```
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        CCCCCCCC    000000    NN      NN  NN      NN
RRRRRRRR    MM      MM      11        CCCCCCCC    000000    NN      NN  NN      NN
RR      RR  MMMM  MMMM    1111      CC            00    00  NN      NN  NN      NN
RR      RR  MMMM  MMMM    1111      CC            00    00  NN      NN  NN      NN
RR      RR  MM  MM  MM      11      CC            00    00  NNNN    NN  NNNN    NN
RR      RR  MM  MM  MM      11      CC            00    00  NNNN    NN  NNNN    NN
RRRRRRRR    MM      MM      11      CC            00    00  NN  NN  NN  NN  NN  NN
RRRRRRRR    MM      MM      11      CC            00    00  NN  NN  NN  NN  NN  NN
RR  RR      MM      MM      11      CC            00    00  NN    NNNN  NN    NNNN
RR  RR      MM      MM      11      CC            00    00  NN    NNNN  NN    NNNN
RR    RR    MM      MM      11      CC            00    00  NN      NN  NN      NN      ....
RR    RR    MM      MM      11      CC            00    00  NN      NN  NN      NN      ....
RR      RR  MM      MM    111111      CCCCCCCC    000000    NN      NN  NN      NN      ....
RR      RR  MM      MM    111111      CCCCCCCC    000000    NN      NN  NN      NN      ....

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

Table of contents

```
0000     1          $BEGIN  RM1CONN,000,RM$RMS1,<SEQUENTIAL AND COMMON CONNECT>
0000     2
0000     3 ;
0000     4 ;****************************************************************************
0000     5 ;*                                                                          *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     8 ;*  ALL RIGHTS RESERVED.                                                    *
0000     9 ;*                                                                          *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    15 ;*  TRANSFERRED.                                                            *
0000    16 ;*                                                                          *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    19 ;*  CORPORATION.                                                            *
0000    20 ;*                                                                          *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    23 ;*                                                                          *
0000    24 ;*                                                                          *
0000    25 ;****************************************************************************
0000    26 ;
```

```
0000     28 ;++
0000     29 ; Facility: rms32
0000     30 ;
0000     31 ; Abstract:
0000     32 ;                routine to perform sequential-specific
0000     33 ;                connect processing.
0000     34 ;
0000     35 ; Environment:
0000     36 ;                star processor running starlet exec.
0000     37 ;
0000     38 ; Author: l f laverdure,         creation date: 5-JAN-1977
0000     39 ;
0000     40 ; Modified By:
0000     41 ;
0000     42 ;       V03-023 SHZ0026         Stephen H. Zalewski,    04-May-1984
0000     43 ;               If we are creating a global buffer section, specify that
0000     44 ;               we do an expand region to make sure it comes out of P0 space.
0000     45 ;
0000     46 ;       V03-022 JEJ0025         J E Johnson             10-Apr-1984
0000     47 ;               Ensure that GBH and GBD lengths are quadword aligned.
0000     48 ;
0000     49 ;       V03-021 SHZ0011         Stephen H. Zalewski,    24-Feb-1984
0000     50 ;               Do not initialize TRC blocks when connecting with global
0000     51 ;               buffers.  This was accidently left on from SHZ0010.
0000     52 ;
0000     53 ;       V03-020 SHZ0010         Stephen H. Zalewski     06-Dec-1983
0000     54 ;               Fix the tracing code to work with multi-threaded RMS.  This
0000     55 ;               is accomplished by using the interlock queue instructions.
0000     56 ;
0000     57 ;       V03-019 SHZ0009         Stephen H. Zalewski     19-Sep-1983
0000     58 ;               Replace call to RMS$INIT_SFSB with RMS$INIT_SFSB_IRB.  This is
0000     59 ;               to allow us to successfully stall using the irab.
0000     60 ;
0000     61 ;       V03-018 SHZ0008         Stephen H. Zalewski     10-Aug-1983
0000     62 ;               Bugcheck if we try to create a global buffers section
0000     63 ;               with global buffer count of zero.
0000     64 ;
0000     65 ;       V03-017 SHZ0007         Stephen H. Zalewski     28-Jul-1983
0000     66 ;               Implement cluster global buffers.
0000     67 ;
0000     68 ;       V03-016 SHZ0006         Stephen H. Zalewski     22-Jun-1983
0000     69 ;               Disable global buffers.
0000     70 ;
0000     71 ;       V03-015 SHZ0005         Stephen H. Zalewski     11-Apr-1983
0000     72 ;               Fix bug that caused a process to incorrectly map a global
0000     73 ;               buffer section.
0000     74 ;
0000     75 ;       V03-014 KPL0001         Peter Lieberwirth       23-Mar-1983
0000     76 ;               Fix v03-013 by reversing sense of branch.
0000     77 ;
0000     78 ;       V03-13  SHZ0004         Stephen H. Zalewski     21-Feb-1983
0000     79 ;               If XQP is being used, ignore any request for global buffers.
0000     80 ;               This is only a temporary restriction.
0000     81 ;
0000     82 ;       V03-012 LJA0055         Laurie J. Anderson      12-Jan-1983
0000     83 ;               Fill in MBF field in IRB with the value which is used
0000     84 ;
```

```
0000     85 ;       V03-011 KBT0413         Keith B. Thompson       30-Nov-1982
0000     86 ;               Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000     87 ;
0000     88 ;       V03-010 SHZ0003         Stephen H. Zalewski,    14-Oct-1982  16:29
0000     89 ;               Prevent a $connect from occuring if there is no device buffer
0000     90 ;               or the real-time device bit is set in the device characteristics
0000     91 ;               field (this is also a patch in 3.2).
0000     92 ;
0000     93 ;       V03-009 SHZ0002         Stephen H. Zalewski,    10-Sep-1982  16:43
0000     94 ;               Remove anything and everthing to do with FRBs, SIFBs and SFDs
0000     95 ;               because they no longer exist.
0000     96 ;
0000     97 ;       V03-008 KBT0341         Keith B. Thompson       16-Sep-1982
0000     98 ;               Don't allocate multiple gbsbs when multistreaming
0000     99 ;
0000    100 ;       V03-007 SHZ0001         Stephen H. Zalewski,    1-Sep-1982  15:29
0000    101 ;               Modify so that global buffer section locking is now done
0000    102 ;               via the lock manger and the GBSB.
0000    103 ;
0000    104 ;       V03-006 KBT0299         Keith B. Thompson       24-Aug-1982
0000    105 ;               Reorganize psects
0000    106 ;
0000    107 ;       V03-005 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    108 ;               Added $PCBDEF.
0000    109 ;
0000    110 ;--
0000    111
```

```
                        0000   113          .SBTTL  DECLARATIONS
                        0000   114
                        0000   115 ;
                        0000   116 ; Include Files:
                        0000   117 ;
                        0000   118
                        0000   119 ;
                        0000   120 ; Macros:
                        0000   121 ;
                        0000   122
                        0000   123          $BDBDEF
                        0000   124          $BLBDEF
                        0000   125          $CCBDEF
                        0000   126          $DEVDEF
                        0000   127          $FABDEF
                        0000   128          $FIBDEF
                        0000   129          $FWADEF
                        0000   130          $GBDDEF
                        0000   131          $GBHDFF
                        0000   132          $GBSBDEF
                        0000   133          $IMPDEF
                        0000   134          $IRBDEF
                        0000   135          $IFBDEF
                        0000   136          $PCBDEF
                        0000   137          $PRVDEF
                        0000   138          $PSLDEF
                        0000   139          $RMSDEF
                        0000   140          $RABDEF
                        0000   141          $SECDEF
                        0000   142          $SSDEF
                        0000   143          $SFSBDEF
                        0000   144          $TRCDEF
                        0000   145          $WCBDEF
                        0000   146
                        0000   147 ;
                        0000   148 ; Equated Symbols:
                        0000   149 ;
                        0000   150
              00000020  0000   151          ROP=RAB$L_ROP*8                 ; bit offset to rop
                        0000   152
                        0000   153 ;
                        0000   154 ; Own Storage:
                        0000   155 ;
                        0000   156
                        0000   157 FAOCNTRL:
4C 58 21 24 53 4D 52 5F 00'     0000   158          .ASCIC  /_RMS$!XL/              ; Control string to FAO for GS name.
                        08      0000
                        0009   159
```

```
0009   161          .SBTTL  RM$CONNECT1 - SEQUENTIAL-SPECIFIC CONNECT ROUTINE
0009   162
0009   163 ;++
0009   164 ; RM$CONNECT
0009   165 ;
0009   166 ;       RM$CONNECT
0009   167 ;
0009   168 ;  this routine performs the following functions required
0009   169 ;  for connecting to sequential files:
0009   170 ;
0009   171 ;       1. perform various validity checks
0009   172 ;       2. if opened for block i/o allocate a lock bdb
0009   173 ;       3. allocate required bdb's and buffers and save count
0009   174 ;
0009   175 ;
0009   176 ; Calling sequence:
0009   177 ;
0009   178 ;       entered via case branch from rm$connect
0009   179 ;
0009   180 ; Input Parameters:
0009   181 ;
0009   182 ;       ap      argument list addr
0009   183 ;       r11     impure area addr
0009   184 ;       r10     ifab addr
0009   185 ;       r9      irab addr
0009   186 ;       r8      rab addr
0009   187 ;
0009   188 ; Implicit Inputs:
0009   189 ;
0009   190 ;       the contents of the rab and irab blocks
0009   191 ;
0009   192 ; Output Parameters:
0009   193 ;
0009   194 ;       r0      status
0009   195 ;
0009   196 ; Implicit Outputs:
0009   197 ;
0009   198 ;       sets various fields in the irab and ifab.
0009   199 ;
0009   200 ; Completion Codes:
0009   201 ;
0009   202 ;       the standard rms status code is set into r0 and
0009   203 ;       return is made to the user (not caller).
0009   204 ;
0009   205 ;       if any errors, all irab-related internal structures
0009   206 ;       are deallocated.
0009   207 ;
0009   208 ; Side Effects:
0009   209 ;
0009   210 ;       none
0009   211 ;
0009   212 ; note:
0009   213 ;
0009   214 ;       only 1 connected irab is allowed on a sequential file.
0009   215 ;       this routine assumes that this is the first irab in
0009   216 ;       ifab's irab chain.
0009   217 ;
```

```
                     0009   218 ;--
                     0009   219
                     0009   220 RMS$CONNECT1::
                     0009   221
                     0009   222 ;
                     0009   223 ;  if open or create was done with bro specified (mixed block & record i/o),
                     0009   224 ;  check the bio rop bit and if set switch to block i/o only
                     0009   225 ;
                     0009   226
          06   E1    0009   227          BBC     #FAB$V_BRO,-
    0C 22 AA         000B   228                  IFB$B_FAC(R10),8$        ; branch if bro not set
          20   8A    000E   229          BICB2   #FAB$M_BIO,-
       22 AA         0010   230                  IFB$B_FAC(R10)           ; indicate not limited to block i/o
 04 68    2B   E1    0012   231          BBC     #RAB$V_BIO+ROP,(R8),8$   ; branch if bio clear in rop
          20   88    0016   232          BISB2   #FAB$M_BIO,-
       22 AA         0018   233                  IFB$B_FAC(R10)           ; switch to block i/o only
                     001A   234 8$:
 0C 6A    3E   E1    001A   235          BBC     #IFB$V_DAP, (R10), 20$   ; branch if network access
00000000'EF   16    001E   236          JSB     NT$CONNECT               ; do network connect
       03 50   E8    0024   237          BLBS    R0, 20$                  ; continue on success
        000C   31    0027   238          BRW     CLN1                     ; cleanup on error
                     002A   239 20$:
 22 AA   05   E1    002A   240          BBC     #IFB$V_BIO,IFB$B_FAC(R10),-
             1B         002E   241                  CHKMBC                   ; branch if not block i/o
             6C   11   002F   242          BRB     ALLOC                    ; go to alloc if block i/o
```

```
                     0031   244
                     0031   245 ;
                     0031   246 ;  error processing
                     0031   247 ;
                     0031   248 ;  record format undefined and doing record i/o processing
                     0031   249 ;
                     0031   250
                     0031   251 ERRRFM:
                     0031   252 RM$CONN_ERRRFM::
                     0031   253         RMSERR  RFM
FFC7'  30            0036   254 CLN1:   BSBW    RM$CCLN1                ; deallocate irab
FFC4'  31            0039   255         BRW     RM$EX_NOSTR             ; and exit
                     003C   256
                     003C   257 ;
                     003C   258 ;  disk buffer size not 512, device is realtime device, or device has a zero
                     003C   259 ;  length device buffer.
                     003C   260 ;
                     003C   261
                     003C   262 ERRDEV: RMSERR  DEV
  F3   11            0041   263         BRB     CLN1
                     0043   264
                     0043   265 ;
                     0043   266 ;  mbc negative. (reserved for later use)
                     0043   267 ;
                     0043   268
                     0043   269 ERRMBC:
                     0043   270         RMSERR  MBC
  EC   11            0048   271         BRB     CLN1
                     004A   272
```

```
                     004A   274          .ENABL  LSB
                     004A   275
                     004A   276 ;
                     004A   277 ;  allocate bdbs and i/o buffers of size = blocksize (* mbc, if disk)
                     004A   278 ;
                     004A   279 ;!!!!!
                     004A   280 ;  \note:  might be nice to change rm$aldbuf to do split-page
                     004A   281 ;   (but not cross-page) allocations for unit record devices.\
                     004A   282 ;!!!!!
                     004A   283 ;
                     004A   284
                     004A   285 CHKMBC:
     55   48 AA   D0 004A   286          MOVL    IFB$L_DEVBUFSIZ(R10),R5 ; length of buffer
          EC   13   004E   287          BEQL    ERRDEV                  ; Cannot connect if no device buffer
          1D   E0   0050   288          BBS     #DEV$V_RTM,-            ;  or device is realtime device.
        E8 6A       0052   289                  IFB$L_PRIM_DEV(R10),ERRDEV
        54   01   D0 0054   290          MOVL    #1,R4                   ; mt offset for default mbf
          0E   E1   0057   291          BBC     #DEV$V_FOD,-
        4B 6A       0059   292                  IFB$L_PRIM_DEV(R10),UNIT; branch if not disk or mt
          18   E0   005B   293          BBS     #DEV$V_FOR,-
        47 6A       005D   294                  IFB$L_PRIM_DEV(R10),UNIT; branch if mounted foreign
        6A   1C   E1 005F   295          BBC     #DEV$V_RND,IFB$L_PRIM_DEV(R10),-
          3A        0062   296                  ALLOC                   ; branch if not disk
                    0063   297
                    0063   298 ;
                    0063   299 ;  check that assumptions regarding disk buffer size are accurate
                    0063   300 ;
                    0063   301 ;  otherwise some of sequential get code won't work
                    0063   302 ;
                    0063   303
0200 8F   55   B1   0063   304          CMPW    R5,#512
          D2   12   0068   305          BNEQ    ERRDEV          ; it's all over if not that magic number!
                    006A   306
                    006A   307 ;
                    006A   308 ;  this is a connect for a disk file.
                    006A   309 ;
                    006A   310 ;  process the mbc (multi-block count) field of the rab to determine
                    006A   311 ;  the size of the buffers to be allocated.
                    006A   312 ;
                    006A   313
          54   D4   006A   314          CLRL    R4                      ; disk offset for default mbf
     50   37 A8   98 006C   315          CVTBL   RAB$B_MBC(R8),R0        ; get mbc
        6A   2E   E1 0070   316          BBC     #IFB$V_PPF_INPUT,(R10),-
          08        0073   317                  120$                    ; branch if not sys$input
          02   50   D1 0074   318          CMPL    R0,#2                   ; mbc at least 2?
          1A   1E   0077   319          BGEQU   130$                    ; branch if yes
          50   02   D0 0079   320          MOVL    #2,R0                   ; set mbc=2 for sys$input
          15   12   007C   321 120$:      BNEQ    130$                    ; branch if speced
50   00000000'9F   98 007E   322          CVTBL   @#PIO$GB_DFMBC,R0       ; else get process default
          0C   12   0085   323          BNEQ    130$                    ; branch if speced
50   00000000'9F   98 0087   324          CVTBL   @#SYS$GB_DFMBC,R0       ; else get system default
          03   12   008E   325          BNEQ    130$                    ; branch if speced
          50   01   D0 0090   326          MOVL    #1,R0                   ; else use a single block
          AE   19   0093   327 130$:      BLSS    ERRMBC                  ; error if mbc negative
 55 A9    50   01   83 0095   328          SUBB3   #1,R0,IRB$B_MBC(R9)     ; store adjusted mbc value
          55   50   A4 009A   329          MULW2   R0,R5                   ; get total size of buffer
             0071   30 009D   330 ALLOC:   BSBW    RM$BDBALLOC             ; go allocate the buffers
```

```
1B 50   E8  00A0    331 150$:   BLBS    R0,SETNXT               ; continue on success
 FF5A'  31  00A3    332         BRW     RM$EX_NOSTR             ; exit on error.  error in
            00A6    333                                         ; rm$bdballoc returns everything
```

```
                     00A6    335 ;
                     00A6    336 ;  buffer allocation for unit record and foreign mounted devices
                     00A6    337 ;
                     00A6    338 ;  allocate a single buffer only
                     00A6    339 ;
                     00A6    340
        56   01   D0  00A6    341 UNIT:   MOVL    #1,R6                       ; get just one buffer/bdb
             02   E1  00A9    342         BBC     #DEV$V_TRM,-
           0C 6A         00AB    343                 IFB$L_PRIM_DEV(R10),160$; go allocate if not term
   55   0200 8F   B1  00AD    344         CMPW    #512,R5                     ; buffer size at least 512
             05   1B  00B2    345         BLEQU   160$                        ; yes, use it
   55   0200 8F   B0  00B4    346         MOVW    #512,R5                     ; use 512 bytes as minimum
          00A2   30  00B9    347 160$:   BSBW    RM$BDBALLOC_ALT             ; go allocate the buffer
             E2   11  00BC    348         BRB     150$                        ; do error check
                     00BE    349         .DSABL  LSB
                     00BE    350
                     00BE    351 ;
                     00BE    352 ;  perform remaining stream setup
                     00BE    353 ;
                     00BE    354
     3C A9   54   D0  00BE    355 SETNXT: MOVL    R4,IRB$L_NXTBDB(R9)         ; set next bdb for seqxfr
                     00C2    356
                     00C2    357 ;
                     00C2    358 ;  position file for stream at beginning of file
                     00C2    359 ;  unless eof bit set in ifab or rop
                     00C2    360 ;
                     00C2    361
             18   E0  00C2    362         BBS     #DEV$V_FOR,-
           39 6A         00C4    363                 IFB$L_PRIM_DEV(R10),65$ ; leave positioned at blk 0;
                     00C6    364                                             ; if non-file structured
           40 A9   D6  00C6    365         INCL    IRB$L_NRP_VBN(R9)           ; assume at beginning of file
        04 6A    21   E0  00C9    366         BBS     #IFB$V_EOF,(R10),20$    ; branch if position to eof flag set
        0A 68    28   E1  00CD    367         BBC     #RAB$V_EOF+ROP,(R8),30$ ; branch if eof not set in rop either
                     00D1    368
                     00D1    369 ;
                     00D1    370 ;  copy the eof position to the next record pointer context
                     00D1    371 ;
                     00D1    372
   40 A9   74 AA   D0  00D1    373 20$:    MOVL    IFB$L_EBK(R10),IRB$L_NRP_VBN(R9); these better be zero
   44 A9   5C AA   B0  00D6    374         MOVW    IFB$W_FFB(R10),IRB$W_NRP_OFF(R9); for unit record devices
                     00DB    375
                     00DB    376 ;
                     00DB    377 ;  check for positioned at or past eof unless unit record
                     00DB    378 ;
                     00DB    379
                     00DB    380 30$:
                     00DB    381         ASSUME  DEV$V_REC EQ 0
           14 6A   E8  00DB    382         BLBS    IFB$L_PRIM_DEV(R10),50$ ; branch if unit record
           40 A9   D1  00DE    383         CMPL    IRB$L_NRP_VBN(R9),-
           74 AA       00E1    384                 IFB$L_EBK(R10)          ; nrp past eof?
              0D   1F  00E3    385         BLSSU   50$                     ; branch if not
              07   1A  00E5    386         BGTRU   40$                     ; branch if definite yes
                     00E7    387
                     00E7    388 ;
                     00E7    389 ;  nrp vbn = eof vbn
                     00E7    390 ;  must check byte in block to determine if at eof
                     00E7    391 ;
```

```
                     00E7   392
5C AA   44 A9   B1   00E7   393          CMPW    IRB$W_NRP_OFF(R9),IFB$W_FFB(R10)
           04   1F   00EC   394          BLSSU   50$                     ; branch if nrp < eof
                     00EE   395 40$:      SSB     #IRB$V_EOF,(R9)         ; set the eof flag
   01      54 A9   91   00F2   396 50$:      CMPB    IRB$B_BCNT(R9),#1       ; just 1 buffer allocated?
           04   1B   00F6   397          BLEQU   60$                     ; branch if yes
                     00F8   398          SSB     #IRB$V_RAHWBH,(R9)      ; enable read ahead & write behind
         FF01'  31   00FC   399 60$:      BRW     RM$EXSUC                ; exit with success
                     00FF   400
                     00FF   401 ;
                     00FF   402 ;   maintains eof context on foreign devices
                     00FF   403 ;
                     00FF   404
   F9 6A   05   E1   00FF   405 65$:      BBC     #DEV$V_SQD,IFB$L_PRIM_DEV(R10),60$; branch if not magtape
   F5 6A   21   E1   0103   406          BBC     #IFB$V_EOF,(R10),60$    ; if not at eof, no problem
                     0107   407          SSB     #IRB$V_EOF,(R9)         ; set irab eof bit
           EF   11   010B   408          BRB     60$                     ; return to mainline
```

```
                     010D   410
                     010D   411 ;++
                     010D   412 ;
                     010D   413 ;  subroutine to allocate bdbs and buffers.  code to lock buffers in working
                     010D   414 ;  set remains no-op'd at time of release 2.  it is felt at this time that
                     010D   415 ;  locking buffers in the working set when the multi-buffer count is positive
                     010D   416 ;  will probably cause problems with existing programs because in fact, rms
                     010D   417 ;  has not been locking them at all.  if this is implemented in a future
                     010D   418 ;  release, the cleanest technique would seem to be the addition of yet
                     010D   419 ;  another (sigh...) rop bit in the rab as input to the $connect operation
                     010D   420 ;  to specifically request rms to lock the buffers.  the current behavior
                     010D   421 ;  of using absolute value of the mbf field or default counts will continue.
                     010D   422 ;
                     010D   423 ;  if this is for magtape with truncate access, only 1 buffer is allocated
                     010D   424 ;
                     010D   425 ;  entry at rm$bdballoc_alt is used when buffer size is already specified in
                     010D   426 ;  r6.  see additional comments there.
                     010D   427 ;
                     010D   428 ;
                     010D   429 ;  inputs:
                     010D   430 ;
                     010D   431 ;       r11             impure area address
                     010D   432 ;       r10             ifab address
                     010D   433 ;       r9              irab address
                     010D   434 ;       r8              rab address
                     010D   435 ;       r5              size of buffers to allocate, in bytes.
                     010D   436 ;       r4              index for defaults, values as follows:
                     010D   437 ;
                     010D   438 ;                               0 - sequential disk file default
                     010D   439 ;                               1 - magtape default
                     010D   440 ;                               2 - unit record default
                     010D   441 ;                               3 - relative file default
                     010D   442 ;                               4 - indexed file default
                     010D   443 ;                               5 - hashed file default
                     010D   444 ;
                     010D   445 ;       rab$b_mbf       explicit # of buffers
                     010D   446 ;
                     010D   447 ;  outputs:
                     010D   448 ;
                     010D   449 ;       r0              status code
                     010D   450 ;       r1-r6           destroyed
                     010D   451 ;       r4              address of last bdb allocated
                     010D   452 ;       irb$b_bcnt      # of buffers allocated - updated only if r9 nonzero.
                     010D   453 ;
                     010D   454 ;       allocation failure when called from connect (r9 nonzero) will
                     010D   455 ;       return all allocated buffers, bdb's, bcb's, and the irab.
                     010D   456 ;
                     010D   457 ;--
                     010D   458
                  55 7C  010D   459 BLKALL: CLRQ    R5                          ; this will get lock bdb only
                  4D 11  010F   460         BRB     RM$BDBALLOC_ALT             ; extended branch
                         0111   461 RM$BDBALLOC::
              22 AA 05  E0 0111   462         BBS     #IFB$V_BIO,IFB$B_FAC(R10),-
                  F7     0115   463                 BLKALL                      ; block i/o then just do bdb
              56 36 A8  98 0116   464         CVTBL   RAB$B_MBF(R8),R6            ; get number of buffers
                  1F 12  011A   465         BNEQ    10$                         ; branch if specified
56 00000000'9F44  98 011C   466         CVTBL   @#PIO$GB_DFMBFSDK[R4],R6; else, pick up process default
```

```
                 15   12  0124    467          BNEQ    10$                             ; branch if specified
     56  00000000'9F44   98  0126    468          CVTBL   @#SYS$GB_DFMBFSDK[R4],R6; else, pick up system default
                 0B   12  012E    469          BNEQ    10$                             ; branch if specified
                 56   01  D0  0130    470          MOVL    #1,R6                           ; else use 1 buffer
                             0133    471
                             0133    472 ;
                             0133    473 ;  if read ahead or write behind spec'd, then need two buffers
                             0133    474 ;
                             0133    475
                             0133    476          ASSUME  <<RAB$M_RAH!RAB$M_WBH>&^XFFFF00FF> EQ 0
              05 A8   06   B3  0133    477          BITW    #<RAB$M_RAH!RAB$M_WBH>@-8,RAB$B_ROP1(R8)
                             0137    478                                                  ; either rab or wbh spec'd?
                             0137    479
                 02   13  0137    480          BEQL    10$                             ; eql don't want rah/wbh
                 56   D6  0139    481          INCL    R6                              ; need min two buffs
                 56   D5  013B    482 10$:     TSTL    R6
                 03   14  013D    483          BGTR    20$                             ; if pos, then ok
                 56   56  CE  013F    484          MNEGL   R6,R6                           ; otherwise make it positive
                             0142    485 20$:
              5C A9   56   90  0142    486          MOVB    R6,IRB$B_MBF(R9)                ; Save MBF value used
              07 6A   36   E1  0146    487          BBC     #IFB$V_TEF,(R10),40$            ; branch if no truncate access
                 05   E1  014A    488          BBC     #DEV$V_SQD,-
                    03 6A        014C    489                  IFB$L_PRIM_DEV(R10),40$ ; branch if not magtape
                 56   01  D0  014E    490          MOVL    #1,R6                           ; allocate 1 buffer
                             0151    491 40$:
                             0151    492
                             0151    493 ;
                             0151    494 ;  since we can't get good indexed defaults any other way
                             0151    495 ;  alter r6 here. indexed files require at least 2 bdb's and buffer's
                             0151    496 ;  so if absolute value of r6 is 1, then need to change it
                             0151    497 ;
                             0151    498
     54   04   D1  0151    499          CMPL    #4,R4                           ; see if indexed
                 08   12  0154    500          BNEQ    80$                             ; if not branch
     02   56   D1  0156    501          CMPL    R6,#2                           ; at least 2 buffers spec'd?
                 03   1E  0159    502          BGEQU   80$                             ; ok if greater than or equal
     56   02   D0  015B    503          MOVL    #2,R6                           ; use 2 otherwise
                             015E    504 80$:
                             015E    505
                             015E    506 ;
                             015E    507 ;  alternate entry point for number of buffers already specified in r6.
                             015E    508 ;
                             015E    509 ;  if r9 is zero, then irb$b_bcnt is not filled in.  this entry point is
                             015E    510 ;  for unit record and foreign devices to allocate a single buffer not
                             015E    511 ;  using the mbf or defaults.  extend and display will use this to allocate
                             015E    512 ;  buffers when no streams are connected (relative or isam only).
                             015E    513 ;
                             015E    514 ;  inputs:
                             015E    515 ;
                             015E    516 ;       r6                      number of buffers to
                             015E    517 ;                               allocate.  0 causes only one buffer to
                             015E    518 ;                               be allocate and bypasses potential allocation
                             015E    519 ;                               of lock bdb for relative and isam orgs.
                             015E    520 ;
                             015E    521 ;       ifb$v_wrtacc            if set, then allocate a lock bdb also for
                             015E    522 ;                               relative and isam files if low word r6 non zero
                             015E    523 ;
```

```
                     015E   524 ;       ifb$l_sfsb_ptr          if non-zero, file is shared and a bcb is
                     015E   525 ;                                allocated for each bdb.
                     015E   526 ;
                     015E   527 ;  outputs:
                     015E   528 ;
                     015E   529 ;       bdb's are linked into the end of the ifab bdb list.
                     015E   530 ;
                     015E   531
                     015E   532 RM$BDBALLOC_ALT::
           7E   D4   015E   533          CLRL    -(SP)                   ; init buffer counter
04 22 AA   05   E1   0160   534          BBC     #IFB$V_BIO, IFB$B_FAC(R10), AGAIN ; Br if not block i/o.
                     0165   535          SSB     #IFB$V_NORECLK, (R10)   ; Make sure noreclk is set for bio.
                     0169   536 AGAIN:
           6E   D6   0169   537          INCL    (SP)                    ; count the buffer
         FE92'  30   016B   538          BSBW    RM$ALDBUF               ; allocate the buffer
        23 50   E9   016E   539          BLBC    R0,DECR_BCNT            ; get out on error
06 6A    33   E0   0171   540          BBS     #IFB$V_NORECLK, (R10), 10$ ; branch if no record locking.
         FE88'  30   0175   541          BSBW    RM$ALBLB                ; Allocate a BLB.
        2E 50   E9   0178   542          BLBC    R0,GIVEBACK             ; branch if error on getting bcb
           55   D5   017B   543 10$:     TSTL    R5                      ; was buffer allocated?
           04   13   017D   544          BEQL    20$                     ; EQL then not, so don't count it.
     0084 CA   B6   017F   545          INCW    IFB$W_AVLCL(R10)        ; note buffer allocated.
        E3 56   F5   0183   546 20$:     SOBGTR  R6,AGAIN                ; decrement counter, go again
                     0186   547                                          ; if still positive
           0C   19   0186   548          BLSS    DECR_BCNT               ; this was last pass to alloc
                     0188   549                                          ; just lock bdb so decr bcnt
                     0188   550                                          ; so it only counts buffers
                     0188   551
                     0188   552 ;
                     0188   553 ; At this point the required number of buffers and bdbs, and blbs (if shared)
                     0188   554 ; have been allocated.  Allocate a lock blb if record locking is being done.
                     0188   555 ;
                     0188   556
0A 6A    33   E0   0188   557          BBS     #IFB$V_NORECLK,(R10),EXIT ; done if no locking.
         FE71'  30   018C   558          BSBW    RM$ALBLB                ; Allocate a lock BLB.
        24 50   E8   018F   559          BLBS    R0, CHKGBL              ; Check out global buffers.
           02   11   0192   560          BRB     EXIT                    ; Exit on error from alblb.
                     0194   561 DECR_BCNT:
                     0194   562
                     0194   563 ;
                     0194   564 ; come here on error and
                     0194   565 ; last pass to get count right
                     0194   566 ;
                     0194   567
           6E   D7   0194   568          DECL    (SP)                    ;
                     0196   569 EXIT:
           02   BA   0196   570          POPR    #^M<R1>                 ; get buffer count off stack
           59   D5   0198   571          TSTL    R9                      ; is there an irab?
           07   13   019A   572          BEQL    10$                     ; no, then don't update bcnt
                     019C   573                                          ; and exit (caller checks error)
54 A9    51   90   019C   574          MOVB    R1,IRB$B_BCNT(R9)       ; store count of buffers
        01 50   E9   01A0   575          BLBC    R0,20$                  ; error on allocation
                     01A3   576                                          ; clean up buffers allocated
                     01A3   577                                          ; and get rid of irab
                     01A3   578                                          ; r9 nonzero means this was
                     01A3   579                                          ; called on a connect
           05   01A3   580 10$:     RSB                                 ; and exit routine
```

```
                50   DD  01A4   581 20$:    PUSHL   R0                          ; save status
            FE57'    31  01A6   582          BRW     RM$COMCLNUP                 ; and branch to cleanup
                     01A9   583
                     01A9   584 ;
                     01A9   585 ; we couldn't get a blb for some reason (e.g., not enough space left).
                     01A9   586 ; therefore, we must return the bdb we just got.
                     01A9   587 ;
                     01A9   588
                     01A9   589 GIVEBACK:
                50   DD  01A9   590          PUSHL   R0                          ; save status code
        54  44 AA   D0  01AB   591          MOVL    IFB$L_BDB_BLNK(R10),R4      ; get back link because
                     01AF   592                                                  ; aldbuf calls albdb which
                     01AF   593                                                  ; links them at end of list
            FE4E'    30  01AF   594          BSBW    RM$RETBDB                   ; deallocates bdb ar4
                01   BA  01B2   595          POPR    #^M<R0>                     ; restore status code
                DE   11  01B4   596          BRB     DECR_BCNT                   ; fix count and exit
                     01B6   597
```

```
                     01B6   599
                     01B6   600 ;
                     01B6   601 ; Code to allocate global buffers if desired, and initialize if necessary.
                     01B6   602 ;
                     01B6   603
                     01B6   604 CHKGBL:
          78 AA      D5 01B6   605          TSTL    IFB$L_SFSB_PTR(R10)      ; If file is not being shared,
             DB      13 01B9   606          BEQL    EXIT                     ;  then don't bother with global buffers
        0088 CA      D5 01BB   607          TSTL    IFB$L_GBH_PTR(R10)       ; If we already have global buffers
             03      13 01BF   608          BEQL    1$                       ;  then multi-streaming.
           00A5      31 01C1   609          BRW     MAP_IT                   ; Go map the section.
                     01C4   610 1$:        ASSUME  IMP$V_IIOS EQ 0
          CF 6B      E9 01C4   611          BLBC    (R11), EXIT              ; Just use local if this is ppf.
       CB 6B  05     E0 01C7   612          BBS     #IMP$V_NOPOBUFS,(R11),EXIT ; Use local if P0 off limits.
             59      D5 01CB   613          TSTL    R9                       ; Is irab present?
             C7      13 01CD   614          BEQL    EXIT                     ; No, then just use local.
    56   3C AB      D0 01CF   615          MOVL    RAB$L_FAB(R8), R6        ; Get address of FAB.
    56   48 A6      32 01D3   616          CVTWL   FAB$W_GBC(R6), R6        ; Get gbl buffer count.
             BD      13 01D7   617          BEQL    EXIT                     ; None desired, so exit.
           FE24'     30 01D9   618          BSBW    RM$INIT_GBSB             ; Allocate GBSB and get EX lock.
          B7 50      E9 01DC   619          BLBC    R0,EXIT                  ; Exit if lock failed.
    54   7C AA      D0 01DF   620          MOVL    IFB$L_GBSB_PTR(R10),R4   ; Get address of GBSB from IFB.
    52   38 A4      D0 01E3   621          MOVL    GBSB$L_GS_SIZE(R4),R2    ; Are global buffers already in use?
             54      13 01E7   622          BEQL    CHK_GBC                  ; No, branch to validate GBC.
           007D      31 01E9   623          BRW     MAP_IT                   ; Yes, go use them.
                     01EC   624
54   009C CA      D0 01EC   625 ERL0:     MOVL    IFB$L_BLBBLNK(R10), R4   ; Get addr of BLB.
           FE0C'     30 01F1   626          BSBW    RM$RETBLB                ; Return it.
54   009C CA      D0 01F4   627 ERL1:     MOVL    IFB$L_BLBBLNK(R10), R4   ; Get BLB addr.
           FE04'     30 01F9   628          BSBW    RM$RETBLB                ; Give back.
  54   44 AA      D0 01FC   629 ERL4:     MOVL    IFB$L_BDB_BLNK(R10), R4  ; Get address of GBPB just alloc'd.
           FDFD'     30 0200   630          BSBW    RM$RETGBPB               ; Give it back.
  54   44 AA      D0 0203   631 ERL3:     MOVL    IFB$L_BDB_BLNK(R10), R4  ; Get address of GBPB just alloc'd.
           FDF6'     30 0207   632          BSBW    RM$RETGBPB               ; Give it back.
54   009C CA      D0 020A   633 ERL2:     MOVL    IFB$L_BLBBLNK(R10), R4   ; Get addr of a BLB (lock BLB).
           FDEE'     30 020F   634          BSBW    RM$RETBLB                ; Give it back.
           FDEB'     30 0212   635          BSBW    RM$RLS_GBSB              ; Dequeue the lock we had on the GBSB
             01      BA 0215   636          POPR    #^M<R0>                  ; Restore error code.
           FF7C      31 0217   637          BRW     EXIT                     ; Go finish up.
                     021A   638
                     021A   639 ALBLBERR:
       5E    08      C0 021A   640          ADDL2   #8, SP                   ; Clean off stack.
             50      DD 021D   641          PUSHL   R0                       ; Save error code.
             D3      11 021F   642          BRB     ERL1                     ; Br and finish up.
                     0221   643 ALBLBERR1:
       5E    08      C0 0221   644          ADDL2   #8, SP                   ; Clean off stack.
             50      DD 0224   645          PUSHL   R0                       ; Save error code.
             D4      11 0226   646          BRB     ERL4                     ; Br and finish up.
                     0228   647 ALGBPERR:
       5E    08      C0 0228   648          ADDL2   #8, SP                   ; Clean off stack.
             50      DD 022B   649          PUSHL   R0                       ; Save error code.
             DB      11 022D   650          BRB     ERL2                     ; Br to give back lock BLB.
                     022F   651 ALGBPERR1:
       5E    08      C0 022F   652          ADDL2   #8, SP                   ; Clean off stack.
             50      DD 0232   653          PUSHL   R0                       ; Save error code.
             CD      11 0234   654          BRB     ERL3                     ; Br to give back one gbpb.
                     0236   655 BAD_GBC:
```

```
                                0236   656          RMSERR  GBC,-(SP)                      ; Note error.
                          CD    11  023B   657          BRB     ERL2                    ; Give back lock BLB.
                                023D   658
                                023D   659          ASSUME  <<GBH$C_BLN/8>*8> EQ GBH$C_BLN  ; Check for quadword alignment
                                023D   660          ASSUME  <<GBD$C_BLN/8>*8> EQ GBD$C_BLN  ; in GBD and GBH sections
                                023D   661
                                023D   662 CHK_GBC:
                     52    56   D0   023D   663          MOVL    R6, R2                  ; Save number of buffers desired.
                           F4   19   0240   664          BLSS    BAD_GBC                 ; Only positive values allowed.
                51   52    55   C5   0242   665          MULL3   R5, R2, R1              ; Total buffer bytes into R1.
                     52    28   C4   0246   666          MULL2   #GBD$C_BLN, R2          ; R2 is now descriptor bytes.
                     52    51   C0   0249   667          ADDL2   R1, R2                  ; Sum of desc and buffers.
          52   00000058    8F   C0   024C   668          ADDL2   #GBH$C_BLN, R2          ; Plus size of header area.
          52   000001FF    8F   C0   0253   669          ADDL    #511,R2                 ; Round up to even pages.
          52   000001FF    8F   CA   025A   670          BICL    #511,R2
                           06   11   0261   671          BRB     MAP_IT                  ; Noop to branch to ADDTRC for tracing.
                00000004'  EF   17   0263   672          JMP     ADDTRC                  ;
                                0269   673 MAP_IT:
                           14   BB   0269   674          PUSHR   #^M<R2,R4>              ; Save registers needed after algbpb.
                     FD92'      30   026B   675          BSBW    RM$ALGBPB               ; Get Global Buffer Pointer Block.
                     B7   50   E9   026E   676          BLBC    R0, ALGBPERR            ; Branch on error.
                     FD8C'      30   0271   677          BSBW    RM$ALGBPB               ; Get Global Buffer Pointer Block.
                     B8   50   E9   0274   678          BLBC    R0, ALGBPERR1           ; Branch on error.
                     FD86'      30   0277   679          BSBW    RM$ALBLB                ; Get a BLB.
                     A4   50   E9   027A   680          BLBC    R0, ALBLBERR1           ; Branch on error.
                     FD80'      30   027D   681          BSBW    RM$ALBLB                ; Get a BLB.
                     97   50   E9   0280   682          BLBC    R0, ALBLBERR            ; Exit on error.
                           14   BA   0283   683          POPR    #^M<R2,R4>              ; Restore registers.
                     0088  CA   D5   0285   684          TSTL    IFB$L_GBH_PTR(R10)      ; Already have gbl buffs?
                           07   13   0289   685          BEQL    1$                      ; No, then go on to map it.
                                028B   686          SSB     #IRB$V_GBLBUFF, (R9)    ; Note irab has extra gbpb, blb.
                        FF04    31   028F   687          BRW     EXIT                    ; Branch to exit.
                                0292   688
                                0292   689 ;
                                0292   690 ; R2 = Number of bytes to allocate (rounded up to full pages)
                                0292   691 ;
                                0292   692
                           7E   7C   0292   693 1$:     CLRQ    -(SP)                   ; Zero INADR forces P0 space to be allocated
                           7E   7C   0294   694          CLRQ    -(SP)                   ; Reserve space for RETADR.
                                0296   695
                                0296   696 ;
                                0296   697 ; The section name will be the ascii text '_RMS$' followed by the
                                0296   698 ; FCB address in hexadecimal.
                                0296   699 ;
                                0296   700
                     5E    10   C2   0296   701          SUBL2   #16, SP                 ; Make room for gsd name.
                           6E   DF   0299   702          PUSHAL  (SP)                    ; Addr part of descriptor.
                           0D   DD   029B   703          PUSHL   #13                     ; Length of GSD name.
                     020C  8F   BB   029D   704          PUSHR   #^M<R2,R3,R9>           ; Save these around GETCCB call.
                     59    5A   D0   02A1   705          MOVL    R10, R9                 ; Need ifab in r9.
                        FD59'      30   02A4   706          BSBW    RM$GETCCB               ; Get CCB addr into R1.
                     020C  8F   BA   02A7   707          POPR    #^M<R2,R3,R9>           ; Restore registers.
                51     04  A1   D0   02AB   708          MOVL    CCB$L_WIND(R1), R1      ; Get ptr to window.
                51     18  A1   D0   02AF   709          MOVL    WCB$L_FCB(R1), R1       ; Get FCB addr into R1.
                     FD4A  CF   DF   02B3   710          PUSHAL  FAOCNTRL+1              ;
             7E      FD45  CF   9A   02B7   711          MOVZBL  FAOCNTRL, -(SP)         ; Build descriptor for control string.
                     50    5E   D0   02BC   712          MOVL    SP, R0                  ; Need to pass addr of desc.
```

```
                                02BF    713          $FAO_S  CTRSTR=(R0),-           ; Address of control string descriptor
                                02BF    714                  OUTBUF=8(R0),-        ; Addr of output buffer descriptor.
                                02BF    715                  P1=R1                 ; FCB addr to show up in output string.
                          6E    7C    02CF    716          CLRQ    (SP)              ; Clear priv mask.
                                02D1    717          SSB     #PRV$V_SYSGBL, (SP)   ; Need sysgbl privilege.
                    51    5E    D0    02D5    718          MOVL    SP, R1            ; Save this stack address.
                                02D8    719          $SETPRV_S ENBFLG=#1,-         ; Turn on sysgbl for crmpsc.
                                02D8    720                  PRVADR=(R1),-
                                02D8    721                  PRVPRV=(R1)           ; Get previous state.
                                02E7    722
              51    08 AE     DE    02E7    723          MOVAL   8(SP), R1         ; Address of gsd name desc.
           50   52    17     9C    02EB    724          ROTL    #23, R2, R0       ; Get page count into r0.
                                02EF    725          $CRMPSC_S INADR = 32(R1),-    ; Point to array on stack.
                                02EF    726                  RETADR = 24(R1),-     ; Point to array on stack.
                                02EF    727                  GSDNAM = (R1),-       ;  "
                                02EF    728                  PAGCNT = R0,-         ; Number of pages in section.
                                02EF    729                  ACMODE = #PSL$C_EXEC,-; Access mode is EXEC.
                                02EF    730          FLAGS = #SEC$M_GBL!SEC$M_SYSGBL!SEC$M_WRT!SEC$M_DZRO!SEC$M_PAGFIL!SEC$M_EXPR
                                0313    731
                 1D 6E    19    E0    0313    732          BBS     #PRV$V_SYSGBL, (SP), 5$ ; If already had sysgbl, skip turnoff.
                          6E    7C    0317    733          CLRQ    (SP)              ; Init priv mask.
                                0319    734          SSB     #PRV$V_SYSGBL,(SP)    ; Turn off sysgbl.
                    51    5E    D0    031D    735          MOVL    SP, R1            ; Address of priv mask.
                          50    DD    0320    736          PUSHL   R0                ; Save status from crmpsc.
                                0322    737          $SETPRV_S PRVADR=(R1)         ; Turn off sysgbl.
                          50 8ED0     0331    738          POPL    R0                ; Restore crmpsc status.
                                0334    739 5$:
                    5E    20    C0    0334    740          ADDL2   #32, SP           ; Clean priv mask+name desc +name.
                      06  50    E8    0337    741          BLBS    R0, 20$           ; Continue if Ok.
                        00ED    31    033A    742          BRW     SEC_ERR           ; Branch to error code.
                        00E3    31    033D    743 10$:     BRW     SEC_ERR1          ; Branch to error code.
                                0340    744 20$:
           51   04 AE     6E    C3    0340    745          SUBL3   (SP), 4(SP), R1     ; Get size allocated - 1.
                          51    D6    0345    746          INCL    R1                ; Size allocated.
                    52    51    D1    0347    747          CMPL    R1, R2            ; Get everything?
                          F1    12    034A    748          BNEQ    10$               ; Br if not.
                    53    6E    D0    034C    749          MOVL    (SP),R3           ; Move starting address of section into R3.
              50    0619  8F    B1    034F    750          CMPW    #SS$_CREATED, R0  ; Was the section just created?
                          0B    13    0354    751          BEQL    30$               ; Then it needs to be initialized.
           08 A3    1611  8F    B1    0356    752          CMPW    #<GBH$C_BID+<GBH$C_BLN/4@8>>, GBH$B_BID(R3) ; Seem legit?
                          DF    12    035C    753          BNEQ    10$               ; NEQ there's an error.
                       0087     31    035E    754          BRW     STORE_PTR         ; Else use it.
                                0361    755
                                0361    756 ;
                                0361    757 ; Initialize newly created section.
                                0361    758 ; R3 = start address of section
                                0361    759 ; R2 = size of section in bytes
                                0361    760 ; R6 = number of buffers in section.
                                0361    761 ;
                                0361    762
                 0C A3    01    CE    0361    763 30$:     MNEGL   #1, GBH$L_HI_VBN(R3)  ; Store hi vbn for scan end check.
                 10 A3    52    D0    0365    764          MOVL    R2, GBH$L_GS_SIZE(R3) ; Store size of section in section.
                 38 A4    52    D0    0369    765          MOVL    R2, GBSB$L_GS_SIZE(R4) ; Store size of section in GBSB.
                 34 A4    56    B0    036D    766          MOVW    R6, GBSB$W_GBC(R4)    ; Store number of buffers in section.
           08 A3    1611  8F    B0    0371    767          MOVW    #<GBH$C_BID+<GBH$C_BLN/4@8>>, GBH$B_BID(R3) ; Store id, bln.
              50    0058  8F    3C    0377    768          MOVZWL  #GBH$C_BLN, R0        ; Offset to first GBD from GBH.
                 04 A3    50    D0    037C    769          MOVL    R0, GBH$L_GBD_BLNK(R3) ; Back link to GBD's.
```

```
                28 A3    50    D0  0380    770          MOVL    R0, GBH$L_GBD_START(R3) ; Save offset to first GBD.
                30 A3    50    D0  0384    771          MOVL    R0, GBH$L_GBD_NEXT(R3) ; First GBD is first victim.
                34 A3    08    D0  0388    772          MOVL    #8, GBH$L_SCAN_NUM(R3)  ; Assume scan size of 8.
                   56    08    D1  038C    773          CMPL    #8, R6                  ; Have at least 8 buffers?
                         04    1B  038F    774          BLEQU   45$                     ; LEQU just use 8.
                34 A3    56    D0  0391    775          MOVL    R6, GBH$L_SCAN_NUM(R3)  ; Else only use # in section.
                   56    D7    D7  0395    776 45$:     DECL    R6                      ; Num - 1.
                   56    28    C4  0397    777          MULL2   #GBD$C_BLN, R6          ; Offset to last GBD from first.
        52   56   0000027F  8F    C1  039A    778          ADDL3   #GBD$C_BLN+GBH$C_BLN+511, R6, R2 ; End of GBD's + page-1 byte.
             52     01FF    8F    AA  03A2    779          BICW2   #511, R2                ; Round off to even page.
                   50    53    C0  03A7    780          ADDL2   R3, R0                  ; Start address of GBD's.
                   56    50    C0  03AA    781          ADDL2   R0, R6                  ; Addr of last GBD.
                                    03AD    782          ASSUME  GBH$L_GBD_FLNK EQ 0
             63    56    53    C3  03AD    783          SUBL3   R3, R6, (R3)            ; Forw link points to last GBD.
                2C A3    63    D0  03B1    784          MOVL    (R3), GBH$L_GBD_END(R3) ; Offset to last GBD.
                                    03B5    785 50$:
                                    03B5    786          ASSUME  GBD$L_FLINK EQ 0
                   60    28    CE  03B5    787          MNEGL   #GBD$C_BLN, (R0)        ; Offset to next GBD.
                04 A0    28    D0  03B8    788          MOVL    #GBD$C_BLN, GBD$L_BLINK(R0) ; Offset to last GBD.
                                    03BC    789          ASSUME  GBD$B_BLN EQ <GBD$B_BID + 1>
          08 A0    0A13  8F    B0  03BC    790          MOVW    #<GBD$C_BID+<GBD$C_BLN/4@8>>, GBD$B_BID(R0) ; Id and bln.
             0C A0    01    CE  03C2    791          MNEGL   #1, GBD$L_VBN(R0)       ; Init VBN to -1.
             1A A0    55    B0  03C6    792          MOVW    R5, GBD$W_SIZE(R0)      ; Store buffer size.
             1C A0    52    D0  03CA    793          MOVL    R2, GBD$L_REL_ADDR(R0)  ; Store offset to buffer.
                52    55    C0  03CE    794          ADDL2   R5, R2                  ; Point to next buffer.
       FFDE 50    28    56    F1  03D1    795          ACBL    R6, #GBD$C_BLN, R0, 50$ ; Loop until past last GBD.
                                    03D7    796          ASSUME  GBH$L_GBD_FLNK EQ 0
             04 A6    63    CE  03D7    797          MNEGL   (R3), GBD$L_BLINK(R6)   ; Last GBD's back link is
                                    03DB    798                                      ; opposite of header's forw link.
                   04 A3    CE  03DB    799          MNEGL   GBH$L_GBD_BLNK(R3),-    ; First GBD's forw link is
                   58 A3        03DE    800                  GBH$C_BLN+GBD$L_FLINK(R3) ; opposite of header's back link.
                                03E0    801
                                03E0    802 ;
                                03E0    803 ; If tracing is to be enabled, noop the following branch.
                                03E0    804 ;
                                03E0    805
                      06    11  03E0    806          BRB     STORE_PTR               ; To make it easy to patch in tracing.
        00000028'EF   17  03E2    807          JMP     INIT_TRC                ; To init tracing blocks.
                                03E8    808 STORE_PTR:
             5E    10    C0  03E8    809          ADDL2   #16, SP                 ; 'Pop' INADR, RETADR arrays off stack.
                1C A3    D6  03EB    810          INCL    GBH$L_USECNT(R3)        ; Increment accessor count for section.
       0088 CA    53    D0  03EE    811          MOVL    R3, IFB$L_GBH_PTR(R10)  ; Point to the section.
                   59    D5  03F3    812          TSTL    R9                      ; Irab present?
                   04    13  03F5    813          BEQL    20$                     ; EQL then no irab.
                               03F7    814          SSB     #IRB$V_GBLBUFF, (R9)    ; Note this irab has extra gbpb, blb.
          1C A3    01    D1  03FB    815 20$:     CMPL    #1,GBH$L_USECNT(R3)     ; Are we first accessor?
                   16    12  03FF    816          BNEQ    30$                     ; No, branch to release lock.
             0080 CA    D0  0401    817          MOVL    IFB$L_PAR_LOCK_ID(R10),-; Save file lock id in global section.
                14 A3        0405    818                  GBH$L_LOCK_ID(R3)
                  FBF6'  30  0407    819          BSBW    RM$LOWER_SYSLOCK        ; Turn file lock into system lock.
                  FBF3'  30  040A    820          BSBW    RM$LOWER_GBS_LOCK       ; Lower lock on global buffer section.
          54    78 AA    D0  040D    821          MOVL    IFB$L_SFSB_PTR(R10),R4  ; Put address of SFSB in R4 for INIT_SFSB.
                  FBEC'  30  0411    822          BSBW    RM$INIT_SFSB_IRB        ; Get a file lock for process using IRB to s
                  FD7F   31  0414    823          BRW     EXIT                    ; Continue.
                14 A3    D0  0417    824 30$:     MOVL    GBH$L_LOCK_ID(R3),-     ; Move the parent lock id for bucket
             0080 CA        041A    825                  IFB$L_PAR_LOCK_ID(R10)  ;  locks into ifab from global buffer header
                  FBE0'  30  041D    826          BSBW    RM$LOWER_GBS_LOCK       ; Do lock mode conversion.
```

```
                FD73   31  0420   827          BRW     EXIT                    ; Continue.
                           0423   828
                           0423   829 ;
                           0423   830 ; An error has been detected.  Disassociate from section, return structures
                           0423   831 ; already allocated.
                           0423   832 ;
                           0423   833
                           0423   834 SEC_ERR1:
     50  000184D4 8F    D0  0423   835          MOVL    #RMS$_DME, R0           ; Give DME error if not all mapped.
                           042A   836 SEC_ERR:
                           042A   837           .ASSUME  FAB$L_STV EQ RAB$L_STV
          0C A8    50    D0  042A   838          MOVL    R0, RAB$L_STV(R8)       ; Save error code.
             50    8E    7D  042E   839          MOVQ    (SP)+, R0               ; Get RETADR off stack into r0 and r1.
             5E    08    C0  0431   840          ADDL2   #8, SP                  ; Pop INADR off stack.
                           0434   841          RMSERR  CRMP, -(SP)             ; Note error.
                FBC4'  30  0439   842          BSBW    RM$UNMAP_GBL_ALT        ; Delete the whole VA.
                FDAD   31  043C   843          BRW     ERL0                    ; Branch to finish up.
                           043F   844
```

```
                                043F    846          $NEWPSECT       RM$TRACE
                    00000190    0000    847 NUMTRC: .LONG   400                     ; Number of trace blocks to allocate.
                                0004    848
                                0004    849 ;
                                0004    850 ; Add in extra bytes for trace blocks after size of section is determined.
                                0004    851 ;
                                0004    852
                                0004    853 ADDTRC:
          50    F9 AF    D0     0004    854          MOVL    NUMTRC, R0             ; Get number of trace blocks desired.
                   02    12     0008    855          BNEQ    10$                    ; Branch if non-zero.
                   50    D6     000A    856          INCL    R0                     ; Get at least one.
    50  00000040 8F    C4     000C    857 10$:     MULL2   #TRC$C_BLN, R0         ; Get size of trace blocks.
    50  000001FF 8F    C0     0013    858          ADDL2   #511, R0               ; Add in almost a page.
       50    01FF 8F    AA     001A    859          BICW2   #511, R0               ; Round to even page's worth.
             52   50    C0     001F    860          ADDL2   R0, R2                 ; Add in to size being requested.
          00000269'EF    17     0022    861          JMP     MAP_IT                 ; And return to mainline.
                                0028    862
                                0028    863 ;
                                0028    864 ; Initialize the trace blocks and pointer from the global buffer header.
                                0028    865 ;
                                0028    866 ; R3 - pointer to GBH
                                0028    867 ; R5 - buffer size
                                0028    868 ;
                                0028    869
                                0028    870 INIT_TRC:
                                0028    871          ASSUME  GBH$L_GBD_FLNK EQ 0
       50   53    63    C1     0028    872          ADDL3   (R3), R3, R0           ; Get address of last GBD in list.
    50  55    1C A0    C1     002C    873          ADDL3   GBD$L_REL_ADDR(R0), R5, R0 ; R0 now first byte after last buff.
    20 A3    50   20    C3     0031    874          SUBL3   #GBH$L_TRC_FLNK, R0, GBH$L_TRC_FLNK(R3) ; Offset to 1st trc blk
             50   53    C0     0036    875          ADDL2   R3, R0                 ; R0 now addr of first trace block.
    51   53    10 A3    C1     0039    876          ADDL3   GBH$L_GS_SIZE(R3), R3, R1 ; Get addr of end of gbl sec.
    51  00000040 8F    C2     003E    877          SUBL2   #TRC$C_BLN, R1         ; Limit for last trace block.
                                0045    878 10$:
                                0045    879          ASSUME  <TRC$C_BLN & 7> EQ 0   ; These will line up on quad boundary.
                                0045    880          ASSUME  TRC$L_FLNK EQ 0
    80  00000040 8F    D0     0045    881          MOVL    #TRC$C_BLN, (R0)+      ; Fwd offset to next block.
                                004C    882          ASSUME  TRC$L_BLNK EQ 4
    80  00000040 8F    CE     004C    883          MNEGL   #TRC$C_BLN, (R0)+      ; Back offset to last block.
                                0053    884          ASSUME  TRC$B_BID EQ 8
                                0053    885          ASSUME  TRC$B_BLN EQ <TRC$B_BID + 1>
      80   1012 8F    B0     0053    886          MOVW    #<TRC$C_BID+<TRC$C_BLN/4@8>>, (R0)+ ; Store id and bln.
FFE7  50   36   51    F1     0058    887          ACBL    R1, #TRC$C_BLN-10, R0, 10$ ; Keep going until past limit.
                                005E    888
    50  00000040 8F    C2     005E    889          SUBL2   #TRC$C_BLN, R0         ; Back up to last trace block.
       51   50   53    C3     0065    890          SUBL3   R3, R0, R1             ; R1 is offset to last trc blk.
     24 A3   51   20    C3     0069    891          SUBL3   #GBH$L_TRC_FLNK, R1, GBH$L_TRC_BLNK(R3) ; Back link in header.
          60   24 A3    CE     006E    892          MNEGL   GBH$L_TRC_BLNK(R3),TRC$L_FLNK(R0) ; Flnk to hdr from last trc.
          50   20 A3    DE     0072    893          MOVAL   GBH$L_TRC_FLNK(R3), R0 ; Addr of flnk from header.
               50   60    C0     0076    894          ADDL2   (R0), R0               ; Get first trace block.
        04 A0   20 A3    CE     0079    895          MNEGL   GBH$L_TRC_FLNK(R3), TRC$L_BLNK(R0) ; Fix it's back link.
          000003E8'EF    17     007E    896          JMP     STORE_PTR              ; Jump back to main line.
                                0084    897
```

```
                                0084   899 ;
                                0084   900 ; Routine called to store information in trace block from initial call
                                0084   901 ; to cache routine.
                                0084   902 ;
                                0084   903 ; AP is destroyed.  All other registers preserved.
                                0084   904
                                0084   905 RM$CACH_IN::
                          03    BB    0084   906          PUSHR   #^M<R0,R1>                  ; Save registers used.
                50  0088 CA    D0    0086   907 10$:     MOVL    IFB$L_GBH_PTR(R10), R0      ; Get pointer to gbh, if any.
                          44    13    008B   908          BEQL    EX2                         ; Exit if none.
                        0236    30    008D   909          BSBW    REMQT                       ; Get a trace block.
                          3F    13    0090   910          BEQL    EX2                         ; Exit if none.
                  50  0A A0    9E    0092   911          MOVAB   TRC$W_FUNCTION(R0), R0      ; Get addr of function cell.
                    80  01    B0    0096   912          MOVW    #GBH$M_CACHE_IN, (R0)+      ; Note this function.
                    80  59    D0    0099   913          MOVL    R9, (R0)+                   ; structure
            5C  00000000'9F    D0    009C   914          MOVL    @#CTL$GL_PCB, AP            ; Get pcb addr.
                  80  60 AC    B0    00A3   915          MOVW    PCB$L_PID(AP), (R0)+        ; pid
                        0219    30    00A7   916          BSBW    CNT                         ; seqnum
                  80  04 AE    D0    00AA   917          MOVL    4(SP), (R0)+                ; vbn
                  80  0C AE    D0    00AE   918          MOVL    12(SP), (R0)+               ; return1
                  80  20 AE    D0    00B2   919          MOVL    32(SP), (R0)+               ; return2
                    80  53    D0    00B6   920          MOVL    R3, (R0)+                   ; arg_flg
                          80    D4    00B9   921          CLRL    (R0)+                       ; bdb_addr
                          80    7C    00BB   922          CLRQ    (R0)+                       ; not used
                          80    7C    00BD   923          CLRQ    (R0)+                       ; not used
                          80    7C    00BF   924          CLRQ    (R0)+                       ; not used
        51  50  00000040 8F    C3    00C1   925          SUBL3   #TRC$C_BLN, R0, R1          ; Get addr of trc blk
                50  0088 CA    D0    00C9   926          MOVL    IFB$L_GBH_PTR(R10), R0      ; Get addr of gbh.
                        01FF    30    00CE   927          BSBW    INSQH                       ; Insert blk at head of list.
                          03    BA    00D1   928 EX2:     POPR    #^M<R0,R1>                  ; Restore registers.
                          05    00D3   929          RSB                                 ; Return to cache
                                00D4   930
                                00D4   931
```

```
                                00D4   933 ;
                                00D4   934 ; Store useful information from cache exit.
                                00D4   935 ;
                                00D4   936
                                00D4   937 RM$CACH_OUT::
                     03    BB   00D4   938          PUSHR   #^M<R0,R1>                  ; Save registers.
           50   0088 CA    D0   00D6   939 1$:      MOVL    IFB$L_GBH_PTR(R10), R0      ; Get GBH ptr, if any.
                     F4    13   00DB   940          BEQL    EX2                         ; Exit if none.
                   01E6    30   00DD   941          BSBW    REMQT                       ; Remove a trc blk from tail.
                     EF    13   00E0   942          BEQL    EX2                         ; Exit if none.
              50    0A A0    9E   00E2   943          MOVAB   TRC$W_FUNCTION(R0), R0      ; Ptr to func field.
                 80    02    B0   00E6   944          MOVW    #GBH$M_CACHE_OUT, (R0)+     ; function
                 80    59    D0   00E9   945          MOVL    R9, (R0)+                   ; structure
        5C   00000000'9F    D0   00EC   946          MOVL    @#CTL$GL_PCB, AP            ; Addr of PCB
                 80    60 AC    B0   00F3   947          MOVW    PCB$L_PID(AP), (R0)+        ; pid
                   01C9    30   00F7   948          BSBW    CNT                         ; structure
                     80    D4   00FA   949          CLRL    (R0)+                       ; vbn
                 80    0C AE    D0   00FC   950          MOVL    12(SP), (R0)+               ; return1
                 80    24 AE    D0   0100   951          MOVL    36(SP), (R0)+               ; return2
                    80    6E    D0   0104   952          MOVL    (SP), (R0)+                 ; arg_flg
                    80    54    D0   0107   953          MOVL    R4, (R0)+                   ; bdb_addr
                     45    13   010A   954          BEQL    10$
           EC A0    1C A4    D0   010C   955          MOVL    BDB$L_VBN(R4), -20(R0)
              80    0C A4    B0   0111   956          MOVW    BDB$W_USERS(R4), (R0)+
              80    0E A4    B0   0115   957          MOVW    BDB$W_BUFF_ID(R4), (R0)+
              80    0B A4    90   0119   958          MOVB    BDB$B_CACHE_VAL(R4), (R0)+
              80    0A A4    90   011D   959          MOVB    BDB$B_FLGS(R4), (R0)+
              80    20 A4    D0   0121   960          MOVL    BDB$L_VBNSEQNO(R4), (R0)+
              51    10 A4    D0   0125   961          MOVL    BDB$L_BLB_PTR(R4), R1
                     2E    13   0129   962          BEQL    20$
              80    0B A1    90   012B   963          MOVB    BLB$B_MODEHELD(R1), (R0)+
              80    0A A1    90   012F   964          MOVB    BLB$B_BLBFLGS(R1), (R0)+
                 80    51    D0   0133   965          MOVL    R1, (R0)+
              80    24 A1    D0   0136   966          MOVL    BLB$L_LOCK_ID(R1), (R0)+
              80    28 A1    D0   013A   967          MOVL    BLB$L_VALSEQNO(R1), (R0)+
                                013E   968 5$:
     51    50   00000040 8F    C3   013E   969          SUBL3   #TRC$C_BLN, R0, R1          ; Get ptr to trc blk to insert.
                 50   0088 CA    D0   0146   970          MOVL    IFB$L_GBH_PTR(R10), R0
                       0182    30   014B   971          BSBW    INSQH                       ; Insert at head of queue.
                       FF80    31   014E   972          BRW     EX2                         ; Branch to exit.
                          80    7C   0151   973 10$:     CLRQ    (R0)+
                          80    7C   0153   974          CLRQ    (R0)+
                          80    7C   0155   975 15$:     CLRQ    (R0)+
                          E5    11   0157   976          BRB     5$
                          80    B4   0159   977 20$:     CLRW    (R0)+
                          80    D4   015B   978          CLRL    (R0)+
                          F6    11   015D   979          BRB     15$
```

```
                                015F   981
                                015F   982 ;
                                015F   983 ; Store trace info for initial call to release.
                                015F   984 ;
                                015F   985
                                015F   986 RM$RLS_IN::
                     07   BB    015F   987          PUSHR   #^M<R0,R1,R2>
               50    0088 CA   D0  0161   988 1$:    MOVL    IFB$L_GBH_PTR(R10), R0
                        03   12  0166   989          BNEQ    3$
                      00A3   31  0168   990          BRW     EX1
                      0158   30  016B   991 3$:     BSBW    REMQT                   ; Get trc blk from end.
                        03   12  016E   992          BNEQ    4$                      ; Branch if got one.
                      009B   31  0170   993          BRW     EX1                     ; Else exit.
                  50   0A A0   9E  0173   994 4$:     MOVAB   TRC$W_FUNCTION(R0), R0
                     80   04   B0  0177   995          MOVW    #GBH$M_RLS_IN, (R0)+    ; function
                     80   59   D0  017A   996          MOVL    R9, (R0)+               ; structure
               5C  00000000'9F   D0  017D   997          MOVL    @#CTL$GL_PCB, AP
                     80   60 AC  B0  0184   998          MOVW    PCB$L_PID(AP), (R0)+    ; pid
                      0138   30  0188   999          BSBW    CNT                     ; seqnum
                                018B  1000
                        51   7C  018B  1001          CLRQ    R1
                        54   D5  018D  1002          TSTL    R4
                        04   12  018F  1003          BNEQ    5$
                                0191  1004
                        80   D4  0191  1005          CLRL    (R0)+                   ; VBN
                        20   11  0193  1006          BRB     50$
                                0195  1007 5$:
                  08 A4   10   91  0195  1008          CMPB    #BLB$C_BID, BLB$B_BID(R4)
                        05   12  0199  1009          BNEQ    20$
                     51   54   D0  019B  1010          MOVL    R4, R1
                        07   11  019E  1011          BRB     30$
               51    10 A4   D0  01A0  1012 20$:     MOVL    BDB$L_BLB_PTR(R4), R1
                  52    54   D0  01A4  1013          MOVL    R4, R2
                        52   D5  01A7  1014 30$:     TSTL    R2                      ; IS THERE BDB?
                        06   13  01A9  1015          BEQL    40$
               80    1C A2   D0  01AB  1016          MOVL    BDB$L_VBN(R2), (R0)+
                        04   11  01AF  1017          BRB     50$
               80    14 A1   D0  01B1  1018 40$:     MOVL    BLB$L_VBN(R1), (R0)+
               80    10 AE   D0  01B5  1019 50$:     MOVL    16(SP), (R0)+           ; RETURN1
               80    20 AE   D0  01B9  1020          MOVL    32(SP), (R0)+           ; RETURN2
                  80    53   D0  01BD  1021          MOVL    R3, (R0)+               ; FLAGS
                  80    52   D0  01C0  1022          MOVL    R2, (R0)+               ; BDB ADDR
                        16   13  01C3  1023          BEQL    60$
               80    0C A2   B0  01C5  1024          MOVW    BDB$W_USERS(R2), (R0)+
               80    0E A2   B0  01C9  1025          MOVW    BDB$W_BUFF_ID(R2), (R0)+
               80    0B A2   90  01CD  1026          MOVB    BDB$B_CACHE_VAL(R2), (R0)+
               80    0A A2   90  01D1  1027          MOVB    BDB$B_FLGS(R2), (R0)+
               80    20 A2   D0  01D5  1028          MOVL    BDB$L_VBNSEQNO(R2), (R0)+
                        04   11  01D9  1029          BRB     70$
                                01DB  1030 60$:
                        80   7C  01DB  1031          CLRQ    (R0)+
                        80   B4  01DD  1032          CLRW    (R0)+
                        51   D5  01DF  1033 70$:     TSTL    R1                      ; IS THERE BLB?
                        15   13  01E1  1034          BEQL    80$
               80    0B A1   90  01E3  1035          MOVB    BLB$B_MODEHELD(R1), (R0)+
               80    0A A1   90  01E7  1036          MOVB    BLB$B_BLBFLGS(R1), (R0)+
                  80    51   D0  01EB  1037          MOVL    R1, (R0)+
```

```
                80   24 A1   D0  01EE  1038          MOVL    BLB$L_LOCK_ID(R1), (R0)+
                80   28 A1   D0  01F2  1039          MOVL    BLB$L_VALSEQNO(R1), (R0)+
                     06   11  01F6  1040          BRB     90$
                                 01F8  1041 80$:
                          80   B4  01F8  1042          CLRW    (R0)+
                          80   D4  01FA  1043          CLRL    (R0)+
                          80   7C  01FC  1044          CLRQ    (R0)+
                                 01FE  1045 90$:
        51   50  00000040 8F   C3  01FE  1046          SUBL3   #TRC$C_BLN, R0, R1
                50   0088 CA   D0  0206  1047          MOVL    IFB$L_GBH_PTR(R10), R0
                        00C2   30  020B  1048          BSBW    INSQH                   ; Insert element at head of queue.
                          07   BA  020E  1049 EX1:     POPR    #^M<R0,R1,R2>
                               05  0210  1050          RSB
```

```
                     0211  1052
                     0211  1053 ;
                     0211  1054 ; Store trace info at exit of release routine.
                     0211  1055 ;
                     0211  1056
                     0211  1057 RMS$RLS_OUT::
            07   BB  0211  1058          PUSHR   #^M<R0,R1,R2>
     50  0088 CA   D0  0213  1059 1$:     MOVL    IFB$L_GBH_PTR(R10), R0
               03   12  0218  1060          BNEQ    3$
             FFF1   31  021A  1061          BRW     EX1
             00A6   30  021D  1062 3$:      BSBW    REMQT                   ; Get trc blk from end of queue.
               03   12  0220  1063          BNEQ    4$                      ; Br if got one
             FFE9   31  0222  1064          BRW     EX1                     ; Else quit.
      50   0A A0   9E  0225  1065 4$:      MOVAB   TRC$W_FUNCTION(R0), R0
         80   08   B0  0229  1066          MOVW    #GBH$M_RLS_OUT, (R0)+   ; function
         80   59   D0  022C  1067          MOVL    R9, (R0)+               ; structure
   5C  00000000'9F   D0  022F  1068          MOVL    @#CTL$GL_PCB, AP
      80   60 AC   B0  0236  1069          MOVW    PCB$L_PID(AP), (R0)+    ; pid
             0086   30  023A  1070          BSBW    CNT                     ; seqnum
                    023D  1071
               51   7C  023D  1072          CLRQ    R1
               54   D5  023F  1073          TSTL    R4
               04   12  0241  1074          BNEQ    5$
                    0243  1075
               80   D4  0243  1076          CLRL    (R0)+                   ; VBN
               20   11  0245  1077          BRB     50$
                    0247  1078 5$:
      08 A4    10   91  0247  1079          CMPB    #BLB$C_BID, BLB$B_BID(R4)
               05   12  024B  1080          BNEQ    20$
         51    54   D0  024D  1081          MOVL    R4, R1
               07   11  0250  1082          BRB     30$
   51    10 A4    D0  0252  1083 20$:      MOVL    BDB$L_BLB_PTR(R4), R1
      52    54   D0  0256  1084          MOVL    R4, R2
               52   D5  0259  1085 30$:      TSTL    R2                      ; IS THERE BDB?
               06   13  025B  1086          BEQL    40$
80    1C A2   D0  025D  1087          MOVL    BDB$L_VBN(R2), (R0)+
               04   11  0261  1088          BRB     50$
80    14 A1   D0  0263  1089 40$:      MOVL    BLB$L_VBN(R1), (R0)+
80    10 AE   D0  0267  1090 50$:      MOVL    16(SP), (R0)+           ; RETURN1
80    20 AE   D0  026B  1091          MOVL    32(SP), (R0)+           ; RETURN2
   80    6E   D0  026F  1092          MOVL    (SP), (R0)+             ; STATUS
   80    52   D0  0272  1093          MOVL    R2, (R0)+               ; BDB ADDR
               16   13  0275  1094          BEQL    60$
80    0C A2   B0  0277  1095          MOVW    BDB$W_USERS(R2), (R0)+
80    0E A2   B0  027B  1096          MOVW    BDB$W_BUFF_ID(R2), (R0)+
80    0B A2   90  027F  1097          MOVB    BDB$B_CACHE_VAL(R2), (R0)+
80    0A A2   90  0283  1098          MOVB    BDB$B_FLGS(R2), (R0)+
80    20 A2   D0  0287  1099          MOVL    BDB$L_VBNSEQNO(R2), (R0)+
               04   11  028B  1100          BRB     70$
                    028D  1101 60$:
               80   7C  028D  1102          CLRQ    (R0)+
               80   B4  028F  1103          CLRW    (R0)+
               51   D5  0291  1104 70$:      TSTL    R1                      ; IS THERE BLB?
               15   13  0293  1105          BEQL    80$
80    0B A1   90  0295  1106          MOVB    BLB$B_MODEHELD(R1), (R0)+
80    0A A1   90  0299  1107          MOVB    BLB$B_BLBFLGS(R1), (R0)+
   80    51   D0  029D  1108          MOVL    R1, (R0)+
```

```
                80    24 A1    D0  02A0  1109          MOVL    BLB$L_LOCK_ID(R1), (R0)+
                80    28 A1    D0  02A4  1110          MOVL    BLB$L_VALSEQNO(R1), (R0)+
                      06       11  02A8  1111          BRB     90$
                                   02AA  1112 80$:
                      80       B4  02AA  1113          CLRW    (R0)+
                      50       D4  02AC  1114          CLRL    (R0)+
                      80       7C  02AE  1115          CLRQ    (R0)+
                                   02B0  1116 90$:
        51    50    00000040 8F  C3  02B0  1117          SUBL3   #TRC$C_BLN, R0, R1
                50    0088 CA  D0  02B8  1118          MOVL    IFB$L_GBH_PTR(R10), R0
                         0010  30  02BD  1119          BSBW    INSQH                   ; Insert at head of queue.
                         FF4B  31  02C0  1120          BRW     EX1
                                   02C3  1121
                                   02C3  1122 CNT:
                      80       B4  02C3  1123          CLRW    (R0)+
                               05  02C5  1124          RSB
                                   02C6  1125
                                   02C6  1126 ;CRASH: RMSPBUG -99
```

```
                     02C6  1128 ;
                     02C6  1129 ; Routine to remove an element from the end of a self relative queue.
                     02C6  1130 ; The forward and back links in the removed element remain intact.
                     02C6  1131 ;
                     02C6  1132 ; Input: R0 - GBH header.
                     02C6  1133 ; Output: R0 - trc blk element to use.
                     02C6  1134 ; R1 destroyed.
                     02C6  1135 ;
                     02C6  1136
                     02C6  1137 REMQT:
      20 A0    D5    02C6  1138         TSTL    GBH$L_TRC_FLNK(R0)      ; Make sure trace blocks exits.
         04    13    02C9  1139         BEQL    10$                     ; EQL there aren't any.
50    20 A0    5F    02CB  1140         REMQTI  GBH$L_TRC_FLNK(R0),R0   ; Remove a trc block from end of queue.
         05    02CF  1141 10$:     RSB                                   ; Return.
                     02D0  1142
                     02D0  1143 ;
                     02D0  1144 ; Routine to insert the trc blk previously removed from the tail of the queue
                     02D0  1145 ; onto the head of the queue.
                     02D0  1146 ;
                     02D0  1147 ; Input:
                     02D0  1148 ; R0 - GBH ptr.
                     02D0  1149 ; R1 - element to insert.
                     02D0  1150 ;
                     02D0  1151
                     02D0  1152 INSQH:
   20 A0    61    5C  02D0  1153         INSQHI  (R1),GBH$L_TRC_FLNK(R0) ; Insert onto front of queue.
         05    02D4  1154         RSB                                   ; And return.
                     02D5  1155
                     02D5  1156         $PSECT_RESTORE
                     043F  1157         .END
```

```
$$.PSECT_EP                = 00000000
$$RMSTEST                  = 0000001A
$$RMS_PBUGCHK              = 00000010
$$RMS_TBUGCHK              = 00000008
$$RMS_UMODE                = 00000004
$$T1                       = 00000000
$$T2                       = 00000004
ADDTRC                       00000004 R      03
AGAIN                        00000169 R      01
ALBLBERR                     0000021A R      01
ALBLBERR1                    00000221 R      01
ALGBPERR                     00000228 R      01
ALGBPERR1                    0000022F R      01
ALLOC                        0000009D R      01
BAD_GBC                      00000236 R      01
BDB$B_CACHE_VAL            = 0000000B
BDB$B_FLGS                 = 0000000A
BDB$L_BLB_PTR              = 00000010
BDB$L_VBN                  = 0000001C
BDB$L_VBNSEQNO             = 00000020
BDB$W_BUFF_ID              = 0000000E
BDB$W_USERS                = 0000000C
BLB$B_BID                  = 00000008
BLB$B_BLBFLGS              = 0000000A
BLB$B_MODEHELD             = 0000000B
BLB$C_BID                  = 00000010
BLB$L_LOCK_ID              = 00000024
BLB$L_VALSEQNO             = 00000028
BLB$L_VBN                  = 00000014
BLKALC                       0000010D R      01
CCB$L_WIND                 = 00000004
CHKGBC                       000001B6 R      01
CHKMBC                       0000004A R      01
CHK_GBC                      0000023D R      01
CLNT                         00000036 R      01
CNT                          000002C3 R      03
CTL$GL_PCB                   ******** X      03
DECR_BCNT                    00000194 R      01
DEV$V_FOD                  = 0000000E
DEV$V_FOR                  = 00000018
DEV$V_REC                  = 00000000
DEV$V_RND                  = 0000001C
DEV$V_RTM                  = 0000001D
DEV$V_SQD                  = 00000005
DEV$V_TRM                  = 00000002
ERL0                         000001EC R      01
ERL1                         000001F4 R      01
ERL2                         0000020A R      01
ERL3                         00000203 R      01
ERL4                         000001FC R      01
ERRDEV                       0000003C R      01
ERRMBC                       00000043 R      01
ERRRFM                       00000031 R      01
EX1                          0000020E R      03
EX2                          000000D1 R      03
EXIT                         00000196 R      01
FAB$L_STV                  = 0000000C
FAB$M_BIO                  = 00000020
FAB$V_BRO                  = 00000006
FAB$W_GBC                  = 00000048
FAOCNTRL                     00000000 R      01
GBD$B_BID                  = 00000008
GBD$B_BLN                  = 00000009
GBD$C_BID                  = 00000013
GBD$C_BLN                  = 00000028
GBD$L_BLINK                = 00000004
GBD$L_FLINK                = 00000000
GBD$L_REL_ADDR             = 0000001C
GBD$L_VBN                  = 0000000C
GBD$W_SIZE                 = 0000001A
GBH$B_BID                  = 00000008
GBH$C_BID                  = 00000011
GBH$C_BLN                  = 00000058
GBH$L_GBD_BLNK             = 00000004
GBH$L_GBD_END              = 0000002C
GBH$L_GBD_FLNK             = 00000000
GBH$L_GBD_NEXT             = 00000030
GBH$L_GBD_START            = 00000028
GBH$L_GS_SIZE              = 00000010
GBH$L_HI_VBN               = 0000000C
GBH$L_LOCK_ID              = 00000014
GBH$L_SCAN_NUM             = 00000034
GBH$L_TRC_BLNK             = 00000024
GBH$L_TRC_FLNK             = 00000020
GBH$L_USECNT               = 0000001C
GBH$M_CACHE_IN             = 00000001
GBH$M_CACHE_OUT            = 00000002
GBH$M_RLS_IN               = 00000004
GBH$M_RLS_OUT              = 00000008
GBSB$C_GS_SIZE             = 00000038
GBSB$W_GBC                 = 00000034
GIVEBACK                     000001A9 R      01
IFB$B_FAC                  = 00000022
IFB$L_BDB_BLNK             = 00000044
IFB$L_BLBBLNK              = 0000009C
IFB$L_DEVBUFSIZ            = 00000048
IFB$L_EBK                  = 00000074
IFB$L_GBH_PTR              = 00000088
IFB$L_GBSB_PTR             = 0000007C
IFB$L_PAR_LOCK_ID          = 00000080
IFB$L_PRIM_DEV             = 00000000
IFB$L_SFSB_PTR             = 00000078
IFB$V_BIO                  = 00000005
IFB$V_DAP                  = 0000003E
IFB$V_EOF                  = 00000021
IFB$V_NORECLK              = 00000033
IFB$V_PPF_INPUT            = 0000002E
IFB$V_TEF                  = 00000036
IFB$W_AVLCL                = 00000084
IFB$W_FFB                  = 0000005C
IMP$V_IIOS                 = 00000000
IMP$V_NOPOBUFS             = 00000005
INIT_TRC                     00000028 R      03
INSQR                        000002D0 R      03
```

```
IRB$B_BCNT                      = 00000054
IRB$B_MBC                       = 00000055
IRB$B_MBF                       = 0000005C
IRB$L_NRP_VBN                   = 00000040
IRB$L_NXTBDB                    = 0000003C
IRB$V_EOF                       = 00000021
IRB$V_GBLBUFF                   = 00000036
IRB$V_RAHWBH                    = 0000002A
IRB$W_NRP_OFF                   = 00000044
MAP_IT                            00000269 R      01
NT$CONNECT                        ******** X      01
NUMTRC                            00000000 R      03
PCB$L_PID                       = 00000060
PIO$GB_DFMBC                      ******** X      01
PIO$GB_DFMBFSDK                   ******** X      01
PRV$V_SYSGBL                    = 00000019
PSL$C_EXEC                      = 00000001
RAB$B_MBC                       = 00000037
RAB$B_MBF                       = 00000036
RAB$B_ROP1                      = 00000005
RAB$L_FAB                       = 0000003C
RAB$L_ROP                       = 00000004
RAB$L_STV                       = 0000000C
RAB$M_RAH                       = 00000200
RAB$M_WBH                       = 00000400
RAB$V_BIO                       = 0000000B
RAB$V_EOF                       = 00000008
REMQT                             000002C6 R      03
RM$ALBLB                          ******** X      01
RM$ALDBUF                         ******** X      01
RM$ALGBPB                         ******** X      01
RM$BDBALLOC                       00000111 RG     01
RM$BDBALLOC_ALT                   0000015E RG     01
RM$CACH_IN                        00000084 RG     03
RM$CACH_OUT                       000000D4 RG     03
RM$CCLNT                          ******** X      01
RM$COMCLNUP                       ******** X      01
RM$CONNECT1                       00000009 RG     01
RM$CONN_ERRRFM                    00000031 RG     01
RM$EXSUC                          ******** X      01
RM$EX_NOSTR                       ******** X      01
RM$GETCCB                         ******** X      01
RM$INIT_GBSB                      ******** X      01
RM$INIT_SFSB_IRB                  ******** X      01
RM$LOWER_GBS_LOCK                 ******** X      01
RM$LOWER_SYSLOCK                  ******** X      01
RM$RETBDB                         ******** X      01
RM$RETBLB                         ******** X      01
RM$RETGBPB                        ******** X      01
RM$RLS_GBSB                       ******** X      01
RM$RLS_IN                         0000015F RG     03
RM$RLS_OUT                        00000211 RG     03
RM$UNMAP_GBL_ALT                  ******** X      01
RMS$_CRMP                       = 0001C14C
RMS$_DEV                        = 000184C4
RMS$_DME                        = 000184D4
RMS$_GBC                        = 000187CC
RMS$_MBC                        = 00018734
RMS$_RFM                        = 00018664
ROP                             = 00000020
SEC$M_DZRO                      = 00000004
SEC$M_EXPREG                    = 00020000
SEC$M_GBL                       = 00000001
SEC$M_PAGFIL                    = 00080C00
SEC$M_SYSGBL                    = 00008000
SEC$M_WRT                       = 00000008
SEC_ERR                           0000042A R      01
SEC_ERR1                          00000423 R      01
SETNXT                            000000BE R      01
SSS_CREATED                     = 00000619
STORE_PTR                         000003E8 R      01
SYS$CRMPSC                        ******** GX     01
SYS$FAO                           ******** X      01
SYS$GB_DFMBC                      ******** X      01
SYS$GB_DFMBFSDK                   ******** X      01
SYS$SETPRV                        ******** GX     01
TRC$B_BID                       = 00000008
TRC$B_BLN                       = 00000009
TRC$C_BID                       = 00000012
TRC$C_BLN                       = 00000040
TRC$L_BLNK                      = 00000004
TRC$L_FLNK                      = 00000000
TRC$W_FUNCTION                  = 0000000A
UNIT                              000000A6 R      01
WCB$L_FCB                       = 00000018
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RMSRMS1	0000043F (1087.)	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
RMSTRACE	000002D5 (725.)	03 (3.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.08	00:00:00.40
Command processing	137	00:00:00.68	00:00:04.36
Pass 1	608	00:00:26.33	00:01:09.37
Symbol table sort	0	00:00:03.90	00:00:04.89
Pass 2	219	00:00:05.36	00:00:13.14
Symbol table output	32	00:00:00.22	00:00:00.74
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1036	00:00:36.60	00:01:32.94

The working set limit was 1950 pages.
145709 bytes (285 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2617 non-local and 71 local symbols.
1157 source lines were read in Pass 1, producing 20 object records in Pass 2.
45 pages of virtual memory were used to define 43 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	19
_255DUA28:[SYS.OBJ]LIB.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	16
TOTALS (all libraries)	39

2809 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM1CONN/OBJ=OBJ$:RM1CONN MSRC$:RM1CONN/UPDATE=(ENH$:RM1CONN)+EXECML$/LIB+LIB$:RMS/LIB
```

RM1CONN
LIS

RM1GET
LIS

RM1INPSCN
LIS

RM1DISCON
LIS

RM1GETNT
LIS

RM1CREATE
LIS

RM1JOURNL
LIS