```
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
```

FILEID**RM1SEQXFR

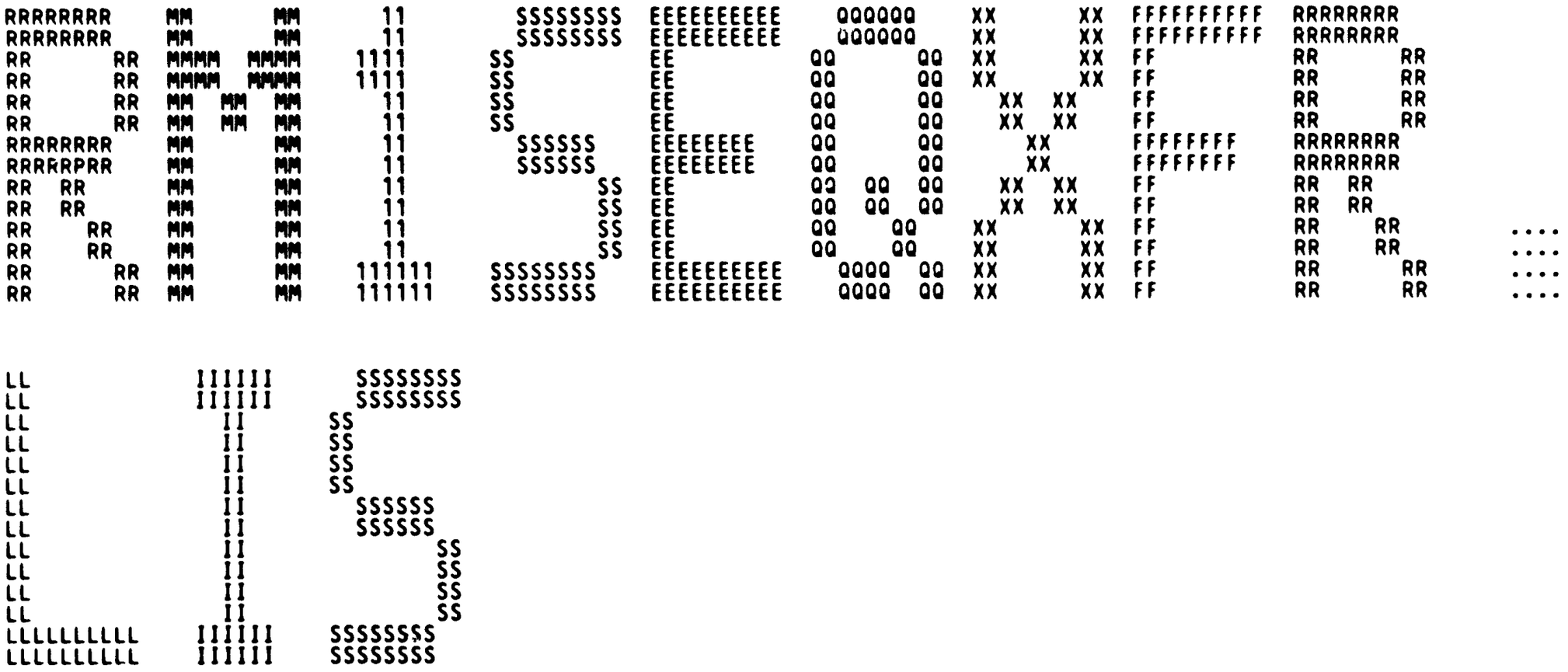

Table of contents

(2) 158 DECLARATIONS
(3) 193 RM$SEQRAH - ROUTINE TO PERFORM READ AHEAD QIO
(4) 279 RM$QUIET_SEQMBF - ROUTINE TO FLUSH OUT ALL RAH/WBH
(5) 429 RM$SEQRD - ROUTINE TO PERFORM SEQUENTIAL READS
(15) 1045 RM$SEQWT - ROUTINE TO PERFORM SEQUENTIAL WRITES

```
0000     1          $BEGIN  RM1SEQXFR,000,RM$RMS1,<TRANSFER BLOCK FOR SEQUENTIAL FILE ORG>
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    27 ;++
0000    28 ; Facility: rms32
0000    29 ;
0000    30 ; Abstract:
0000    31 ;                this module performs i/o transfers for the
0000    32 ;                sequential file organization.
0000    33 ;
0000    34 ; Environment:
0000    35 ;                star processor running starlet exec.
0000    36 ;
0000    37 ; Author: L F Laverdure,         creation date: 3-FEB-1977
0000    38 ;
0000    39 ; Modified By:
0000    40 ;
0000    41 ;       V03-012 JEJ0023         J E Johnson             06-Apr-1984
0000    42 ;                In the event of an attempt to append to a magtape file back
0000    43 ;                up over the tape mark and write the block.
0000    44 ;
0000    45 ;       V03-011 SHZ0001         Stephen H. Zalewski     05-Dec-1983
0000    46 ;                No longer attempt to allocate different event flags
0000    47 ;                in RM$SETEFN.  The new scheme is to always return EFN 29.
0000    48 ;
0000    49 ;       V03-010 RAS0187         Ron Schaefer             7-Sep-1983
0000    50 ;                Restore TSK0001 lost by KPL000x.
0000    51 ;                Force journal entries for the file immediately
0000    52 ;                before a buffer is written to disk.  Fix a broken branch.
0000    53 ;                Improve register save/restore performance.
0000    54 ;
0000    55 ;       V03-009 RAS0182         Ron Schaefer            29-Aug-1983
0000    56 ;                Correct KPL0003 for non-terminal/non-disk devices;
0000    57 ;                BDB$L_VBN and BDB$W_NUMB were not being filled in.
```

```
0000     58 ;
0000     59 ;       V03-008 KPL0003         Peter Lieberwirth       12-Aug-1983
0000     60 ;               Re-work KPL0002 because it wasn't quite right.  RAH from
0000     61 ;               magtapes broke.  Add some explanatory commentary now
0000     62 ;               that I understand this a little more.
0000     63 ;
0000     64 ;       V03-007 KPL0002         Peter Lieberwirth       12-Jul-1983
0000     65 ;               Fix dependancy on order of QIO completion.  RMS required
0000     66 ;               that QIOs finish in the same order as they were started.
0000     67 ;               UDAs and HSCs broke this assumption, resulting in
0000     68 ;               intermittent "stale data" reads on RAH/WBE IO.
0000     69 ;
0000     70 ;       V03-006 RAS0142         Ron Schaefer             7-Apr-1983
0000     71 ;               Correct spurious data returned on error path for
0000     72 ;               terminal input with a bad prompt buffer.
0000     73 ;
0000     74 ;       V03-005 DAS0002         David Solomon           24-Feb-1983
0000     75 ;               Return new error RMS$_XNF if an XABTRM is required
0000     76 ;               (e.g. ROP ETO bit set), but not found.
0000     77 ;
0000     78 ;       V03-004 DAS0002         David Solomon           10-Feb-1983
0000     79 ;               Add support for extended terminal I/O (XABTRM).
0000     80 ;
0000     81 ;       V03-003 KBT0148         Keith B. Thompson       20-Aug-1982
0000     82 ;               Reorganize psects
0000     83 ;
0000     84 ;       V03-002 KBT0089         Keith B. Thompson       13-Jul-1982
0000     85 ;               Clean up psects
0000     86 ;
0000     87 ;       V03-001 KPL0001         P Lieberwirth   22-Mar-1982
0000     88 ;               Move a word of the BDB$L_IOSB as system error code
0000     89 ;               on errors encountered in read-ahead/write-behind,
0000     90 ;               not a longword, as before, which led to bad messages.
0000     91 ;
0000     92 ;       V02-039 CDS0003         C D Saether     20-Aug-1981
0000     93 ;               Remove references to BDB$M_LKDINWS flag.
0000     94 ;
0000     95 ;       V02-038 CDS0002         C D Saether     7-May-1981
0000     96 ;               Clear DRT before qio on write to block device
0000     97 ;               like it used to before V02-037.
0000     98 ;
0000     99 ;       V02-037 CDS0001         C D Saether     6-Feb-1981
0000    100 ;               Always set bdb$v_iop for duration of i/o on
0000    101 ;               buffer reads and writes so multi-streaming
0000    102 ;               cache interlocks work correctly
0000    103 ;
0000    104 ;       V02-036 KRM0001         K R Malik       19-NOV-1980
0000    105 ;               Check for new status codes, SS$_LINKABORT and SS$_LINKDISCON
0000    106 ;               and map to RMS$_EOF if found (leave in check for SS$_ABORT
0000    107 ;               for the time being).
0000    108 ;
0000    109 ;       V02-035 REFORMAT        P S Knibbe      24-Jul-1980
0000    110 ;
0000    111 ;       V034    CDS0078         C D Saether     26-MAR-1980     11:10
0000    112 ;               zero bdb$w_numb on sys error, branch to errwrt on
0000    113 ;               qio error for task to task
0000    114 ;
```

```
0000    115 ;       V033    CDS0077         C D Saether     21-MAR-1980     15:20
0000    116 ;               set bdb$w_numb to smaller of iosb count field and
0000    117 ;               buffer size on errors.
0000    118 ;
0000    119 ;       V032    JAK0041         J A Krycka      19-MAR-1980     15:30
0000    120 ;               continuation of v031.
0000    121 ;
0000    122 ;       V031    JAK0041         J A Krycka      03-MAR-1980     11:00
0000    123 ;               clean up of ntget and ntput.
0000    124 ;
0000    125 ;       V030    PSK0013         P S Knibbe      18-FEB-1980     17:50
0000    126 ;               make sure bdb$w_numb is always valid after a sequential
0000    127 ;               read.
0000    128 ;
0000    129 ; Revision history:
0000    130 ;
0000    131 ;       V029    PSK0011         P S Knibbe      23-JAN-1980
0000    132 ;       when a magtape reads eof, the eof bit flag should be set and
0000    133 ;       the nrp_vbn should be set equal to the ebk in order to force
0000    134 ;       checkin the flag on subsequent operations.
0000    135 ;
0000    136 ;       V028    CDS0076         C D Saether     22-JAN-1980     15:20
0000    137 ;       fix write behind bug causing wait for completion of i/o on
0000    138 ;       on all buffers when unnecessary
0000    139 ;
0000    140 ;       V027    CDS0061         C D Saether     6-DEC-1979      19:50
0000    141 ;       set now for mailboxes if 0 timeout spec'd.  use lesser of
0000    142 ;       bdb$w_numb or bdb$w_size for reads from terminal.
0000    143 ;
0000    144 ;       V026    CDS0030         C D Saether     11-SEP-1979     16:30
0000    145 ;       take out terminal checks in put_unit_rec for size of record
0000    146 ;
0000    147 ;       V025    CDS0026         C D Saether     18-AUG-1979     00:03
0000    148 ;       write to unit record devices from user buffer
0000    149 ;       don't write records larger than devbufsiz to terminals checks
0000    150 ;       have moved here from rm$put_unit_rec
0000    151 ;
0000    152 ;       V024    CDS0011         C D Saether     26-JUN-1979     17:00
0000    153 ;       take cntrl z checking for terminals out of rm$seqrd
0000    154 ;
0000    155 ;--
0000    156
```

```
                     0000   158          .SBTTL  DECLARATIONS
                     0000   159
                     0000   160 ;
                     0000   161 ; Include Files:
                     0000   162 ;
                     0000   163 ;
                     0000   164 ; Macros:
                     0000   165 ;
                     0000   166
                     0000   167          $IFBDEF
                     0000   168          $IRBDEF
                     0000   169          $BDBDEF
                     0000   170          $IMPDEF
                     0000   171          $PIODEF
                     0000   172          $RABDEF
                     0000   173          $DEVDEF
                     0000   174          $IODEF
                     0000   175          $QIODEF
                     0000   176          $RMSDEF                         ; Define the RMS$_xxx symbols
                     0000   177          $TTDEF
                     0000   178          $SSDEF
                     0000   179          $XABTRMDEF
                     0000   180
                     0000   181 ;
                     0000   182 ; Equated Symbols:
                     0000   183 ;
                     0000   184
00000020             0000   185          ROP=RAB$L_ROP*8                 ; bit offset to rop
0000001A             0000   186          CTRLZ   = 26
                     0000   187
                     0000   188 ;
                     0000   189 ; Own Storage:
                     0000   190 ;
                     0000   191
```

```
0000   193          .SBTTL  RM$SEQRAH - ROUTINE TO PERFORM READ AHEAD QIO
0000   194
0000   195 ;++
0000   196 ; RM$SEQRAH - Read ahead QIO
0000   197 ;
0000   198 ; this routine performs read ahead for the sequential file organization.
0000   199 ; currently used for files-11 disk and magtape only.
0000   200 ;
0000   201 ; Calling sequence:
0000   202 ;
0000   203 ;       bsbw    rm$seqrah
0000   204 ;
0000   205 ; Input Parameters:
0000   206 ;
0000   207 ;       r11     impure area addr
0000   208 ;       r10     ifab addr
0000   209 ;       r9      irab addr
0000   210 ;       r8      rab addr
0000   211 ;       r4      bdb address for read ahead
0000   212 ;       r2      # of bytes to read
0000   213 ;       r1      starting vbn
0000   214 ;
0000   215 ; Implicit Inputs:
0000   216 ;
0000   217 ;       ifb$w_chnl      i/o channel for qio
0000   218 ;
0000   219 ; outputs:
0000   220 ;
0000   221 ;       r0              status code
0000   222 ;       r1,ap           destroyed
0000   223 ;
0000   224 ; Implicit Outputs:
0000   225 ;
0000   226 ;       bdb$w_numb              # of bytes transferred
0000   227 ;       bdb$l_vbn               starting vbn for buffer
0000   228 ;       bdb$v_val               cleared
0000   229 ;       bdb$l_iosb,bdb$l_iosb+4 user as system iosb for qio
0000   230 ;
0000   231 ;
0000   232 ; status codes:
0000   233 ;
0000   234 ;       standard qio system service codes
0000   235 ;
0000   236 ; Side Effects:
0000   237 ;
0000   238 ;       the ast for i/o completion will occur at rm$rahwbhast with r4
0000   239 ;       as the ast parameter.
0000   240 ;
0000   241 ; note:
0000   242 ;
0000   243 ; The following is an historical note:
0000   244 ;
0000   245 ;       the read ahead/write behind code assumes that qios will occur in the
0000   246 ;       same order as they are queued, otherwise the following sequence may fail:
0000   247 ;       wbh vbn n, read vbn m, read vbn n; a stale copy of vbn n may be read.
0000   248 ;       if this ordering assumption becomes invalid, additional code will have
0000   249 ;       to be written.
```

```
                                0000   250 ;
                                0000   251 ; End of historical note.
                                0000   252 ;
                                0000   253 ;       This problem is fixed in the following way:  When an IO is initiated,
                                0000   254 ;       all BDBs are examined to see if they have IOP for a buffer that contains
                                0000   255 ;       any VBNs in the range of the IO about to be initiated.  If any such
                                0000   256 ;       overlap occurs, all RAH/WBE IO is quieted.  This probably slows the
                                0000   257 ;       code down some, but is the only way to assure data integrity, short
                                0000   258 ;       of writing a much more complicated sequential cache manager.
                                0000   259 ;--
                                0000   260
                                0000   261 RM$SEQRAH::
                                0000   262
                     0A A4   94 0000   263         CLRB    BDB$B_FLGS(R4)          ; clear all bdb flags
                                0003   264         SSB     #BDB$V_IOP,-
                                0003   265                 BDB$B_FLGS(R4)          ; set io in progress
                  1C A4   51    D0 0008   266         MOVL    R1,BDB$L_VBN(R4)        ; store vbn in bdb
                  14 A4   52    B0 000C   267         MOVW    R2,BDB$W_NUMB(R4)       ; store size of xfer in bdb
                        02B2    30 0010   268         BSBW    SETP6_P3                ; build qio parm block
                        02BB    30 0013   269         BSBW    SETP2_EFN_RAH
                                0016   270
                                0016   271         ASSUME  QIO$_NARGS EQ 12
                                0016   272
          00000000'9F    0C    FB 0016   273         CALLS   #12,@#SYS$QIO           ; do qio
                     05 50    E8 001D   274         BLBS    R0,10$                  ; branch if qio was ok
                                0020   275         CSB     #BDB$V_IOP,-
                                0020   276                 BDB$B_FLGS(R4)          ; note no io in progress
                           05   0025   277 10$:    RSB
```

```
                     0026   279          .SBTTL  RM$QUIET_SEQMBF - ROUTINE TO FLUSH OUT ALL RAH/WBH
                     0026   280
                     0026   281 ;++
                     0026   282 ; RM$QUIET_SEQMBF - Flush out all RAH/WBH
                     0026   283 ;
                     0026   284 ; this routine scans down the bdb chain and wait for all
                     0026   285 ; pending io to complete. if a write behind error occurs it saves
                     0026   286 ; the status and reports it after all io is quiet.
                     0026   287 ;
                     0026   288 ; Calling sequence:
                     0026   289 ;
                     0026   290 ;       bsbw    rm$quiet_seqmbf
                     0026   291 ;
                     0026   292 ; Input Parameters:
                     0026   293 ;
                     0026   294 ;       r11     impure area addr
                     0026   295 ;       r10     ifab addr
                     0026   296 ;       r9      irab addr
                     0026   297 ;       r8      rab addr (or fab)
                     0026   298 ;
                     0026   299 ; Implicit Inputs:
                     0026   300 ;
                     0026   301 ;       ifb$w_chnl      i/o channel for qio
                     0026   302 ;
                     0026   303 ; outputs:
                     0026   304 ;
                     0026   305 ;       r0              status code
                     0026   306 ;       r1 thru r3,ap are destroyed
                     0026   307 ;
                     0026   308 ; Implicit Outputs:
                     0026   309 ;
                     0026   310 ;       all bdb's in chain have been initialized
                     0026   311 ;
                     0026   312 ;
                     0026   313 ; status codes:
                     0026   314 ;
                     0026   315 ;       wbe
                     0026   316 ;
                     0026   317 ; Side Effects:
                     0026   318 ;
                     0026   319 ;
                     0026   320 ;--
                     0026   321
                     0026   322 RM$QUIET_SEQMBF::
                  20 A9   D4  0026   323          CLRL    IRB$L_CURBDB(R9)        ; there is no longer a current bdb
                7E   54   7D  0029   324          MOVQ    R4,-(SP)                ; save registers
                     01   DD  002C   325          PUSHL   #1                      ; anticipate success
54   69   00000040 8F   C1  002E   326          ADDL3   #IFB$L_BDB_FLNK,IRB$L_IFAB_LNK(R9),R4; get bdb list head addr
                55   54   D0  0036   327          MOVL    R4,R5                   ; save it
                          0039   328          ASSUME  BDB$L_FLINK EQ 0
                54   64   D0  0039   329 10$:     MOVL    (R4),R4                 ; get next bdb
                55   54   D1  003C   330          CMPL    R4,R5                   ; at end?
                     15   13  003F   331          BEQL    30$                     ; branch if yes
                     02   E1  0041   332          BBC     #BDB$V_IOP,-
             08 0A   A4        0043   333                  BDB$B_FLGS(R4),20$      ; branch if no io in progress
                     11   10  0046   334          BSBB    RM$STALLRAHWBH          ; wait for io to complete
                   03 50   E8  0048   335          BLBS    R0,20$                  ; branch if no wbe errors
```

```
                6E    50    D0  004B    336            MOVL    R0,(SP)                 ; remember error
                  0A  A4    94  004E    337 20$:       CLRB    BDB$B_FLGS(R4)          ; clear flags
                  1C  A4    D4  0051    338            CLRL    BDB$L_VBN(R4)           ; and vbn
                      E3    11  0054    339            BRB     10$                     ; branch back for next bdb
                            0056    340
                      31    BA  0056    341 30$:       POPR    #^M<R0,R4,R5>           ; restore status and r4,r5
                            05  0058    342            RSB
                            0059    343
                            0059    344 ;++
                            0059    345 ; rm$stallrahwbh - STALL Read Ahead and Write behind
                            0059    346 ;
                            0059    347 ; this routine waits for io done for a bdb for which a rah or wbh was done.
                            0059    348 ;
                            0059    349 ; Calling sequence:
                            0059    350 ;
                            0059    351 ;       bsbw    rm$stallrahwbh
                            0059    352 ;
                            0059    353 ; inputs:
                            0059    354 ;
                            0059    355 ;       same as rm$quiet_seqmbf
                            0059    356 ;       r4 = bdb address to stall for
                            0059    357 ;
                            0059    358 ; outputs:
                            0059    359 ;
                            0059    360 ;       r0 = status     either suc or wbe
                            0059    361 ;
                            0059    362 ;       r1 thru r3,ap are destroyed only if really stalls
                            0059    363 ;
                            0059    364 ;  note:         the bdb flags bit 'ast_dcl' is used to interlock rah/wbh
                            0059    365 ;       i/o done processing with the ast side of the system.  when the i/o
                            0059    366 ;       is initialized ast_dcl is zeroed.  when the ast routine for i/o
                            0059    367 ;       completion is executed, the bit is set, and if already set the waiting
                            0059    368 ;       stream is restarted (subject to imp$v_inhast), else the ast is simply
                            0059    369 ;       dismissed.  when testing for i/o done, this routine first sets
                            0059    370 ;       bdb$l_wait to the address of the irab and then tests the bit, setting
                            0059    371 ;       it.  if the bit was clear, the routine must stall for i/o done,
                            0059    372 ;       otherwise the buffer may be used immediately.
                            0059    373 ;
                            0059    374 ;--
                            0059    375
                            0059    376 RM$STALLRAHWBH::
            24  A4    59    D0  0059    377            MOVL    R9,BDB$L_WAIT(R4)       ; set ast param in case of wait
                        06    E2  005D    378            BBSS    #BDB$V_AST_DCL,-
                03  0A  A4        005F    379                    BDB$B_FLGS(R4),20$      ; branch if i/o done
                            0062    380                                            ; indicates waiting otherwise
                    FF9B'   30  0062    381            BSBW    RM$STALL                ; stall for io complete
                            0065    382 20$:       RMSSUC                          ; assume success for wbh or rah
                            0068    383
                            0068    384            ASSUME  BDB$V_VAL EQ 0
                            0068    385
                        00    E5  0068    386            BBCC    #BDB$V_VAL,-
                07  0A  A4        006A    387                    BDB$B_FLGS(R4),25$      ; branch if rah
                   1C   A4    D4  006D    388            CLRL    BDB$L_VBN(R4)           ; cause blk not to be reused
                01  48  A4    E9  0070    389            BLBC    BDB$L_IOSB(R4),30$      ; branch if errors
                            05  0074    390 25$:       RSB
                   48   A4    3C  0075    391 30$:       MOVZWL  BDB$L_IOSB(R4),-
                   0C   A8        0078    392                    RAB$L_STV(R8)           ; return system error code to user
```

```
                     007A   393          RMSERR  WBE                     ; and report write behind error
                  05 007F   394          RSB
                     0080   395
                     0080   396
                     0080   397 ;++
                     0080   398 ;
                     0080   399 ; routine to get the next bdb via the flnk of a bdb thru the list head
                     0080   400 ;
                     0080   401 ; Calling sequence:
                     0080   402 ;
                     0080   403 ;       bsbw    rm$seqflnkbdb
                     0080   404 ;
                     0080   405 ; inputs:
                     0080   406 ;
                     0080   407 ;       r9 = irab address
                     0080   408 ;       r4 = bdb address
                     0080   409 ;
                     0080   410 ; outputs:
                     0080   411 ;
                     0080   412 ;       r4 = new bdb address via flnk
                     0080   413 ;       r0 = list head address for bdb chain
                     0080   414 ;
                     0080   415 ;--
                     0080   416
                     0080   417 RM$SEQFLNKBDB::
           00000040 8F    C1 0080   418          ADDL3   #IFB$L_BDB_FLNK,-
                50   69       0086   419                  IRB$L_IFAB_LNK(R9),R0   ; get bdb list head addr
                              0088   420
                              0088   421          ASSUME  BDB$L_FLINK EQ 0
                              0088   422
                54   64   D0 0088   423          MOVL    (R4),R4                 ; get next bdb
                50   54   D1 008B   424          CMPL    R4,R0                   ; at list head
                     03   12 008E   425          BNEQ    XIT                     ; branch if no
                54   64   D0 0090   426          MOVL    (R4),R4                 ; yes go one more
                     05   0093   427 XIT:       RSB
```

```
0094   429          .SBTTL  RM$SEQRD - ROUTINE TO PERFORM SEQUENTIAL READS
0094   430
0094   431 ;++
0094   432 ; RM$SEQRD - Perform sequential reads
0094   433 ;
0094   434 ; this routine performs read virtual block for the sequential
0094   435 ; file organization.  one of several flavors is performed
0094   436 ; depending upon whether the device for the read is a
0094   437 ; terminal and whether a read-ahead (ie multi-buffering)
0094   438 ; was done or is in progress.
0094   439 ;
0094   440 ; Calling sequence:
0094   441 ;
0094   442 ;       bsbw    rm$seqrd
0094   443 ;
0094   444 ; Input Parameters:
0094   445 ;
0094   446 ;       r11     impure area addr
0094   447 ;       r10     ifab addr
0094   448 ;       r9      irab addr
0094   449 ;       r8      rab addr
0094   450 ;       r3      option to avoid read if bit 0 set
0094   451 ;               (currently simply checks for i/o in progress)
0094   452 ;       r2      # of bytes to read
0094   453 ;       r1      starting vbn
0094   454 ;
0094   455 ; Implicit Inputs:
0094   456 ;
0094   457 ;       irb$l_curbdb    addr of old current bdb
0094   458 ;       irb$l_nxtbdb    bdb addr for which last rah was done
0094   459 ;       dev$v_trm       terminal device flag
0094   460 ;                       if set, the various rop terminal option
0094   461 ;                       bits are used along with irb$b_tmo,
0094   462 ;                       irb$_l_pbf and irb$b_psz if specified
0094   463 ;       ifb$w_chnl      i/o channel for qio
0094   464 ;
0094   465 ; outputs:
0094   466 ;
0094   467 ;       irb$l_curbdb    new current bdb = r4 = flnk of old curbdb or
0094   468 ;                       if old curbdb = 0 then nxtbdb
0094   469 ;       r4              addr of new current bdb
0094   470 ;       r0              status code
0094   471 ;       r1 thru r3,ap   destroyed
0094   472 ;
0094   473 ; Implicit Outputs:
0094   474 ;
0094   475 ;       bdb$w_numb              # of bytes transferred
0094   476 ;       bdb$l_vbn               starting vbn for buffer
0094   477 ;       bdb$v_val               set unless an error occurred
0094   478 ;       irb$l_ios,irb$l_ios4    system specified i/o status block
0094   479 ;                               (useful for detecting unit record
0094   480 ;                               eof or reporting errors)
0094   481 ;       rab$l_stv               set to terminator if device is terminal
0094   482 ;                               or system error code if sys error
0094   483 ;
0094   484 ;
0094   485 ; status codes:
```

```
                     0094   486 ;
                     0094   487 ;       standard rms, in particular, suc, ecf, tmo, wbe, and sys.
                     0094   488 ;
                     0094   489 ; Side Effects:
                     0094   490 ;
                     0094   491 ;       may have switched to running at ast level
                     0094   492 ;       bdb$v_iop set for duration of i/o
                     0094   493 ;
                     0094   494 ;--
                     0094   495
                     0094   496 RM$SEQRD::                                ; perform sequential reads
                     0094   497         $TSTPT  SEQRD
                     009A   498
                     009A   499 ; Get new Current BDB.  New current BDB is the forward link of the old
                     009A   500 ; current BDB unless there was no old current BDB (CURBDB=0).  In that
                     009A   501 ; case, the new current BDB is taken from NXTBDB.
                     009A   502 ;
                     009A   503 ; RM$NXTBLK1 fires off read-aheads when releasing a buffer.  Each available
                     009A   504 ; buffer is used in that routine to fire off a RAH.  That routine and this
                     009A   505 ; routine share assumptions about the use of CURBDB and NXTBDB.  RM$NXTBLK1
                     009A   506 ; sets up the RAH for the BDB pointed to by the forward link of NXTBDB
                     009A   507 ; (after setting up NXTBDB=CURBDB, of course...) and this routine expects
                     009A   508 ; to find it in the BDB pointed to be the forward link of CURBDB.
                     009A   509 ;
                     009A   510 ; (The reason for this last bit of baroqueness is that the readahead code
                     009A   511 ; in RM1NXTBLK continues playing with NXTBDB, firing off as many RAHs as
                     009A   512 ; possible.  When its done, NXTBDB is no longer the same as CURBDB, it
                     009A   513 ; may point to some BDB farther down the chain.
                     009A   514 ;
                     009A   515
    54    20 A9  D0  009A   516         MOVL    IRB$L_CURBDB(R9),R4      ; get current BDB
          06     12  009E   517         BNEQ    5$                       ; if there is one, get its forward link
    54    3C A9  D0  00A0   518         MOVL    IRB$L_NXTBDB(R9),R4      ; no current BDB, use NXTBDB
          02     11  00A4   519         BRB     6$                       ;    and continue
          D8     10  00A6   520 5$:     BSBB    RM$SEQFLNKBDB            ; get forward link of CURBDB
    20 A9 54     D0  00A8   521 6$:     MOVL    R4,IRB$L_CURBDB(R9)      ; update CURBDB with new value
                     00AC   522
                     00AC   523 ;
                     00AC   524 ; If this BDB is not involved in a read-ahead operation, we're on our
                     00AC   525 ; own. If its involved in a read-ahead operation for the buffer we want, we
                     00AC   526 ; must wait for it.  Note - we may not need to actually wait, if AST_DCL is set,
                     00AC   527 ; clearing IOP will put things just right, and RM$STALLRAHWBH will come
                     00AC   528 ; right back.  If the BDB involved in an RAH for some other IO,
                     00AC   529 ; quiet all IO and go for it.
                     00AC   530 ;
                     00AC   531 ; IOP is set on all RAH BDBs, even if the actual IO has completed.
                     00AC   532 ; AST_DCL (AST has been declared) is set if the the actual QIO
                     00AC   533 ; operation has completed.  RM$STALLRAHWBH understands this.  I didn't.
                     00AC   534 ;
2F 0A A4     02  E5  00AC   535         BBCC    #BDB$V_IOP,BDB$B_FLGS(R4),40$; branch if not a rah bdb
                     00B1   536
          0E     BB  00B1   537         PUSHR   #^M<R1,R2,R3>
          A4     10  00B3   538         BSBB    RM$STALLRAHWBH           ; wait for io done, note that IO may
          0E     BA  00B5   539         POPR    #^M<R1,R2,R3>            ; be done ever though IOP was set.
       D9 50     E9  00B7   540         BLBC    R0,XIT                   ; branch if wbe error
                     00BA   541
       23 53     E8  00BA   542         BLBS    R3,40$                   ; branch if write, claim bdb
```

```
                1C A4    51    D1   00BD    543          CMPL    R1,BDB$L_VBN(R4)        ; is this the desired vbn
                      0C    12   00C1    544          BNEQ    35$                     ; branch if no
          0C A9    48 A4    7D   00C3    545          MOVQ    BDB$L_IOSB(R4),IRB$L_IOS(R9); put io status in irab
          50    48 A4    3C   00C8    546          MOVZWL  BDB$L_IOSB(R4),R0       ; get completion code
                00C7    31   00CC    547          BRW     QIODONE                 ; and go join common code
                             00CF    548
                             00CF    549
                             00CF    550 ; Ran into a RAH IO for some other buffer.  Quiet all this IO and then
                             00CF    551 ; proceed with our IO.
                             00CF    552 ;
                             00CF    553
                      0E    BB   00CF    554 35$:     PUSHR   #^M<R1,R2,R3>
                    FF52    30   00D1    555          BSBW    RM$QUIET_SEQMBF         ; quiet activity
                      0E    BA   00D4    556          POPR    #^M<R1,R2,R3>
                20 A9    54    D0   00D6    557          MOVL    R4,IRB$L_CURBDB(R9)     ; save BDB address for later
                   03 50    E8   00DA    558          BLBS    R0,40$                  ; continue on success
                    FFB3    31   00DD    559          BRW     XIT                     ; pass on error
                             00E0    560
                             00E0    561
                             00E0    562 ; Claim free bdb.
                             00E0    563 ;
                             00E0    564
                      52    D5   00E0    565 40$:     TSTL    R2                      ; any transfer?
                      12    13   00E2    566          BEQL    SUCCESS                 ; get out if none
                   0F 53    E8   00E4    567          BLBS    R3,SUCCESS              ; all done if no read flagged
                      02    E1   00E7    568          BBC     #DEV$V_TRM,-
                      6A         00E9    569                  IFB$L_PRIM_DEV(R10),-
                      17         00EA    570                  CHECK_IO                ; branch if not terminal
                1C A4    51    D0   00EB    571          MOVL    R1,BDB$L_VBN(R4)        ; save vbn in bdb
                14 A4    52    B0   00EF    572          MOVW    R2,BDB$W_NUMB(R4)       ; and # bytes used
                    013D    31   00F3    573          BRW     TRMREAD
                             00F6    574
                             00F6    575 ;
                             00F6    576 ; successful return for no read required
                             00F6    577 ;
                             00F6    578
                             00F6    579 SUCCESS:
                1C A4    51    D0   00F6    580          MOVL    R1,BDB$L_VBN(R4)        ; save vbn in bdb
                14 A4    52    B0   00FA    581          MOVW    R2,BDB$W_NUMB(R4)       ; and # bytes used
                             00FE    582          RMSSUC
                      05    0101    583          RSB
                             0102    584
                             0102    585 CHECK_IO:                                ;
                             0102    586
                             0102    587 ; We're going to do an IO.  We must make sure our IO does not overlap any
                             0102    588 ; outstanding IO.  If it does, wait, because otherwise we may get into trouble
                             0102    589 ; with QIO order dependancy and UDA/HSC disks.
                             0102    590 ;
                             0102    591 ; Look through all the BDBs. If any have an IO in progress (IOP) for any VBN
                             0102    592 ; in the range of our target request, wait for the RAH/WBH to quiet and then
                             0102    593 ; continue.
                             0102    594 ;
                             0102    595 ; By doing so, we are no longer dependant on the ordering of QIOs by ACPs or
                             0102    596 ; devices.
                             0102    597 ;
                             0102    598
             53 6A    1C    E1   01(     599          BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R10),100$ ; only check on disks
```

```
                7E    55    7D  0106    600          MOVQ    R5,-(SP)                ; save work registers used in
                                0109    601                                          ; calculation of VBN range of request
                                0109    602                                          ; or IOP
                                0109    603
                                0109    604 ; R1 has the starting VBN of the target request, calculate the ending VBN by
                                0109    605 ; turning then NUMB of bytes in R2 into a number of blocks and adding that to
                                0109    606 ; the starting VBN.  The ending VBN of the target request will be kept in R5.
                                0109    607 ;
                                0109    608
          55    52    F7 8F     78  0109    609          ASHL    #-9,R2,R5               ; 512 = 2^9 = # of bytes in one block
                55    51        C0  010E    610          ADDL2   R1,R5                   ; last VBN in target range request
                                0111    611
              00000040 8F       C1  0111    612          ADDL3   #IFB$L_BDB_FLNK,-       ; also, get BDB list head in R0
                50    69              0117    613                  IRB$L_IFAB_LNK(R9),R0  ; (save for end test)
                54    60        D0  0119    614          MOVL    (R0),R4                 ; get first BDB address
                                011C    615
                                011C    616 ; See if IO is in progress on buffer described by this BDB.  If so, see if
                                011C    617 ; it conflicts with our request.  If not, save the BDB addre.s and keep
                                011C    618 ; looking for conflicting IO.
                                011C    619 ;
                                011C    620
          2D 0A A4    02        E1  011C    621 10$:     BBC     #BDB$V_IOP,BDB$B_FLGS(R4),20$ ; branch if no IO in progress
          28 0A A4    06        E0  0121    622          BBS     #BDB$V_AST_DCL,BDB$B_FLGS(R4),20$ ; or if AST side says done
                                0126    623
                                0126    624 ;
                                0126    625 ; If there is IOP, see if any VBN in the range of the IO for this BDB/BUFFER
                                0126    626 ; is in the range of our target request.  First, describe the range of the IO
                                0126    627 ; for this buffer as a low-VBN/high-VBN pair as was down above for the target
                                0126    628 ; request.
                                0126    629 ;
                56    14 A4     3C  0126    630          MOVZWL  BDB$W_NUMB(R4),R6       ; get size of buffer in bytes
          56    56    F7 8F     78  012A    631          ASHL    #-9,R6,R6               ; 512 = 2^9 = # bytes in one block
                56    1C A4     C0  012F    632          ADDL2   BDB$L_VBN(R4),R6        ; r6 now has high VBN of IOP
                                0133    633
                                0133    634
                                0133    635 ; Do comparison:
                                0133    636 ;
                                0133    637 ;       If (low VBN in target request > high VBN for IOP)
                                0133    638 ;          or
                                0133    639 ;          (high VBN in target request < low VBN for IOP)
                                0133    640 ;       then
                                0133    641 ;           no_conflict.
                                0133    642 ;
             56    51        D1  0133    643          CMPL    R1,R6                   ; is base of target > high end of IO?
                   16        14  0136    644          BGTR    30$                     ; if GTR yes, no possible conflict
          1C A4    55        D1  0138    645          CMPL    R5,BDB$L_VBN(R4)        ; is top of target < low end of IO?
                   10        19  013C    646          BLSS    30$                     ; if LSS yes, no possible conflict
                                013E    647
                                013E    648
                                013E    649 ; Conflict, clean up and quiet IO.
                                013E    650 ;
                   07        BB  013E    651          PUSHR   #^M<R0,R1,R2>           ; save bdb list head address, etc
                 FF16        30  0140    652          BSBW    RM$STALLRAHWBH          ; wait for this IO to complete
                   50        E8  0143    653          BLBS    R0,15$                  ; proceed on success
             5E    14        C0  0146    654          ADDL2   #20,SP                  ; clean stack
                 FF47        31  0149    655          BRW     XIT                     ; pass on WBE errors
                                014C    656
```

```
                 07   BA  014C    657 15$:    POPR    #^M<R0,R1,R2>           ; restore bdb list head address
                          014E    658
                          014E    659 ; we just solved any problems we had with that IO, now keep looking.
                          014E    660 ;
                          014E    661
                          014E    662 20$:
                          014E    663
                          014E    664 ; Get next BDB address for IOP test.
                          014E    665 ;
                          014E    666         ASSUME  BDB$L_FLINK     EQ      0
                          014E    667
              54   64   D0  014E    668 30$:    MOVL    (R4),R4                 ; get next BDB address
              50   54   D1  0151    669         CMPL    R4,R0                   ; back at head of list?  (ie, done?)
                   C6   12  0154    670         BNEQ    10$                     ; if NEQ, no, another buffer to look at
                            0156    671
                            0156    672
                            0156    673 ; All BDBs in the list have been scanned.  Any IO in the way has been completed.
                            0156    674 ;
                            0156    675
              55   8E   7D  0156    676         MOVQ    (SP)+,R5                ; restore work registers
        54      20 A9   D0  0159    677 100$:   MOVL    IRB$L_CURBDB(R9),R4     ; use saved BDB
     1C A4      51   D0  015D    678         MOVL    R1,BDB$L_VBN(R4)        ; save vbn in bdb
     14 A4      52   B0  0161    679         MOVW    R2,BDB$W_NUMB(R4)       ; and # bytes used
                            0165    680                                     ; Fall into RM$RDBUFWT
```

```
                          0165   682 RM$RDBUFWT::
                          0165   683
                          0165   684 ;
                          0165   685 ; perform remote file access using network specific code. however,
                          0165   686 ; perform task-to-task communication using standard rms code.
                          0165   687 ;
                          0165   688
                3A 6A  3E 50  0165   689         BBS     #IFB$V_DAP,(R10),NTGET  ; branch if remote access via dap
                          0169   690
                          0169   691 ;
                          0169   692 ; issue read qio (not for terminal) and wait for completion
                          0169   693 ;
                          0169   694 ; start building argument list on stack
                          0169   695 ;
                          0169   696
                    0159  30 0169   697         BSBW    SETP6_P3                ; build p6 thru p3 on stack
                    017C  30 016C   698         BSBW    SETP2_EFN               ; build p2 thru efn on stack
                      14  E1 016F   699         BBC     #DEV$V_MBX,-
                      6A     0171   700                 IFB$L_PRIM_DEV(R10),-
                      0E     0172   701                 DOQIO                   ; branch if not mailbox
                      39  E1 0173   702         BBC     #RAB$V_TMO+ROP,-
                  0A 68      0175   703                 (R8),DOQIO              ; or if timeout not wanted
                  1F A8  95  0177   704         TSTB    RAB$B_TMO(R8)           ; 0 timeout spec'd
                      05  12 017A   705         BNEQ    DOQIO                   ; neq don't want now modifier
                          017C   706         SSB     #IO$V_NOW,-
                          017C   707                 QIO$_FUNC-4(SP)         ; set qio sub func to do now
                          0181   708 DOQIO:
                          0181   709         ASSUME  QIO$_NARGS EQ 12
                          0181   710
                0A A4  04 88 0181   711         BISB2   #BDB$M_IOP,BDB$B_FLGS(R4) ; note i/o in progress
      00000000'9F  0C FB 0185   712         CALLS   #12,@#SYS$QIO           ; do the qio
              2B 50  E9 018C   713         BLBC    R0,ERRSYS               ; branch on error
               FE6E' 30 018F   714         BSBW    RM$STALL                ; await completion
                0A A4 04 8A 0192   715         BICB2   #BDB$M_IOP,BDB$B_FLGS(R4) ; note i/o complete
                          0196   716 QIODONE:
                0E A9  B0 0196   717         MOVW    IRB$W_IOS2(R9),-
                14 A4     0199   718                 BDB$W_NUMB(R4)          ; store real byte count
                2A 50  E9 019B   719         BLBC    R0,ERR_READ             ; branch on i/o error
                   01  88 019E   720 SETVAL: BISB2   #BDB$M_VAL,-
                0A A4     01A0   721                 BDB$B_FLGS(R4)          ; set buffer valid
                          01A2   722 RET:     RSB
                          01A3   723
                          01A3   724 ;++
                          01A3   725 ;
                          01A3   726 ; perform network get/find function via the remote fal
                          01A3   727 ; note that network find does not return a record
                          01A3   728 ;
                          01A3   729 ;--
                          01A3   730
                          01A3   731 NTGET:
                08 69  29 E1 01A3   732         BBC     #IRB$V_FIND,(R9) 10$    ; branch if get operation
            00000000'EF  16 01A7   733         JSB     NT$FIND                 ; perform network find
                   06  11 01AD   734         BRB     20$                     ;
            00000000'EF  16 01AF   735 10$:     JSB     NT$GET                  ; perform network get
                  EA 50 E9 01B5   736 20$:     BLBC    R0,RET                  ; branch on failure
                     E4  11 01B8   737         BRB     SETVAL                  ; branch on success
```

```
                     01BA   739
                     01BA   740 ;
                     01BA   741 ;  error on qio - give default error of 'sys'
                     01BA   742 ;
                     01BA   743
                     01BA   744 ERRSYS:
 0A A4    04    8A   01BA   745          BICB2   #BDB$M_IOP,BDB$B_FLGS(R4) ; make sure clear on error
      14 A4    B4   01BE   746          CLRW    BDB$W_NUMB(R4)             ; zero on error so no data moved
                     01C1   747          RMSERR  SYS,RT                     ; set default error code
         11    11   01C6   748          BRB     CHKEOF                     ; go see if error is eof
                     01C8   749
                     01C8   750 ;
                     01C8   751 ; error on read - check for eof
                     01C8   752 ;
                     01C8   753
                     01C8   754 ERR_READ:
                     01C8   755          RMSERR  RER,R1                     ; set default error code
      0E A9    B1   01CD   756          CMPW    IRB$W_IOS2(R9),-
      16 A4                01D0   757                  BDB$W_SIZE(R4)             ; check 'byte count' against
         05    1B   01D2   758          BLEQU   CHKEOF                     ; size of buffer. dataoverrun
                     01D4   759                                             ; err from magtape has size of
                     01D4   760                                             ; of block here instead of bytes
                     01D4   761                                             ; actually transferred.
      16 A4    B0   01D4   762          MOVW    BDB$W_SIZE(R4),-
      14 A4                01D7   763                  BDB$W_NUMB(R4)             ; use buff size instead
0870 8F       50    B1   01D9   764 CHKEOF: CMPW    R0,#SS$_ENDOFFILE          ; is it eof?
         28    13   01DE   765          BEQL    EOF                        ; branch if yes
   0E 6A    3F    E0   01E0   766          BBS     #IFB$V_NSP,(R10),10$       ; branch if task-to-task operation
022C 8F       50    B1   01E4   767          CMPW    R0,#SS$_TIMEOUT            ; perhaps it's time out?
         1A    12   01E9   768          BNEQ    ERRQIO                     ; branch if not
                     01EB   769          RMSERR  TMO                        ; yes - change error code
         31    11   01F0   770          BRB     TSTSIZ
20E4 8F       50    B1   01F2   771 10$:    CMPW    R0,#SS$_LINKABORT          ; Did partner issue disconnect abort?
         0F    13   01F7   772          BEQL    EOF                        ; Branch if yes
20EC 8F       50    B1   01F9   773          CMPW    R0,#SS$_LINKDISCON         ; Did partner issue sync disconnect?
         08    13   01FE   774          BEQL    EOF                        ; Branch if yes
      2C 50    B1   0200   775          CMPW    R0,#SS$_ABORT              ; Was logical link destroyed (old code)?
         03    13   0203   776          BEQL    EOF                        ; Branch if yes
                     0205   777
                     0205   778 ;
                     0205   779 ; some unexpected qio error
                     0205   780 ;
                     0205   781
       FDF8'   31    0205   782 ERRQIO: BRW     RM$MAPERR                  ; see if error mappable to rms code
         05    E1   0208   783 EOF:    BBC     #DEV$V_SQD,-
   12 6A                020A   784                  IFB$L_PRIM_DEV(R10),10$    ; if magtape
         18    E0   020C   785          BBS     #DEV$V_FOR,-
   0E 6A                020E   786                  IFB$L_PRIM_DEV(R10),10$    ; and not foreign
                     0210   787          SSB     #IRB$V_EOF,(R9)            ; set eof flag
   74 AA    D0   0214   788          MOVL    IFB$L_EBK(R10),-
   40 A9                0217   789                  IRB$L_NRP_VBN(R9)          ; set vbn = ebk to force
                     0219   790                                             ; op's tc look at flag
   5C AA    B0   0219   791          MOVW    IFB$W_FFB(R10),-
   44 A9                021C   792                  IRB$W_NRP_OFF(R9)
                     021E   793 10$:    RMSERR  EOF                        ; change error code
   0E A9    B5   0223   794 TSTSIZ: TSTW    IRB$W_IOS2(R9)             ; zero data length?
         01    12   0226   795          BNEQ    SETVAL1                    ; treat as valid if so
```

```
                 05  0228   796          RSB
                     0229   797
                     0229   798 SETVAL1:
             FF72    31  0229   799          BRW     SETVAL
                     022C   800
```

```
                        802
                        803 ;
                        804 ;
                        805 ; either no next bdb or iop was set
                        806 ; neither should ever happen (gulp!)
                        807 ;
                        808 ; note: on read ahead or write behind, iop can be set
                        809 ;       in which case must wait for the buffer to be freed.
                        810 ;
                        811
                        812 ERRBUG: RMSTBUG FTL$_NONXTBDB
```

```
                     0233   814
                     0233   815 ;
                     0233   816 ; issue read qio for terminal
                     0233   817 ;
                     0233   818 ; must build argument list to include timeout period
                     0233   819 ; and prompt buffer address and size
                     0233   820 ;
                     0233   821
                     0233   822 TRMREAD:
                     0233   823
                     0233   824 ;
                     0233   825 ; start building the qio parameter list
                     0233   826 ;
                     0233   827
     14 A4    B1     0233   828          CMPW    BDB$W_NUMB(R4),-
     16 A4           0236   829                  BDB$W_SIZE(R4)          ; buffer big enough?
        05    1B     0238   830          BLEQU   5$                      ; lequ it is
     16 A4    B0     023A   831          MOVW    BDB$W_SIZE(R4),-
     14 A4           023D   832                  BDB$W_NUMB(R4)          ; else use buffer size
3A 68   3C    E0     023F   833 5$:       BBS     #RAB$V_ETO+ROP,(R8),TRMREAD_EXTEND
                     0243   834                                          ; handle extended terminal operation
7E   34 A8    9A     0243   835          MOVZBL  RAB$B_PSZ(R8),-(SP)     ; p6 = prompt buffer size
     30 B8    9F     0247   836          PUSHAB  @RAB$L_PBF(R8)          ; p5 = prompt buffer addr
        05    12     024A   837          BNEQ    10$                     ; branch if specified
     04 AE    D4     024C   838          CLRL    QIO$_P6-QIO$_P5(SP)     ; insure 0 size
        09    11     024F   839          BRB     20$
                     0251   840
                     0251   841 ;
                     0251   842 ; probe prompt buffer for caller readability
                     0251   843 ;
                     0251   844
                     0251   845 10$:
                     0251   846          IFNORD  QIO$_P6-QIO$_P5(SP),-
                     0251   847                  @0(SP),-
                     0251   848                  ERRPBF,IRB$B_MODE(R9)   ; branch if not readable
        7E   D4      025A   849 20$:     CLRL    -(SP)                   ; p4 = terminator class
                     025C   850                                          ;  = default (tc$m_standard)
7E   1F A8    9A     025C   851          MOVZBL  RAB$B_TMO(R8),-(SP)     ; p3 = timeout period
      0088    30     0260   852          BSBW    SETP2_EFN               ; finish build of arg list
                     0263   853
                     0263   854 ;
                     0263   855 ;  check for read prompt and if found change the i/o function code
                     0263   856 ;
                     0263   857
        3E   E1      0263   858          BBC     #RAB$V_PMT+ROP,-
     04 68           0265   859                  (R8),30$                ; branch if not read prompt
        37   D0      0267   860          MOVL    #IO$_READPROMPT,-
     08 AE           0269   861                  QIO$_FUNC-4(SP)         ; substitute function code
                     026B   862                                          ; or in the terminal read options bits
                     026B   863
                     026B   864 ;
                     026B   865 ; the following assumes check that the rms definitions for
                     026B   866 ; the terminal read sub functions match those of
                     026B   867 ; the starlet system
                     026B   868 ;
                     026B   869
                     026B   870          ASSUME  <RAB$V_RNE-24> EQ <IO$V_NOECHO - 6>
```

```
                                026B   871          ASSUME  <RAB$V_PTA-24> EQ <IO$V_PURGE - 6>
                                026B   872          ASSUME  <RAB$V_TMO-24> EQ <IO$V_TIMED - 6>
                                026B   873          ASSUME  <RAB$V_RNF-24> EQ <IO$V_NOFILTR - 6>
                                026B   874          ASSUME  <RAB$V_CVT-24> EQ <IO$V_CVTLOW - 6>
                                026B   875
                                026B   876 ;
                                026B   877
     50    68    06    38    EF  026B   878 30$:     EXTZV   #RAB$V_RNE+ROP,#6,(R8),R0; get the option bits
  08 AE    06    06    50    F0  0270   879          INSV    R0,#6,#6,QIO$_FUNC-4(SP); and set them into the i/o func
                   FF08    31  0276   880          BRW     DOQIO                   ; common finish
                               0279   881
                               0279   882 ;+
                               0279   883 ; Extended terminal operation QIO argument setup (assumes ROP ETO set
                               0279   884 ; plus an XABTRM to specify the item list for the QIO).
                               0279   885 ;-
                               0279   886
                               0279   887 TRM_XAB_ARGS:                          ; Dispatching info for RM$XAB_SCAN.
                               0279   888
                               0279   889          ;       XAB code        XAB length      CASE index
                         00'24 1F  0279   890          .BYTE   XAB$C_TRM,      XAB$C_TRMLEN,   XBC$C_GETPUTTRM
                            00     027C   891          .BYTE   0                       ; End of table flag.
                               027D   892
                               027D   893 TRMREAD_EXTEND:
                               027D   894
                               027D   895          ;+
                               027D   896          ; Scan for XABTRM, returning address in R5. If none, return an error
                               027D   897          ; (if ROP ETO is set, an XABTRM must be present).
                               027D   898          ;-
                               027D   899
        5C    F9 AF    9E  027D   900          MOVAB   TRM_XAB_ARGS, AP        ; Set XAB scan arguments.
              3E   BB  0281   901          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save registers.
           FD7A'   30  0283   902          BSBW    RM$XAB_SCAN             ; Scan for XABTRM; return address in R5.
        2A 50      E9  0286   903          BLBC    R0, 30$                 ; If error, then return.
     07 54   00'   E0  0289   904 10$:     BBS     S^#XBC$C_GETPUTTRM, R4, 20$
                               028D   905                                  ; If XABTRM found then continue
                               028D   906          RMSERR  XNF             ; else set error to XAB not found and
           1F    11  0292   907          BRB     30$                     ;  return.
                               0294   908
                               0294   909          ;+
                               0294   910          ; Push QIO arguments for extended terminal read.
                               0294   911          ;-
                               0294   912
        50    55   D0  0294   913 20$:     MOVL    R5, R0                  ; Save address of XABTRM.
              3E   BA  0297   914          POPR    #^M<R1,R2,R3,R4,R5>     ; Restore registers.
     7E    0C A0    3C  0299   915          MOVZWL  XAB$W_ITMLST_LEN(R0), -(SP)
                               029D   916                                  ; P6 = item list length in bytes.
           08 B0    9F  029D   917          PUSHAB  @XAB$L_ITMLST(R0)       ; P5 = item list address.
              7E    D4  02A0   918          CLRL    -(SP)                   ; P4 = must be zero.
     7E    0A A9    9A  02A2   919          MOVZBL  IRB$B_MODE(R9), -(SP)   ; P3 = mode to probe item list.
              43    10  02A6   920          BSBB    SETP2_EFN               ; Finish build of QIO args (P2 -> efn).
08 AE    00008000 8F    C8  02A8   921          BISL2   #IO$M_EXTEND, QIO$_FUNC-4(SP)
                               02B0   922                                  ; Set extended I/O bit in function code.
              FECE   31  02B0   923          BRW     DOQIO                   ; Continue at common QIO code.
                               02B3   924
                               02B3   925          ;+
                               02B3   926          ; Error exit from TRMREAD_EXTEND.
                               02B3   927          ;-
```

```
                02B3    928
     3E   BA    02B3    929 30$:    POPR    #^M<R1,R2,R3,R4,R5>     ; Restore registers.
  14 A4   B4    02B5    930         CLRW    BDB$W_NUMB(R4)          ; no data in this case
          05    02B8    931         RSB                             ; Return with error from XAB scan.
                02B9    932
                02B9    933 ;
                02B9    934 ; prompt buffer not readable
                02B9    935 ;
                02B9    936
                02B9    937 ERRPBF:
50   8E   7D    02B9    938         MOVQ    (SP)+,R0                ; clean up stack
  14 A4   B4    02BC    939         CLRW    BDB$W_NUMB(R4)          ; no data in this case
                02BF    940         $RMSERR PBF
          05    02C4    941         RSB
                02C5    942
```

```
                     02C5   944
                     02C5   945 ;
                     02C5   946 ; routine to build qio args on stack
                     02C5   947 ; p3 = vbn, p4 = p5 = p6 = 0
                     02C5   948 ; input:        bdb$l_vbn(r4) = vbn
                     02C5   949 ; note:         r0,r1 destroyed
                     02C5   950 ;
                     02C5   951
                     02C5   952 SETP6_P3:
                  50 8ED0   02C5   953         POPL    R0                      ; get return pc
                  7E   7C   02C8   954         CLRQ    -(SP)                   ; p6, p5 = 0
                  00   DD   02CA   955         PUSHL   #0                      ; p4 = 0
               1C A4   DD   02CC   956         PUSHL   BDB$L_VBN(R4)           ; p3 = vbn
                  60   17   02CF   957         JMP     (R0)                    ; return
                     02D1   958
                     02D1   959 ;
                     02D1   960 ; routine to build qio args on stack for a read ahead or write behind qio
                     02D1   961 ; set p2 thru efn
                     02D1   962 ;
                     02D1   963
                     02D1   964         ASSUME  QIO$_ASTPRM EQ <QIO$_P1 - 4>
                     02D1   965         ASSUME  QIO$_ASTADR EQ <QIO$_ASTPRM - 4>
                     02D1   966         ASSUME  QIO$_IOSB EQ <QIO$_ASTADR - 4>
                     02D1   967
                     02D1   968 SETP2_EFN_RAH:
                  50 8ED0   02D1   969         POPL    R0                      ; get return pc
            7E    14 A4    3C   02D4   970         MOVZWL  BDB$W_NUMB(R4),-(SP)    ; p2 = xfer size
               18 A4   DD   02D8   971         PUSHL   BDB$L_ADDR(R4)          ; p1 = buffer addr from bdb
                  54   DD   02DB   972         PUSHL   R4                      ; astprm = bdb addr
            0000'CF    DF   02DD   973         PUSHAL  W^RM$RAHWBHAST          ; set ast service address
               48 A4   DF   02E1   974         PUSHAL  BDB$L_IOSB(R4)          ; use iosb in bdb
                     02E4   975         CSB     #BDB$V_AST_DCL,-
                     02E4   976                 BDB$B_FLGS(R4)                  ; clear i/o done flag
                  17   11   02E9   977         BRB     SETFUNC                 ; go join common code
                     02EB   978
                     02EB   979 ;
                     02EB   980 ; routine to build qio args on stack
                     02EB   981 ; sets p2 thru efn
                     02EB   982 ;
                     02EB   983
                     02EB   984 SETP2_EFN:
                  50 8ED0   02EB   985         POPL    R0                      ; get return pc
            7E    14 A4    3C   02EE   986         MOVZWL  BDB$W_NUMB(R4),-(SP)    ; p2 = xfer size
               18 A4   DD   02F2   987         PUSHL   BDB$L_ADDR(R4)          ; p1 = buffer addr from bdb
                     02F5   988
                     02F5   989         ASSUME  QIO$_ASTPRM EQ <QIO$_P1 - 4>
                     02F5   990         ASSUME  QIO$_ASTADR EQ <QIO$_ASTPRM - 4>
                     02F5   991         ASSUME  QIO$_IOSB EQ <QIO$_ASTADR - 4>
                     02F5   992         ASSUME  QIO$_FUNC EQ <QIO$_IOSB - 4>
                     02F5   993
                     02F5   994 SETAST_EFN:                                  ; entry point from rm$segwtur
                  59   DD   02F5   995         PUSHL   R9                      ; astprm = irab addr = r9
            0000'CF    DF   02F7   996         PUSHAL  W^RM$STALLAST           ; ast addr
               0C A9   DF   02FB   997         PUSHAL  IRB$L_IOS(R9)           ; iosb addr
                     02FE   998
                  18   E0   02FE   999         BBS     #DEV$V_FOR,-
                  6A        0300  1000                 IFB$L_PRIM_DEV(R10),-
```

```
                 2D        0301  1001                    READ_LOG                ; branch if foreign dev
                           0302  1002 SETFUNC:
                 31   DD   0302  1003            PUSHL   #IO$_READVBLK           ; i/o function code
                           0304  1004
                           0304  1005            ASSUME  QIO$_CHAN EQ <QIO$_FUNC - 4>
                           0304  1006
           7E    20 AA   3C   0304  1007 SETCHN: MOVZWL  IFB$W_CHNL(R10),-(SP)   ; i/o channel
                           0308  1008
                           0308  1009            ASSUME  QIO$_EFN EQ <QIO$_CHAN - 4>
                           0308  1010            ASSUME  IFB$B_EFN EQ IRB$B_EFN
                           0308  1011 SETEFN:
                           0308  1012            ASSUME  IRB$C_BID&1 EQ 0
                           0308  1013            ASSUME  IFB$C_BID&1 EQ 1
                           0308  1014            ASSUME  IRB$B_BID EQ IFB$B_BID
                           0308  1015
        08 08 A9   E8      0308  1016            BLBS    IRB$B_BID(R9),2$        ; branch if ifab
                 23   E0   030C  1017            BBS     #IRB$V_ASYNC,-
              1B 69         030E  1018                    (R9),ASYNCEFN           ; branch if async operation
                 2A   E0   0310  1019            BBS     #IRB$V_RAHWBH,-
              17 69         0312  1020                    (R9),ASYNCEFN           ; branch if rah/wbh call
           7E    0B A9   9A 0314  1021 2$:      MOVZBL  IRB$B_EFN(R9),-(SP)     ; use allocated efn
                 0F   12   0318  1022            BNEQ    10$                     ; branch if one allocated
              6E    1D   90 031A  1023            MOVB    #29,(SP)                ; always use efn 29
                 02   E2   031D  1024            BBSS    #PIO$V_SYNC1,-
        00 00000000'9F      031F  1025                    @#PIO$GW_STATUS,5$      ; claim efn
           0B A9    6E   90 0325  1026 5$:      MOVB    (SP),IRB$B_EFN(R9)      ; save efn
                 60   17   0329  1027 10$:     JMP     (R0)                    ; return
                           032B  1028
                           032B  1029 ASYNCEFN:
                 1F   DD   032B  1030            PUSHL   #IMP$C_ASYQIOEFN        ; use (throw away) async efn
                 60   17   032D  1031            JMP     (R0)                    ; return
                           032F  1032
                           032F  1033 ;
                           032F  1034
                           032F  1035 READ_LOG:
           21    DD   032F  1036            PUSHL   #IO$_READLBLK           ; set read logical for foreign dev
           D1    11   0331  1037            BRB     SETCHN                  ; rejoin main line
                           0333  1038
                           0333  1039 ;
                           0333  1040
                           0333  1041 RM$SETEFN::
        50 8ED0   0333  1042            POPL    R0                      ; get return pc
           D0    11   0336  1043            BRB     SETEFN
```

```
0338  1045          .SBTTL  RM$SEQWT - ROUTINE TO PERFORM SEQUENTIAL WRITES
0338  1046
0338  1047 ;++
0338  1048 ; RM$SEQWT
0338  1049 ;
0338  1050 ; this routine performs write virtual block for the sequential
0338  1051 ; file organization. one of several flavors is performed
0338  1052 ; depending upon whether the device is a unit record
0338  1053 ; device and whether write-behind multi-buffer is in progress.
0338  1054 ;
0338  1055 ; Calling sequence:
0338  1056 ;
0338  1057 ;       bsbw    rm$seqwtbd - write to a block device
0338  1058 ;       bsbw    rm$seqwtur - write to a unit record device
0338  1059 ;
0338  1060 ; Input Parameters:
0338  1061 ;
0338  1062 ;       r11     impure area addr
0338  1063 ;       r10     ifab addr
0338  1064 ;       r9      irab addr
0338  1065 ;       r8      rab addr
0338  1066 ;       r4      bdb addr of buffer to write
0338  1067 ;
0338  1068 ; Implicit Inputs:
0338  1069 ;
0338  1070 ;       rab$v_wbh       user wants write behind option
0338  1071 ;       irb$v_rahwbh    write behind processing flag
0338  1072 ;       bdb$l_addr      address of buffer
0338  1073 ;       bdb$l_vbn       vbn to write (block device)
0338  1074 ;       bdb$w_numb      # bytes to write
0338  1075 ;       bdb$b_pre_cctl  carriage control byte (unit record device)
0338  1076 ;       bdb$b_post_cctl
0338  1077 ;       ifb$l_chnl      i/o channel
0338  1078 ;
0338  1079 ; outputs:
0338  1080 ;
0338  1081 ;       r0              status code
0338  1082 ;       r1-r3,ap        destroyed
0338  1083 ;
0338  1084 ; Implicit Outputs:
0338  1085 ;
0338  1086 ;       bdb$v_drt               cleared
0338  1087 ; if not write behind:
0338  1088 ;       irb$l_ios,irb$l_ios4    system specified i/o status block data
0338  1089 ;       rab$l_stv               system error code if sys error
0338  1090 ; if write behind:
0338  1091 ;       bdb$v_iop               set
0338  1092 ;
0338  1093 ; status codes:
0338  1094 ;
0338  1095 ;       standard rms, in particular, suc, eof, and sys.
0338  1096 ;
0338  1097 ; Side Effects:
0338  1098 ;
0338  1099 ;       if write behind, an ast will occur at rm$rahwbhast
0338  1100 ;       upon i/o completion
0338  1101 ;
```

```
                     0338  1102 ;       if not write behind, may have switched to running at ast level.
                     0338  1103 ;
                     0338  1104 ;       bdb$v_iop is set for duration of i/o
                     0338  1105 ;
                     0338  1106 ;       Note that AP is used as RAHWBH/SYNC flag and as an EOF loop detector
                     0338  1107 ;       for magtape append error recovery.
                     0338  1108 ;
                     0338  1109 ;--
                     0338  1110 ;
                     0338  1111 ; write to a unit record device - build arg list on stack
                     0338  1112 ; note: not doing write behind to unit record device
                     0338  1113 ;
                     0338  1114 ;--
                     0338  1115
                     0338  1116 RM$SEQWTUR::
                     0338  1117         $TSTPT  SEQWTUR
                     033E  1118
                     033E  1119 ;
                     033E  1120 ; perform remote file access using network specific code. however,
                     033E  1121 ; perform task-to-task communication using standard rms code.
                     033E  1122 ;
                     033E  1123
 03 6A    3E    E1   033E  1124         BBC     #IFB$V_DAP,(R10),5$      ; is this remote access via dap?
       0095    31   0342  1125         BRW     NTPUT                    ; branch if so
         7E    7C   0345  1126 5$:     CLRQ    -(SP)                    ; p6,p5 = 0
         7E    7C   0347  1127         CLRQ    -(SP)                    ; p4,p3 = 0
                     0349  1128
                     0349  1129         ASSUME  BDB$B_PRE_CCTL+1 EQ BDB$B_POST_CCTL
                     0349  1130
 06 AE    4A A4    B0   0349  1131         MOVW    BDB$B_PRE_CCTL(R4),6(SP); p4 = carriage control
    50    7A'AF    9E   034E  1132         MOVAB   B^WTCOM0,R0              ; ret addr after setast_efn
    7E    14 A4    3C   0352  1133         MOVZWL  BDB$W_NUMB(R4),-(SP)     ; push transfer size
          4C A4    DD   0356  1134         PUSHL   BDB$L_CURBUFADR(R4)      ; address of buffer
             9A    11   0359  1135         BRB     SETAST_EFN               ; and set up rest
                     035B  1136
                     035B  1137 ;
                     035B  1138 ; write to a block device - build arg list on stack
                     035B  1139 ;
                     035B  1140
                     035B  1141 RM$SEQWTBD::
                     035B  1142         $TSTPT  SEQWTBD
             2A    E1   0361  1143         BBC     #IRB$V_RAHWBH,-
          0F 69           0363  1144                 (R9),RM$WTBUFWT          ; branch if wbh not enabled
             2A    E1   0365  1145         BBC     #RAB$V_WBH+ROP,-
          0B 68           0367  1146                 (R8),RM$WTBUFWT          ; or user does not buy in
           FF59    30   0369  1147         BSBW    SETP6_P3                 ; setup qio call arg block on stack
           FF62    30   036C  1148         BSBW    SETP2_EFN_RAH
       5C    01    D0   036F  1149         MOVL    #1,AP                    ; set write behind flag
             08    11   0372  1150         BRB     WTCOM1                   ; join common code
                     0374  1151
                     0374  1152 RM$WTBUFWT::
           FF4E    30   0374  1153         BSBW    SETP6_P3                 ; p6=p5=p4=0, p3 = vbn
                     0377  1154
                     0377  1155 ;
                     0377  1156 ; build remaining argument list on stack
                     0377  1157 ;
                     0377  1158
```

```
                FF71   30   0377  1159 WTCOMM: BSBW    SETP2_EFN               ; build p2 thru efn on stack
                  5C   D4   037A  1160 WTCOM0: CLRL    AP                      ; clear write behind flag
                       037C  1161
                       037C  1162          ASSUME  IO$_WRITEVBLK EQ IO$_READVBLK-1
                       037C  1163          ASSUME  IO$_WRITELBLK EQ IO$_READLBLK-1
                       037C  1164
             08 AE     D7   037C  1165 WTCOM1: DECL    QIO$_FUNC-4(SP)         ; change to write
                02     E0   037F  1166          BBS     #DEV$V_TRM,-
             40 6A          0381  1167                  IFB$L_PRIM_DEV(R10),30$ ; branch if terminal
                14     E0   0383  1168          BBS     #DEV$V_MBX,-
             45 6A          0385  1169                  IFB$L_PRIM_DEV(R10),40$ ; branch if mailbox
                       0387  1170
                       0387  1171          ASSUME  QIO$_NARGS EQ 12
                       0387  1172
                       0387  1173 10$:
          0A A4    02   8A   0387  1174          BICB2   #BDB$M_DRT,BDB$B_FLGS(R4) ; note not dirty anymore
          0A A4    04   88   038B  1175          BISB2   #BDB$M_IOP,BDB$B_FLGS(R4) ; note i/o in progress
             00A0  CA   95   038F  1176          TSTB    IFB$B_JNLFLG(R10)       ; are we journaling?
                11   13   0393  1177          BEQL    15$                     ; nope, no need to force jnl records
                7E   54   7D   0395  1178          MOVQ    R4, -(SP)               ; save R4 and R5 from being clobbered
                54   D4   0398  1179          CLRL    R4                      ; force all jnl entries for this process
       00000000'EF   16   039A  1180          JSB     RM$FRCJNL               ; do the force
                54 50   E9   03A0  1181          BLBC    R0, ERRWRT              ; if entries weren't be forced, get out
                54   8E   7D   03A3  1182          MOVQ    (SP)+, R4               ; restore R4 and R5
       00000000'9F   0C   FB   03A6  1183 15$:    CALLS   #12,@#SYS$QIO           ; do the qio
                3C 50   E9   03AD  1184          BLBC    R0,ERRQIO1
                0F 5C   E8   03B0  1185          BLBS    AP,20$                  ; branch if write behind call
                5C   DD   03B3  1186          PUSHL   AP                      ; Keep for check on EOF loop
              FC48'  30   03B5  1187          BSBW    RM$STALL                ; await completion
                5C 8ED0   03B8  1188          POPL    AP                      ; Restore EOF recovery flag.
          0A A4    04   8A   03BB  1189          BICB2   #BDB$M_IOP,BDB$B_FLGS(R4) ; note i/o completion
                35 50   E9   03BF  1190          BLBC    R0,ERRWRT               ; branch on error
                       05   03C2  1191 20$:    RSB
          C0 68    3F   E1   03C3  1192 30$:    BBC     #RAB$V_CCO+ROP,(R8),10$ ; branch if cco not speced
                06   E3   03C7  1193          BBCS    #IO$V_CANCTRLO,-
            BB 08 AE          03C9  1194                  QIO$_FUNC-4(SP),10$     ; cancel control o. this
                       03CC  1195                                                  ; won't be set already.
          B7 68    39   E1   03CC  1196 40$:    BBC     #RAB$V_TMO+ROP,(R8),10$ ; don't want now checked
                1F A8   95   03D0  1197          TSTB    RAB$B_TMO(R8)           ; 0 timeout spec'd?
                B2   12   03D3  1198          BNEQ    10$                     ; nope, do normal read
                06   E3   03D5  1199          BBCS    #IO$V_NOW,-
            AD 08 AE          03D7  1200                  QIO$_FUNC-4(SP),10$     ; set now sub func and branch.
                       03DA  1201                                                  ; won't be set already.
                       03DA  1202
                       03DA  1203 ;++
                       03DA  1204 ;
                       03DA  1205 ; perform network put/update function via the remote fal
                       03DA  1206 ;
                       03DA  1207 ;--
                       03DA  1208
                       03DA  1209 NTPUT:
          07 69    33   E4   03DA  1210          BBSC    #IRB$V_UPDATE,(R9),10$  ; branch if update operation
                       03DE  1211
                       03DE  1212 ;
                       03DE  1213 ;  and clear flag
                       03DE  1214 ;
                       03DE  1215
```

```
00000000'EF   16  03DE  1216          JSB     NT$PUT                          ; perform network put
              05  03E4  1217          RSB                                     ; exit with rms code in r0
                  03E5  1218
00000000'EF   16  03E5  1219 10$:     JSB     NT$UPDATE                       ; perform network update
              05  03EB  1220          RSB                                     ; exit with rms code in r0
                  03EC  1221
                  03EC  1222 ;
                  03EC  1223 ; error on qio call
                  03EC  1224 ;
                  03EC  1225
                  03EC  1226 ERRQIO1:
 0A A4    04  8A  03EC  1227          BICB2   #BDB$M_IOP,BDB$B_FLGS(R4)       ; make sure iop is not set
 03 6A    3F  E0  03F0  1228          BBS     #IFB$V_NSP,(R10),ERRWRT         ; branch if task to task net
                  03F4  1229
                  03F4  1230 ;
                  03F4  1231 ; operation so abort error
                  03F4  1232 ; doesn't map to eof
                  03F4  1233 ;
                  03F4  1234
        FDC3  31  03F4  1235          BRW     ERRSYS
                  03F7  1236
                  03F7  1237 ;
                  03F7  1238 ; error writing - set default error code and go map if possible, unless
                  03F7  1239 ; we are magtape and error is EOF.  Then we want to back up before the tape
                  03F7  1240 ; mark and try again.
                  03F7  1241 ;
                  03F7  1242
        05  E1  03F7  1243 ERRWRT: BBC     #DEV$V_SQD,-                    ; Only backspace if the
     32 6A        03F9  1244                  IFB$L_PRIM_DEV(R10),20$         ; device is a magtape
0870 8F    50  B1  03FB  1245          CMPW    R0,#SS$_ENDOFFILE               ; Was this an EOF?
        2B  12  0400  1246          BNEQ    20$                             ; Branch if not.
     02 5C  D1  0402  1247          CMPL    AP,#2                           ; Is this our first time here?
        26  13  0405  1248          BEQL    20$                             ; Branch if not.
                  0407  1249
     0070 8F    BB  0407  1250          PUSHR   #^M<R4,R5,R6>                   ; Save working registers
     56   01    CE  040B  1251          MNEGL   #1,R6                           ; Number of blocks to space
        FBEF'  30  040E  1252          BSBW    RM$SPACE_MT                     ;  Back up before last block
       06 50   E9  0411  1253          BLBC    R0,10$                          ; If it failed, give up now
     56   01    D0  0414  1254          MOVL    #1,R6                           ; Otherwise, now bump to end of
        FBE6'  30  0417  1255          BSBW    RM$SPACE_MT                     ; the magtape file.
     0070 8F    BA  041A  1256 10$:     POPR    #^M<R4,R5,R6>                   ; Restore the working regs.
       0C 50   E9  041E  1257          BLBC    R0,20$                          ; Check if space forward worked
                  0421  1258
        FEA1  30  0421  1259          BSBW    SETP6_P3                        ; Set up p3 - p6 of qio
        FEC4  30  0424  1260          BSBW    SETP2_EFN                       ; and now set up the rest
     5C   02    D0  0427  1261          MOVL    #2,AP                           ; Signal we've hit EOF once.
        FF4F  31  042A  1262          BRW     WTCOM1                          ; And try it again.
                  042D  1263
                  042D  1264 20$:     RMSERR  WER,R1
        FBCB'  31  0432  1265          BRW     RM$MAPERR
                  0435  1266
                  0435  1267          .END
```

```
$$.PSECT_EP                     = 00000000
$$ARGS                          = 0000000C
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
$$T1                            = 00000034
ASYNCEFN                          0000032B R      01
BDB$B_FLGS                      = 0000000A
BDB$B_POST_CCTL                 = 0000004B
BDB$B_PRE_CCTL                  = 0000004A
BDB$L_ADDR                      = 00000018
BDB$L_CURBUFADR                 = 0000004C
BDB$L_FLINK                     = 00000000
BDB$L_IOSB                      = 00000048
BDB$L_VBN                       = 0000001C
BDB$L_WAIT                      = 00000024
BDB$M_DRT                       = 00000002
BDB$M_IOP                       = 00000004
BDB$M_VAL                       = 00000001
BDB$V_AST_DCL                   = 00000006
BDB$V_IOP                       = 00000002
BDB$V_VAL                       = 00000000
BDB$W_NUMB                      = 00000014
BDB$W_SIZE                      = 00000016
CHECK_IO                          00000102 R      01
CHKEOF                            000001D9 R      01
CTRLZ                           = 0000001A
DEV$V_FOR                       = 00000018
DEV$V_MBX                       = 00000014
DEV$V_RND                       = 0000001C
DEV$V_SQD                       = 00000005
DEV$V_TRM                       = 00000002
DOQIO                             00000181 R      01
EOF                               00000208 R      01
ERRBUG                            0000022C R      01
ERRPBF                            000002B9 R      01
ERRQIO                            00000205 R      01
ERRQIO1                           000003EC R      01
ERRSYS                            000001BA R      01
ERRWRT                            000003F7 R      01
ERR_READ                          000001C8 R      01
FTL$_NONXTBDB                   = FFFFFFF4
IFB$B_BID                       = 00000008
IFB$B_EFN                       = 0000000B
IFB$B_JNLFLG                    = 000000A0
IFB$C_BID                       = 0000000B
IFB$L_BDB_FLNK                  = 00000040
IFB$L_EBK                       = 00000074
IFB$L_PRIM_DEV                  = 00000000
IFB$V_DAP                       = 0000003E
IFB$V_NSP                       = 0000003F
IFB$W_CHNL                      = 00000020
IFB$W_FFB                       = 0000005C
IMP$C_ASYQIOEFN                 = 0000001F
IO$M_EXTEND                     = 00008000
IO$V_CANCTRLO                   = 00000006
IO$V_CVTLOW                     = 00000008
IO$V_NOECHO                     = 00000006
IO$V_NOFILTR                    = 00000009
IO$V_NOW                        = 00000006
IO$V_PURGE                      = 0000000B
IO$V_TIMED                      = 00000007
IO$_READLBLK                    = 00000021
IO$_READPROMPT                  = 00000037
IO$_READVBLK                    = 00000031
IO$_WRITELBLK                   = 00000020
IO$_WRITEVBLK                   = 00000030
IRB$B_BID                       = 00000008
IRB$B_EFN                       = 0000000B
IRB$B_MODE                      = 0000000A
IRB$C_BID                       = 0000000A
IRB$L_CURBDB                    = 00000020
IRB$L_IFAB_LNK                  = 00000000
IRB$L_IOS                       = 0000000C
IRB$L_NRP_VBN                   = 00000040
IRB$L_NXTBDB                    = 0000003C
IRB$V_ASYNC                     = 00000023
IRB$V_EOF                       = 00000021
IRB$V_FIND                      = 00000029
IRB$V_RAHWBH                    = 0000002A
IRB$V_UPDATE                    = 00000033
IRB$W_IOS2                      = 0000000E
IRB$W_NRP_OFF                   = 00000044
NT$FIND                           ********    X   01
NT$GET                            ********    X   01
NT$PUT                            ********    X   01
NT$UPDATE                         ********    X   01
NTGET                             000001A3 R      01
NTPUT                             000003DA R      01
PIO$A_TRACE                       ********    X   01
PIO$GW_STATUS                     ********    X   01
PIO$V_SYNC1                     = 00000002
QIO$_ASTADR                     = 00000014
QIO$_ASTPRM                     = 00000018
QIO$_CHAN                       = 00000008
QIO$_EFN                        = 00000004
QIO$_FUNC                       = 0000000C
QIO$_IOSB                       = 00000010
QIO$_NARGS                      = 0000000C
QIO$_P1                         = 0000001C
QIO$_P2                         = 00000020
QIO$_P3                         = 00000024
QIO$_P4                         = 00000028
QIO$_P5                         = 0000002C
QIO$_P6                         = 00000030
QIODONE                           00000196 R      01
RAB$B_PSZ                       = 00000034
RAB$B_TMO                       = 0000001F
RAB$L_PBF                       = 00000030
RAB$L_ROP                       = 00000004
RAB$L_STV                       = 0000000C
RAB$V_CCO                       = 0000001F
RAB$V_CVT                       = 0000001A
```

```
RAB$V_ETO                       = 0000001C
RAB$V_PMT                       = 0000001E
RAB$V_PTA                       = 0000001D
RAB$V_RNE                       = 00000018
RAB$V_RNF                       = 0000001B
RAB$V_TMO                       = 00000019
RAB$V_WBH                       = 0000000A
READ_LOG                          0000032F R      01
RET                               000001A2 R      01
RM$BUG                            ********   X    01
RM$FRCJNL                         ********   X    01
RM$MAPERR                         ********   X    01
RM$QUIET_SEQMBF                   00000026 RG     01
RM$RAHWBRAST                      ********   X    01
RM$RDBUFWT                        00000165 RG     01
RM$SEQFLNKBDB                     00000080 RG     01
RM$SEQRAH                         00000000 RG     01
RM$SEQRD                          00000094 RG     01
RM$SEQWTBD                        0000035B RG     01
RM$SEQWTUR                        00000338 RG     01
RM$SETEFN                         00000333 RG     01
RM$SPACE_MT                       ********   X    01
RM$STALL                          ********   X    01
RM$STALLAST                       ********   X    01
RM$STALLRAHWBH                    00000059 RG     01
RM$WTBUFWT                        00000374 RG     01
RM$XAB_SCAN                       ********   X    01
RMS$_EOF                        = 0001827A
RMS$_PBF                        = 00018614
RMS$_RER                        = 0001C0F4
RMS$_SYS                        = 0001C10C
RMS$_TMO                        = 000181B0
RMS$_WBE                        = 0001C12C
RMS$_WER                        = 0001C114
RMS$_XNF                        = 00018704
ROP                             = 00000020
SETAST_EFN                        000002F5 R      01
SETCHN                            00000304 R      01
SETEFN                            00000308 R      01
SETFUNC                           00000302 R      01
SETP2_EFN                         000002EB R      01
SETP2_EFN_RAH                     000002D1 R      01
SETP6_P3                          000002C5 R      01
SETVAL                            0000019E R      01
SETVAL1                           00000229 R      01
SS$_ABORT                       = 0000002C
SS$_ENDOFFILE                   = 00000870
SS$_LINKABORT                   = 000020E4
SS$_LINKDISCON                  = 000020EC
SS$_TIMEOUT                     = 0000022C
SUCCESS                           000000F6 R      01
SYS$QIO                           ********   X    01
TPT$L_SEQRD                       ********   X    01
TPT$L_SEQWTBD                     ********   X    01
TPT$L_SEQWTUR                     ********   X    01
TRMREAD                           00000233 R      01
TRMREAD_EXTEND                    0000027D R      01
TRM_XAB_ARGS                      00000279 R      01
TSTSIZ                            00000223 R      01
WTCOM0                            0000037A R      01
WTCOM1                            0000037C R      01
WTCOMM                            00000377 R      01
XAB$C_TRM                       = 0000001F
XAB$C_TRMLEN                    = 00000024
XAB$L_ITMLST                    = 00000008
XAB$W_ITMLST_LEN                = 0000000C
XBC$C_GETPUTTRM                   ********   X    01
XIT                               00000093 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$RMS1	00000435 (1077.)	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	34	00:00:00.11	00:00:00.87
Command processing	128	00:00:00.72	00:00:04.14
Pass 1	484	00:00:18.91	00:00:47.45
Symbol table sort	0	00:00:02.85	00:00:03.18
Pass 2	215	00:00:04.45	00:00:10.33
Symbol table output	22	00:00:00.20	00:00:00.36
Psect synopsis output	2	00:00:00.04	00:00:00.12
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	887	00:00:27.28	00:01:06.45

The working set limit was 1800 pages.
106587 bytes (209 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1903 non-local and 44 local symbols.
1267 source lines were read in Pass 1, producing 17 object records in Pass 2.
33 pages of virtual memory were used to define 32 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	16
_255DUA28:[SYS.OBJ]LIB.MLB;1	2
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	28

2015 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM1SEQXFR/OBJ=OBJ$:RM1SEQXFR MSRC$:RM1SEQXFR/UPDATE=(ENH$:RM1SEQXFR)+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