```
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    RRRRRRRR      SSSSSSSS  EEEEEEEEEE  TTTTTTTTTT
RRRRRRRR    MM       MM    000000    RRRRRRRR      SSSSSSSS  EEEEEEEEEE  TTTTTTTTTT
RR      RR  MMMM   MMMM  00      00  RR      RR  SS          EE              TT
RR      RR  MMMM   MMMM  00      00  RR      RR  SS          EE              TT
RR      RR  MM MM MM MM  00    0000  RR      RR  SS          EE              TT
RR      RR  MM  MM  MM  00    0000  RR      RR  SS          EE              TT
RRRRRRRR    MM      MM  00  00  00  RRRRRRRR      SSSSSS    EEEEEEEE        TT
RRRRRRRR    MM      MM  00  00  00  RRRRRRRR      SSSSSS    EEEEEEEE        TT
RR  RR      MM      MM  0000    00  RR  RR              SS  EE              TT
RR  RR      MM      MM  0000    00  RR  RR              SS  EE              TT
RR    RR    MM      MM  00      00  RR    RR            SS  EE              TT          ....
RR    RR    MM      MM  00      00  RR    RR            SS  EE              TT          ....
RR      RR  MM      MM    000000    RR      RR  SSSSSSSS    EEEEEEEEEE      TT          ....
RR      RR  MM      MM    000000    RR      RR  SSSSSSSS    EEEEEEEEEE      TT          ....

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          $BEGIN  RMORSET,000,RM$RMS0,<SETUP FOR A RAB FUNCTION>,<NOWRT,QUAD>
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 ;                routine to perform common setup for a rab function
0000     33 ;
0000     34 ; Environment:
0000     35 ;                star processor running starlet exec.
0000     36 ;
0000     37 ; Author: L F Laverdure,         creation date: 5-JAN-1977
0000     38 ;
0000     39 ; Modified By:
0000     40 ;
0000     41 ;       V03-005 RAS0269         Ron Schaefer            21-Mar-1984
0000     42 ;               Improve performance by not performing the useless
0000     43 ;               probe of the ERR= or SUC= routines here.
0000     44 ;               Arglist gets probed when those routines are used anyway.
0000     45 ;
0000     46 ;       V03-004 DAS0001         David Solomon           2-Feb-1984
0000     47 ;               Don't call RMS$RAISE_LOCK unless sharing.
0000     48 ;
0000     49 ;       V03-003 KBT0487         Keith B. Thompson       2-Feb-1983
0000     50 ;               Don't take the file lock out for ISAM
0000     51 ;
0000     52 ;       V03-002 KBT0318         Keith B. Thompson       8-Sep-1982
0000     53 ;               Remove all S0 sharing code
0000     54 ;
0000     55 ;       V03-001 KBT0215         Keith B. Thompson       23-Aug-1982
0000     56 ;               Reorganize psects
0000     57 ;
0000     58 ;       V02-017 CDS0001         C Saether       10-Dec-1981
0000     59 ;               Rename psect.
0000     60 ;
0000     61 ;       V02-016 REFORMAT        P S Knibbe      23-Jul-1980
0000     62 ;
0000     63 ;--
0000     64
```

```
0000     66          .SBTTL  DECLARATIONS
0000     67
0000     68 ;
0000     69 ; Include Files:
0000     70 ;
0000     71
0000     72 ;
0000     73 ; Macros:
0000     74 ;
0000     75
0000     76          $IMPDEF
0000     77          $RABDEF
0000     78          $IRBDEF
0000     79          $IFBDEF
0000     80          $RMSDEF
0000     81
0000     82 ;
0000     83 ; Equated Symbols:
0000     84 ;
0000     85
0000     86 ;
0000     87 ; Own Storage:
0000     88 ;
0000     89
```

```
0000     91          .SBTTL  RM$RSET - COMMON SETUP FOR RAB FUNCTION ROUTINE
0000     92
0000     93 ;++
0000     94 ;
0000     95 ;  RM$RSET
0000     96 ;  RM$RSET_ALT
0000     97 ;
0000     98 ;      this routine performs common setup for a rab function call
0000     99 ;      including the following:
0000    100 ;
0000    101 ;       1. call rm$rabchk to check arglist, set base regs
0000    102 ;       2. check for valid isi and set irab and ifab addr
0000    103 ;       3. check for stream idle and set to busy
0000    104 ;       4. clear the sts and stv fields of the rab
0000    105 ;       5. set asy irab bit from rab
0000    106 ;       6. store the arglist addr and caller's mode in the irab
0000    107 ;          and save sp entry value in imp$l_saved_sp
0000    108 ;       7. perform additional setups including checking fac for
0000    109 ;          operation and clearing find_last flag
0000    110 ;
0000    111 ;
0000    112 ; Calling sequence:
0000    113 ;
0000    114 ;  called via the $rabset macro which expands into:
0000    115 ;
0000    116 ;       bsbw    rm$rset
0000    117 ;       .byte   function_code
0000    118 ;
0000    119 ;  alternate entry at rm$rset_alt to perform functions  4, 5, 6, & 7 only.
0000    120 ;  r7, r8, r9, r10, and r11 must be set as per output prior to call.
0000    121 ;
0000    122 ;
0000    123 ; Input Parameters:
0000    124 ;
0000    125 ;       sp      stack pointer
0000    126 ;       ap      argument list addr
0000    127 ;
0000    128 ; Implicit Inputs:
0000    129 ;
0000    130 ;       the in-line 1-byte function code.
0000    131 ;       the contents of the rab
0000    132 ;
0000    133 ; Output Parameters:
0000    134 ;
0000    135 ;       r11     impure area address
0000    136 ;       r10     ifab address
0000    137 ;       r9      irab address
0000    138 ;       r8      rab address
0000    139 ;       r7      caller's mode
0000    140 ;       r0 thru r5 destroyed
0000    141 ;
0000    142 ; Implicit Outputs:
0000    143 ;
0000    144 ;       imp$l_saved_sp is set to the value of sp+4
0000    145 ;       the sts and stv fields of the rab are zeroed
0000    146 ;
0000    147 ; Completion Codes:
```

```
                     0000   148 ;
                     0000   149 ;       none.  if an error is detected returns to user (not caller)
                     0000   150 ;       after appropriate cleanup.  the user receives a standard
                     0000   151 ;       rms error code, in particular, isi, act, and those returned by
                     0000   152 ;       rm$rabchk.
                     0000   153 ;
                     0000   154 ;       if all o.k., returns to caller after the 1-byte
                     0000   155 ;       in-line function code.
                     0000   156 ;
                     0000   157 ; Side Effects:
                     0000   158 ;
                     0000   159 ;       none
                     0000   160 ;
                     0000   161 ;--
                     0000   162
                     0000   163 ;
                     0000   164 ;  entry point to rm$rset
                     0000   165 ;
                     0000   166 ;  validate rab and perform other basic setups
                     0000   167 ;
                     0000   168
                     0000   169 RM$RSET::
             FFFD'   30  0000   170         BSBW    RM$RABCHK                       ; valid rab? (exits if invalid)
                     0003   171
                     0003   172 ;
                     0003   173 ;       NOTE: SETS R11 TO IMPURE ADDR
                     0003   174 ;             r9 to isi
                     0003   175 ;             r8 to rab addr
                     0003   176 ;             r7 to caller's mode
                     0003   177 ;
                     0003   178 ;  get the irab address and check for a valid irab
                     0003   179 ;
                     0003   180
               0A    13  0003   181         BEQL    100$                            ; branch if isi value bad
            0F   59    B1  0005   182         CMPW    R9,#IMP$C_ENTPERSEG             ; simple isi case (in 1st segment)?
               0A    1A  0008   183         BGTRU   10$                             ; branch if so
         59    64 AB49    D0  000A   184         MOVL    IMP$L_IRBTBLINK(R11)[R9],R9     ; get irab address
               0E    12  000F   185 100$:   BNEQ    5$                              ; branch if irab allocated
             0094    31  0011   186         BRW     ERRISI                          ; error is irab not allocated
                     0014   187
                     0014   188 ;
                     0014   189 ;  convert isi value to irab address (general case)
                     0014   190 ;
                     0014   191
            50   07    D0  0014   192 10$:    MOVL    #IMP$L_IRABTBL/4,R0             ; irab table offset/4
                FFE6'   30  0017   193         BSBW    RM$GTIADR                       ; get irab addr
                  F3    11  001A   194         BRB     100$                            ; continue
                     001C   195
                0090    31  001C   196 15$:    BRW     ERRBUG                          ; error...invalid irab
                     001F   197
         0A    08 A9    91  001F   198 5$:     CMPB    IRB$B_BID(R9),#IRB$C_BID
                  F7    12  0023   199         BNEQ    15$                             ; branch if valid irab
                     0025   200
                     0025   201 ;++
                     0025   202 ;
                     0025   203 ;  set r10 = ifab address
                     0025   204 ;
```

```
                     0025   205 ;--
                     0025   206
   5A   69   D0      0025   207          MOVL    IRB$L_IFAB_LNK(R9),R10          ; get ifab address
0B   08 AA   91      0028   208          CMPB    IFB$B_BID(R10),#IFB$C_BID       ; really an ifab?
         EE   12      002C   209          BNEQ    15$
67 6A    20   E0      002E   210          BBS     #IFB$V_BUSY,(R10),ERRACT        ; branch if ifab busy
                     0032   211
                     0032   212 ;++
                     0032   213 ;
                     0032   214 ;  set busy, checking if already active, and clear async and ppf_image flags
                     0032   215 ;
                     0032   216 ;--
                     0032   217
6A 69    20   E2      0032   218          BBSS    #IRB$V_BUSY,(R9),ERRRSA         ; set busy, branch if busy already
                     0036   219
           00000004  0036   220          IRB$M_PPF_IMAGE = 1@<IRB$V_PPF_IMAGE-<IRB$L_BKPBITS*8>>
           00000008  0036   221          IRB$M_ASYNC     = 1@<IRB$V_ASYNC-<IRB$L_BKPBITS*8>>
                     0036   222
04 A9    0C   8A      0036   223 20$:    BICB2   #IRB$M_PPF_IMAGE!IRB$M_ASYNC,IRB$L_BKPBITS(R9); clear flags
57 68    1E   E0      003A   224          BBS     #RAB$V_PPF_IND+<RAB$W_ISI*8>,(R8),CHKIND; branch if indirect ppf
                     003E   225
                     003E   226 ;++
                     003E   227 ;
                     003E   228 ;  alternate entry from rms0conn here
                     003E   229 ;
                     003E   230 ;  clear rab$l_sts and rab$l_stv
                     003E   231 ;  set asynchronous flag if required and probe async arg list
                     003E   232 ;  store caller's mode and arglist addr in irab
                     003E   233 ;
                     003E   234 ;--
                     003E   235
                     003E   236 RM$RSET_ALT::
                     003E   237
                     003E   238          ASSUME  RAB$L_STV       EQ      RAB$L_STS+4
                     003E   239
   08 A8    7C       003E   240          CLRQ    RAB$L_STS(R8)                   ; zero sts and stv in rab
                     0041   241
                     0041   242 ;
                     0041   243 ;  perform asynchronous setup
                     0041   244 ;
                     0041   245
                     0041   246          ASSUME  RAB$V_ASY       EQ      0
                     0041   247
07 04 A8    E9       0041   248          BLBC    RAB$L_ROP(R8),50$               ; branch if sync
                     0045   249
                     0045   250          ASSUME  IMP$W_RMSSTATUS EQ      0
                     0045   251          ASSUME  IMP$V_IIOS      EQ      0
                     0045   252
   04 68    E9       0045   253          BLBC    (R11),50$                       ; branch if this is a ppf forcing
                     0048   254                                                  ; synchronous operation
                     0048   255
                     0048   256          SSB     #IRB$V_ASYNC,(R9)               ; flag async operation
                     004C   257
                     004C   258 ;
                     004C   259 ;  store caller's mode and arglist addr in irab
                     004C   260 ;
                     004C   261
```

```
                0A A9   57   90  004C   262 50$:    MOVB    R7,IRB$B_MODE(R9)       ; save caller's mode
                18 A9   5C   D0  0050   263          MOVL    AP,IRB$L_ARGLST(R9)     ; save pointer to arglist
                24 A9   58   D0  0054   264          MOVL    R8,IRB$L_LAST_RAB(R9)   ; save addr this rab
                                 0058   265
                                 0058   266 ;++
                                 0058   267 ;
                                 0058   268 ;  pick up in-line byte specifying optional functions and checking
                                 0058   269 ;
                                 0058   270 ;--
                                 0058   271
                51   00 BE   9A  0058   272          MOVZBL  @0(SP),R1               ; get byte
                        1E   13  005C   273          BEQL    SAVESP                  ; branch if nothing to do
                04 51   04   E5  005E   274          BBCC    #4,R1,60$               ; branch if no need to clr find_last
                                 0062   275
                                 0062   276 ;++
                                 0062   277 ;
                                 0062   278 ;  cflg (bit 4 set) - clear last-operation-was-a-find flag
                                 0062   279 ;
                                 0062   280 ;--
                                 0062   281
                                 0062   282          CSB     #IRB$V_FIND_LAST,(R9)
                                 0066   283
                                 0066   284 ;++
                                 0066   285 ;
                                 0066   286 ;  verify accessed for block i/o or not based on bio (bit 3) parameter
                                 0066   287 ;
                                 0066   288 ;--
                                 0066   289
                5B 51   03   E4  0066   290 60$:    BBSC    #3,R1,CHKBIO            ; branch if a block i/o function
                        05   E0  006A   291          BBS     #IFB$V_BIO,-            ; branch if file accessed
                     22 AA       006C   292                  IFB$B_FAC(R10),-         ;  for block i/o
                        47       006E   293                  RM$ERRIOP               ;  (invalid operation)
                        27   E4  006F   294          BBSC    #IRB$V_BIO_LAST,-       ; this is not a block operation so
                     00 69       0071   295                  (R9),CHKFAC             ;  clear block i/o last
                                 0073   296
                                 0073   297 ;++
                                 0073   298 ;
                                 0073   299 ;  check for appropriate access for function
                                 0073   300 ;  (note: r1 now has bit offset to required fac access)
                                 0073   301 ;
                                 0073   302 ;--
                                 0073   303
                05 51   05   E0  0073   304 CHKFAC: BBS     #5,R1,SAVESP                    ; branch if flag says any fac ok
              41 22 AA   51   E1  0077   305          BBC     R1,IFB$B_FAC(R10),RM$ERRFAC     ; branch if needed access not on
                                 007C   306
            14 AB   5E   04   C1  007C   307 SAVESP: ADDL3   #4,SP,IMP$L_SAVED_SP(R11)       ; save stack entry value
                                 0081   308
                                 0081   309 ;+
                                 0081   310 ;  Take lock on file if sharing and if not ISAM.
                                 0081   311 ;-
                                 0081   312
                     78 AA   D5  0081   313          TSTL    IFB$L_SFSB_PTR(R10)             ; are we sharing?
                        0C   13  0084   314          BEQL    10$                             ; no, no need to lock file
                  23 AA   02   91  0086   315          CMPB    #IFB$C_IDX,IFB$B_ORGCASE(R10)   ; is this ISAM?
                        06   13  008A   316          BEQL    10$                             ; if so don't lock yet
                      FF71'  30  008C   317          BSBW    RM$RAISE_LOCK                   ; get lock on the file
                     13 50   E9  008F   318          BLBC    R0,ERROR                        ; get out on error
```

```
                     0092   319
                     0092   320 ;
                     0092   321 ;  bump return pc past the in-line function byte and return
                     0092   322 ;
                     0092   323
                6E   D6 0092   324 10$:    INCL    (SP)
                     05 0094   325         RSB
                     0095   326
                     0095   327 ;++
                     0095   328 ;
                     0095   329 ;  this is an indirect operation on a ppf.  set ppf_image flag.
                     0095   330 ;
                     0095   331 ;--
                     0095   332
          A5 69      22   E3 0095   333 CHKIND: BBCS    #IRB$V_PPF_IMAGE,(R9),RM$RSET_ALT; say its indirect and branch
```

```
                     0099   335
                     0099   336 ;++
                     0099   337 ;
                     0099   338 ;  error returns
                     0099   339 ;
                     0099   340 ;--
                     0099   341
                     0099   342 ERRACT:
                     0099   343          RMSERR  ACT                     ; stream already active
          05    11   009E   344          BRB     ERROR
                     00A0   345
                     00A0   346 ERRRSA:
                     00A0   347          RMSERR  RSA                     ; record stream active
                     00A5   348
      FF58'    31   00A5   349 ERROR:   BRW     RMS$EX_NOSTR
                     00A8   350
                     00A8   351 ERRISI:
                     00A8   352          RMSERR  ISI                     ; invalid isi value
          F6    11   00AD   353          BRB     ERROR
                     00AF   354
                     00AF   355 ;
                     00AF   356 ;  internal rms problem  -  irab table pointed to an invalid irab
                     00AF   357 ;                           or irab pointed to invalid ifab!
                     00AF   358 ;
                     00AF   359
                     00AF   360 ERRBUG: RMSTBUG FTL$_BADIFAB
                     00B6   361
                     00B6   362 ;++
                     00B6   363 ;
                     00B6   364 ;  entry point from rms$delete (attempted delete for seq. file org)
                     00B6   365 ;
                     00B6   366 ;--
                     00B6   367
                     00B6   368 RM$ERRIOP::
                     00B6   369          RMSERR  IOP                     ; wrong type of access re. bio
          05    11   00BB   370          BRB     ERROR1
                     00BD   371
                     00BD   372 RM$ERRFAC::
                     00BD   373          RMSERR  FAC                     ; not accessed for function
      FF3B'    31   00C2   374 ERROR1: BRW     RM$EXRMS                ; get out
                     00C5   375
```

```
                     00C5   377
                     00C5   378 ;++
                     00C5   379 ;
                     00C5   380 ;  file must be accessed to allow block i/o
                     00C5   381 ;
                     00C5   382 ;--
                     00C5   383
A9 22 AA   05   E0   00C5   384 CHKBIO: BBS     #IFB$V_BIO,IFB$B_FAC(R10),CHKFAC        ; branch if file accessed fo
E7 22 AA   06   E1   00CA   385         BBC     #IFB$V_BRO,IFB$B_FAC(R10),RM$ERRIOP     ; branch if also not block
                     00CF   386                                                         ;  and record access
                     00CF   387 ;++
                     00CF   388 ;
                     00CF   389 ;  file is accessed for mixed block and record i/o
                     00CF   390 ;
                     00CF   391 ;  set the block i/o last flag and also set the 'switched between block
                     00CF   392 ;  and record i/o' flag if last operation was not a block i/o operation,
                     00CF   393 ;  else clear flag
                     00CF   394 ;
                     00CF   395 ;--
                     00CF   396
                     00CF   397         CSB     #IRB$V_BRO_SW,(R9)
   9C 69   27   E2   00D3   398         BBSS    #IRB$V_BIO_LAST,(R9),CHKFAC
   98 69   28   E3   00D7   399         BBCS    #IRB$V_BRO_SW,(R9),CHKFAC               ; set bro_sw and branch
                     00DB   400
                     00DB   401         .END
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
CHKBIO                            000000C5 R      01
CHKFAC                            00000073 R      01
CHKIND                            00000095 R      01
ERRACT                            00000099 R      01
ERRBUG                            000000AF R      01
ERRISI                            000000A8 R      01
ERROR                             000000A5 R      01
ERROR1                            000000C2 R      01
ERRRSA                            000000A0 R      01
FTL$_BADIFAB                    = FFFFFFFD
IFB$B_BID                       = 00000008
IFB$B_FAC                       = 00000022
IFB$B_ORGCASE                   = 00000023
IFB$C_BID                       = 0000000B
IFB$C_IDX                       = 00000002
IFB$L_SFSB_PTR                  = 00000078
IFB$V_BIO                       = 00000005
IFB$V_BRO                       = 00000006
IFB$V_BUSY                      = 00000020
IMP$C_ENTPERSEG                 = 0000000F
IMP$L_IRABTBL                   = 0000001C
IMP$L_IRBTBLINK                 = 00000064
IMP$L_SAVED_SP                  = 00000014
IMP$V_IIOS                      = 00000000
IMP$W_RMSSTATUS                 = 00C00000
IRB$B_BID                       = 00000008
IRB$B_MODE                      = 0000000A
IRB$C_BID                       = 0000000A
IRB$L_ARGLST                    = 00000018
IRB$L_BKPBITS                   = 00000004
IRB$L_IFAB_LNK                  = 00000000
IRB$L_LAST_RAB                  = 00000024
IRB$M_ASYNC                     = 00000008
IRB$M_PPF_IMAGE                 = 00000004
IRB$V_ASYNC                     = 00000023
IRB$V_BIO_LAST                  = 00000027
IRB$V_BRO_SW                    = 00000028
IRB$V_BUSY                      = 00000020
IRB$V_FIND_LAST                 = 00000025
IRB$V_PPF_IMAGE                 = 00000022
RAB$L_ROP                       = 00000004
RAB$L_STS                       = 00000008
RAB$L_STV                       = 0000000C
RAB$V_ASY                       = 00J00000
RAB$V_PPF_IND                   = 0000000E
RAB$W_ISI                       = 00000002
RM$BUG                            ********    X   01
RM$ERRFAC                         000000BD RG     01
RM$ERRIOP                         000000B6 RG     01
RM$EXRMS                          ********    X   01
RM$EX_NOSTR                       ********    X   01
RM$GTTADR                         ********    X   01
RM$RABCHK                         ********    X   01
RM$RAISE_LOCK                     ********    X   J1
RM$RSET                           00000000 RG     01
RM$RSET_ALT                       0000003E RG     01
RMS$_ACT                        = 0001825A
RMS$_FAC                        = 00018514
RMS$_IOP                        = 00018574
RMS$_ISI                        = 00018584
RMS$_RSA                        = 000182DA
SAVESP                            0000007C 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
RM$RMSO	000000DB (219.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC QUAD
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	30	00:00:00.10	00:00:00.98
Command processing	115	00:00:00.71	00:00:04.55
Pass 1	271	00:00:07.65	00:00:20.70
Symbol table sort	0	00:00:01.04	00:00:01.69
Pass 2	79	00:00:01.66	00:00:04.07
Symbol table output	9	00:00:00.08	00:00:00.09
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	508	00:00:11.27	00:00:32.12

The working set limit was 1500 pages.
42299 bytes (83 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 799 non-local and 11 local symbols.
401 source lines were read in Pass 1, producing 13 object records in Pass 2.
21 pages of virtual memory were used to define 20 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	11
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	16

904 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM0RECLCK
LIS

RM0SET
LIS

RM0SCAN
LIS

RM0PRFLNM
LIS

RM0RCLCK2
LIS

RM0RABCHK
LIS

RM0RELEAS
LIS

RM0NAMSTR
LIS