```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM       MM      11        SSSSSSSS  TTTTTTTTTT  MM       MM  FFFFFFFFFF  MM       MM  TTTTTTTTTT
RRRRRRRR    MM       MM      11        SSSSSSSS  TTTTTTTTTT  MM       MM  FFFFFFFFFF  MM       MM  TTTTTTTTTT
RR      RR  MMMM   MMMM    1111      SS              TT      MMMM   MMMM  FF          MMMM   MMMM      TT
RR      RR  MMMM   MMMM    1111      SS              TT      MMMM   MMMM  FF          MMMM   MMMM      TT
RR      RR  MM  MM  MM      11        SS              TT      MM  MM  MM  FF          MM  MM  MM      TT
RR      RR  MM  MM  MM      11        SS              TT      MM  MM  MM  FF          MM  MM  MM      TT
RRRRRRRR    MM       MM      11          SSSSSS        TT      MM      MM  FFFFFFFF    MM      MM      TT
RRRRRRRR    MM       MM      11          SSSSSS        TT      MM      MM  FFFFFFFF    MM      MM      TT
RR  RR      MM       MM      11                SS      TT      MM      MM  FF          MM      MM      TT
RR  RR      MM       MM      11                SS      TT      MM      MM  FF          MM      MM      TT
RR    RR    MM       MM      11                SS      TT      MM      MM  FF          MM      MM      TT      ....
RR    RR    MM       MM      11                SS      TT      MM      MM  FF          MM      MM      TT      ....
RR      RR  MM       MM    111111    SSSSSSSS        TT      MM      MM  FF          MM      MM      TT      ....
RR      RR  MM       MM    111111    SSSSSSSS        TT      MM      MM  FF          MM      MM      TT      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0000     1          $BEGIN  RM1STMFMT,000,RM$RMS1,<STREAM FORMAT SUPPORT>
0000     2
0000     3 ;
0000     4 ;****************************************************************************
0000     5 ;*                                                                          *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     8 ;*  ALL RIGHTS RESERVED.                                                    *
0000     9 ;*                                                                          *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    15 ;*  TRANSFERRED.                                                            *
0000    16 ;*                                                                          *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    19 ;*  CORPORATION.                                                            *
0000    20 ;*                                                                          *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    23 ;*                                                                          *
0000    24 ;*                                                                          *
0000    25 ;****************************************************************************
0000    26 ;
```

```
0000     28 ;++
0000     29 ; Facility: RMS32
0000     30 ;
0000     31 ; Abstract:
0000     32 ;                    This module provides stream format
0000     33 ;                    specific processing for the sequential file organization.
0000     34 ;
0000     35 ;
0000     36 ; Environment:
0000     37 ;                    Star processor running Starlet exec.
0000     38 ;
0000     39 ; Author:            Ron Schaefer                                    32-Jul-1981
0000     40 ;
0000     41 ; Modified By:
0000     42 ;
0000     43 ;           V03-013 DAS0001         David Solomon           16-Jul-1984
0000     44 ;                   Fix accvio due to RM$RELBLK1 being called with R4=0 (QAR 581).
0000     45 ;
0000     46 ;           V03-012 JEJ0034         J E Johnson             02-May-1984
0000     47 ;                   The previous fix missed the case where there is no
0000     48 ;                   current BDB (eof on a bucket boundry).  In this case
0000     49 ;                   do not attempt to update the NRP offset.
0000     50 ;
0000     51 ;           V03-011 JEJ0021         J E Johnson             02-Apr-1984
0000     52 ;                   Correctly return RMS$_RTB error in the event of no
0000     53 ;                   stream terminator before eof which occurs on a block
0000     54 ;                   boundry.
0000     55 ;
0000     56 ;           V03-010 RAS0278         Ron Schaefer            20-Mar-1984
0000     57 ;                   Side-effect of improving seq $GET performance; eliminate
0000     58 ;                   call to RM$NEXT_BLK, by calling RM$NXTBLK1 directly.
0000     59 ;
0000     60 ;           V03-009 RAS0210         Ron Schaefer            7-Nov-1983
0000     61 ;                   Fix append problem where remainder of block was
0000     62 ;                   skipped when reading to EOF followed by $put.
0000     63 ;
0000     64 ;           V03-008 RAS0159         Ron Schaefer            1-Jul-1983
0000     65 ;                   Modify PPF input record checking to properly deal
0000     66 ;                   with EOF and other edge conditions.  Re-position
0000     67 ;                   only if there is less than 16 bytes left and this
0000     68 ;                   is not the last block.
0000     69 ;
0000     70 ;           V03-007 KBT0418         Keith B. Thompson       30-Nov-1982
0000     71 ;                   Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000     72 ;
0000     73 ;           V03-006 KBT0149         Keith B. Thompson       20-Aug-1982
0000     74 ;                   Reorganize psects
0000     75 ;
0000     76 ;           V03-005 KBT0084         Keith B. Thompson       13-Jul-1982
0000     77 ;                   Clean up psects
0000     78 ;
0000     79 ;           V03-004 JWH0001         Jeffrey W. Horn         29-Jun-1982
0000     80 ;                   Fix yet another BSBW to JSB broken branch.
0000     81 ;
0000     82 ;           V03-003 TMK0001         Todd M. Katz            12-Jan-1982
0000     83 ;                   Fix a broken branch by changing a BSBW (to RM$NEXT_BLK)
0000     84 ;                   to a JSB.
```

```
0000     85 ;
0000     86 ;       V03-002 RAS0083         Ron Schaefer            2-Apr-1982
0000     87 ;               Correct RAS0081 to do local EOF checking for blocks
0000     88 ;               between EBK and HBK, since RMS$NEXT_BLK does not check.
0000     89 ;
0000     90 ;       V03-001 RAS0081         Ron Schaefer            22-Mar-1982
0000     91 ;               Correct end-of-file checking to treat EOF as a std
0000     92 ;               terminator and latch the EOF flag.
0000     93 ;               Clean up some other flow paths.
0000     94 ;
0000     95 ;       V02-010 RAS0066         Ron Schaefer            9-Feb-1982
0000     96 ;               Drop ^Z and ESC as stream file terminators.
0000     97 ;
0000     98 ;       V02-009 KPL0002         Peter Lieberwirth       31-Dec-1981
0000     99 ;               Fix yet another broken branch.
0000    100 ;
0000    101 ;       V02-008 RAS0054         Ron Schaefer    15-Dec-1981
0000    102 ;               Change the stream terminator set to match RMS-11 and
0000    103 ;               make it a global symbol to patch.
0000    104 ;
0000    105 ;       V02-007 RAS0046         Ron Schaefer    12-Nov-1981
0000    106 ;               Fix another broken branch.
0000    107 ;
0000    108 ;       V02-006 KPL0001         P Lieberwirth   30-Oct-1981
0000    109 ;               Fix another broken branch.
0000    110 ;
0000    111 ;       V02-005 RAS0041         Ron Schaefer    27-Oct-1981
0000    112 ;               Fix broken branches caused by root directories.
0000    113 ;
0000    114 ;       V02-004 RAS0028         Ron Schaefer    20-Aug-1981
0000    115 ;               Change FAB$C_STM11 to FAB$C_STM.
0000    116 ;
0000    117 ;       V02-003 RAS0024         Ron Schaefer    17-Aug-1981
0000    118 ;               Correct CTRL/Z checking for stream-11 files
0000    119 ;               used for PPF indirect input.
0000    120 ;
0000    121 ;--
0000    122 ;
0000    123
```

```
                     0000   125          .SBTTL  DECLARATIONS
                     0000   126
                     0000   127 ;
                     0000   128 ; Include Files:
                     0000   129 ;
                     0000   130
                     0000   131 ;
                     0000   132 ; Macros:
                     0000   133 ;
                     0000   134          $FABDEF
                     0000   135          $RABDEF
                     0000   136          $RMSDEF
                     0000   137          $IFBDEF
                     0000   138          $IRBDEF
                     0000   139          $BDBDEF
                     0000   140          $DEVDEF
                     0000   141
                     0000   142 ;
                     0000   143 ; Equated Symbols:
                     0000   144 ;
00000000             0000   145          NUL=0                   ; null character
0000000A             0000   146          LF=10                   ; line feed
0000000B             0000   147          VT=11                   ; vertical tab
0000000C             0000   148          FF=12                   ; form feed
0000000D             0000   149          CR=13                   ; carriage return
00000010             0000   150          DLE=16                  ; data link error
00000011             0000   151          DC1=17                  ; device control 1
00000012             0000   152          DC2=18                  ; device control 2
00000013             0000   153          DC3=19                  ; device control 3
00000014             0000   154          DC4=20                  ; device control 4
0000001A             0000   155          CTRLZ=26                ; control z
0000001B             0000   156          ESC=27                  ; escape
                     0000   157
                     0000   158 ;
                     0000   159 ; Own Storage:
                     0000   160 ;
                     0000   161 ; A 32-bit bitmask for each stream format, indexed by character value,
                     0000   162 ; where each bit is a 1 if that ASCII character is a terminator:
                     0000   163 ;
                     0000   164 ;       STM:    LF, FF, VT
                     0000   165 ;
                     0000   166 ; Character values greater than 31 are eliminated earlier.
                     0000   167 ; CRLF is handled as a special case.
                     0000   168 ;
                     0000   169 RM$STM_MASK::
00001C00             0000   170          .LONG   <<1@LF> + <1@FF> + <1@VT>>
```

```
0004   172          .SBTTL  RMSGET_STM_FMT - Get a stream format record
0004   173
0004   174 ;++
0004   175 ; RMSGFT_STM_FMT
0004   176 ;
0004   177 ;       this module includes the routine to perform get/find
0004   178 ;       record processing for stream formats the sequential file organization.
0004   179 ;
0004   180 ; Calling sequence:
0004   181 ;
0004   182 ;       bsbw    rm$get_stm_fmt  ; for stream record formats
0004   183 ;
0004   184 ; Input Parameters:
0004   185 ;
0004   186 ;       r11     impure area addr
0004   187 ;       r10     ifab addr
0004   188 ;       r9      irab addr
0004   189 ;       r8      rab addr
0004   190 ;       r7      end of buffer addr
0004   191 ;       r1      addr of next datum
0004   192 ;
0004   193 ; Implicit Inputs:
0004   194 ;
0004   195 ;       the contents  f the various structures.
0004   196 ;
0004   197 ; Output Parameters:
0004   198 ;
0004   199 ;       r0      status code
0004   200 ;       r1      updated addr of next datum
0004   201 ;       r2-r6   destroyed
0004   202 ;       r7      end of buffer addr
0004   203 ;
0004   204 ; Implicit Outputs:
0004   205 ;
0004   206 ;       the internal structures are updated to reflect the
0004   207 ;       results of the get or find. the rab fields
0004   208 ;       are updated to correspond to the found or gotten
0004   209 ;       record (see functional spec for list).
0004   210 ;
0004   211 ; Completion Codes:
0004   212 ;
0004   213 ;       standard rms.
0004   214 ;
0004   215 ; Side Effects:
0004   216 ;
0004   217 ;       none
0004   218 ;
0004   219 ;--
```

```
                     0004   221 RM$GET_STM_FMT::
                     0004   222 ;
                     0004   223 ; quit if EOF flag is latched for STM format
                     0004   224 ;
03 69   21   E1      0004   225         BBC     #IRB$V_EOF, (R9), 10$   ; quit if EOF
       0156   31     0008   226         BRW     EOF                     ; return end-of-file
                     000B   227
   6A   2E   E1      000B   228 10$:     BBC     #IFB$V_PPF_INPUT, (R10),-
             7D      000E   229                 NO_PPF_IN               ; br if not 'sys$input'
   69   22   E1      000F   230         BBC     #IRB$V_PPF_IMAGE, (R9),-
             79      0012   231                 NO_PPF_IN               ; branch if not indirect
   6A   02   E0      0013   232         BBS     #DEV$V_TRM, IFB$L_PRIM_DEV(R10),-
             75      0016   233                 NO_PPF_IN               ; branch if terminal
                     0017   234
                     0017   235 ;++
                     0017   236 ;
                     0017   237 ;  Fix up the buffer pointer so that the $EOD scan will complete
                     0017   238 ;  succesfully for the worst case record.
                     0017   239 ;  The worst case is defined as a 257 byte record, so that strategy is to
                     0017   240 ;  use the min(257,actual size to terminator)
                     0017   241 ;
                     0017   242 ;  This may involve re-positioning the buffer so that the current record is
                     0017   243 ;  in the first block.   We assume the RM$CONNECT1 has forced MBC>0 so that
                     0017   244 ;  at least 2 blocks are present in each buffer.
                     0017   245 ;
                     0017   246 ;  Check if entire record is contained in the buffer.
                     0017   247 ;  Note that the current record is not entirely contained within the buffer
                     0017   248 ;  can only happen with disk files.
                     0017   249 ;
                     0017   250 ;  This check assumes that mbc is non-zero for disk files where records cross
                     0017   251 ;  block boundaries.  in this case, bdb$b_val_vbns gives the number of valid
                     0017   252 ;  blocks in the buffer, and bdb$b_rel_vbn specifies which block within the
                     0017   253 ;  buffer is currently being processed, in the range of 0 to irb$b_mbc.
                     0017   254 ;
                     0017   255 ;  Calculate in r5 the total # of bytes remaining in this and any subsequent
                     0017   256 ;  blocks in the buffer.
                     0017   257 ;
                     0017   258 ;       note: if sys$input is from a disk file and records are allowed to
                     0017   259 ;             cross block boundaries, irb$b_mbc must be greater than 0,
                     0017   260 ;             otherwise the processing of a record crossing a block boundary
                     0017   261 ;             will cause rms to loop.  it is assumed that rm$connect1 has
                     0017   262 ;             forced mbc > 0.
                     0017   263 ;
                     0017   264 ;  If the record is not contained in the current buffer,
                     0017   265 ;  cause the buffer to be read again, but with the current block as the
                     0017   266 ;  first block in the buffer.  Since the buffer is a multiple block buffer,
                     0017   267 ;  the new read will cause the next part of the current record to be
                     0017   268 ;  resident also.
                     0017   269 ;
                     0017   270 ;--
                     0017   271
57   51   D1         0017   272         CMPL    R1, R7                  ; is this the end of the buffer?
     08   12         001A   273         BNEQ    20$                     ; branch if not
     53   D4         001C   274         CLRL    R3                      ; flag read required
   FFDF'  30         001E   275         BSBW    RM$NXTBLK1              ; go to next block routine
 58 50    E9         0021   276         BLBC    R0, ERR10               ; get out on error
                     0024   277
```

```
                56   57   51   C3  0024    278 20$:    SUBL3   R1, R7, R6                  ; compute # bytes left this block
                  54   20 A9   D0  0028    279         MOVL    IRB$L_CURBDB(R9), R4        ; get current bdb address
                    48 A4      83  002C    280         SUBB3   BDB$B_REL_VBN(R4),-         ; # blocks to end of buffer
                  50   49 A4          002F  281                 BDB$B_VAL_VBNS(R4),R0     ; # blocks in buffer
                       50   50   9A  0032    282         MOVZBL  R0,R0                       ; correct data type
                            50   D7  0035    283         DECL    R0                          ; adjust for current block
                  50   48 AA   C4  0037    284         MULL2   IFB$L_DEVBUFSIZ(R10),R0     ; # bytes after current block
                       56   50   C0  003B    285         ADDL2   R0,R6                       ; total # valid bytes left
                       50   56   D0  003E    286         MOVL    R6, R0                      ; build a descriptor of the memory buff
                  54   50 AA   9A  0041    287         MOVZBL  IFB$B_RFMORG(R10), R4       ; get stream format type
                            51   DD  0045    288         PUSHL   R1                          ; save original buffer addr
                         01DB   30  0047    289         BSBW    RM$STM_TERM                 ; discover if there was a terminator
                       51   8E   D0  004A    290         MOVL    (SP)+, R1                   ; restore buff addr
                         05 50   E9  004D    291         BLBC    R0, 30$                     ; no terminator, so use buffer size
                       56   54   D0  0050    292         MOVL    R4, R6                      ; use size till terminator
                            21   11  0053    293         BRB     40$                         ; for the EOD check
                                     0055    294
                       56   00'  D1  0055    295 30$:    CMPL    S^#PIO$S_EODSTR, R6         ; are there enough chars within buff?
                            1C   1B  0058    296         BLEQU   40$                         ; branch if so
                         40 A9   D1  005A    297         CMPL    IRB$L_NRP_VBN(R9),-         ; is this the last block?
                         74 AA          005D  298                 IFB$L_EBK(R10)            ; use what there is if so
                            15   1E  005F    299         BGEQU   40$                         ; since there is nothing to re-map
                  54   20 A9   D0  0061    300         MOVL    IRB$L_CURBDB(R9), R4        ; get current bdb address
                        FF98'   30  0065    301         BSBW    RM$RELBLK1                  ; release buffer
                         11 50  E9  0068    302         BLBC    R0, ERR10                   ; get out on error
                         48 A9  7D  006B    303         MOVQ    IRB$L_RP_VBN(R9) -
                         40 A9          006E  304                 IRB$L_NRP_VBN(R9)         ; re-read this record
                         20 A9  D4  0070    305         CLRL    IRB$L_CURBDB(R9)            ; make sure this bdb not reused
                        FF8A'   31  0073    306         BRW     RM$GET_BLK_DEV              ; start over, re-mapping the buffer
                                     0076    307
                        FF87'   30  0076    308 40$:    BSBW    RM$INPUT_SCAN               ; compare string for
                                     0079    309                                         ; match with eod string
                                     0079    310                                         ; or '$deck' or '$eod'
                                     0079    311
                      10 50     E8  0079    312         BLBS    R0, NO_PPF_IN               ; continue on success
                                05   007C    313 ERR10:  RSB
                                     007D    314
                                     007D    315 ;++
                                     007D    316 ;
                                     007D    317 ; user buffer errors
                                     007D    318 ;
                                     007D    319 ;--
                                     007D    320
                                     007D    321 ERRUSZ: RMSERR  USZ                     ; user buffer size = 0
                                05   0082    322         RSB
                                     0083    323
                5E   0C      C0  0083    324 ERRUBF1:ADDL2   #12, SP                     ; clear the stack
                                     0086    325 ERRUBF: RMSERR  UBF                     ; no access to user buffer
                                05   008B    326         RSB
```

```
                     008C    328 ;++
                     008C    329 ;
                     008C    330 ; now deal with a regular data record
                     008C    331 ; currently stream format is always returned in movemode as locate mode
                     008C    332 ; is exceedingly complicated.
                     008C    333 ;
                     008C    334 ;--
                     008C    335 NO_PPF_IN:
                     008C    336
                     008C    337 ;
                     008C    338 ; get set up for copying the data
                     008C    339 ;
          013A   30  008C    340          BSBW    CHKEOF                  ; check for short buffer
        EA 50   E9   008F    341          BLBC    R0, ERRIO               ; quit if at EOF
50  57    51   C3   0092    342          SUBL3   R1, R7, R0              ; source buffer length
            0C   12   0096    343          BNEQ    30$                     ; if  zero, need a new blk
          0120   30   0098    344 20$:     BSBW    NXT_BLK                 ; get the next block
        DE 50   E9   009B    345          BLBC    R0, ERRIO               ; quit on error
50  57    51   C3   009F    346          SUBL3   R1, R7, R0              ; compute new buffer length
            F4   13   00A2    347          BEQL    20$                     ; go again to get the eof error
                     00A4    348 ;
                     00A4    349 ; now if we are doing STM, we have to ignore leading NULs (yecch)
                     00A4    350 ;
      50 AA   91   00A4    351 30$:     CMPB    IFB$B_RFMORG(R10),-     ; STM format?
            04     00A7    352                  #FAB$C_STM
            06   12   00A8    353          BNEQ    40$                     ; proceed if not
61  50    00   3B   00AA    354          SKPC    #NUL, R0, (R1)          ; skip over NULs
            E8   13   00AE    355          BEQL    20$                     ; skipped a whole block
                     00B0    356
                     00B0    357 ;
                     00B0    358 ; probe the user buffer and setup the ptrs.
                     00B0    359 ;
18 69    29   E0   00B0    360 40$:     BBS     #IRB$V_FIND, (R9), 50$  ; no need for buffers on $FIND
53   24 A8   D0   00B4    361          MOVL    RAB$L_UBF(R8), R3       ; get user buffer
52   20 A8   3C   00B8    362          MOVZWL  RAB$W_USZ(R8), R2       ; and size
            BF   13   00BC    363          BEQL    ERRUSZ                  ; can't be 0 length
28 A8    53   D0   00BE    364          MOVL    R3, RAB$L_RBF(R8)       ; set up result buffer
62 A9    52   B0   00C2    365          MOVW    R2, IRB$W_CSIZ(R9)      ; assume we'll fill 'er up
          0125   30   00C6    366          BSBW    PROBE_WRITE             ; probe the user buffer
        BA 50   E9   00C9    367          BLBC    R0,ERRUBF               ; quit on error
                     00CC    368
                     00CC    369 ;
                     00CC    370 ; at this point, we have a descripter of the source buffer in R0/R1
                     00CC    371 ; and a descriptor of the user's destination buffer in R2/R3.
                     00CC    372 ; Now copy stream record fragments to the user buffer stopping when
                     00CC    373 ; a terminator is encountered (return RMS$_NORMAL).
                     00CC    374 ; If no terminator is seen before the end of the user buffer,
                     00CC    375 ; then return RTB and skip to the terminator.
                     00CC    376 ;
            7E   D4   00CC    377 50$:     CLRL    -(SP)                   ; accumulated record length
            01   DD   00CE    378          PUSHL   #1                      ; assume success status
            7E   D4   00D0    379          CLRL    -(SP)                   ; terminator seen flag
                     00D2    380 MOVEDATA:
      5D 6E    E8   00D2    381          BLBS    (SP), DONE_1            ; quit on terminator seen
50  57    51   C3   00D5    382          SUBL3   R1, R7, R0              ; re-generate buffer length
            16   12   00D9    383          BNEQ    20$                     ; if there is any
          00DD   30   00DB    384 5$:      BSBW    NXT_BLK                 ; get the next block
```

```
                56 50    E9  00DE    385          BLBC    R0, ERRIO1                      ; quit on error
             06 69 29    E0  00E1    386          BBS     #IRB$V_FIND, (R9), 10$          ; no need for buffers on $FIND
                0106     30  00E5    387          BSBW    PROBE_WRITE                     ; probe the user buffer
                98 50    E9  00E8    388          BLBC    R0,ERRUBF1                      ; quit on error
          50    57 51    C3  00EB    389 10$:     SUBL3   R1, R7, R0                      ; get new source data length
                   EA    13  00EF    390          BEQL    5$                              ; go again to get the eof error
          54    50 AA    9A  00F1    391 20$:     MOVZBL  IFB$B_RFMORG(R10), R4           ; get stream format type
                012D     30  00F5    392          BSBW    RM$STM_TERM                     ; find a terminator
             56    51    D0  00F8    393          MOVL    R1, R6                          ; save the update buffer ptr
             08 AE 54    C0  00FB    394          ADDL2   R4, 8(SP)                       ; add in the chars seen
                6E 50    D0  00FF    395          MOVL    R0, (SP)                        ; and terminator seen flag
                69 29    E0  0102    396          BBS     #IRB$V_FIND, (R9),-             ; no data copy needed for $FIND
                   CC        0105    397                  MOVEDATA
                             0106    398
                             0106    399 ;
                             0106    400 ; now for a real bizarre check!  If we are doing STM and the last
                             0106    401 ; byte of the old buffer was CR, and the first byte of this buffer
                             0106    402 ; is LF, then we must delete the CR and finish with terminator encountered.
                             0106    403 ;
                   5E 50     E9  0106    404          BLBC    R0, COPY                        ; no terminator in this buffer
                   50 AA     91  0109    405          CMPB    IFB$B_RFMORG(R10),-             ; STM format?
                      04             010C    406                  #FAB$C_STM
                      58     12  010D    407          BNEQ    COPY                            ; not STM
                01    54     B1  010F    408          CMPW    R4, #1                          ; 1-byte record?
                      53     12  0112    409          BNEQ    COPY                            ; nope
                0A    65     91  0114    410          CMPB    (R5), #LF                       ; is it LF?
                      4E     12  0117    411          BNEQ    COPY                            ; nope
             28 A8    53     D1  0119    412          CMPL    R3, RAB$L_RBF(R8)               ; first time thru?
                      48     13  011D    413          BEQL    COPY                            ; no checking then
                             011F    414          IFNORD  #1, -1(R3), ERRUBF2,-           ; probe the user buffer
                             011F    415                  IRB$B_MODE(R9)
             0D    FF A3     91  0127    416          CMPB    -1(R3), #CR                     ; was the last byte CR?
                      3A     12  012B    417          BNEQ    COPY                            ; nope
                      52     B6  012D    418          INCW    R2                              ; remove the CR from the count
                   08 AE     D7  012F    419          DECL    8(SP)                           ; and chars seen
                      57     11  0132    420 DONE_1:  BRB     DONE_TERM                       ; and quit with terminator seen
                             0134    421
                   FF4C      31  0134    422 ERRUBF2: BRW     ERRUBF1                         ; access failure
                             0137    423
          827A 8F    50     B1  0137    424 ERRIO1:  CMPW    R0, #<RMS$_EOF&^XFFFF>          ; was error EOF?
                      1F     12  013C    425          BNEQ    10$                             ; quit if not
                             013E    426          SSB     #IRB$V_EOF, (R9)                ; set EOF flag
                45 04 AE     E9  0142    427          BLBC    4(SP),DONE_TERM                 ; If we have an error already, give
                             0146    428                                                  ; that one back to the user.
             17 69    29     E0  0146    429          BBS     #IRB$V_FIND, (R9), EOF          ; always give EOF on $FIND
22 A8        62 A9    52     A3  014A    430          SUBW3   R2, IRB$W_CSIZ(R9), -           ; any data?
                             0150    431                  RAB$W_RSZ(R8)
                      0F     13  0150    432          BEQL    EOF                             ; if so, return the record, no error
                51    56     D0  0152    433          MOVL    R6,R1                           ; restore buffer ptr
                   48 A9     7D  0155    434          MOVQ    IRB$L_RP_VBN(R9),-              ; set user RFA
                   10 A8         0158    435                  RAB$W_RFA(R8)
                             015A    436          RMSSUC                                  ; claim success
                5E   0C      C0  015D    437 10$:     ADDL2   #12, SP                         ; clear the stack
                             05  0160    438          RSB                                     ; and return
                             0161    439
                5E   0C      C0  0161    440 EOF:     ADDL2   #12, SP                         ; clear the stack
                   FE99'     31  0164    441          BRW     RM$GET_EOF                      ; return EOF
```

```
                        0167   442
                        0167   443 ;
                        0167   444 ; now copy the smaller of the source buffer (upto terminator) or dest buffer
                        0167   445 ;
      52   54   B1   0167   446 COPY:   CMPW    R4, R2                      ; source <> dest?
           0E   1F   016A   447         BLSSU   60$                         ; branch if dest if larger
           06   13   016C   448         BEQL    40$                         ; data segment = user buffer
                     016E   449         RMSERR  RTB, 4(SP)                  ; set return status
63    65   52   28   0174   450 40$:    MOVC3   R2, (R5), (R3)              ; copy the data
           0B   11   0178   451         BRB     80$
                     017A   452
                     017A   453 60$:
7E    52   54   A3   017A   454         SUBW3   R4, R2, -(SP)               ; subtract this data portion
63    65   54   28   017E   455         MOVC3   R4, (R5), (R3)              ; copy it
      52   8E   B0   0182   456         MOVW    (SP)+, R2                   ; updated dest size
      51   56   D0   0185   457 80$:    MOVL    R6, R1                      ; restore buffer ptr
         FF47   31   0188   458         BRW     MOVEDATA                    ; and get another segment
                     018B   459
                     018B   460 DONE_TERM:
   06 69   29   E0   018B   461         BBS     #IRB$V_FIND, (R9), 10$      ; no record length for $FIND
   62 A9   52   A3   018F   462         SUBW3   R2, IRB$W_CSIZ(R9),-        ; compute and store record length
      22 A8             0193   463                 RAB$W_RSZ(R8)
      5E   04   C0   0195   464 10$:    ADDL2   #4, SP                      ; discard terminator status
      50   8E   7D   0198   465         MOVQ    (SP)+, R0                   ; get return status and total recl
        04 50   E8   019B   466         BLBS    R0, 20$                     ; skip STV update on success
   0C A8   51   D0   019E   467         MOVL    R1, RAB$L_STV(R8)           ; tell user how big it was
   62 A9   51   B0   01A2   468 20$:    MOVW    R1, IRB$W_CSIZ(R9)          ; mark current record
      51   56   D0   01A6   469         MOVL    R6, R1                      ; update buffer ptr past terminator
   54   20 A9   D0   01A9   470         MOVL    IRB$L_CURBDB(R9), R4        ; get current bdb address
           06   13   01AD   471         BEQL    30$                         ; Branch if there is no current bdb
      4C A4   C3   01AF   472         SUBL3   BDB$L_CURBUFADR(R4),-
44 A9    51             01B2   473                 R1, IRB$L_NRP_OFF(R9)   ; make byte pointer relative
      48 A9   7D   01B5   474 30$:    MOVQ    IRB$L_RP_VBN(R9),-
      10 A8             01B8   475                 RAB$W_RFA(R8)           ; set rfa
           05   01BA   476         RSB                                     ; and return
                     01BB   477
                     01BB   478 ;
                     01BB   479 ; get the next block in the file and check for logical EOF
                     01BB   480 ;
   7E   52   7D   01BB   481 NXT_BLK:MOVQ    R2, -(SP)                   ; save dest buffer desc
        53   D4   01BE   482         CLRL    R3                          ; flag read required
     FE3D' 30   01C0   483         BSBW    RM$NXTBLK1                  ; go to next block routine
   52   8E   7D   01C3   484         MOVQ    (SP)+, R2                   ; restore buffer desc
     1A 50   E9   01C6   485         BLBC    R0, NXT                     ; quit on errors
                     01C9   486
                     01C9   487 ;
                     01C9   488 ; check for logical eof occuring in this buffer and correct
                     01C9   489 ; the buffer size appropriately.
                     01C9   490 ; R1 is the start addr of the record in the NRP_VBN block.
                     01C9   491 ; R7 is the end-of-buffer + 1 address.
                     01C9   492 ;
      40 A9   D1   01C9   493 CHKEOF: CMPL    IRB$L_NRP_VBN(R9),-         ; in the eof block?
      74 AA             01CC   494                 IFB$L_EBK(R10)
         13   1F   01CE   495         BLSSU   NXT                         ; nope, continue
         12   1A   01D0   496         BGTRU   EOF1                        ; already past eof
      44 A9   A3   01D2   497         SUBW3   IRB$W_NRP_OFF(R9),-         ; re-compute buffer end at eof
57    5C AA             01D5   498                 IFB$W_FFB(R10),R7
```

```
                0A   1B  01D8   49$          BLEQU   EOF1                    ; past FFB means EOF
57              57   3C  01DA   500          MOVZWL  R7,R7                   ; make into addr
57              51   C0  01DD   501          ADDL2   R1,R7                   ; set new end addr
                     01E0   502              RMSSUC                          ; successful
                     05  01E3   503 NXT:     RSB
                     01E4   504
                     01E4   505 EOF1:    SSB     #IRB$V_EOF,(R9)             ; set eof flag
                     01E8   506          RMSERR  EOF                         ; and say eof error
                     05  01ED   507          RSB
```

```
                                01EE    509          .SBTTL  PROBE_WRITE - probe user buffers
                                01EE    510 ;
                                01EE    511 ; probe the user buffer
                                01EE    512 ;
                                01EE    513 ; inputs
                                01EE    514 ;       r2 - user buffer size
                                01EE    515 ;       r3 - user buffer addr
                                01EE    516 ;
                                01EE    517 ; outputs
                                01EE    518 ;       r0 - status
                                01EE    519 ;
                                01EE    520 ; calling sequence
                                01EE    521 ;       BSBW    PROBE_WRITE
                                01EE    522 ;
                                01EE    523
                                01EE    524 PROBE_WRITE:
0200 8F    52    B1    01EE    525          CMPW    R2, #512                ; short enough buffer?
           09    1A    01F3    526          BGTRU   10$
                             01F5    527          IFNOWRT R2, (R3), PROBE_FAIL,-
                             01F5    528                  IRB$B_MODE(R9)
           20    11    01FC    529          BRB     30$
                             01FE    530
                             01FE    531 ;
                             01FE    532 ; probe for buffers greater than 512 bytes long
                             01FE    533 ;
                             01FE    534
     7E    52    7D    01FE    535 10$:     MOVQ    R2, -(SP)               ; save byte count and address
50   FE00  8F    32    0201    536          CVTWL   #-512, R0               ; address computation constant
                             0206    537 20$:     IFNOWRT R2, (R3), PROBE_FAIL,-
                             0206    538                  IRB$B_MODE(R9)          ; branch if not writeable
     53    50    C2    020D    539          SUBL2   R0, R3                  ; get address next page
   52    6240    3E    0210    540          MOVAW   (R2)[R0], R2            ; adjust count (- 2 pages)
           F0    14    0214    541          BGTR    20$                     ; branch if more to probe
     52    50    C2    0216    542          SUBL2   R0, R2                  ; 'add' back in 1 page
           EB    14    0219    543          BGTR    20$                     ; branch if more to probe
52   8E    7D    021B    544          MOVQ    (SP)+, R2               ; restore byte count and address
                             021E    545 30$:     RMSSUC
           05    0221    546          RSB
                             0222    547
                             0222    548 PROBE_FAIL:
           50    D4    0222    549          CLRL    R0                      ; failure
           05    0224    550          RSB
                             0225    551
```

```
0225    553          .SBTTL  RM$STM_TERM - Locate terminator for stream format
0225    554
0225    555 ;++
0225    556 ; RM$STM_TERM - Locate the terminator in a stream format buffer.
0225    557 ;
0225    558 ; This module performs the following functions:
0225    559 ;
0225    560 ;       Locate the terminator (if any) in a buffer.
0225    561 ;       Return a string descriptor of the stream record identified.
0225    562 ;       Compute the buffer address after the terminator.
0225    563 ;       Include the terminator or not per format type and terminator type.
0225    564 ;
0225    565 ; Calling Sequence:
0225    566 ;
0225    567 ;        BSBW   RM$STM_TERM
0225    568 ;
0225    569 ; Input Parameters:
0225    570 ;
0225    571 ;       R0      buffer length
0225    572 ;       R1      buffer address
0225    573 ;       R4      stream format code (uses FAB RFM field codes)
0225    574 ;
0225    575 ; Implicit Inputs:
0225    576 ;
0225    577 ;       none
0225    578 ;
0225    579 ; Output Parameters:
0225    580 ;
0225    581 ;       R0      status --  true IFf a terminator was found
0225    582 ;       R1      updated buffer address, past record or past all of buffer
0225    583 ;       R4      length of stream record identified
0225    584 ;       R5      start address of stream record
0225    585 ;
0225    586 ; Implicit Outputs:
0225    587 ;
0225    588 ;       none
0225    589 ;
0225    590 ; Completion Codes:
0225    591 ;
0225    592 ;       R0      status
0225    593 ;
0225    594 ; Side Effects:
0225    595 ;
0225    596 ;       none
0225    597 ;
0225    598 ;--
0225    599
```

```
                     0225   601 RM$STM_TERM::
                     0225   602
          7E  50  7D 0225   603         MOVQ    R0, -(SP)               ; Save the original buffer desc
                     0228   604
                     0228   605         ASSUME  <FAB$C_STM+1> EQ FAB$C_STMLF
                     0228   606         ASSUME  <FAB$C_STMLF+1> EQ FAB$C_STMCR
                     0228   607         CASE    TYPE=B, SRC=R4,-        ; Dispatch based on stream format type
                     0228   608                 LIMIT=#FAB$C_STM,-      ;  first stream format is STM
                     0228   609                 DISPLIST=< -
                     0228   610                 STM,-                   ; FAB$C_STM
                     0228   611                 STM_LF,-                ; FAB$C_STMLF
                     0228   612                 STM_CR>                 ; FAB$C_STMCR
                     0232   613
                     0232   614 ;
                     0232   615 ;  all other format types have no terminator
                     0232   616 ;
                     0232   617 NO_TERM:
          54  8E  7D 0232   618         MOVQ    (SP)+, R4               ; retrieve original buffer
     51   55  54  C1 0235   619         ADDL3   R4, R5, R1              ; point R1 past entire buffer
              50  D4 0239   620         CLRL    R0                      ; return failure
                  05 023B   621         RSB
                     023C   622
                     023C   623 STM_LF:
          54  0A  90 023C   624         MOVB    #LF, R4                 ; only 1 terminator char
              03  11 023F   625         BRB     SIMPLE_STM              ; makes for an easy scan
                     0241   626
                     0241   627 STM_CR:
          54  0D  90 0241   628         MOVB    #CR, R4                 ; likewise for this mode
                     0244   629 SIMPLE_STM:
     61   50  54  3A 0244   630         LOCC    R4, R0, (R1)            ; was the terminator in the buffer?
              E8  13 0248   631         BEQL    NO_TERM                 ; nope, so return failure
              51  D6 024A   632         INCL    R1                      ; skip over the DFT
                     024C   633 FND_TERM:
          6E  50  C2 024C   634         SUBL2   R0, (SP)                ; compute recl
          54  8E  7D 024F   635         MOVQ    (SP)+, R4               ; get original buffer desc
                     0252   636         RMSSUC                          ; flag success
                  05 0255   637         RSB
                     0256   638
                     0256   639 ;
                     0256   640 ; deal with the more complicated case of RMS-11 stream format
                     0256   641 ;
                     0256   642 STM:
          54  81  9A 0256   643         MOVZBL  (R1)+, R4               ; get the next char
          1F  54  91 0259   644         CMPB    R4, #31                 ; eliminate non-control chars
              06  1A 025C   645         BGTRU   10$                     ;
FD9D CF   54  E0 025E   646         BBS     R4, W^RM$STM_MASK,-
              06     0263   647                 TRM_11                  ; bit index by char to see if this
                     0264   648                                         ; is a terminator
              50  D7 0264   649 10$:    DECL    R0                      ; count this char
              CA  13 0266   650         BEQL    NO_TERM                 ; buffer is all gone without a term
              EC  11 0268   651         BRB     STM                     ; try next char
                     026A   652
                     026A   653 ;
                     026A   654 ; found one of the STM terminators, check for CRLF
                     026A   655 ; which are deleted, otherwise include the terminator in the data
                     026A   656 ; note that R1 points past the terminator but R0 hasn't counted it yet
                     026A   657 ;
```

```
                     026A   658 TRM_11:
          50   D7    026A   659          DECL    R0                      ; include the terminator
     54   0A   91    026C   660          CMPB    #LF, R4                 ; was the terminator LF?
          DB   12    026F   661          BNEQ    FND_TERM
                     0271   662
                     0271   663 ;
                     0271   664 ; found LF in STM, now drop both if the preceding char is CR.
                     0271   665 ;
                     0271   666 DFT_11:
55   50   01   C1    0271   667          ADDL3   #1, R0, R5              ; was LF first char?
     6E   55   D1    0275   668          CMPL    R5, (SP)
          D2   13    0278   669          BEQL    FND_TERM                ; if so, just ignore it
   FE A1   0D   91    027A   670          CMPB    #CR, -2(R1)             ; is preceding char CR?
          CC   12    027E   671          BNEQ    FND_TERM                ; if not, leave it in the data
     50   02   C0    0280   672          ADDL2   #2, R0                  ; discard CR and LF
          C7   11    0283   673          BRB     FND_TERM                ; and return success
                     0285   674
                     0285   675          .END
```

```
$$.PSECT_EP                     =  00000000
$$RMSTEST                       =  0000001A
$$RMS_PBUGCHK                   =  00000010
$$RMS_TBUGCHK                   =  00000008
$$RMS_UMODE                     =  00000004
BDB$B_REL_VBN                   =  00000048
BDB$B_VAL_VBNS                  =  00000049
BDB$L_CURBUFADR                 =  0000004C
CHKEOF                             000001C9 R      01
COPY                               00000167 R      01
CR                              =  0000000D
CTRLZ                           =  0000001A
DC1                             =  00000011
DC2                             =  00000012
DC3                             =  00000013
DC4                             =  00000014
DEV$V_TRM                       =  00000002
DFT_1T                             00000271 R      01
DLE                             =  00000010
DONE_1                             00000132 R      01
DONE_TERM                          0000018B R      01
EOF                                00000161 R      01
EOF1                               000001E4 R      01
ERRIO                              0000007C R      01
ERRIO1                             00000137 R      01
ERRUBF                             00000086 R      01
ERRUBF1                            00000083 R      01
ERRUBF2                            00000134 R      01
ERRUSZ                             0000007D R      01
ESC                             =  0000001B
FAB$C_STM                       =  00000004
FAB$C_STMCR                     =  00000006
FAB$C_STMLF                     =  00000005
FF                              =  0000000C
FND_TERM                           0000024C R      01
IFB$B_RFMORG                    =  00000050
IFB$L_DEVBUFSIZ                 =  00000048
IFB$L_EBK                       =  00000074
IFB$L_PRIM_DEV                  =  00000000
IFB$V_PPF_INPUT                 =  0000002E
IFB$W_FFB                       =  0000005C
IRB$B_MODE                      =  0000000A
IRB$L_CURBDB                    =  00000020
IRB$L_NRP_OFF                   =  00000044
IRB$L_NRP_VBN                   =  00000040
IRB$L_RP_VBN                    =  00000048
IRB$V_EOF                       =  00000021
IRB$V_FIND                      =  00000029
IRB$V_PPF_IMAGE                 =  00000022
IRB$W_CSIZ                      =  00000062
IRB$W_NRP_OFF                   =  00000044
LF                              =  0000000A
MOVEDATA                           000000D2 R      01
NO_PPF_IN                          0000008C R      01
NO_TERM                            00000232 R      01
NUL                             =  00000000
NXT                                000001E3 R      01
NXT_BLK                            000001BB R      01
PIO$S_EODSTR                       ********   X    01
PROBE_FAIL                         00000222 R      01
PROBE_WRITE                        000001EE R      01
RAB$L_RBF                       =  00000028
RAB$L_STV                       =  0000000C
RAB$L_UBF                       =  00000024
RAB$W_RFA                       =  00000010
RAB$W_RSZ                       =  00000022
RAB$W_USZ                       =  00000020
RM$GET_BLK_DEV                     ********   X    01
RM$GET_EOF                         ********   X    01
RM$GET_STM_FMT                     00000004 RG     01
RM$INPUT_SCAN                      ********   X    01
RM$NXTBLK1                         ********   X    01
RM$RELBLK1                         ********   X    01
RM$STM_MASK                        00000000 RG     01
RM$STM_TERM                        00000225 RG     01
RMS$_EOF                        =  000182A8
RMS$_RTB                        =  000181A8
RMS$_UBF                        =  000186EC
RMS$_USZ                        =  000186F4
SIMPLE_STM                         00000244 R      01
STM                                00000256 R      01
STM_CR                             00000241 R      01
STM_LF                             0000023C R      01
TRM_11                             0000026A R      01
VT                              =  0000000B
```

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	00000285 (645.)	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	30	00:00:00.07	00:00:00.91
Command processing	125	00:00:00.74	00:00:03.56
Pass 1	313	00:00:10.26	00:00:26.63
Symbol table sort	0	00:00:01.30	00:00:01.69
Pass 2	129	00:00:02.37	00:00:08.69
Symbol table output	11	00:00:00.10	00:00:00.22
Psect synopsis output	3	00:00:00.03	00:00:00.13
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	613	00:00:14.89	00:00:41.84

The working set limit was 1650 pages.
59088 bytes (116 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1057 non-local and 26 local symbols.
675 source lines were read in Pass 1, producing 14 object records in Pass 2.
24 pages of virtual memory were used to define 23 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	11
_255DUA28:[SYS.OBJ]LIB.MLB;1	3
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	19

1153 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM1STMFMT/OBJ=OBJ$:RM1STMFMT MSRC$:RM1STMFMT/UPDATE=(ENH$:RM1STMFMT)+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