompson       30-Nov-1982
0000    101 ;               Change fwa$t_shrfildev to fwa$t_shrfilbuf
0000    102 ;
0000    103 ;       V03-004 RAS0103         Ron Schaefer            19-Nov-1982
0000    104 ;               Correct saving of the caller's access mode so that
0000    105 ;               exits via RM$EX_NOSTR have the caller's mode in R7.
0000    106 ;
0000    107 ;       V03-003 DMW4003         DMWalp                  2-Sep-1982
0000    108 ;               Added code so that RM$FABCHK was not called twice;
0000    109 ;               it was called once directly and a second time via RM$FSETI
0000    110 ;
0000    111 ;       V03-002 KBT0188         Keith B. Thompson       23-Aug-1982
0000    112 ;               Reorganize psects and rename entry point to single '$'
0000    113 ;
0000    114 ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    115 ;               Added $PSLDEF.
0000    116 ;
0000    117 ;--
```

```
                     0000   119          .SBTTL  DEFINITIONS
                     0000   120
                     0000   121 ;
                     0000   122 ;       symbol definitions
                     0000   123 ;
                     0000   124
                     0000   125          $DEVDEF                         ; device characteristics
                     0000   126          $FABDEF                         ; fab definitions
                     0000   127          $FIBDEF                         ; fib definitions
                     0000   128          $FIDDEF                         ; fid definitions
                     0000   129          $FWADEF                         ; fwa definitions
                     0000   130          $IFBDEF                         ; ifab definitions
                     0000   131          $IMPDEF                         ; impure area definitions
                     0000   132          $NAMDEF                         ; nam definitions
                     0000   133          $PIODEF                         ; i/o control page definitions
                     0000   134          $PSLDEF                         ; program status longword definitions
                     0000   135          $RMSDEF                         ; RMS error codes
                     0000   136          $SSDEF                          ; system error codes
                     0000   137
                     0000   138 ;
                     0000   139 ; Symbols
                     0000   140 ;
                     0000   141
                     0000   142 ;
                     0000   143 ; Stack offsets for saved context (RM$PARSE_FILE)
                     0000   144 ;
                     0000   145
00000000             0000   146          ESL     = 0     ;       NAM$B_ESL
00000001             0000   147          RSL     = 1     ;       NAM$B_RSL
00000002             0000   148          FNB     = 2     ;       NAM$L_FNB
00000006             0000   149          STV     = 6     ;       FAB$L_STV
0000000A             0000   150          ERR     = 10    ;       R0
                     0000   151
0000000E             0000   152          STACK_SIZE = 14 ;       Size of stack to allocate
                     0000   153
```

```
                     0000   155          .SBTTL  RMS$PARSE, Initiate Wildcard Sequence
                     0000   156
                     0000   157 ;--
                     0000   158 ;
                     0000   159 ; PARSE
                     0000   160 ;       RMS$PARSE
                     0000   161 ;
                     0000   162 ;       This routine initiates wildcarding within rms.
                     0000   163 ;       it allocates swb and fwa buffers to handle context
                     0000   164 ;       while traversing the directory tree.  They remain
                     0000   165 ;       allocated until the wildcard sequence is terminated
                     0000   166 ;       via either another parse or nmf error condition.
                     0000   167 ;
                     0000   168 ; inputs:
                     0000   169 ;
                     0000   170 ;       file spec from fab
                     0000   171 ;
                     0000   172 ; outputs:
                     0000   173 ;
                     0000   174 ;       r0 = status
                     0000   175 ;       expanded name string set
                     0000   176 ;       did set in nam block if non-wild directory
                     0000   177 ;
                     0000   178 ;--
                     0000   179
                     0000   180          $ENTRY  RMS$PARSE
                     0000   181          $TSTPT  PARSE
                     0006   182
                     0006   183 ;
                     0006   184 ; If another wildcard sequence was in progress using this
                     0006   185 ; ifab, then cleanup the previous one.
                     0006   186 ;
                     0006   187
                FFF7' 30  0006   188          BSBW    RM$FABCHK                       ; check fab validity returns only if ok
                          0009   189                                                  ; r11 = impure area
                          0009   190                                                  ; r8 = fab address
                          0009   191                                                  ; r7 = caller's access mode
                   52 12  0009   192          BNEQ    10$                             ; error if IFI non-zero
                   57 DD  000B   193          PUSHL   R7                              ; save caller's mode
             57 28 A8 D0  000D   194          MOVL    FAB$L_NAM(R8),R7                ; get nam address
                FFEC' 30  0011   195          BSBW    RM$CHKNAM                       ; check nam validity
                57 56 D0  0014   196          MOVL    R7,R6                           ; copy NAM block address
                8E 57 D0  0017   197          MOVL    (SP)+,R7                        ; restore caller's mode
             58 50 E9     001A   198          BLBC    R0,40$                          ; branch if invalid (no error)
                          001D   199
             59 30 A6 D0  001D   200          MOVL    NAM$L_WCC(R6),R9                ; get ifi of previous ifab
                   52 13  0021   201          BEQL    40$                             ; branch if none
                          0023   202
                32 A6 B3  0023   203          BITW    NAM$L_WCC+2(R6),-               ; check that there are no spurious
                3FFE 8F   0026   204                  #^C<<NAM$M_SVCTX!-              ; bits set in NAM$L_WCC other than
                          0029   205                       NAM$M_SRCHNMF!-            ; the save context bit, the IFI bit,
                          0029   206                       NAM$M_IFI>@-16>             ; or the search NMF bit
                   3A 12  0029   207          BNEQ    20$                             ; if so, error in wcc param
                          002B   208
             46 59 10 E5  002B   209          BBCC    #NAM$V_IFI,R9,40$               ; branch if not 'search' ifi
                          002F   210
                50 06 D0  002F   211          MOVL    #IMP$L_IFABTBL/4,R0             ; ifab table longword offset
```

```
                FFCB'  30  0032   212          BSBW    RM$GTIADR                ; get ifab address
                  3E   13  0035   213          BEQL    40$                      ; branch if illegal ifi
          3A 69   39   E1  0037   214          BBC     #IFB$V_SEARCH,(R9),40$   ; branch if not ours
                               003B   215
                               003B   216 ;
                               003B   217 ; See if the user wants to go on to the next searchlist element.
                               003B   218 ;
                               003B   219
       02 A8   30 A6   B0  003B   220          MOVW    NAM$L_WCC(R6),FAB$W_IFI(R8); set FAB as busy
                FFBD'  30  0040   221          BSBW    RM$FSET_ALT              ; set up remaining state
          5A   38 A9   D0  0043   222          MOVL    IFB$L_FWA_PTR(R9),R10    ; get fwa ptr
                  24   13  0047   223          BEQL    30$                      ; start anew if none
                  05   E1  0049   224          BBC     #NAM$V_SLPARSE,-         ; see if special parse request
            1F 08 A6           004B   225                  NAM$B_NOP(R6),30$        ; start anew if not
                               004E   226          RMSERR  NMF                      ; assume at end of list
          44 6A   38   E1  0053   227          BBC     #FWA$V_SLPRESENT,(R10),EXIT      ; no more files if not a list
                               0057   228          SSB     #FWA$V_SL_PASS,(R10)     ; doing searchlist processing
                  1D   11  005B   229          BRB     50$                      ; and do it
                               005D   230
                               005D   231 ;
                               005D   232 ; error returns
                               005D   233 ;
                               005D   234
                               005D   235 10$:     RMSERR  IFI                      ; invalid ifi (must be zero)
                FF9B'  31  0062   236          BRW     RM$EX_NOSTR              ; exit without ifab
                               0065   237
                               0065   238 20$:     RMSERR  WCC                      ; error in wcc value
                FF93'  31  006A   239          BRW     RM$EX_NOSTR              ; exit without ifab
                               006D   240
                               006D   241 ;
                               006D   242 ; Cleaning up the previous context (IFAB, FWA, etc...) save the current
                               006D   243 ; Note that during the cleanup, stalling may take place.
                               006D   244 ;
                               006D   245
                  5C   DD  006D   246 30$:     PUSHL   AP                       ; save ap over cleanup call
              FF8E'  30  006F   247          BSBW    RM$CLEANUP               ; terminate previous sequence
                5C 8ED0  0072   248          POPL    AP                       ; restore ap
                               0075   249
                               0075   250 ;
                               0075   251 ; Allocate an ifab for internal context
                               0075   252 ;
                               0075   253
              FF88'  30  0075   254 40$:     BSBW    RM$FSETI_ALT             ; allocate ifab/ifi
                               0078   255
                               0078   256 ;
                               0078   257 ; Parse the name, assign channel, and fill in nam fields
                               0078   258 ;
                               0078   259
                  5A   D4  0078   260          CLRL    R10                      ; signal initial call
                  53   10  007A   261 50$:     BSBB    RM$PARSE_FILE            ; do the heavy work
          1C 50   E9  007C   262          BLBC    R0,EXIT                  ; branch if error
                  36   E0  007F   263          BBS     #FWA$V_SYNTAX_CHK,-      ; exit cleaning up if syntax check only
          18 6A           0081   264                  (R10),EXIT
                               0083   265
                               0083   266 ;
                               0083   267 ; Fill in (in the FAB) the primary and secondary device characteristics.
                               0083   268 ;
```

```
                     0083   269
          FF7A'  30  0083   270          BSBW    RM$RET_DEV_CHAR         ; return characteristics
                     0086   271
                     0086   272 ;
                     0086   273 ; If wcc was -1 on entry, then set a "save context" flag
                     0086   274 ; as the top bit of the ifi and save the ifi of the ifab/fwa
                     0086   275 ; for the current context in the nam block so we can pick
                     0086   276 ; it up later when the user calls search.  the save context flag
                     0086   277 ; enables keeping context around over parse/search calls
                     0086   278 ; and causes directory files to be read when possible.
                     0086   279 ;
                     0086   280
57   28 A8   D0      0086   281          MOVL    FAB$L_NAM(R8),R7        ; get NAM block
       FF73'  30     008A   282          BSBW    RM$CHKNAM               ; check if nam valid
     0E 50    E8     008D   283          BLBS    R0,SVCTX                ; branch if ok
   6A   1C    E0     0090   284          BBS     #FWA$V_WILD_DIR,(R10),- ; if wild dir, must have nam block
          07         0093   285          EXIT
03 6A   38    E0     0094   286          BBS     #FWA$V_SLPRESENT,(R10),EXIT     ; or if search list
                     0098   287          RMSSUC                          ; else, set success
          FF62'  31  009B   288 EXIT:     BRW     RM$CLSCU                ; cleanup ifab,etc and exit with status
                     009E   289                                          ; and without saving context
                     009E   290
16 6A   19    E0     009E   291 SVCTX:   BBS     #FWA$V_NODE,(R10),70$   ; always keep context for networks
12 6A   38    E0     00A2   292          BBS     #FWA$V_SLPRESENT,(R10),70$      ; or if search list
          1C    E1     00A6   293          BBC     #DEV$V_RND,-            ; never keep context for devices with
        F1 69          00A8   294                  IFB$L_PRIM_DEV(R9),EXIT ; nonrandom primary characteristics
          06    E0     00AA   295          BBS     #DEV$V_SPL,-            ; never keep context for devices
EB 008C  C9            00AC   296                  IFB$L_AS_DEV(R9),EXIT   ; that are spooled
04 6A   1C    E0     00B0   297          BBS     #FWA$V_WILD_DIR,(R10),70$; if wild directories, keep context
E3 69   39    E1     00B4   298          BBC     #IFB$V_SEARCH,(R9),EXIT ; cleanup if svctx not requested
                     00B8   299
     02 A8   3C      00B8   300 70$:     MOVZWL  FAB$W_IFI(R8),-
     30 A7          00BB   301                  NAM$L_WCC(R7)           ; save ifi of current context
                     00BD   302          SSB     #IFB$V_SEARCH,(R9)      ; mark as search-type ifab
                     00C1   303          SSB     #NAM$V_IFI,NAM$L_WCC(R7); bit 16 set to indicate ifi, not wcc
     02 A8   B4      00C6   304          CLRW    FAB$W_IFI(R8)           ; mbz for subsequent operations on fab
     30 AA   D4      00C9   305          CLRL    FWA$L_DIRBDB(R10)       ; init directory bdb address
      FF31'  31      00CC   306          BRW     RM$EXSUC                ; exit with success -- leave ifab alone
```

```
                     00CF    308          .SBTTL  RM$PARSE_FILE, Parse a File Specification
                     00CF    309
                     00CF    310 ;--
                     00CF    311 ;
                     00CF    312 ; RM$PARSE_FILE
                     00CF    313 ;
                     00CF    314 ;       This routine parses the file specification and sets up
                     00CF    315 ;       the channel and did for the file.  If this routine is called
                     00CF    316 ;       for a search list operation and there are no more search list
                     00CF    317 ;       elements to parse R0, FAB$L_STV, NAM$L_FNB, NAM$B_ESL and NAM$B_RSL
                     00CF    318 ;       (if any) are NOT affected.
                     00CF    319 ;
                     00CF    320 ;       RMS$RENAME calls this routine twice for each file specification
                     00CF    321 ;       If the channel is already assigned, then the did must not be set.
                     00CF    322 ;
                     00CF    323 ;       RMS$SEARCH calls this routine when ever it gets a FNF or NMF error.
                     00CF    324 ;       It sets FWA$V_SL_PASS in order to look for a new file spec from
                     00CF    325 ;       a search list.
                     00CF    326 ;
                     00CF    327 ; Inputs:
                     00CF    328 ;
                     00CF    329 ;       R8      = fab address
                     00CF    330 ;       R9      = ifab address
                     00CF    331 ;       R10     = fwa addr (if search list) or 0 (if not)
                     00CF    332 ;       R11     = impure area
                     00CF    333 ;       R0      = input error status (if FWA$V_SL_PASS set)
                     00CF    334 ;       FAB$L_STV =                                 ''
                     00CF    335 ;       NAM$L_FNB =                                 ''
                     00CF    336 ;       NAM$B_ESL =                                 ''
                     00CF    337 ;       NAM$B_RSL =
                     00CF    338 ;
                     00CF    339 ; Outputs:
                     00CF    340 ;
                     00CF    341 ;       R0      = status (see explination above)
                     00CF    342 ;       R10     = fwa address
                     00CF    343 ;
                     00CF    344 ;       IFB$V_SEARCH is set if the user requested context to be saved
                     00CF    345 ;
                     00CF    346 ;       Registers r1-r7,ap are destroyed, device characteristics if PPF
                     00CF    347 ;
                     00CF    348 ;--
                     00CF    349
                     00CF    350 RM$PARSE_FILE::
                     00CF    351
                     00CF    352 ;
                     00CF    353 ; Make room on stack to save error codes and name block string lengths
                     00CF    354 ;
                     00CF    355 ;       ERR(SP) =>      R0                  error code
                     00CF    356 ;       STV(SP) =>      FAB$L_STV(R8)
                     00CF    357 ;       FNB(SP) =>      NAM$L_FNB
                     00CF    358 ;       RSL(SP) =>      NAM$B_RSL
                     00CF    359 ;       ESL(SP) =>      NAM$B_ESL
                     00CF    360 ;
                     00CF    361
5E   0E   C2         00CF    362          SUBL2   #STACK_SIZE,SP                  ; adjust stack
     5A   D5         00D2    363          TSTL    R10                             ; any fwa?
     55   13         00D4    364          BEQL    PRS                             ; nope so parse as is
```

```
                     00D6   365
                     00D6   366 ;
                     00D6   367 ; If this is a search list operation see if a channel was assigned, if so
                     00D6   368 ; deassign it
                     00D6   369 ;
                     00D6   370
    51 6A   02   E1  00D6   371 LOOP:   BBC     #FWA$V_SL_PASS,(R10),PRS        ; is this a search list pass
                     00DA   372
                     00DA   373 ;
                     00DA   374 ; See if the input error is one that allows for continuation
                     00DA   375 ;
                     00DA   376
      51   00'  D0  00DA   377          MOVL    S^#<RM$SLIST_ERR_CNT/2>,R1      ; get number of errs to check
FFFE'CF41    50  B1  00DD   378 10$:     CMPW    R0,W^RM$SLIST_ERRS-2[R1]        ; continue from this err?
          06   13  00E3   379          BEQL    20$                             ; yes
        F5 51  F5  00E5   380          SOBGTR  R1,10$                          ; try another
         017E  31  00E8   381          BRW     PREXIT                          ; return previous input status
                     00EB   382
    0A AE   50   D0  00EB   383 20$:     MOVL    R0,ERR(SP)                      ; save error status
 06 AE   0C A8   D0  00EF   384          MOVL    FAB$L_STV(R8),STV(SP)           ; save stv secondary code
    0C A8   D4  00F4   385          CLRL    FAB$L_STV(R8)                   ; zero to avoid confusion
 57 28 A8   D0  00F7   386          MOVL    FAB$L_NAM(R8),R7                ; get nam address
       14   13  00FB   387          BEQL    30$                             ; branch if none
     FF00'  30  00FD   388          BSBW    RM$CHKNAM                       ; check nam validity
       0E 50  E9  0100   389          BLBC    R0,30$                          ; branch if illegal (no error)
 02 AE   34 A7   D0  0103   390          MOVL    NAM$L_FNB(R7),FNB(SP)           ; save file name status
 01 AE   03 A7   90  0108   391          MOVB    NAM$B_RSL(R7),RSL(SP)           ; save result string
    6E   0B A7   90  010D   392          MOVB    NAM$B_ESL(R7),ESL(SP)           ;  and expanded string lens
       20 A9   B5  0111   393 30$:     TSTW    IFB$W_CHNL(R9)                  ; yes, was a channel assigned?
          15   13  0114   394          BEQL    PRS                             ; no, continue
 03 69   25   E5  0116   395          BBCC    #IFB$V_ACCESSED,(R9),40$        ; deaccess any open file or
     FEE3'  30  011A   396          BSBW    RM$DEACCESS                     ; network links
                 011D   397 40$:     $DASSGN_S       CHAN=IFB$W_CHNL(R9)     ; deassign the channel
       20 A9   B4  0128   398          CLRW    IFB$W_CHNL(R9)                  ; clear it
                 012B   399
                 012B   400 ;
                 012B   401 ; Zero the fid and did in nam block for rm$setdid to work
                 012B   402 ;
                 012B   403
 57 28 A8   D0  012B   404 PRS:     MOVL    FAB$L_NAM(R8),R7                ; get nam address
       24   13  012F   405          BEQL    10$                             ; branch if none
     FECC'  30  0131   406          BSBW    RM$CHKNAM                       ; check nam validity
       1E 50  E9  0134   407          BLBC    R0,10$                          ; branch if illegal (no error)
       14 A7   94  0137   408          CLRB    NAM$T_DVI(R7)                   ; clear device name
                 013A   409
                 013A   410          ASSUME  NAM$W_DID       EQ      NAM$W_FID+6
                 013A   411
       24 A7   7C  013A   412          CLRQ    NAM$W_FID(R7)                   ; zero fid and did fields
       2C A7   D4  013D   413          CLRL    NAM$W_DID+2(R7)
```

```
                     0140   415 ;
                     0140   416 ; Zero expanded string length and resultant string length fields to avoid
                     0140   417 ; leaving these strings lying around from previous parses and consequently
                     0140   418 ; using the wrong filespec in an error message.
                     0140   419 ;
                     0140   420 ; Zero resultant string length and file name status fields to support network
                     0140   421 ; (simulated) open by nam block (see expand_name and setnam in rm0xpfn).
                     0140   422 ;
                     0140   423 ; Zero the wildcard context field to avoid the situation whereby the WCC
                     0140   424 ; context of the current PARSE is OR'd in with the WCC context of the previous
                     0140   425 ; PARSE, but save the fact that the user requested context to be saved
                     0140   426 ; (if the user requested context to be saved), by setting IFB$V_SEARCH.
                     0140   427 ;
                     0140   428
          0B A7   94 0140   429          CLRB    NAM$B_ESL(R7)                   ; preset expanded string null
          03 A7   94 0143   430          CLRB    NAM$B_RSL(R7)                   ; and result string too
          34 A7   D4 0146   431          CLRL    NAM$L_FNB(R7)                   ; zero file name status bits
04 30 A7  1F   E1 0149   432          BBC     #NAM$V_SVCTX,NAM$L_WCC(R7),5$   ; if the user requested context to b
                     014E   433          SSB     #IFB$V_SEARCH,(R9)              ;  saved, then set IFB$V_SEARCH
          30 A7   D4 0152   434 5$:     CLRL    NAM$L_WCC(R7)                   ; clear NAM wildcard bits
                     0155   435
                     0155   436 ;
                     0155   437 ; Parse the input file name and store the pattern in SWB and
                     0155   438 ; initialize the FWA which will contain the result directory specification
                     0155   439 ;
                     0155   440
00000000'EF   16 0155   441 10$:     JSB     RM$XPFN                         ; expand the file spec.
        07 50   E8 015B   442          BLBS    R0,15$                          ; branch if ok
           50   D5 015E   443          TSTL    R0                              ; did we exaust search list?
           12   12 0160   444          BNEQ    20$                             ; no, so other error
         0108   31 0162   445          BRW     RESTORE_ERROR                   ; restore old error and exit
                     0165   446
                     0165   447 ;
                     0165   448 ; If the file is a PPF, retrieve its IFAB and move the device characteristics
                     0165   449 ; into the IFAB that has been allocated for this parse.
                     0165   450 ;
                     0165   451
          0C AA   95 0165   452 15$:     TSTB    FWA$B_ESCFLG(R10)               ; if this file is not a PPF then
           45   13 0168   453          BEQL    60$                             ;  go assign a channel otherwise
08 0E AA   0F   E0 016A   454          BBS     #15,FWA$W_ESCIFI(R10),30$       ; make sure the escape sequence
                     016F   455                                                  ;  is for a PPF IFI and if it
                     016F   456          RMSERR  LNE                             ;  is not, go return am error
          00F2   31 0174   457 20$:     BRW     PREXIT                          ;  invalid equivalence string
                     0177   458
        0A00 8F   BB 0177   459 30$:     PUSHR   #^M<R9,R11>                     ; save IFAB and impure area addr
        57   01   9A 017B   460          MOVZBL  #PSL$C_EXEC,R7                  ; this a executive mode request - NO
    59   0E AA   3C 017E   461          MOVZWL  FWA$W_ESCIFI(R10),R9            ; move ifi into R9
        50   06   D0 0182   462          MOVL    #IMP$C_IFABTBL/4,R0             ; ifab table offset/4
                     0185   463
          FE78'  30 0185   464          BSBW    RM$GTIADR                       ; get ifab address
           06   13 0188   465          BEQL    40$                             ; no IFAB returned?
    08 A9   0B   91 018A   466          CMPB    #IFB$C_BID,IFB$B_BID(R9)        ; is this a valid ifab
           0B   13 018E   467          BEQL    50$                             ; go move device characteristics
                     0190   468
        0A00 8F   BA 0190   469 40$:     POPR    #^M<R9,R11>                     ; restore IFAB and impure addrs
                     0194   470          RMSERR  IFI                             ; return an error of
           D9   11 0199   471          BRB     20$                             ; invalid equivalence string IFI
```

```
                     019B   472
50    59    D0       019B   473 50$:    MOVL    R9,R0                           ; save PPF IFAB address in R0
0A00  8F    BA       019E   474          POPR    #^M<R9,R11>                     ; restore IFAB and impure area addrs
      60    D0       01A2   475          MOVL    IFB$L_PRIM_DEV(R0),-            ; move primary device characteristic
      69             01A4   476                  IFB$L_PRIM_DEV(R9)              ; into IFAB from PPF IFAB
008C  C0    D0       01A5   477          MOVL    IFB$L_AS_DEV(R0),-              ; move secondary device characterist
008C  C9             01A9   478                  IFB$L_AS_DEV(R9)                ; into IFAB from PPF IFAB
   009E     31       01AC   479          BRW     SUC                             ; go return device information
                     01AF   480
                     01AF   481 ;
                     01AF   482 ; If channel is already assigned (must have been called by $rename),
                     01AF   483 ; then exit immediately without assigning channel or setdid
                     01AF   484 ;
                     01AF   485
 20   A9    B5       01AF   486 60$:    TSTW    IFB$W_CHNL(R9)                  ; any channel assigned?
      C0    12       01B2   487          BNEQ    20$                             ; exit successfully if so
      36    E0       01B4   488          BBS     #FWA$V_SYNTAX_CHK,-             ; exit cleaning up if
BC    6A             01B6   489                  (R10),20$                       ; syntax check only
                     01B8   490
                     01B8   491
                     01B8   492 ;
                     01B8   493 ; Assign a channel to the device
                     01B8   494 ;
                     01B8   495
 FE45'     30       01B8   496          BSBW    RM$ASSIGN                       ; assign channel to device
03    50    E8       01BB   497          BLBS    R0,65$                          ; continure if ok
  0081     31       01BE   498          BRW     CHKLST                          ; branch if error
                     01C1   499
                     01C1   500 ;
                     01C1   501 ; For nondisk, foreign mounted, and spooled devices, set FWA$B_DIRLEN to 0,
                     01C1   502 ; clear all directory related bits in the NAM$L_FNB and the wild card summary
                     01C1   503 ; bit NAM$V_WILDCARD, if no wildcards remain. Leave the expanded-name
                     01C1   504 ; string alone, for SDI
                     01C1   505 ; devices initialize the FIB's DID to the MFD. if the file specification
                     01C1   506 ; contained node names, skip all of this and go immediately fill in the NAM.
                     01C1   507 ;
                     01C1   508
      0D     E0      01C1   509 65$:    BBS     #DEV$V_NET,-                    ; if a network operation,
   69 69             01C3   510                 IFB$L_PRIM_DEV(R9),95$          ;  skip the directory stuff
      1C     E0      01C5   511          BBS     #DEV$V_RND,-                    ; go initialize SWB if device
   3D 69             01C7   512                 IFB$L_PRIM_DEV(R9),85$          ; is a random (disk) device
   2E AA     94      01C9   513 70$:    CLRB    FWA$B_DIRLEN(R10)               ; init the number of dirs to 0
                     01CC   514          CSB     #FWA$V_DIR,(R10)                ; clear directory specification
57 28 A8     D0      01D0   515          MOVL    FAB$L_NAM(R8),R7                ; get nam address
      19     13      01D4   516          BEQL    80$                             ; branch if none
  FE27'      30      01D6   517          BSBW    RM$CHKNAM                       ; check nam validity
   13 50     E9      01D9   518          BLBC    R0,80$                          ; branch if illegal (no error)
             CA      01DC   519          BICL2   #NAM$M_GRP_MBR!-                ; clear group-member dir bit
                     01DD   520                  NAM$M_WILD_DIR!-                ;  wild dir summary bit
                     01DD   521                  NAM$M_DIR_LVLS!-                ;  set directory sublevels to 0
                     01DD   522                  ^XFF000000,-                    ;  and clear all wild directory
34 A7  FFF80000 8F   01DD   523                  NAM$L_FNB(R7)                   ;  bits
             D3      01E4   524          BITL    #NAM$M_WILD_NAME!-              ; if either the file name
                     01E5   525                  NAM$M_WILD_TYPE!-               ;  or the file type
                     01E5   526                  NAM$M_WILD_VER,-                ;  or the file version number
   34 A7     38      01E5   527                  NAM$L_FNB(R7)                   ;  is wild
                05   12     01E8   528          BNEQ    80$                     ;  keep the wildcard summary bit
```

```
                     01EA   529          CSB     #NAM$V_WILDCARD,-                       ; otherwise, clear it
                     01EA   530                   NAM$L_FNB(R7)                           ;  whether it was set or not
          18   E0    01EF   531  80$:     BBS     #DEV$V_FOR,-                            ; if device is foreign mounted
       3B 69          01F1   532                   IFB$L_PRIM_DEV(R9),95$                  ;  then go fill in the NAM block
          04   E1    01F3   533           BBC     #DEV$V_SDI,-                            ; if not on a mag tape then
       37 69          01F5   534                   IFB$L_PRIM_DEV(R9),95$                  ;  fill in the NAM block, otherwise,
 00040004   8F   D0    01F7   535           MOVL    #<FID$C_MFD@16>+FID$C_MFD,-             ; initialize the FIB's DID to
     01FE   CA          01FD   536                   FIB$W_DID_NUM+FWA$T_FIBBUF(R10)         ;  the mag tape's MFD
     0202   CA   B4     0200   537           CLRW    FIB$W_DID_RVN+FWA$T_FIBBUF(R10)
          24   11    0204   538           BRB     90$                                     ; and go clear the FIB's FID
```

```
                     0206   540
                     0206   541 ;
                     0206   542 ; Initialize the swb to process the directory pattern.
                     0206   543 ;
                     0206   544
            18   E0  0206   545 85$:    BBS     #DEV$V_FOR,-                        ; if device is foreign then
          43 69      0208   546                 IFB$L_PRIM_DEV(R9),SUC              ;  don't do directory lookup
B9 008C C9   06   E0  020A   547         BBS     #DEV$V_SPL,IFB$L_AS_DEV(R9),70$     ; spooled devices not treated
                     0210   548                                                     ;  the same as disk devices
      39 6A   0E   E1  0210   549         BBC     #FWA$V_DIR,(R10),SUC                ; skip if no dir in spec
00000000'EF   16   0214   550         JSB     RM$INIT_SWB                         ; initialize swb context
        4C 50   E9   021A   551         BLBC    R0,PREXIT                           ; give up on errors
                     021D   552
                     021D   553 ;
                     021D   554 ; Note: RM$NEXTDIR clobbers R8
                     021D   555 ;
                     021D   556
00000000'EF   16   021D   557         JSB     RM$NEXTDIR                          ; get DID of first directory
     58   24 A9   D0   0223   558         MOVL    IFB$L_LAST_FAB(R9),R8               ; restore fab address
        06 50   E9   0227   559         BLBC    R0,100$                             ; go handle any errors otherwise
      0204 CA   D4   022A   560 90$:    CLRL    FWA$T_FIBBUF+FIB$L_WCC(R10)         ; start at first file in directory
                     022E   561                                                     ;  by clear FIB's FID
            1D   11  022E   562 95$:    BRB     SUC                                 ; go fill in the NAM block
                     0230   563
                     0230   564 ;
                     0230   565 ; There was some sort of directory error
                     0230   566 ;
                     0230   567
  82CA 8F   50   B1  0230   568 100$:   CMPW    R0,#RMS$_NMF&^XFFFF                 ; any directory found at all?
            0B   12  0235   569         BNEQ    CHKLST                              ; branch if some other error
                     0237   570         RMSERR  DNF                                 ; set directory not found
0C A8   0910 8F   3C  023C   571         MOVZWL  #SS$_NOSUCHFILE,FAB$L_STV(R8)       ; set stv secondary code
                     0242   572
                     0242   573 ;
                     0242   574 ; There was some error other then a file specification error, so check to
                     0242   575 ; see if there is a search list, if so try the parse again
                     0242   576 ;
                     0242   577
      23 6A   38   E1  0242   578 CHKLST: BBC     #FWA$V_SLPRESENT,(R10),PREXIT       ; no search list, exit
                     0246   579         SSB     #FWA$V_SL_PASS,(R10)                ; flag this as search list pass
          FE89   31  024A   580         BRW     LOOP                                ; go try again
                     024D   581
                     024D   582 ;
                     024D   583 ; We have successfully parsed a name, assigned a channel and/or found
                     024D   584 ; a directory
                     024D   585 ;
                     024D   586
      57   28 A8   D0  024D   587 SUC:    MOVL    FAB$L_NAM(R8),R7                    ; get nam address
            13   13  0251   588         BEQL    10$                                 ; branch if none
          FDAA'  30   0253   589         BSBW    RM$CHKNAM                           ; check nam validity
        0D 50   E9   0256   590         BLBC    R0,10$                              ; branch if illegal (no error)
09 6A     19   E0  0259   591         BBS     #FWA$V_NODE,(R10),10$               ; skip dvi, did if node found
          FDA0'  30   025D   592         BSBW    RM$WRITE_DVI                        ; write DVI into NAM block
                     0260   593
                     0260   594         ASSUME  NAM$L_WCC       EQ      NAM$W_DID+6
                     0260   595
      01FE CA   7D  0260   596         MOVQ    FWA$T_FIBBUF+FIB$W_DID(R10),-       ; copy did and top word of wcc
```

```
                2A A7        0264   597                     NAM$W_DID(R7)                   ;  for fun
                             0266   598
                             0266   599 10$:      RMSSUC
                5E   0E  C0  0269   600 PREXIT:  ADDL2   #STACK_SIZE,SP                  ; readjust stack
                         05  026C   601          RSB
                             026D   602
                             026D   603 ;
                             026D   604 ; XPFN exited with RMS$_NOMLIST, no more search list to parse, so restore
                             026D   605 ; the original error code and name block string lengths
                             026D   606 ;
                             026D   607
                             026D   608 RESTORE_ERROR:
                57   28 A8  D0  026D   609          MOVL    FAB$L_NAM(R8),R7                ; get nam address
                       14    13  0271   610          BEQL    20$                             ; branch if none
                     FD8A'   30  0273   611          BSBW    RM$CHKNAM                       ; check nam validity
                     0E 50   E9  0276   612          BLBC    R0,20$                          ; branch if illegal (no error)
                             0279   613
                             0279   614          ASSUME  ESL     EQ      0
                             0279   615          ASSUME  RSL     EQ      ESL+1
                             0279   616          ASSUME  FNB     EQ      RSL+1
                             0279   617          ASSUME  STV     EQ      FNB+4
                             0279   618          ASSUME  ERR     EQ      STV+4
                             0279   619
                0B A7   6E   90  0279   620 10$:      MOVB    ESL(SP),NAM$B_ESL(R7)           ; restore expanded string
             03 A7   01 AE   90  027D   621          MOVB    RSL(SP),NAM$B_RSL(R7)           ;  and result string
             34 A7   02 AE   D0  0282   622          MOVL    FNB(SP),NAM$L_FNB(R7)           ; restore file name flags
                5E      06   C0  0287   623 20$:      ADDL2   #STV,SP                         ; restore stack past NAM fields
                0C A8   8E   D0  028A   624          MOVL    (SP)+,FAB$L_STV(R8)             ; set stv secondary code
                50      8E   D0  028E   625          MOVL    (SP)+,R0                        ; restore error status
                             05  0291   626          RSB                                     ; exit
                             0292   627
                             0292   628          .END
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
CHKLST                            00000242 R      01
DEV$V_FOR                       = 00000018
DEV$V_NET                       = 0000000D
DEV$V_RND                       = 0000001C
DEV$V_SDI                       = 00000004
DEV$V_SPL                       = 00000006
ERR                             = 0000000A
ESL                             = 00000000
EXIT                              0000009B R      01
FAB$L_NAM                       = 00000028
FAB$L_STV                       = 0000000C
FAB$W_IFI                       = 00000002
FIB$L_WCC                       = 00000010
FIB$W_DID                       = 0000000A
FIB$W_DID_NUM                   = 0000000A
FIB$W_DID_RVN                   = 0000000E
FID$C_MFD                       = 00000004
FNB                             = 00000002
FWA$B_DIRLEN                    = 0000002E
FWA$B_ESCFLG                    = 0000000C
FWA$L_DIRBDB                    = 00000030
FWA$T_FIBBUF                    = 000001F4
FWA$V_DIR                       = 0000000E
FWA$V_NODE                      = 00000019
FWA$V_SLPRESENT                 = 00000038
FWA$V_SL_PASS                   = 00000002
FWA$V_SYNTAX_CHK                = 00000036
FWA$V_WILD_DIR                  = 0000001C
FWA$W_ESCIFI                    = 0000000E
IFB$B_BID                       = 00000008
IFB$C_BID                       = 0000000B
IFB$L_AS_DEV                    = 0000008C
IFB$L_FWA_PTR                   = 00000038
IFB$L_LAST_FAB                  = 00000024
IFB$L_PRIM_DEV                  = 00000000
IFB$V_ACCESSED                  = 00000025
IFB$V_SEARCH                    = 00000039
IFB$W_CHNL                      = 00000020
IMP$L_IFABTBL                   = 00000018
LOOP                              000000D6 R      01
NAM$B_ESL                       = 0000000B
NAM$B_NOP                       = 00000008
NAM$B_RSL                       = 00000003
NAM$L_FNB                       = 00000034
NAM$L_WCC                       = 00000030
NAM$M_DIR_LVLS                  = 00E00000
NAM$M_GRP_MBR                   = 00080000
NAM$M_IFI                       = 00010000
NAM$M_SRCHNMF                   = 40000000
NAM$M_SVCTX                     = 80000000
NAM$M_WILD_DIR                  = 00100000
NAM$M_WILD_NAME                 = 00000020
NAM$M_WILD_TYPE                 = 00000010
NAM$M_WILD_VER                  = 00000008
NAM$T_DVI                       = 00000014
NAM$V_IFI                       = 00000010
NAM$V_SLPARSE                   = 00000005
NAM$V_SVCTX                     = 0000001F
NAM$V_WILDCARD                  = 00000008
NAM$W_DID                       = 0000002A
NAM$W_FID                       = 00000024
PIO$A_TRACE                       ********    X   01
PREXIT                            00000269 R      01
PRS                               0000012B R      01
PSL$C_EXEC                      = 00000001
RESTORE_ERROR                     0000026D R      01
RM$ASSIGN                         ********    X   01
RM$CHKNAM                         ********    X   01
RM$CLEANUP                        ********    X   01
RM$CLSCU                          ********    X   01
RM$DEACCESS                       ********    X   01
RM$EXSUC                          ********    X   01
RM$EX_NOSTR                       ********    X   01
RM$FABCHK                         ********    X   01
RM$FSETI_ALT                      ********    X   01
RM$FSET_ALT                       ********    X   01
RM$GTIADR                         ********    X   01
RM$INIT_SWB                       ********    X   01
RM$NEXTDIR                        ********    X   01
RM$PARSE_FILE                     000000CF RG     01
RM$RET_DEV_CHAR                   ********      X 01
RM$SLIST_ERRS                     ********      X 01
RM$SLIST_ERR_CNT                  ********      X 01
RM$WRITE_DVI                      ********      X 01
RM$XPFN                           ********      X 01
RMS$PARSE                       = FFFFFFFE RG     01
RMS$_DNF                        = 0001C04A
RMS$_IFI                        = 00018564
RMS$_LNE                        = 000185BC
RMS$_NMF                        = 000182CA
RMS$_WCC                        = 000182EA
RSL                             = 00000001
SS$_NOSUCHFILE                  = 00000910
STACK_SIZE                      = 0000000E
STV                             = 00000006
SUC                               0000024D R      01
SVCTX                             0000009E R      01
SYS$DASSGN                        ********   GX   01
TPT$L_PARSE                       ********    X   01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM$RMS	00000292 (658.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.09	00:00:00.87
Command processing	139	00:00:00.82	00:00:07.61
Pass 1	442	00:00:17.20	00:00:35.26
Symbol table sort	0	00:00:02.64	00:00:03.31
Pass 2	122	00:00:03.32	00:00:07.75
Symbol table output	13	00:00:00.14	00:00:00.44
Psect synopsis output	1	00:00:00.02	00:00:00.07
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	754	00:00:24.24	00:00:55.39

The working set limit was 1800 pages.
95512 bytes (187 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1842 non-local and 35 local symbols.
628 source lines were read in Pass 1, producing 15 object records in Pass 2.
30 pages of virtual memory were used to define 29 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	14
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	25

1982 GETS were required to define 25 macros.

There were no errors, warnings or information messages.

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