```
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      11      NN      NN  XX      XX  TTTTTTTTTT  BBBBBBBB    LL          KK      KK
RRRRRRRR    MM       MM      11      NN      NN  XX      XX  TTTTTTTTTT  BBBBBBBB    LL          KK      KK
RR      RR  MMMM   MMMM    1111      NN      NN  XX      XX      TT      BB      BB  LL          KK      KK
RR      RR  MMMM   MMMM    1111      NN      NN  XX      XX      TT      BB      BB  LL          KK      KK
RR      RR  MM MM MM MM      11      NNNN    NN    XX  XX        TT      BB      BB  LL          KK    KK
RR      RR  MM MM MM MM      11      NNNN    NN    XX  XX        TT      BB      BB  LL          KK    KK
RRRRRRRR    MM  MMM  MM      11      NN NN   NN      XX          TT      BBBBBBBB    LL          KKKKKK
RRRRRRRR    MM  MMM  MM      11      NN NN   NN      XX          TT      BBBBBBBB    LL          KKKKKK
RR  RR      MM       MM      11      NN   NNNN    XX  XX        TT      BB      BB  LL          KK    KK
RR  RR      MM       MM      11      NN   NNNN    XX  XX        TT      BB      BB  LL          KK    KK
RR    RR    MM       MM      11      NN      NN  XX      XX      TT      BB      BB  LL          KK      KK      ....
RR    RR    MM       MM      11      NN      NN  XX      XX      TT      BB      BB  LL          KK      KK      ....
RR      RR  MM       MM  111111      NN      NN  XX      XX      TT      BBBBBBBB    LLLLLLLLLL  KK      KK      ....
RR      RR  MM       MM  111111      NN      NN  XX      XX      TT      BBBBBBBB    LLLLLLLLLL  KK      KK      ....

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          $BEGIN  RM1NXTBLK,000,RM$RMS1,<GET NEXT BLOCK FOR SEQUENTIAL FILE ORG>,-
0000     2          <NOWRT,QUAD>
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
```

```
0000     29 ;++
0000     30 ; Facility: RMS32
0000     31 ;
0000     32 ; Abstract:
0000     33 ;                    this module performs get next block processing
0000     34 ;                    for the sequential file organization.
0000     35 ;
0000     36 ; Environment:
0000     37 ;                    star processor running starlet exec.
0000     38 ;
0000     39 ; Author:            L F Laverdure,   creation date: 3-FEB-1977
0000     40 ;
0000     41 ; Modified By:
0000     42 ;
0000     43 ;           V03-004 JWT0167         Jim Teague              15-Mar-1984
0000     44 ;                   Redo buffer offset implementation.
0000     45 ;
0000     46 ;           V03-003 KBT0415         Keith B. Thompson       30-Nov-1982
0000     47 ;                   Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000     48 ;
0000     49 ;           V03-002 KBT0141         Keith B. Thompson       20-Aug-1982
0000     50 ;                   Reorganize pects
0000     51 ;
0000     52 ;           V03-001 KBT0088         Keith B. Thompson       13-Jul-1982
0000     53 ;                   Clean up psects
0000     54 ;
0000     55 ;           V02-023 REFORMAT        Keith B. Thompson       29-Jul-1980
0000     56 ;
0000     57 ;--
0000     58 ;
0000     59
```

```
                     0000     61          .SBTTL  DECLARATIONS
                     0000     62
                     0000     63 ;
                     0000     64 ; Include Files:
                     0000     65 ;
                     0000     66
                     0000     67 ;
                     0000     68 ; Macros:
                     0000     69 ;
                     0000     70
                     0000     71          $IFBDEF
                     0000     72          $DEVDEF
                     0000     73          $IRBDEF
                     0000     74          $BDBDEF
                     0000     75          $RABDEF
                     0000     76          $RMSDEF
                     0000     77
                     0000     78 ;
                     0000     79 ; Equated Symbols:
                     0000     80 ;
                     0000     81
00000020             0000     82          ROP     = RAB$L_ROP * 8         ; bit offset to rop
                     0000     83
                     0000     84 ;
                     0000     85 ; Own Storage:
                     0000     86 ;
                     0000     87
```

```
0000     89
0000     90 ;++
0000     91 ; notes on the multi-block buffering scheme
0000     92 ;
0000     93 ; this routine causes multiple blocks to be read together
0000     94 ; (as specified by mbc) but returned one at a time for
0000     95 ; processing by the calling routines.
0000     96 ;
0000     97 ; assumptions:
0000     98 ;
0000     99 ;       1.  mbc is never > 0 except for disk(= # vbn's - 1)
0000    100 ;       2.  records are always written at eof (only updates
0000    101 ;           may occur elsewhere in the file).
0000    102 ;       3.  all sequential i/o calls go thru one of the
0000    103 ;           following routines:
0000    104 ;               RM$NXTBLK1
0000    105 ;               RM$WTLST1
0000    106 ;               RM$RELBLK1
0000    107 ;       4.  there is no write sharing for sequential files.
0000    108 ;       5.  a direct release will be done only when there
0000    109 ;           is no i/o for the buffer.
0000    110 ;
0000    111 ; bdb field usage:
0000    112 ;
0000    113 ;       1.  BDB$L_VBN = vbn of first block in buffer
0000    114 ;       2.  (!RB$L_RP_VBN = vbn of current block)
0000    115 ;       3.  BDB$B_REL_VBN = current vbn rel to start vbn for buffer
0000    116 ;       4.  BDB$B_VAL_VBNS = # of valid vbns in buffer
0000    117 ;       5.  BDB$B_FLGS:
0000    118 ;               -BDB$V_DRT:all blocks up to the greater of the current vbn
0000    119 ;                and the number of val_vbns are dirty
0000    120 ;               -BDB$V_VAL:the current vbn is valid
0000    121 ;
0000    122 ;       6.  the relative vbn = requested vbn - start vbn
0000    123 ;       7.  current block buffer addr = buff addr + (REL_VBN*512)
0000    124 ;       8.  bdb$w_numb = # bytes in current block
0000    125 ;               on reads = min(# blocks desired, (IRB$B_MBC+1)) * 512
0000    126 ;               on writes = (max(VAL_VBN,REL_VBN+1))*512
0000    127 ;       9.  requested vbn is in buffer if its REL_VBN < VAL_VBNS
0000    128 ;           (VAL_VBNS will be recalculated from REL_VBN if the val bit is on
0000    129 ;           for calls to RM$NXTBLK1 and RM$GETBLKNRP)
0000    130 ;       10. if read required and REL_VBN < VAL_VBNS ok,
0000    131 ;               else release buffer and reread
0000    132 ;       11. on release (RM$RELBLK1) if BDB$V_VAL is off and the
0000    133 ;           BDB$V_DRT bit is set, merely decrement the
0000    134 ;           current vbn and set the valid bit.
0000    135 ;--
0000    136
```

```
0000    138          .SBTTL  RM$NXTBLK1 - GET NEXT BLOCK SEQUENTIAL ROUTINE
0000    139
0000    140 ;++
0000    141 ;
0000    142 ;  RM$NXTBLK1   -
0000    143 ;  RM$GETBLKNRP -
0000    144 ;  GETBLK1      -
0000    145 ;  GETBLK1ALT-
0000    146 ;  RM$MAPBLK1   -
0000    147 ;
0000    148 ; this routine locates the next block of a sequential file,
0000    149 ; reading it if necessary, and updating the appropriate fields
0000    150 ; in the irab and BDB.
0000    151 ;
0000    152 ; Calling sequence:
0000    153 ;
0000    154 ; there are six entry points:
0000    155 ;
0000    156 ;       BSBW    RM$NXTBLK1              ; return the next block
0000    157 ;       BSBW    RM$GETBLKNRP            ; return the block specified by nrp
0000    158 ;       BSBW    GETBLK1                 ; return the block specified by r1
0000    159 ;       BSBW    GETBLK1ALT              ; read record for unit record device
0000    160 ;                                       ; (note: r2=size of buffer)
0000    161 ;       BSBW    RM$MAPBLK1              ; set start and end
0000    162 ;                                       ; address for current
0000    163 ;                                       ; block of BDB whose
0000    164 ;                                       ; address is in r4
0000    165 ;
0000    166 ; Input Parameters:
0000    167 ;
0000    168 ;       R11     impure area addr
0000    169 ;       R10     ifab addr
0000    170 ;       R9      irab addr
0000    171 ;       R8      rab addr
0000    172 ;       R4      BDB addr (entry at RM$MAPBLK1 only)
0000    173 ;       R3      option to avoid read if bit 0 set (not input to RM$MAPBLK1)
0000    174 ;               option to do short disk read if bit 1 set
0000    175 ;       R2      # of blocks to read minus 1 (only if bit 1 set in r3)
0000    176 ;       R1      vbn to read (entry at GETBLK1 only)
0000    177 ;
0000    178 ; Implicit Inputs:
0000    179 ;
0000    180 ;       the various fields of the ifab, irab, and associated BDB.
0000    181 ;
0000    182 ; Output Parameters:
0000    183 ;
0000    184 ;       R7      addr of end of block buffer+1
0000    185 ;       R4      addr of current bdb
0000    186 ;       R1      addr of current block in buffer
0000    187 ;       R0      status code (except for entry at any of the RM$MAPxxx entries)
0000    188 ;       R2-R3   destroyed (except for entry at any of the RM$MAPxxx entries)
0000    189 ;
0000    190 ; Implicit Outputs:
0000    191 ;
0000    192 ;       various fields of the irab and associated BDB are updated.
0000    193 ;
0000    194 ; Completion Codes:
```

```
0000    195 ;
0000    196 ;       standard rms. upon error, IRAB$L_IOS & _IOS4 have
0000    197 ;       error information.
0000    198 ;       if RM$_EOF, then IRB$V_EOF set.
0000    199 ;
0000    200 ; Side Effects:
0000    201 ;
0000    202 ;       none
0000    203 ;
0000    204 ;--
0000    205
```

```
                     0000   207
                     0000   208 RM$NXTBLK1::
                     0000   209         $TSTPT  NXTBLK1
                     0006   210
                     0006   211 ;
                     0006   212 ; get new value for NRP
                     0006   213 ;
                     0006   214
        40 A9   D6   0006   215         INCL    IRB$L_NRP_VBN (R9)      ; bump vbn
        44 A9   B4   0009   216         CLRW    IRB$W_NRP_OFF(R9)       ; zero offset
                     000C   217
                     000C   218 ;
                     000C   219 ;  entry to return block specified by nrp
                     000C   220 ;
                     000C   221
                     000C   222 RM$GETBLKNRP::
54      20 A9   D0   000C   223         MOVL    IRB$L_CURBDB(R9),R4     ; get BDB address
           58   13   0010   224         BEQL    READAHEAD               ; branch if no current BDB
                     0012   225
                     0012   226 ;
                     0012   227 ; is the requested block already in the buffer?
                     0012   228 ;
                     0012   229
        1C A4   C3   0012   230         SUBL3   BDB$L_VBN(R4),-         ; compute rel vbn
51      40 A9        0015   231                 IRB$L_NRP_VBN(R9),R1
   51      51   F6   0018   232         CVTLB   R1,R1                   ; mbc is at most 127
           46   1D   001B   233         BVS     RELEASE                 ; so branch if not in range
                     001D   234
                     001D   235 ;
                     001D   236 ; if read requested, must be within the valid count
                     001D   237 ; otherwise, it need merely be within mbc
                     001D   238 ;
                     001D   239
                     001D   240         ASSUME  BDB$V_VAL EQ 0
        35 0A A4   E9   001D   241         BLBC    BDB$B_FLGS(R4),20$      ; branch if buffer not valid
                     0021   242
                     0021   243 ;
                     0021   244 ;  recompute # valid blocks = max(VAL_VBNS,REL_VBN+1)
                     0021   245 ;
                     0021   246
50      48 A4   01   81   0021   247 2$:     ADDB3   #1,BDB$B_REL_VBN(R4),R0 ; compute min. # valid vbns
        49 A4   50   91   0026   248         CMPB    R0,BDB$B_VAL_VBNS(R4)   ; need to adjust # valid vbns?
                04   15   002A   249         BLEQ    5$                      ; branch if not
        49 A4   50   90   002C   250         MOVB    R0,BDB$B_VAL_VBNS(R4)   ; set new value for # valid vbns
                     0030   251 5$:
                     0030   252
                     0030   253 ;
                     0030   254 ;  check validity of desired block
                     0030   255 ;
                     0030   256
        49 A4   51   91   0030   257         CMPB    R1,BDB$B_VAL_VBNS(R4)   ; is the block valid?
                16   1F   0034   258         BLSSU   10$                     ; yes - go use it
                2B   1A   0036   259         BGTRU   RELEASE                 ; no - go release current bdb
           28 53   E9   0038   260         BLBC    R3,RELEASE              ; branch if read required
                     003B   261
                     003B   262 ;
                     003B   263 ;  block desired is the 'next' block and it's for output.
```

```
                     003B   264 ;  merely add it to the current buffer if < or = mbc.
                     003B   265 ;
                     003B   266
55 A9    51    91    003B   267          CMPB    R1,IRB$B_MBC(R9)        ; will block fit in the buffer?
         22    1A    003F   268          BGTRU   RELEASE                 ; branch if not
                     0041   269
                     0041   270 ;
                     0041   271 ;  check for auto extend required
                     0041   272 ;
                     0041   273
   40 A9    D1       0041   274          CMPL    IRB$L_NRP_VBN(R9),-     ; is next block allocated?
   70 AA                0044   275                  IFB$L_HBK(R10)
         1B    1A     0046   276          BGTRU   RELEASE                 ; branch if not, writing buffer
                     0048   277
                     0048   278 ;
                     0048   279 ; (note: will allocate space below)
                     0048   280 ;
                     0048   281
      01    8A       0048   282          BICB    #1@BDB$V_VAL,-          ; invalidate the buffer
   0A A4                004A   283                  BDB$B_FLGS(R4)
                     004C   284
                     004C   285 ;
                     004C   286 ; (flags current blk as not yet valid)
                     004C   287 ;
                     004C   288
48 A4    51    90    004C   289 10$:     MOVB    R1,BDB$B_REL_VBN(R4)    ; set new current rel vbn
                     0050   290          RMSSUC                          ; show success
      00D0    31     0053   291          BRW     MAPBLK1ALT
```

```
                     0056   293
                     0056   294 ;
                     0056   295 ;  buffer valid flag is off.
                     0056   296 ;  if buffer is dirty decrement REL_VBN, set valid, & try for a hit.
                     0056   297 ;
                     0056   298
      01   E1        0056   299 20$:    BBC     #BDB$V_DRT,-            ; branch if not dirty
  0A A4              0058   300                 BDB$B_FLGS(R4),-
      08             005A   301                 RELEASE
  48 A4    97        005B   302         DECB    BDB$B_REL_VBN(R4)       ; adjust REL_VBN
      00   E3        005E   303         BBCS    #BDB$V_VAL,-
BE 0A A4             00C0   304                 BDB$B_FLGS(R4),2$       ; set valid and branch
                     00C3   305
                     0063   306 ;
                     0063   307 ; required block not in this buffer
                     0063   308 ; release current contents of buffer before reusing
                     0063   309 ;
                     0063   310
                     0063   311 RELEASE:
    FF9A'    30      0063   312         BSBW    RM$RELBLK1              ; release the buffer
    01 50    E8      0066   313         BLBS    R0,READAHEAD            ; continue on success
             05      0069   314         RSB
                     006A   315 READAHEAD:
79 69  2A    E1      006A   316         BBC     #IRB$V_RAHWBH,(R9),80$  ; branch if rah not enabled
    76 53    E8      006E   317         BLBS    R3,80$                  ; branch if not a real read
                     0071   318         ASSUME  RAB$C_SEQ EQ 0
    1E A8    95      0071   319         TSTB    RAB$B_RAC(R8)           ; only do rah if rac=seq
       71    12      0074   320         BNEQ    80$                     ; branch if not seq
6D 68  29    E1      0076   321         BBC     #RAB$V_RAH+ROP,(R8),80$ ; branch if user does not want rah
                     007A   322
                     007A   323 ;
                     007A   324 ; all is go for read ahead so fire up as many as needed or possible
                     007A   325 ;
                     007A   326
       54    DD      007A   327         PUSHL   R4                      ; set last BDB for RAH limit
54   3C A9    D0      007C   328 20$:    MOVL    IRB$L_NXTBDB(R9),R4     ; get BDB for next RAH
     FF7D'    30      0080   329 25$:    BSBW    RM$SEQFLNKBDB
       02     E0      0083   330         BBS     #BDB$V_IOP,-            ; branch if io in progress
  48 0A A4             0085   331                 BDB$B_FLGS(R4),60$
       6E     D5      0088   332 28$:    TSTL    (SP)                    ; first time call ?
       0E     12      008A   333         BNEQ    30$                     ; branch if no
51   40 A9    D0      008C   334         MOVL    IRB$L_NRP_VBN(R9),R1    ; start rah at NRP_VBN
6E   3C A9    D0      0090   335         MOVL    IRB$L_NXTBDB(R9),(SP)   ; set last BDB for RAH limit
20 A9  6E    D0      0094   336         MOVL    (SP),IRB$L_CURBDB(R9)   ; set up CURBDB to pickup the
                     0098   337
                     0098   338 ;
                     0098   339 ; BDB for which this RAH will
                     0098   340 ; be issued (ie NXTBDB=CURBDB
                     0098   341 ; rah is for flnk of NXTBDB and
                     0098   342 ; rm$seqrd uses flnk of CURBDB)
                     0098   343 ;
                     0098   344
       15    11      0098   345         BRB     50$
50   3C A9    D0      009A   346 30$:    MOVL    IRB$L_NXTBDB(R9),R0     ; get BDB addr for last RAH
   51   01    D0      009E   347         MOVL    #1,R1                   ; assume 1 vbn xfer
       1C    E1      00A1   348         BBC     #DEV$V_RND,-            ; branch if not disk
     06 6A            00A3   349                 IFB$L_PRIM_DEV(R10),40$
```

```
        51   14 A0   07   09   EF  00A5          EXTZV   #9,#7,BDB$W_NUMB(R0),R1 ; get # of vbn's in buffer
                                   00AB
                                   00AB     ;
                                   00AB     ; (assumes 512 byte block size)
                                   00AB     ;
                                   00AB
               51    1C A0    C0  00AB     40$:   ADDL2   BDB$L_VBN(R0),R1        ; r1 = start vbn for xfer
               52    55 A9    9A  00AF     50$:   MOVZBL  IRB$B_MBC(R9),R2        ; calc xfer size (mbc = # of
                        52    D6  00B3            INCL    R2                      ; blocks - 1 so increment)
               52    48 AA    C4  00B5            MULL2   IFB$L_DEVBUFSIZ(R10),R2 ; make bytes
               74 AA    51    D1  00B9            CMPL    R1,IFB$L_EBK(R10)       ; don't read past eof block
                        25    1A  00BD            BGTRU   70$                     ; branch if start vbn past eof block
                     FF3E'    30  00BF            BSBW    RM$SEQRAH               ; issue read ahead
                     1F 50    E9  00C2            BLBC    R0,70$                  ; branch if errors
               3C A9    54    D0  00C5            MOVL    R4,IRB$L_NXTBDB(R9)     ; update NXTBDB
                  6E    54    D1  00C9            CMPL    R4,(SP)                 ; reached limit ?
                        B2    12  00CC            BNEQ    25$                     ; branch if no
                        14    11  00CE            BRB     70$                     ; otherwise exit rah loop
                                  00D0
                     48 A4    D5  00D0     60$:   TSTL    BDB$L_IOSB(R4)          ; we bumped into a wbh
                        0F    13  00D3            BEQL    70$                     ; branch if not done
                     FF28'    30  00D5            BSBW    RM$STALLRAHWBH          ; otherwise process bdb
                                  00D8
                                  00D8     ;
                                  00D8     ; (note: does not stall)
                                  00D8     ;
                                  00D8
                                  00D8            CSB     #BDB$V_IOP,-            ; clear i/o in progress
                                  00D8                    BDB$B_FLGS(R4)
                     A8 50    E8  00DD            BLBS    R0,28$                  ; branch if all ok
                  5E    04    C0  00E0            ADDL2   #4,SP                   ; pop stack
                              05  00E3            RSB                             ; and exit with wbe error
                                  00E4
                  5E    04    C0  00E4     70$:   ADDL2   #4,SP                   ; pop limit off stack
               51    40 A9    D0  00E7     80$:   MOVL    IRB$L_NRP_VBN(R9),R1    ; set block to read
                                  00EB
                                  00EB     ;
                                  00EB     ; and  all into GETBLK1
                                  00EB     ;
                                  00EB
```

```
                     00EB    391
                     00EB    392 ;
                     00EB    393 ; entry to read specified block   r1 = vbn
                     00EB    394 ;
                     00EB    395
                     00EB    396 GETBLK1:
                     00EB    397
                     00EB    398 ;
                     00EB    399 ; compute size for xfer
                     00EB    400 ;
                     00EB    401
06 53    01    E1    00EB    402          BBC     #1,R3,5$                ; branch unless flag indicates
                     00EF    403
                     00EF    404 ;
                     00EF    405 ; a short read for random i/o
                     00EF    406 ;
                     00EF    407
55 A9    52    91    00EF    408          CMPB    R2,IRB$B_MBC(R9)        ; r2 < mbc?
         04    1B    00F3    409          BLEQU   10$                     ; branch if yes
52   55 A9    9A    00F5    410 5$:      MOVZBL  IRB$B_MBC(R9),R2        ; mbc = # blocks -1
         52    D6    00F9    411 10$:     INCL    R2                      ; get # of blocks
52   48 AA    C4    00FB    412          MULL2   IFB$L_DEVBUFSIZ(R10),R2 ; and size in bytes
                     00FF    413
                     00FF    414 ;
                     00FF    415 ;  entry point for unit record device read ( and foreign unblocked mt)
                     00FF    416 ;
                     00FF    417
                     00FF    418 GETBLK1ALT:
   53    DD    00FF    419          PUSHL   R3                      ; save flags
 FEFC'   30    0101    420          BSBW    RM$SEQRD                ; and go read the record
   08    BA    0104    421          POPR    #^M<R3>
               0106    422          ASSUME  <BDB$B_REL_VBN+1> EQ BDB$B_VAL_VBNS
48 A4    B4    0106    423          CLRW    BDB$B_REL_VBN(R4)       ; reset relative vbn
5D 50    E9    0109    424          BLBC    R0,ERRXFR               ; go process error
   1C    E1    010C    425          BBC     #DEV$V_RND,-            ; branch if not disk
   6A          010E    426                  IFB$L_PRIM_DEV(R10),-
   12          010F    427                  RM$MAPBLK1
44 53    E8    0110    428          BLBS    R3,CHK_EXTEND           ; branch if not reading
                     0113    429
                     0113    430 SET_VBNS:
                     0113    431
                     0113    432 ;+!!!!!
                     0113    433 ; note:  this code assumes that disk has a block size of 512 bytes
                     0113    434 ;        number of valid blocks = # bytes xferred divided by 512
                     0113    435 ;--
                     0113    436
51   14 A4    07    09    EF    0113    437          EXTZV   #9,#7,BDB$W_NUMB(R4),R1 ; get # of vbns xferred
        49 A4    51    90    0119    438          MOVB    R1,BDB$B_VAL_VBNS(R4)   ; set # valid blocks
                     011D    439
                     011D    440 ;
                     011D    441 ;-!!!!
                     011D    442 ;
                     011D    443
                     011D    444 SET_NUMB:
48 AA    B0    011D    445          MOVW    IFB$L_DEVBUFSIZ(R10),-  ; reset numb to blocksize
14 A4          0120    446                  BDB$W_NUMB(R4)
               0122    447
```

```
                     0122   446 ;
                     0122   449 ; entry to set beginning and end of current buffer addresses
                     0122   450 ; r4 = bdb address
                     0122   451 ;
                     0122   452
                     0122   453 RM$MAPBLK1::
 20 A9   54   D0     0122   454         MOVL    R4,IRB$L_CURBDB(R9)     ; set current bdb
                     0126   455
                     0126   456 ;
                     0126   457 ;  entry to set beginning and end of current buffer addresses but without
                     0126   458 ;  setting irb$l_curbdb to the bdb address in r4
                     0126   459 ;
                     0126   460
                     0126   461 MAPBLK1ALT:
 51   48 A4   9A     0126   462         MOVZBL  BDB$B_REL_VBN(R4),R1    ; get current relative block
       05   E0     012A   463         BBS     #DEV$V_SQD,-            ; branch if magtape
            6A     012C   464                 IFB$L_PRIM_DEV(R10),-
            23     012D   465                 MTABLS
 57   48 AA   D0     012E   466         MOVL    IFB$L_DEVBUFSIZ(R10),R7 ; get length of block
    51   57   C4     0132   467         MULL2   R7,R1                   ; convert to buffer offset
 51   18 A4   C0     0135   468 SETR1:  ADDL2   BDB$L_ADDR(R4),R1       ; get buffer addr for block
    00A8 CA   B5     0139   469         TSTW    IFB$W_BUFFER_OFFSET(R10); is there a non-0 buffer offset?
       0A   13     013D   470         BEQLU   NO_BUFF_OFF             ; if not, branch around
51  00A8 CA   A0     013F   471         ADDW2   IFB$W_BUFFER_OFFSET(R10),R1  ; add offset to buffer addr
57  00A8 CA   A2     0144   472         SUBW2   IFB$W_BUFFER_OFFSET(R10),R7  ; subtract from buff length
                     0149   473
                     0149   474 NO_BUFF_OFF:
 4C A4   51   D0     0149   475         MOVL    R1,BDB$L_CURBUFADR(R4)  ; save current buffer address
    57   51   C0     014D   476         ADDL2   R1,R7                   ; set ending address
            05     0150   477 EXIT:    RSB                             ; and return
                     0151   478
                     0151   479 ;
                     0151   480 ;  set block size for magtape (it may have been a short block)
                     0151   481 ;
                     0151   482
 57   14 A4   3C     0151   483 MTABLS: MOVZWL  BDB$W_NUMB(R4),R7       ; set block size from # bytes read
       DE   11     0155   484         BRB     SETR1                   ; rejoin common code
```

```
                     0157   486
                     0157   487 ;
                     0157   488 ;  this is a "no read" request (to get a buffer for a write operation).
                     0157   489 ;  check for past high block and, if so, allocate the file space now, before
                     0157   490 ;  the buffer is used.
                     0157   491 ;
                     0157   492
                     0157   493 CHK_EXTEND:
      70 AA    C3    0157   494          SUBL3   IFB$L_HBK(R10),-        ; calc # of blks past HBK
52    1C A4          015A   495                  BDB$L_VBN(R4),R2
         BE    1B    015D   496          BLEQU   SET_NUMB                ; branch if not past HBK
         53    DD    015F   497          PUSHL   R3                      ; save flags
       FE9C'   30    0161   498          BSBW    RM$AUTOEXTEND           ; go do the extend
         08    BA    0164   499          POPR    #^M<R3>                 ; restore flags
      B4 50    E8    0166   500          BLBS    R0,SET_NUMB             ; branch on success
                     0169   501
                     0169   502 ;
                     0169   503 ; fall thru to handle error
                     0169   504 ;
                     0169   505
                     0169   506 ;
                     0169   507 ;  RM$AUTOEXTEND or RM$SEQRD returned an error
                     0169   508 ; check for eof error and if eof check for non-zero
                     0169   509 ; xfer size, in which case return the partial buffer
                     0169   510 ;
                     0169   511
                     0169   512 ERRXFR:
827A 8F    50    B1  0169   513          CMPW    R0,#RMS$_EOF&^XFFFF     ; is error eof?
         28    12    016E   514          BNEQ    10$                     ; branch if not
      0E A9    B0    0170   515          MOVW    IRB$W_IOS2(R9),-        ; set xfer size in bdb
      14 A4          0173   516                  BDB$W_NUMB(R4)
         14    13    0175   517          BEQL    5$                      ; and branch if zero
                     0177   518
                     0177   519 ;
                     0177   520 ; eof was seen but some data was transferred
                     0177   521 ; compute the number of valid blocks
                     0177   522 ;
                     0177   523
         01    88    0177   524          BISB    #1@BDB$V_VAL,-          ; say buffer valid
      0A A4          0179   525                  BDB$B_FLGS(R4)
      49 A4    96    017B   526          INCB    BDB$B_VAL_VBNS(R4)      ; say 1 block valid
               00    017E   527          RMSSUC                          ; indicate success
      55 A9    95    0181   528          TSTB    IRB$B_MBC(R9)           ; multi-block buffer?
         02    13    0184   529          BEQL    1$                      ; branch if not
                     0186   530
                     0186   531 ;
                     0186   532 ; recompute # of valid blocks for disk (intergral # of blocks)
                     0186   533 ;
                     0186   534
         8B    11    0186   535          BRB     SET_VBNS
       FF97    31    0188   536 1$:      BRW     RM$MAPBLK1
                     018B   537
                     018B   538 ;
                     018B   539 ; true eof
                     018B   540 ;
                     018B   541
01100004 8F    D3    018B   542 5$:      BITL    #DEV$M_TRM!DEV$M_MBX!DEV$M_FOR,-
```

```
                                                H 3
                                0191    543                     IFB$L_PRIM_DEV(R10)
    6A      12  0192    544             BNEQ    10$                     ; don't latch eof for terminal,
                0194    545
                0194    546 ;
                0194    547 ;  -ailbox, or foreign devices
                0194    548 ;
                0194    549
                0194    550             SSB     #IRB$V_EOF,(R9)         ; else set eof flag
    01      BB  0198    551 10$:        PUSHR   #^M<R0>
  FE63'     30  019A    552             BSBW    RM$QUIET_SEQMBF         ; flush all rah/wbh
03 50       E9  019D    553             BLBC    R0,20$                  ; branch if wbe errors
    01      BA  01A0    554             POPR    #^M<R0>
            05  01A2    555             RSB
    02      BA  01A3    556 20$:        POPR    #^M<R1>                 ; get rid of saved status
            05  01A5    557             RSB                             ; and exit
                01A6    558
                01A6    559             .END
```

```
$$.PSECT_EP                     =  00000000
$$RMSTEST                       =  0000001A
$$RMS_PBUGCHK                   =  00000010
$$RMS_TBUGCHK                   =  00000008
$$RMS_UMODE                     =  00000004
BDB$B_FLGS                      =  0000000A
BDB$B_REL_VBN                   =  00000048
BDB$B_VAL_VBNS                  =  00000049
BDB$L_ADDR                      =  00000018
BDB$L_CURBUFADR                 =  0000004C
BDB$L_IOSB                      =  00000048
BDB$L_VBN                       =  0000001C
BDB$V_DRT                       =  00000001
BDB$V_IOP                       =  00000002
BDB$V_VAL                       =  00000000
BDB$W_NUMB                      =  00000014
CHK_EXTEND                        00000157 R      01
DEV$M_FOR                       =  01000000
DEV$M_MBX                       =  00100000
DEV$M_TRM                       =  00000004
DEV$V_RND                       =  0000001C
DEV$V_SQD                       =  00000005
ERRXFR                            00000169 R      01
EXIT                              00000150 R      01
GETBLK1                           000000EB R      01
GETBLK1ALT                        000000FF R      01
IFB$L_DEVBUFSIZ                 =  00000048
IFB$L_EBK                       =  00000074
IFB$L_HBK                       =  00000070
IFB$L_PRIM_DEV                  =  00000000
IFB$W_BUFFER_OFFSET             =  000000A8
IRB$B_MBC                       =  00000055
IRB$L_CURBDB                    =  00000020
IRB$L_NRP_VBN                   =  00000040
IRB$L_NXTBDB                    =  0000003C
IRB$V_EOF                       =  00000021
IRB$V_RAHWBH                    =  0000002A
IRB$W_IOS2                      =  0000000E
IRB$W_NRP_OFF                   =  00000044
MAPBLK1ALT                        00000126 R      01
MTABLS                            00000151 R      01
NO_BUFF_OFF                       00000149 R      01
PIO$A_TRACE                       ********    X   01
RAB$B_RAC                       =  0000001E
RAB$C_SEQ                       =  00000000
RAB$L_ROP                       =  00000004
RAB$V_RAH                       =  00000009
READAHEAD                         0000006A R      01
RELEASE                           00000063 R      01
RM$AUTOEXTEND                     ********    X   01
RM$GETBLKNRP                      0000000C RG     01
RM$MAPBLK1                        00000122 RG     01
RM$NXTBLK1                        00000000 RG     01
RM$QUIET_SEQMBF                   ********    X   01
RM$RELBLK1                        ********    X   01
RM$SEQFLNKBDB                     ********    X   01
RM$SEQRAH                         ********    X   01
RM$SEQRD                          ********    X   01
RM$STALLRAHWBH                    ********    X   01
RMS$_EOF                        =  0001827A
ROP                             =  00000020
SETR1                             00000135 R      01
SET_NUMB                          0000011D R      01
SET_VBNS                          00000113 R      01
TPT$L_NXTBLK1                     ********    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
RMS$RMS1	000001A6 (422.)	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	35	00:00:00.08	00:00:00.63
Command processing	132	00:00:00.77	00:00:05.40
Pass 1	282	00:00:08.37	00:00:23.30
Symbol table sort	0	00:00:01.08	00:00:02.32
Pass 2	109	00:00:02.02	00:00:06.11
Symbol table output	9	00:00:00.09	00:00:00.19
Psect synopsis output	2	00:00:00.03	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	571	00:00:12.44	00:00:38.01

The working set limit was 1350 pages.
47211 bytes (93 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 858 non-local and 21 local symbols.
559 source lines were read in Pass 1, producing 14 object records in Pass 2.
20 pages of virtual memory were used to define 19 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	10
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	15

952 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM1NXTBLK/OBJ=OBJ$:RM1NXTBLK MSRC$:RM1NXTBLK/UPDATE=(ENH$:RM1NXTBLK)+EXECML$/LIB+LIB$:RMS/LIB
```

RM1PUTREC
LIS

RM1PUTSET
LIS

RM1UPDATE
LIS

RM1NXTBLK
LIS

RM1PUTBLD
LIS

RM1RELBLK
LIS

RM1SEQXFR
LIS

RM1PUT
LIS

RM2CONN
LIS

RM1OPEN
LIS

RM1WTLST
LIS

RM1STMFMT
LIS