```
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    SSSSSSSS    000000    BBBBBBBB    LL          KK      KK    IIIIII      000000
RRRRRRRR    MM      MM    SSSSSSSS    000000    BBBBBBBB    LL          KK      KK    IIIIII      000000
RR      RR  MMMM  MMMM  SS          00      00  BB      BB  LL          KK      KK      II      00      00
RR      RR  MMMM  MMMM  SS          00      00  BB      BB  LL          KK      KK      II      00      00
RR      RR  MM  MM  MM  SS          00    0000  BB      BB  LL          KK    KK        II      00      00
RR      RR  MM  MM  MM  SS          00    0000  BB      BB  LL          KK    KK        II      00      00
RRRRRRRR    MM      MM    SSSSSS    00  00  00  BBBBBBBB    LL          KKKKKK          II      00      00
RRRRRRRR    MM      MM    SSSSSS    00  00  00  BBBBBBBB    LL          KKKKKK          II      00      00
RR  RR      MM      MM          SS  0000    00  BB      BB  LL          KK    KK        II      00      00
RR  RR      MM      MM          SS  0000    00  BB      BB  LL          KK    KK        II      00      00
RR    RR    MM      MM          SS  00      00  BB      BB  LL          KK      KK      II      00      00      ....
RR    RR    MM      MM          SS  00      00  BB      BB  LL          KK      KK      II      00      00      ....
RR      RR  MM      MM    SSSSSSSS    000000    BBBBBBBB    LLLLLLLLLL  KK      KK    IIIIII      000000      ....
RR      RR  MM      MM    SSSSSSSS    000000    BBBBBBBB    LLLLLLLLLL  KK      KK    IIIIII      000000      ....

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          $BEGIN  RMSOBLKIO,000,RM$RMS,<BLOCK I/O ROUTINES>
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 module performs the $read and $write functions.
0000     33 ;
0000     34 ; Environment:
0000     35 ;                    star processor running starlet exec.
0000     36 ;
0000     37 ; Author:            L F Laverdure,   creation date: 23-MAY-1977
0000     38 ;
0000     39 ; Modified By:
0000     40 ;
0000     41 ;       V03-018 JWT0167         Jim Teague              15-Mar-1984
0000     42 ;               Back out buffer offset for block i/o.
0000     43 ;
0000     44 ;       V03-017 JWT0165         Jim Teague              10-Mar-1984
0000     45 ;               ANSI Magtape buffer offset for block i/o.
0000     46 ;
0000     47 ;       V03-016 DAS0006         David Solomon           6-Feb-1984
0000     48 ;               Fix BI journaling of $WRITE to reset correct value for
0000     49 ;               journal BDB BDB$W_NUMB field. Improve comments and formatting.
0000     50 ;
0000     51 ;       V03-015 DAS0005         David Solomon           15-Dec-1983
0000     52 ;               Don't blindly overwrite IRB$L_JNLBDB if it's already filled in.
0000     53 ;               Reuse existing JNLBDB if it's big enough. Also, don't deallocate
0000     54 ;               JNLBDB after each block I/O write.
0000     55 ;
0000     56 ;       V03-014 KPL0009         Peter Lieberwirth       20-Oct-1983
0000     57 ;               Page-align request for ALDJNLBUF.  Fix block-IO
0000     58 ;               journaling so that AT journaling structures are not
0000     59 ;               deallocated on a block-IO operation.
0000     60 ;
0000     61 ;       V03-013 TSK0004         Tamar Krichevsky        5-Oct-1983
0000     62 ;               Use RM$ALDJNLBUF instead of RM$ALDBUF for allocating
0000     63 ;               journal buffer and bdb.
0000     64 ;
0000     65 ;       V03-012 DAS0004         David Solomon           13-Jul-1983
0000     66 ;               Fix AI block I/O journaling.
0000     67 ;
0000     68 ;       V03-011 KPL0008         Peter Lieberwirth       26-May-1983
0000     69 ;               New format of RJR.
0000     70 ;
0000     71 ;       V03-010 RAS0151         Ron Schaefer            29-Apr-1983
0000     72 ;               Fix broken branches to RM$RDBUFWT and RM$WTBUFWT.
0000     73 ;
0000     74 ;       V03-009 RAS0132         Ron Schaefer            16-Mar-1983
0000     75 ;               Merge $RMSRDEF into $RJRDEF and revise the interface
0000     76 ;               for RM$WRTJNL for easier use from ISAM.
0000     77 ;
0000     78 ;       V03-008 DAS0003         David Solomon           18-Feb-1983
0000     79 ;               Fix date in V03-007.
0000     80 ;
0000     81 ;       V03-007 JWH0186         Jeffrey W. Horn         14-Feb-1983
0000     82 ;               Fix truncation error.
0000     83 ;
0000     84 ;       V03-006 JWH0157         Jeffrey W. Horn         17-Dec-1982
```

```
0000    85 ;               Add journaling support for block/io.
0000    86 ;
0000    87 ;       V03-005 KBT0420         Keith B. Thompson       30-Nov-1982
0000    88 ;               Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000    89 ;
0000    90 ;       V03-004 KBT0174         Keith B. Thompson       23-Aug-1982
0000    91 ;               Reorganize psects
0000    92 ;
0000    93 ;       V03-003 RAS0093         Ron Schaefer    27-Jul-1982
0000    94 ;               Correct block skipping for mixed block/record and/or connect
0000    95 ;               to EOF.  Zero the NRP offset within block after any block I/O
0000    96 ;               operation.
0000    97 ;
0000    98 ;       V03-002 RAS0091         Ron Schaefer    14-Jun-1982
0000    99 ;               Fix incorrect datatype usage.
0000   100 ;
0000   101 ;       V03-001 CDS0002         C Saether       2-Apr-1982
0000   102 ;               Clear IRB$V_BIO_LAST on buffer errors so that
0000   103 ;               subsequent disconnect will correctly release buffer.
0000   104 ;
0000   105 ;       V02-029 RAS0065         Ron Schaefer    8-Feb-1982
0000   106 ;               Position to end of file on block mode append.
0000   107 ;
0000   108 ;       V02-028 CDS0001         C Saether       30-Aug-1981
0000   109 ;               Remove calls to cache and release.
0000   110 ;
0000   111 ;       V02-027 RAS0021         Ron Schaefer    7-Aug-1981
0000   112 ;               Fix block i/o support for stream files.  This is an
0000   113 ;               interim fix to even-align write transfers by asking
0000   114 ;               if the user buffer can be extended 1 byte.
0000   115 ;               If not, the transfer fails.
0000   116 ;
0000   117 ;       V02-026 PSK00018        P S Knibbe      24-Jul-1981
0000   118 ;               Fix yet more broken branches, by keeping the network code in
0000   119 ;               the local PSECT.
0000   120 ;
0000   121 ;       V02-025 KPL0007         Peter Lieberwirth       7-May-1981
0000   122 ;               Fix more broken branches.
0000   123 ;
0000   124 ;       V02-024 PSK0017         P S Knibbe      19-Jan-1981
0000   125 ;               Fix broken branches
0000   126 ;
0000   127 ;       V02-023 REFORMAT        C Saether       30-Jul-1980     23:20
0000   128 ;
0000   129 ;       V022    PSK0016         P S Knibbe      11-Mar-1980      3:15
0000   130 ;               successful read from magtape should clear irb$v_eof bit
0000   131 ;               only successful write should set irb$v_eof bit
0000   132 ;
0000   133 ;       V021    PSK0012         P S Knibbe      14-Feb-1980      2:30
0000   134 ;               write to magtape should set irb$v_eof bit
0000   135 ;
0000   136 ;       V020    JAK0020         J A Krycka      05-Sep-1979     12.00
0000   137 ;               release 2.0 work.
0000   138 ;
0000   139 ;       V019    RAN0003         R A Newell      20-Dec-1978     03:10
0000   140 ;               file sharing enhancements.
0000   141 ;--
```

```
                        0000   142 ;
                        0000   143
```

```
                     0000   145          .SBTTL  DECLARATIONS
                     0000   146
                     0000   147 ;
                     0000   148 ; Include Files:
                     0000   149 ;
                     0000   150
                     0000   151 ;
                     0000   152 ; Macros:
                     0000   153 ;
                     0000   154
                     0000   155          $IFBDEF
                     0000   156          $IRBDEF
                     0000   157          $RABDEF
                     0000   158          $FABDEF
                     0000   159          $DEVDEF
                     0000   160          $RLSDEF
                     0000   161          $BDBDEF
                     0000   162          $RMSDEF
                     0000   163          $CJFDEF
                     0000   164          $RJRDEF
                     0000   165
                     0000   166 ;
                     0000   167
                     0000   168 ;
                     0000   169 ; Equated Symbols:
                     0000   170 ;
                     0000   171
00000020             0000   172          ROP=RAB$L_ROP*8                 ; bit offset to record options
                     0000   173
                     0000   174 ;
                     0000   175 ; Own Storage:
                     0000   176 ;
                     0000   177
```

```
0000   179          .SBTTL  RMS$READ - $READ ROUTINE
0000   180
0000   181 ;++
0000   182 ; RMS$READ - highest level block i/o read routine
0000   183 ;
0000   184 ; this routine performs the $read specific code.
0000   185 ; it checks the user's parameters, calls the
0000   186 ; common block i/o setup, reads the buffer
0000   187 ; awaiting completion, sets the next block
0000   188 ; pointer and exits via the common block i/o
0000   189 ; co-routine.
0000   190 ;
0000   191 ; Calling sequence:
0000   192 ;
0000   193 ; entered from exec as a result of user's calling
0000   194 ; SYS$READ (e.g., by using the $READ macro).
0000   195 ;
0000   196 ; Input Parameters:
0000   197 ;
0000   198 ;       ap      user's argument list addr.
00C0   199 ;
0000   200 ; Implicit Inputs:
0000   201 ;
0000   202 ; the contents of the various rab fields
0000   203 ; (see functional spec for detailed list).
0000   204 ;
0000   205 ; Output Parameters:
0000   206 ;
0000   207 ;       r0      status code
0000   208 ;       r1      destroyed
0000   209 ;
0000   210 ; Implicit Outputs:
0000   211 ;
0000   212 ;       various fields of the rab are filled in
0000   213 ;       to reflect the status of the $read operation.
0000   214 ;       (see rms functional spec for list.)
0000   215 ;
0000   216 ;       the irab is similarly updated.
0000   217 ;
0000   218 ;       a completion ast is queued if specified in the user arg list.
0000   219 ;
0000   220 ; Completion Codes:
0000   221 ;
0000   222 ;       standard rms.
0000   223 ;
0000   224 ; Side Effects:
0000   225 ;
0000   226 ;       if file was opened with 'bro' option specifying
0000   227 ;       mixed block and record i/o operations, the
0000   228 ;       current and next record context is destroyed.
0000   229 ;
0000   230 ; Note on BRO_SW and BIO_LAST:
0000   231 ;
0000   232 ;       The BRO_SW is set in RMORSET when the current operation is a block i/o
0000   233 ;       operation and BIO_LAST is not set.  This is the signal to the blkio
0000   234 ;       co-routine to release the current buffer first before using the BDB
0000   235 ;       for the block i/o operation, as the buffer may be dirty and need to
```

```
0000    236 ;       be written first.  In addition, a disconnect will look at the BIO_LAST
0000    237 ;       for the same reason.  If BIO_LAST is set, it assumes that any necessary
0000    238 ;       release has already taken place.
0000    239 ;
0000    240 ;       BIO_LAST is set by RM0RSET before entering these routines.
0000    241 ;       For that reason, any errors that occur before the release section of
0000    242 ;       the BLKIO co-routine should clear BIO_LAST before exit, as the effect
0000    243 ;       of releasing the buffer from a record operation has not yet occured.
0000    244 ;
0000    245 ;--
0000    246
```

```
                     0000   248          $ENTRY  RMS$READ
                     0000   249          $TSTPT  READ
                     0006   250
                     0006   251          $RABSET FAC=IFB$V_GET,BIO=1,CFLG=1
                     000A   252
                     000A   253 ;
                     000A   254 ; initialize rbf and rsz and call blkio to do the i/o
                     000A   255 ;
                     000A   256
 56    20 A8    3C   000A   257          MOVZWL  RAB$W_USZ(R8),R6        ; get buffer size
          66    13   000E   258          BEQL    ERRUSZ                  ; branch if zero
 57    24 A8    D0   0010   259          MOVL    RAB$L_UBF(R8),R7        ; get buffer addr
 28 A8    57    D0   0014   260          MOVL    R7,RAB$L_RBF(R8)        ; set buffer addr
       22 A8    B4   0018   261          CLRW    RAB$W_RSZ(R8)           ; init read size
                     0013   262          CSB     #IRB$V_WRITE,(R9)       ; flag as read
         00A3   30   001F   263          BSBW    BLKIO                   ; do common block i/o setup
 03 6A    3E    E1   0022   264          BBC     #IFB$V_DAP,(R10),5$     ; branch if not network operation
         021A   31   0026   265          BRW     NTREAD                  ; branch to NTREAD if network operation
00000000'EF    16   0029   266 5$:      JSB     RM$RDBUFWT              ; issue the read and await completion
       5E 50    E9   002F   267          BLBC    R0,CHKEOF               ; branch on error
                     0032   268
                     0032   269 ;
                     0032   270 ;   on magtapes - clear the end of file bit on read
                     0032   271 ;
                     0032   272
 04 6A    05    E1   0032   273          BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R10),10$ ; branch if not magtape
                     0036   274          CSB     #IRB$V_EOF,(R9)         ; clear the eof bit
                     003A   275 10$:
                     003A   276
                     003A   277 ;
                     003A   278 ;  check for logical eof if sequential file org
                     003A   279 ;
                     003A   280
                     003A   281 CHKLEOF:
 32 6A    1C    E1   003A   282          BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R10),SETRSZ ; branch if not disk
          61    10   003E   283          BSBB    SETNBP                  ; set next block pointer
                     0040   284          ASSUME  FAB$C_SEQ EQ 0
       23 AA    95   0040   285          TSTB    IFB$B_ORGCASE(R10)      ; sequential org?
          2B    12   0043   286          BNEQ    SETRSZ                  ; branch if not
                     0045   287
 74 AA    51    D1   0045   288          CMPL    R1,IFB$L_EBK(R10)       ; last block xferred > eof?
          25    1F   0049   289          BLSSU   SETRSZ                  ; branch if not
 74 AA    55    D1   004B   290          CMPL    R5,IFB$L_EBK(R10)       ; 1st block past eof?
          0C    1F   004F   291          BLSSU   20$                     ; branch if definitely not
          05    1A   0051   292          BGTRU   10$                     ; branch if yes
       5C AA    B5   0053   293          TSTW    IFB$W_FFB(R10)          ; 1st block is eof block
          05    12   0056   294          BNEQ    20$                     ; branch if any bytes in use
                     0058   295                                          ; (i.e., not yet eof)
                     0058   296 ;
                     0058   297 ;  past logical eof
                     0058   298 ;  reset numb to reflect the number of bytes transferred before logical eof.
                     0058   299 ;
                     0058   300
                     0058   301 10$:     RMSERR  EOF
51   74 AA    55    C3   005D   302 20$:     SUBL3   R5,IFB$L_EBK(R10),R1    ; get # full vbn's before eof
          11    19   0062   303          BLSS    RETURN                  ; if < 0 just return (rsz = 0)
   51    51    09    9C   0064   304          ROTL    #9,R1,R1                ; get # of bytes
```

```
                14 A4   51   5C AA   A1  0068   305          ADDW3   IFB$W_FFB(R10),R1,BDB$W_NUMB(R4) ; + last block offset = total # byt
                                31   10  006E   306          BSBB    SETNBP                          ; reset next block pointer
                                                0070   307
                                                0070   308 ;
                                                0070   309
                    22 A8   14 A4   B0  0070   310 SETRSZ: MOVW    BDB$W_NUMB(R4),RAB$W_RSZ(R8) ; set length of read
                                                05  0075   311 RETURN: RSB                             ; return to block i/o co-routine
```

```
                     0076   313
                     0076   314 ;++
                     0076   315 ;
                     0076   316 ; handle errors
                     0076   317 ;
                     0076   318 ;--
                     0076   319
                     0076   320 ERRUSZ:
                     0076   321          RMSERR  USZ
          05   11    007B   322          BRB     EXIT1
                     007D   323
                     007D   324 ERRUBF:
                     007D   325          RMSERR  UBF
                     0082   326
                     0082   327 EXIT1:   CSB     #IRB$V_BIO_LAST, (R9)   ; No buffers were released, so
                     0086   328                                           ; don't consider this operation as last
        FF77'   31   0086   329          BRW     RM$EXRMS
                     0089   330
                     0089   331 ;++
                     0089   332 ;
                     0089   333 ;  handle $write errors
                     0089   334 ;
                     0089   335 ;--
                     0089   336
                     0089   337 ERRRBF:
                     0089   338          RMSERR  RBF                     ; bad buffer address
          F2   11    008E   339          BRB     EXIT1
                     0090   340
                     0090   341 ;++
                     0090   342 ;
                     0090   343 ;  handle eof error
                     0090   344 ;
                     0090   345 ;--
                     0090   346
    827A 8F    50   B1   0090   347 CHKEOF: CMPW    R0,#RMS$_EOF&^XFFFF     ; was error eof?
               DE   12   0095   348          BNEQ    RETURN                  ; branch if not
          14   A4   B5   0097   349          TSTW    BDB$W_NUMB(R4)          ; any input?
               D4   13   009A   350          BEQL    SETRSZ                  ; branch if none (it's eof)
                    009C   351          RMSSUC                          ; return success instead
               99   11   009F   352          BRB     CHKLEOF                 ; go check for logical eof
```

```
                     00A1   354
                     00A1   355 ;++
                     00A1   356 ;
                     00A1   357 ;  subroutine to set next block pointer
                     00A1   358 ;
                     00A1   359 ; inputs:
                     00A1   360 ;       r9              irab addr
                     00A1   361 ;       r5              starting vbn for xfer
                     00A1   362 ;       r4              bdb addr
                     00A1   363 ;       bdb$w_numb      # bytes xfered
                     00A1   364 ;
                     00A1   365 ;  outputs:
                     00A1   366 ;       irb$l_nrp_vbn   vbn of block following last transferred
                     00A1   367 ;       r1              last vbn xferred
                     00A1   368 ;       r2              # bytes xferred in last blk
                     00A1   369 ;
                     00A1   370 ; \assumes vbn significant for disk devices only\
                     00A1   371 ;--
                     00A1   372
      51   14 A4   3C 00A1   373 SETNBP: MOVZWL  BDB$W_NUMB(R4),R1        ; get xfer len
52    51 FE00 8F   AB 00A5   374         BICW3   #^XFE00,R1,R2             ; get # bytes in last blk
              05   12 00AB   375         BNEQ    10$                       ; branch if non-zero
   52    0200 8F   B0 00AD   376         MOVW    #512,R2                   ; must have been 512!
              51   D7 00B2   377 10$:     DECL    R1                        ; round down # bytes
 51   51   F7 8F   78 00B4   378         ASHL    #-9,R1,R1                 ; get # vbn's xferred - 1
         51   55   C0 00B9   379         ADDL2   R5,R1                     ; set last vbn xferred
40 A9    01   51   C1 00BC   380         ADDL3   R1,#1,IRB$L_NRP_VBN(R9)   ; set nbp
            44 A9   B4 00C1   381         CLRW    IRB$W_NRP_OFF(R9)         ; zero offset within vbn
              05   00C4   382         RSB
```

```
00C5          .SBTTL  BLKIO - COMMON $READ - $WRITE ROUTINE
00C5    ;++
00C5    ; BLKIO - common block i/o read/write routine
00C5    ;
00C5    ;       this routine performs common $read - $write function
00C5    ;       setup and cleanup.  it functions as a co-routine
00C5    ;       so that the $read or $write routine need
00C5    ;       merely do an rsb to execute the common
00C5    ;       block i/o cleanup.
00C5    ;
00C5    ; Calling sequence:
00C5    ;
00C5    ;       BSBW    BLKIO
00C5    ;
00C5    ; Input Parameters:
00C5    ;
00C5    ;       r11     impure area address
00C5    ;       r10     ifab addr
00C5    ;       r9      irab addr
00C5    ;       r8      rab addr
00C5    ;       r7      user buffer addr
00C5    ;       r6      user buffer length
00C5    ;
00C5    ; Implicit Inputs:
00C5    ;
00C5    ;       irb$l_curbdb
00C5    ;       irb$l_nrp_vbn
00C5    ;       rab$l_bkt
00C5    ;       irb$w_bkpbits
00C5    ;
00C5    ; Output Parameters:
00C5    ;
00C5    ;       r5      vbn for 1st block of xfer
00C5    ;       r4      current bdb addr
00C5    ;       r0-r3   destroyed
00C5    ;
00C5    ; Implicit Outputs:
00C5    ;
00C5    ;       current bdb possibly released
00C5    ;       new current bdb locked
00C5    ;       bdb setup for xfer
00C5    ;       rab$w_rfa set to starting vbn
00C5    ;
00C5    ; Completion Codes:
00C5    ;
00C5    ;       none.  exits rms if any errors.
00C5    ;
00C5    ; Side Effects:
00C5    ;
00C5    ;       bdb buffer information saved in irab,
00C5    ;       hence co-routine must be called in
00C5    ;       order to clean up on exit.
00C5    ;--
00C5
00C5    BLKIO:
00C5
00C5    ;
```

```
                        00C5   441 ;  probe the user's buffer for appropriate access
                        00C5   442 ;
                        00C5   443
              56   B5   00C5   444          TSTW    R6                      ; zero length?
              28   13   00C7   445          BEQL    40$                     ; branch if yes (avoiding probe)
        54    56   7D   00C9   446          MOVQ    R6,R4                   ; copy size and address
50      FE00  8F   32   00CC   447          CVTWL   #-512,R0                ; addressing constant
  08    69    29   E0   00D1   448 10$:     BBS     #IRB$V_WRITE,(R9),20$   ; branch if write
65      54    00   0D   00D5   449          PROBEW  #0,R4,(R5)              ; probe writeability
              08   12   00D9   450          BNEQ    30$                     ; branch if o.k.
              A0   11   00DB   451          BRB     ERRUBF                  ; go handle error
                        00DD   452 20$:     IFNORD  R4,(R5),ERRRBF          ; branch if not readable
        55    50   C2   00E3   453 30$:     SUBL2   R0,R5                   ; get address of next page
     54     6440  3E   00E6   454          MOVAW   (R4)[R0],R4             ; calculate new length
              E5   14   00EA   455          BGTR    10$                     ; branch if more to probe
        54    50   C2   00EC   456          SUBL2   R0,R4                   ; need to handle last page?
              E0   14   00EF   457          BGTR    10$                     ; branch if yes
                        00F1   458 40$:
                        00F1   459
                        00F1   460 ;
                        00F1   461 ;  get block number for i/o
                        00F1   462 ;
                        00F1   463
  55    38    A8   D0   00F1   464          MOVL    RAB$L_BKT(R8),R5        ; get block #
              0F   12   00F5   465          BNEQ    RELBDB                  ; branch if non-zero
                        00F7   466
                        00F7   467 ;
                        00F7   468 ; sequential operation indicated - set nbp from saved value
                        00F7   469 ;
                        00F7   470
  55    40    A9   D0   00F7   471          MOVL    IRB$L_NRP_VBN(R9),R5
        44    A9   B5   00FB   472          TSTW    IRB$L_NRP_OFF(R9)
              06   13   00FE   473          BEQL    RELBDB
02      69    21   E1   0100   474          BBC     #IRB$V_EOF,(R9),RELBDB
              55   D6   0104   475          INCL    R5
                        0106   476
                        0106   477 ;
                        0106   478 ; release the current bdb if any, unless
                        0106   479 ; it is for this vbn and we're writing
                        0106   480 ;
                        0106   481
                        0106   482 RELBDB:
54      20    A9   D0   0106   483          MOVL    IRB$L_CURBDB(R9),R4     ; get current bdb
              2A   13   010A   484          BEQL    GETBDB                  ; branch if none
0A      69    28   E0   010C   485          BBS     #IRB$V_BRO_SW,(R9),RELEASE ; always release if
                        0110   486                                          ; record i/o operation last
06      69    29   E1   0110   487          BBC     #IRB$V_WRITE,(R9),RELEASE ; always release on read
55      1C    A4   D1   0114   488          CMPL    BDB$L_VBN(R4),R5        ; same block?
              39   13   0118   489          BEQL    SETBDB                  ; yes (omit release)
                        011A   490
                        011A   491 ;
                        011A   492 ; must release bdb - use rm$wtlst1 for sequential org if irb$v_bro_sw set,
                        011A   493 ;  else rm$release
                        011A   494 ;
                        011A   495
                        011A   496 RELEASE:
                        011A   497          ASSUME  FAB$C_SEQ EQ 0
```

```
                23 AA   95   011A   498          TSTB    IFB$B_ORGCASE(R10)       ; seq org?
                   1E   12   011D   499          BNEQ    REL23                    ; branch if not
            09 69   28   E1   011F   500          BBC     #IRB$V_BRO_SW,(R9),GBDB0 ; branch if block i/o last
                                0123   501                                           ; (omitting unneeded release)
          00000000'EF   16   0123   502          JSB     RM$WTLST1                ; release last block (with padding)
                51 50   E9   0129   503          BLBC    R0,RELERR                ; branch on error
            54    3C A9   D0   012C   504 GBDB0:   MOVL    IRB$L_NXTBDB(R9),R4      ; get bdb addr to use
            1C A4   55   D0   0130   505          MOVL    R5,BDB$L_VBN(R4)         ; and set vbn into it
                   1D   11   0134   506          BRB     SETBDB
                                0136   507
                                0136   508 ;
                                0136   509 ;  need merely get a bdb (no release required).  branch based on file org.
                                0136   510 ;
                                0136   511
                                0136   512          ASSUME  FAB$C_SEQ EQ 0
                23 AA   95   0136   513 GETBDB: TSTB    IFB$B_ORGCASE(R10)       ; seq. file org?
                   F1   13   0139   514          BEQL    GBDB0                    ; branch if yes
                   03   11   013B   515          BRB     GBDB23                   ; branch for rel. and idx.
                                013D   516
                                013D   517 ;
                                013D   518 ;  release last block for relative and indexed file orgs
                                013D   519 ;  and get a bdb for the new operation
                                013D   520 ;
                                013D   521
                                013D   522 REL23:
                20 A9   D4   013D   523          CLRL    IRB$L_CURBDB(R9)         ; note no bdb
            54    40 AA   DE   0140   524 GBDB23: MOVAL   IFB$L_BDB_FLNK(R10), R4 ; get bdb for operation
                                0144   525          ASSUME  BDB$L_FLINK EQ 0
               54   64   D0   0144   526 10$:     MOVL    (R4), R4                 ; get next one.
                 0C A4   B5   0147   527          TSTW    BDB$W_USERS(R4)          ; in use?
                   F8   12   014A   528          BNEQ    10$                      ; NEQ then someone has it.
                 0C A4   B6   014C   529          INCW    BDB$W_USERS(R4)          ; bump count (to 1)
            1C A4   55   D0   014F   530          MOVL    R5, BDB$L_VBN(R4)        ; note VBN to transfer.
                                0153   531 SETBDB:
            20 A9   54   D0   0153   532          MOVL    R4,IRB$L_CURBDB(R9)      ; note new current bdb
                                0157   533
                                0157   534 ;
                                0157   535 ; save size and address of buffer associated
                                0157   536 ; with bdb in rp field of irab
                                0157   537 ;
                                0157   538
        48 A9   18 A4   D0   0157   539          MOVL    BDB$L_ADDR(R4),IRB$L_RP_VBN(R9)
        4C A9   16 A4   B0   015C   540          MOVW    BDB$W_SIZE(R4),IRB$W_RP_OFF(R9)
                                0161   541
                                0161   542 ;
                                0161   543 ; store user-supplied buffer info
                                0161   544 ;
                                0161   545
            16 A4   56   B0   0161   546          MOVW    R6, BDB$W_SIZE(R4)
            14 A4   56   B0   0165   547          MOVW    R6, BDB$W_NUMB(R4)
            18 A4   57   D0   0169   548          MOVL    R7, BDB$L_ADDR(R4)
            10 A8   55   D0   016D   549          MOVL    R5, RAB$W_RFA(R8)        ; set rfa from vbn
                   9E   16   0171   550          JSB     @(SP)+                   ; co-routine call for caller
                                0173   551 ;
                                0173   552 ; finish block i/o co-routine
                                0173   553 ;
                                0173   554 ; called when read or write does 'rsb'
```

```
                     0173   555 ;
                     0173   556 ; restore real buffer info
                     0173   557 ; (r0 has status code)
                     0173   558 ;
                     0173   559
18 A4   48 A9   D0   0173   560          MOVL    IRB$L_RP_VBN(R9),BDB$L_ADDR(R4)
16 A4   4C A9   B0   0178   561          MOVW    IRB$W_RP_OFF(R9),BDB$W_SIZE(R4)
   0A A4    03   8A   017D   562 RELERR: BICB2   #BDB$M_VAL!BDB$M_DRT,BDB$B_FLGS(R4) ; clear valid and dirty
        1C A4   D4   0181   563          CLRL    BDB$L_VBN(R4)              ; clear vbn field
        0C A4   B4   0184   564          CLRW    BDB$W_USERS(R4)            ; not is use now.
        20 A9   D4   0187   565          CLRL    IRB$L_CURBDB(R9)           ; say no bdb
         FE73'  31   018A   566 EXIT:    BRW     RM$EXRMS
```

```
                        .SBTTL  RMS$WRITE - $WRITE ROUTINE

;++
; RMS$WRITE - highest level block i/o write routine
;
; this routine performs the $write specific code.
; it checks the user's parameters, calls the
; common block i/o setup, writes the buffer
; awaiting completion, sets the next block
; pointer and exits via the common block i/o
; co-routine.
;
; Calling sequence:
;
; entered from exec as a result of user's calling
; sys$write (e.g., by using the $write macro).
;
; Input Parameters:
;
;       ap      user's argument list addr.
;
; Implicit Inputs:
;
; the contents of the various rab fields
; (see functional spec for detailed list).
;
; Output Parameters:
;
;       r0      status code
;       r1      destroyed
;
; Implicit Outputs:
;
;       various fields of the rab are filled in
;       to reflect the status of the $write operation.
;       (see rms functional spec for list.)
;
;       the irab is similarly updated.
;
;       a completion ast is queued if specified in the user arg list.
;
; Completion Codes:
;
;       standard rms.
;
; Side Effects:
;
;       if file was opened with 'bro' option specifying
;       mixed block and record i/o operations, the
;       current and next record context is destroyed.
;
; Notes:
;       since the UNIBUS disks only allow even sized transfers, we
;       must lie to the driver about how many bytes are to be transfered
;       if the user gives an odd-sized buffer.
;       Currently the only case that is handled is $WRITE operations where
;       the next byte after the user-buffer is accessible, so that we can
```

```
                     018D   625 ;       just write one more byte(!)  The EOF mark is however properly
                     018D   626 ;       maintained.  This case occurs for the $COPY command for stream files.
                     018D   627 ;       SOMEDAY, we should fix all cases!!!
                     018D   628 ;
                     018D   629 ;       Also see note in the READ routine header regarding BRO_SW and BIO_LAST.
                     018D   630 ;
                     018D   631 ;--
                     018D   632
                     018D   633         $ENTRY  RMS$WRITE
                     018D   634         $TSTPT  WRITE
                     0193   635         $RABSET FAC=IFB$V_PUT,BIO=1,CFLG=1
                     0197   636
                     0197   637 ;
                     0197   638 ; get record size and address and probe the buffer
                     0197   639 ;
                     0197   640
56   22 A8   3C     0197   641         MOVZWL  RAB$W_RSZ(R8),R6        ; get size
                     019B   642
                     019B   643 ;
                     019B   644 ;  cope with the inadequacies of certain block-structured devices that
                     019B   645 ;  only permit even sized transfers (e.g. RK06/7, RL01/2, RX01/2,...)
                     019B   646 ;
   6A   1C   E1      019B   647         BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R10),-
                05   019E   648                 5$                      ; block structured device?
     02 56   E9      019F   649         BLBC    R6,5$                   ; even size buffer?
        56   D6      01A2   650         INCL    R6                      ; round buffer up if odd
                     01A4   651
57   28 A8   D0      01A4   652 5$:      MOVL    RAB$L_RBF(R8),R7        ; get addr
                     01A8   653         SSB     #IRB$V_WRITE,(R9)       ; flag as write
      FF16   30      01AC   654         BSBW    BLKIO                   ; do common block i/o setup
22 6A   3E   E0      01AF   655         BBS     #IFB$V_DAP,(R10),BR_AID ; WRITE ; branch if network operation
66 6A   1C   E1      01B3   656         BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R10),WT1 ; branch if not disk
                     01B7   657         ASSUME  FAB$C_SEQ EQ 0
      23 AA   95     01B7   658         TSTB    IFB$B_ORGCASE(R10)      ; is this the seq. file org?
         10   12     01BA   659         BNEQ    WRTBLK                  ; branch if not (no auto extend)
                     01BC   660
                     01BC   661 ;
                     01BC   662 ;  check for file extend needed
                     01BC   663 ;
                     01BC   664
         56   D7     01BC   665         DECL    R6                      ; round down # of bytes
52   56   F7 8F     78      01BE   666         ASHL    #-9,R6,R2               ; get # blks to xfer - 1
     52   55   C0     01C3   667         ADDL2   R5,R2                   ; get ending vbn for xfer
     52   70 AA   C2     01C6   668         SUBL2   IFB$L_HBK(R10),R2       ; compute # blocks to extend
             63   1A     01CA   669         BGTRU   EXTEND                  ; and branch if > 0
                     01CC   670
      00A0 CA   95     01CC   671 WRTBLK:  TSTB    IFB$B_JNLFLG(R10)       ; any journaling?
             06   13     01D0   672         BEQL    DOWRT                   ; skip if not
           0083   31     01D2   673         BRW     WRTJNL                  ; branch if so
                     01D5   674
           0061   31     01D5   675 BR_AID:  BRW     NTWRITE                 ; out-of-line branch aid
                     01D8   676
00000000'EF   16     01D8   677 DOWRT:   JSB     RMSWTBUFWT              ; write the block
      30 50   E9     01DE   678         BLBC    R0,10$                  ; branch on error
                     01E1   679
                     01E1   680 ;
                     01E1   681 ; if requested block is past current eof, reset eof
```

```
                     01E1    68  ;
          22 A8      B0  01E1    683          MOVW    RAB$W_RSZ(R8),-          ; reset transfer size
          14 A4          01E4    684                  BDB$W_NUMB(R4)           ; to number of real bytes
           FEB8      30  01E6    685          BSBW    SETNBP                   ; set next block pointer
       25 68   21    E0  01E9    686          BBS     #RAB$V_TPT+ROP,(R8),20$  ; branch if new eof wanted
       51   74 AA    D1  01ED    687          CMPL    IFB$L_EBK(R10),R1        ; past eof?
             1E      1A  01F1    688          BGTRU   10$                      ; branch if not
             06      1F  01F3    689          BLSSU   5$                       ; branch if definite yes
       5C AA   52    B1  01F5    690          CMPW    R2,IFB$W_FFB(R10)        ; past current eof byte?
             16      1B  01F9    691          BLEQU   10$                      ; branch if not
                     01FB    692
                     01FB    693  ;
                     01FB    694  ;  store the new eof
                     01FB    695  ;
                     01FB    696
                     01FB    697  5$:    SSB     #IFB$V_RW_ATTR,(R10)     ; flag attribute rewrite needed
       48 AA   52    B1  01FF    698          CMPW    R2,IFB$L_DEVBUFSIZ(R10)  ; is last block full?
             04      1F  0203    699          BLSSU   7$                       ; branch if not
             52      B4  0205    700          CLRW    R2                       ; yes - clear offset in block
             51      D6  0207    701          INCL    R1                       ; and bump eof vbn
       74 AA   51    D0  0209    702  7$:     MOVL    R1,IFB$L_EBK(R10)
       5C AA   52    B0  020D    703          MOVW    R2,IFB$W_FFB(R10)
             05      11  0211    704  10$:    BRB     RSB
                     0212    705
                     0212    706  ;
                     0212    707  ;  verify tpt option allowed by fac
                     0212    708  ;
                     0212    709
    E4 22 AA   04    E0  0212    710  20$:    BBS     #FAB$V_TRN,IFB$B_FAC(R10),5$ ; branch if allowed
                     0217    711          RMSERR  FAC                      ; give error
             05      21C     712          RSB                              ; and return to co-routine
                     021D    713
                     021D    714  ;
                     021D    715  ;  write for non disk devices
                     021D    716  ;
                     021D    717
 00000000'EF         16  021D    718  WT1:    JSB     RM$WTBUFWT
          08 50      E9  0223    719          BLBC    R0,10$                   ; go away if error
       04 6A   05    E1  0226    720          BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R10),10$ ; if magtape
                     022A    721          SSB     #IRB$V_EOF,(R9)          ; set the eof bit
                     022E    722  10$:
             05      22E     723          RSB
                     022F    724
                     022F    725  ;
                     022F    726  ;  extend file (i.e., for disk only)
                     022F    727  ;
                     022F    728
 00000000'EF         16  022F    729  EXTEND: JSB     RM$AUTOEXTEND
          94 50      E8  0235    730          BLBS    R0,WRTBLK                ; if successful, rewrite
             05      238     731          RSB
                     0239    732
                     0239    733  ;++
                     0239    734  ;
                     0239    735  ;  perform network write function
                     0239    736  ;
                     0239    737  ;--
                     0239    738
```

```
                     0239   739 NTWRITE:
 FDC4'  30           0239   740          BSBW    NT$WRITE                ; write the block
03 50   E9           023C   741          BLBC    R0,10$                  ; branch on error
 FE5F   30           023F   742          BSBW    SETNBP                  ; update next block pointer
        05           0242   743 10$:     RSB                             ; return to co-routine
                     0243   744
                     0243   745 ;++
                     0243   746 ;
                     0243   747 ;  perform network read function
                     0243   748 ;
                     0243   749 ;--
                     0243   750
                     0243   751 NTREAD:
 FDBA'  30           0243   752          BSBW    NT$READ                 ; read the block
06 50   E9           0246   753          BLBC    R0,10$                  ; branch on error
 FE55   30           0249   754          BSBW    SETNBP                  ; update next block pointer
 FE21   31           024C   755          BRW     SETRSZ                  ; rejoin mainline
 FE3E   31           024F   756 10$:     BRW     CHKEOF                  ; branch aid
                     0252   757
```

```
                                0252   759 ;
                                0252   760 ; Journal the $WRITE.
                                0252   761 ;
                                0252   762
                                0252   763 ERRJNS: RMSERR  JNS                         ; give error
                             05 0257   764         RSB
                                0258   765
                         68  21 E0 0258   766 WRTJNL: BBS     #RAB$V_TPT+ROP,(R8),-    ; branch if truncate put
                             F6        025B   767                ERRJNS                ;
                             3C BB     025C   768         PUSHR   #^M<R2,R3,R4,R5>      ; save registers
                             01 DD     025E   769         PUSHL   #1                    ; preset success
                      55  14 A4 3C     0260   770         MOVZWL  BDB$W_NUMB(R4),R5     ; get size of user request
                55    00000044 8F C0 0264   771         ADDL    #RJR$C_BLKLEN,R5      ; compute size of journal buffer needed
                55    000001FF 8F C0 026B   772         ADDL2   #511,R5               ; round request to a page boundary
                55    000001FF 8F CA 0272   773         BICL2   #511,R5               ;
                      54  30 A9 D0 0279   774         MOVL    IRB$L_JNLBDB(R9),R4   ; get JNLBDB address
                         1D   13 027D   775         BEQL    7$                    ; none allocated; go allocate one
                                027F   776
                                027F   777 ;
                                027F   778 ; See if existing JNLBDB is big enough for this request. If so, re-use it.
                                027F   779 ; If not, deallocate it and allocate a bigger one.
                                027F   780 ;
                                027F   781
                      55  14 A4 B1 027F   782         CMPW    BDB$W_NUMB(R4),R5     ; compare buffer size to size needed
                         2E   13 0283   783         BEQLU   15$                   ; all set up, go use it
                      55  2C A4 B1 0285   784         CMPW    BDB$W_ALLOC_SIZE(R4),R5 ; compare allocation size to size needed
                         06   1F 0289   785         BLSSU   5$                    ; too small; deallocate it
                   14 A4    55   B0 028B   786         MOVW    R5,BDB$W_NUMB(R4)     ; size ok; set new buffer size
                         22   11 028F   787         BRB     15$                   ;  and go use it
                         55   DD 0291   788 5$:     PUSHL   R5                    ; save R5
                00000000'EF   16 0293   789         JSB     RM$RETJNLBDB          ; deallocate existing JNLBDB
                         55 8ED0 0299   790         POPL    R5                    ; restore R5
                00000000'EF   16 029C   791 7$:     JSB     RM$ALDJNLBUF          ; get BDB and buffer
                      06 50   E8 02A2   792         BLBS    R0,10$                ; continue on succcess
                   6E    50   D0 02A5   793         MOVL    R0,(SP)               ; replace status code
                         0098 31 02A8   794         BRW     40$
                14 A4    55   B0 02AB   795 10$:    MOVW    R5,BDB$W_NUMB(R4)     ; set buffer size
                30 A9    54   D0 02AF   796         MOVL    R4,IRB$L_JNLBDB(R9)   ; save journal BDB address
                                02B3   797
                                02B3   798 ;
                                02B3   799 ; Set up journaling record
                                02B3   800 ;
                                02B3   801
                55   18 A4    D0 02B3   802 15$:    MOVL    BDB$L_ADDR(R4),R5     ; get buffer address
                                02B7   803
                                02B7   804 ;
                                02B7   805 ; Fill in RJR overhead
                                02B7   806 ;
                                02B7   807
                      03   90   02B7   808         MOVB    #RJR$C_BLOCK,-        ; journal entry type
                   03 A5         02B9   809                 RJR$B_ENTRY_TYPE(R5)
                      1E   90   02BB   810         MOVB    #RJR$_WRITE,-         ; operation
                   05 A5         02BD   811                 RJR$B_OPER(R5)
                   23 AA   90   02BF   812         MOVB    IFB$B_ORGCASE(R10),-  ; file organization
                   04 A5         02C2   813                 RJR$B_ORG(R5)
                                02C4   814
                53   0C AE    D0 02C4   815         MOVL    12(SP),R3             ; get data BDB address
```

```
                1C A3    D0  02C8   816          MOVL    BDB$L_VBN(R3),-          ; fill in VBN
                3C A5        02CB   817                  RJR$L_BLOCK_VBN(R5)
                14 A3    3C  02CD   818          MOVZWL  BDB$W_NUMB(R3),-         ; fill in block size
                40 A5        02D0   819                  RJR$L_BLOCK_SIZE(R5)
                             02D2   820
                             02D2   821 ;
                             02D2   822 ; To BI journal this we need to read old block(s) into journal BDB.
                             02D2   823 ;
                             02D2   824
                    02   E1  02D2   825          BBC     #IFB$V_BI,-              ; branch if no BI
             3D 00A0 CA          02D4   826                  IFB$B_JNLFLG(R10),20$
                14 A3    B0  02D8   827          MOVW    BDB$W_NUMB(R3),-         ; copy number of bytes of user transfer
                14 A4        02DB   828                  BDB$W_NUMB(R4)
                1C A3    D0  02DD   829          MOVL    BDB$L_VBN(R3),-          ; copy vbn
                1C A4        02E0   830                  BDB$L_VBN(R4)
            00000044 8F      C0  02E2   831          ADDL    #RJR$T_BLOCK,-           ; read old data into RJR$T_BLOCK
                18 A4        02E8   832                  BDB$L_ADDR(R4)
            00000000'EF      16  02EA   833          JSB     RM$RDBUFWT               ; read the old block(s)
            00000044 8F      C2  02F0   834          SUBL2   #RJR$T_BLOCK,-           ; get back real top of buff
                18 A4        02F6   835                  BDB$L_ADDR(R4)
                0044 8F      A0  02F8   836          ADDW2   #RJR$T_BLOCK,-           ; restore correct jBDB buffer size
                14 A4        02FC   837                  BDB$W_NUMB(R4)           ; (user buffer size + RJR overhead)
                3F 50    E9  02FE   838          BLBC    R0,30$                   ; get out on error
                    54   DD  0301   839          PUSHL   R4                       ; jBDB addr
                    02   DD  0303   840          PUSHL   #CJF$_BI                 ; set BI
             54     14   AE  0305   841          MOVL    12+8(SP),R4              ; get data BDB address from saved R4
            00000000'EF      16  0309   842          JSB     RM$WRTJNL                ; write journal entry
                5E   08      C0  030F   843          ADDL2   #8,SP                    ; discard arglist
                2B 50    E9  0312   844          BLBC    R0,30$                   ; get out on error
                             0315   845
                    03   E1  0315   846 20$:     BBC     #IFB$V_AI,-              ; branch if no AI
             25 00A0 CA          0317   847                  IFB$B_JNLFLG(R10),30$
             54     30   A9  D0  031B   848          MOVL    IRB$L_JNLBDB(R9),R4      ; get journal BDB address
             51     18   A4  D0  031F   849          MOVL    BDB$L_ADDR(R4),R1        ; get journal buffer address
             53     0C   AE  D0  0323   850          MOVL    12(SP),R3                ; get data BDB address from saved R4
                14 A3    28  0327   851          MOVC3   BDB$W_NUMB(R3),-         ; copy user buffer to journal buffer
                18 B3        032A   852                  @BDB$L_ADDR(R3),-
                44 A1        032C   853                  RJR$T_BLOCK(R1)
                30 A9    DD  032E   854          PUSHL   IRB$L_JNLBDB(R9)         ; jBDB addr
                    03   DD  0331   855          PUSHL   #CJF$_AI                 ; set AI
             54     14   AE  D0  0333   856          MOVL    12+8(SP),R4              ; get data BDB address from saved R4
            00000000'EF      16  0337   857          JSB     RM$WRTJNL                ; write journal entry
                5E   08      C0  033D   858          ADDL2   #8,SP                    ; discard arglist
                6E   50      D0  0340   859 30$:     MOVL    R0,(SP)                  ; update return status
                    3D   BA  0343   860 40$:     POPR    #^M<R0,R2,R3,R4,R5>      ; set return status and restore regs
                03 50    E9  0345   861          BLBC    R0,50$                   ; get out on error
                 FE8D    31  0348   862          BRW     DOWRT                    ; rejoin $WRITE code
                             034B   863
                         05  034B   864 50$:     RSB
                             034C   865
                             034C   866          .END
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
BDB$B_FLGS                      = 0000000A
BDB$L_ADDR                      = 00000018
BDB$L_FLINK                     = 00000000
BDB$L_VBN                       = 0000001C
BDB$M_DRT                       = 00000002
BDB$M_VAL                       = 00000001
BDB$W_ALLOC_SIZE                = 0000002C
BDB$W_NUMB                      = 00000014
BDB$W_SIZE                      = 00000016
BDB$W_USERS                     = 0000000C
BLKIO                             000000C5 R      01
BR_AID                            000001D5 R      01
CHKEOF                            00000090 R      01
CHKLEOF                           0000003A R      01
CJF$_AI                         = 00000003
CJF$_BI                         = 00000002
DEV$V_RND                       = 0000001C
DEV$V_SQD                       = 00000005
DOWRT                             000001D8 R      01
ERRJNS                            00000252 R      01
ERRRBF                            00000089 R      01
ERRUBF                            0000007D R      01
ERRUSZ                            00000076 R      01
EXIT                              0000018A R      01
EXIT1                             00000082 R      01
EXTEND                            0000022F R      01
FAB$C_SEQ                       = 00000000
FAB$V_TRN                       = 00000004
GBDB0                             0000012C R      01
GBDB23                            00000140 R      01
GETBDB                            00000136 R      01
IFB$B_FAC                       = 00000022
IFB$B_JNLFLG                    = 000000A0
IFB$B_ORGCASE                   = 00000023
IFB$L_BDB_FLNK                  = 00000040
IFB$L_DEVBUFSIZ                 = 00000048
IFB$L_EBK                       = 00000074
IFB$L_HBK                       = 00000070
IFB$L_PRIM_DEV                  = 00000000
IFB$V_AI                        = 00000003
IFB$V_BI                        = 00000002
IFB$V_DAP                       = 0000003E
IFB$V_GET                       = 00000001
IFB$V_PUT                       = 00000000
IFB$V_RW_ATTR                   = 00000034
IFB$W_FFB                       = 0000005C
IRB$L_CURBDB                    = 00000020
IRB$L_JNLBDB                    = 00000030
IRB$L_NRP_OFF                   = 00000044
IRB$L_NRP_VBN                   = 00000040
IRB$L_NXTBDB                    = 0000003C
IRB$L_RP_VBN                    = 00000048
IRB$V_BIO_LAST                  = 00000027
IRB$V_BRO_SW                    = 00000028
IRB$V_EOF                       = 00000021
IRB$V_WRITE                     = 00000029
IRB$W_NRP_OFF                   = 00000044
IRB$W_RP_OFF                    = 0000004C
NT$READ                           ********     X    01
NT$WRITE                          ********     X    01
N_READ                            00000243 R        01
NTWRITE                           00000239 R        01
PIO$A_TRACE                       ********     X    01
RAB$L_BKT                       = 00000038
RAB$L_RBF                       = 00000028
RAB$L_ROP                       = 00000004
RAB$L_UBF                       = 00000024
RAB$V_TPT                       = 00000001
RAB$W_RFA                       = 00000010
RAB$W_RSZ                       = 00000022
RAB$W_USZ                       = 00000020
REL23                             0000013D R      01
RELBDB                            00000106 R      01
RELEASE                           0000011A R      01
RELERR                            0000017D R      01
RETURN                            00000075 R      01
RJR$B_ENTRY_TYPE                = 00000003
RJR$B_OPER                      = 00000005
RJR$B_ORG                       = 00000004
RJR$C_BLKLEN                    = 00000044
RJR$C_BLOCK                     = 00000003
RJR$L_BLOCK_SIZE                = 00000040
RJR$L_BLOCK_VBN                 = 0000003C
RJR$T_BLOCK                     = 00000044
RJR$_WRITE                      = 0000001E
RM$ALDJNLBUF                      ********     X    01
RM$AUTOEXTEND                     ********     X    01
RM$EXRMS                          ********     X    01
RM$RDBUFWT                        ********     X    01
RM$RETJNLBDB                      ********     X    01
RM$RSET                           ********     X    01
RM$WRTJNL                         ********     X    01
RM$WTBUFWT                        ********     X    01
RM$WTLST1                         ********     X    01
RMS$READ                        = FFFFFFFE RG     01
RMS$WRITE                       = 0000018B RG     01
RMS$_EOF                        = 0001827A
RMS$_FAC                        = 00018514
RMS$_JNS                        = 000187F4
RMS$_RBF                        = 00018654
RMS$_UBF                        = 000186EC
RMS$_USZ                        = 000186F4
ROP                             = 00000020
SETBDB                            00000153 R      01
SETNBP                            000000A1 R      01
SETRSZ                            00000070 R      01
TPT$L_READ                        ********     X    01
TPT$L_WRITE                       ********     X    01
WRTBLK                            000001CC R      01
```

```
WRTJNL                          00000258 R      01
WT1                             0000021D 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$RMS                          0000034C  (  844.)  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	00:00:00.10	00:00:00.71
Command processing	139	00:00:00.74	00:00:04.25
Pass 1	370	00:00:13.42	00:00:33.83
Symbol table sort	0	00:00:01.84	00:00:02.38
Pass 2	157	00:00:03.11	00:00:09.97
Symbol table output	15	00:00:00.12	00:00:00.14
Psect synopsis output	1	00:00:00.03	00:00:00.29
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	719	00:00:19.37	00:00:51.58

```
The working set limit was 1650 pages.
76994 bytes (151 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1380 non-local and 32 local symbols.
866 source lines were read in Pass 1, producing 15 object records in Pass 2.
29 pages of virtual memory were used to define 28 macros.
```

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	17
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	24

```
1503 GETS were required to define 24 macros.

There were no errors, warnings or information messages.

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

RMS0FSCN
LIS

RM3XALLO
LIS

RMS0CONN
LIS

RMS0ENTER
LIS

RMS0ERASE
LIS

RMS0CREAT
LIS

RMS0DELET
LIS

RMS0FIND
LIS

RM3XSUMO
LIS

RMS0DISPL
LIS

RM3XKEYO
LIS

RMS0GET
LIS

RMS0CLOSE
LIS

RMS0DISC
LIS

RMS0EXTEN
LIS

RMS0BLKIO
LIS