```
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
```

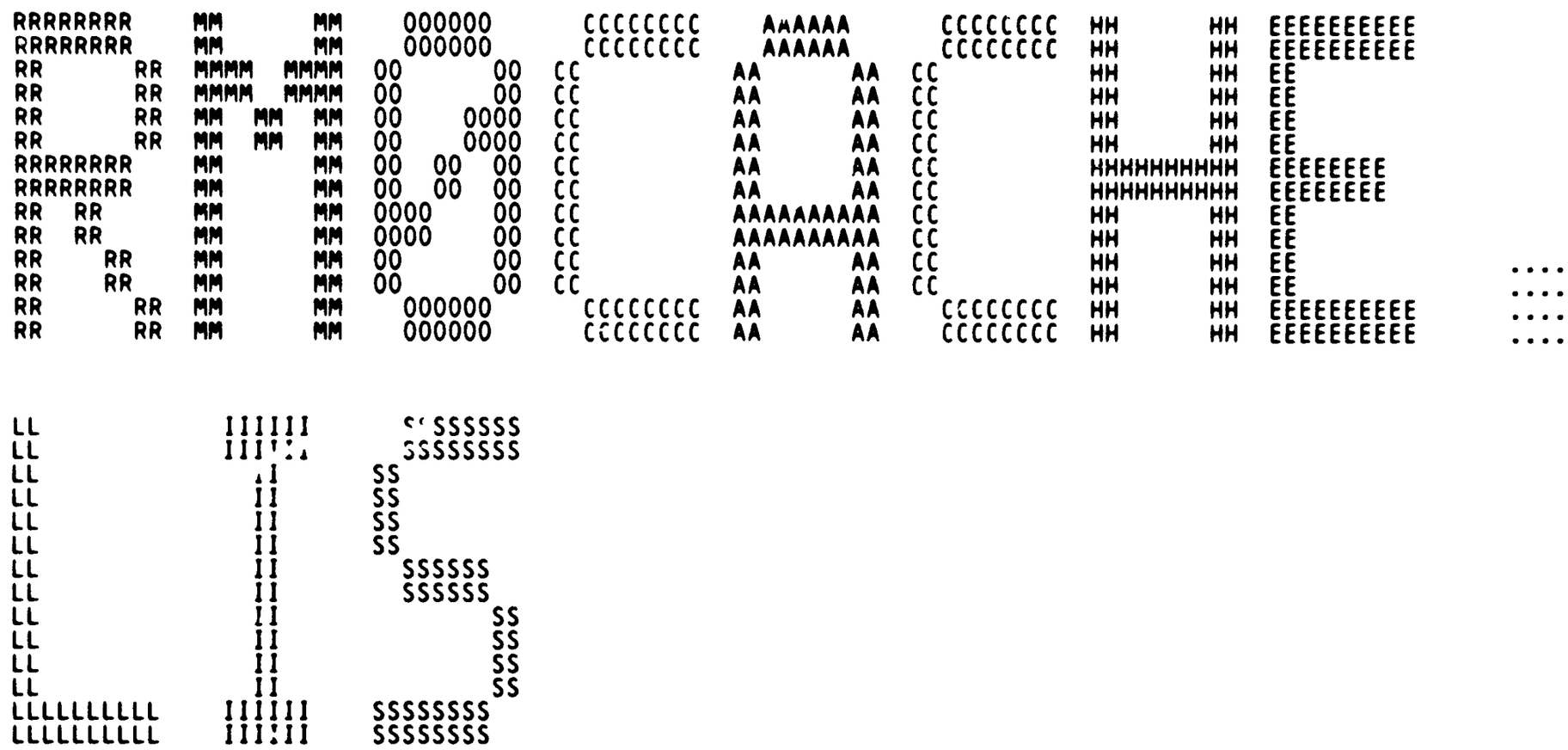
RMOCACHE
VOLUME

Table of contents

```
0000     1          $BEGIN  RMOCACHE,000,RM$RMS0,<IO CACHE ROUTINE>
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 ;
0000     30 ; Facility: rms32
0000     31 ;
0000     32 ; Abstract:
0000     33 ;
0000     34 ;       This module provides a block cache and access control
0000     35 ;       to the buckets of the relative and indexed file organizations
0000     36 ;
0000     37 ; Environment:
0000     38 ;       VAX/VMS
0000     39 ;
0000     40 ; Author: E.H. Marison  15-SEP-1977
0000     41 ;
0000     42 ; Modified By:
0000     43 ;
0000     44 ;       V03-023 JEJ0044         J E Johnson             21-Jun-1984
0000     45 ;               Tweak the instructions a little for a performance boost.
0000     46 ;
0000     47 ;       V03-022 SHZ0011         Stephen H. Zalewski,    30-Apr-1984
0000     48 ;               If we stall in CACHE, do not set the event flag unless it
0000     49 ;               is nonzero.  In async I/O case it will be zero.
0000     50 ;
0000     51 ;       V03-021 JEJ0007         J E Johnson             16-Mar-1984
0000     52 ;               Add global buffer quota accounting to limit the number of
0000     53 ;               system-wide locks taken out by the users.
0000     54 ;
0000     55 ;       V03-020 SHZ0010         Stephen H. Zalewski,    13-Mar-1984
0000     56 ;               Make sure we are record locking before we try to scan
0000     57 ;               the BLB list in RMS$FREE_LCL.  Thus, in the case of
0000     58 ;               the local buffer count being wrong, we will not
0000     59 ;               try to scan the BLB list to try and free up a BDB.
0000     60 ;
0000     61 ;       V03-019 JWT0160         Jim Teague              29-Feb-1984
0000     62 ;               Remove call to RMS$DEALLEFN.
0000     63 ;
0000     64 ;       V03-018 SHZ0009         Stephen H. Zalewski     26-Jan-1984
0000     65 ;               If we must stall waiting for a writeback to occur on
0000     66 ;               a blb we want to throw out of cache, we must set the
0000     67 ;               event flag after the stall, or we wait forever on the flag.
0000     68 ;
0000     69 ;       V03-017 SHZ0008         Stephen H. Zalewski     5-Dec-1983
0000     70 ;               If we stall waiting for a writeback to occur, set the
0000     71 ;               event flag after the stall has completed, or we will
0000     72 ;               wait forever on the event flag to be set.
0000     73 ;
0000     74 ;       V03-016 KPL0001         Peter Lieberwirth       28-Oct-1983
0000     75 ;               Fix problem with BI journaling.  Before-image copy of
0000     76 ;               the buffer was only made if the buffer was read in with
0000     77 ;               intent to write.  However, if the buffer was cached for
0000     78 ;               read, and is found in the cache to write out, no before
0000     79 ;               image copy was made.
0000     80 ;
0000     81 ;               This fix unconditionally copies the before-image of the
0000     82 ;               bucket to the before-image buffer - read or write.  While
0000     83 ;               this is good enough for FT1, it should be changed for FT2
0000     84 ;               such that the copy is only made when the bucket is to be
```

```
0000     85 ;          dirtied.  The CPU performance consequences of failing to
0000     86 ;          do so would be high.
0000     87 ;
0000     88 ;  V03-015 SHZ0007         Stephen H. Zalewski     17-Oct-1983
0000     89 ;          After finding or obtaining a buffer in a global buffer
0000     90 ;          cache, do not lower lock on section until the user count
0000     91 ;          in the GBD has been incremented.  This prevents a window
0000     92 ;          where 2 accessors point to the same GBD, but each thinks it
0000     93 ;          contains a different VBN.
0000     94 ;
0000     95 ;          After finding a buffer to throw out of a global buffer cache,
0000     96 ;          mark the sequence number invalid to prevent a second process
0000     97 ;          from thinking the buffer valid while first process does the io.
0000     98 ;
0000     99 ;  V03-014 DAS0001         David Solomon           18-Oct-1983
0000    100 ;          Restore lost BI journaling code.
0000    101 ;
0000    102 ;  V03-013 SHZ0006         Stephen H. Zalewski     28-Jul-1983
0000    103 ;          Modify to allow cluster global buffers.
0000    104 ;
0000    105 ;  V03-012 SHZ0005         Stephen H. Zalewski     17-Apr-1983
0000    106 ;          Add cluster failover capability for bucket locking.
0000    107 ;
0000    108 ;  V03-011 TMK0001         Todd M. Katz            02-Apr-1983
0000    109 ;          Add support for BI Journalling of ISAM files. Whenever, an
0000    110 ;          ISAM file is marked for BI Journalling, and an EXclusive lock
0000    111 ;          has been requested on a bucket, then move the contents of the
0000    112 ;          bucket (before they are potentially modified) into the buffer
0000    113 ;          controlled by the BI BDB associated with the BDB that is about
0000    114 ;          to be returned. Also modify the routines within this module
0000    115 ;          so that the cache flags within R3 are not destroyed. This is
0000    116 ;          because they will be needed to decide whether to save the
0000    117 ;          bucket or not.
0000    118 ;
0000    119 ;  V03-010 SHZ0004         Stephan H. Zalewski,    11-Feb-1983
0000    120 ;          Update the VBN sequence number of a buffer if the NOREAD
0000    121 ;          flag is set in the BLB.
0000    122 ;
0000    123 ;  V03-009 KBT0446         Keith B. Thompson       5-Dec-1982
0000    124 ;          Fix a case where the gbsb lock was not being released
0000    125 ;
0000    126 ;  V03-008 SHZ0003         Stephen H. Zalewski,    22-Sep-1982  13:49
0000    127 ;          Take EXCLUSIVE lock on Global buffer cache when searching
0000    128 ;          for a buffer, or updating the position of a buffer in the
0000    129 ;          cache.
0000    130 ;
0000    131 ;  V03-007 SHZ0002         Stephen H. Zalewski,    6-Sep-1982  20:18
0000    132 ;          Use the interlocked self-relative queue instructions when
0000    133 ;          placing and removing GBDs.
0000    134 ;
0000    135 ;  V03-006 KBT0200         Keith B. Thompson       23-Aug-1982
0000    136 ;          Reorganize psects
0000    137 ;
0000    138 ;  V03-005 SHZ0001         Stephen H. Zalewski,    29-Jun-1982  15:38
0000    139 ;          When forcing the writeback of a BLB, make sure the IFAB is
0000    140 ;          not one from a $OPEN command.  If it is, do not attempt to
0000    141 ;          write the BLB back.
```

```
0000   142 ;
0000   143 ;       V03-004 CDS0028         C Saether                13-Apr-1982
0000   144 ;               Modify lock_bkt routine to attempt to toss something
0000   145 ;               out of the local cache when the enq results in
0000   146 ;               either an exenqlm or nolockid error.  Hopefully this
0000   147 ;               frees up a lock so the operation can continue.
0000   148 ;
0000   149 ;       V03-003 CDS0027         C Saether                30-Mar-1982
0000   150 ;               Correct problem when LOCK, NOREAD, NOBUFFER is
0000   151 ;               specified for a bucket which is already accessed
0000   152 ;               with a buffer.  This was causing the new flags to
0000   153 ;               overwrite those from the initial access and not
0000   154 ;               store the value block on release.
0000   155 ;
0000   156 ;               Set default error in R1 before call to MAPERR.
0000   157 ;               Increment use count in GBD only after GBPB accessed.
0000   158 ;
0000   159 ;               Check if BDB was present before looking at it.
0000   160 ;
0000   161 ;       V03-002 CDS0026         C Saether                22-Mar-1982
0000   162 ;               Modify SCAN_GBL routine to avoid end test and speed
0000   163 ;               up by looping in line.
0000   164 ;
0000   165 ;       V03-001 CDS0025         C Saether                 1-Mar-1982
0000   166 ;               Count hits and misses for global buffers.
0000   167 ;               Fix incorrect register use in SCAN_LOCKS checking
0000   168 ;               for GBPB.
0000   169 ;                                                        26-Feb-1982
0000   170 ;               Don't bother trying to get read locks on buckets in
0000   171 ;               a compatible mode.  It causes conversion deadlock
0000   172 ;               problems when other streams (processes) are attempting
0000   173 ;               to modify the same bucket simultaneously.  This is
0000   174 ;               the same problem only partially corrected in V02-036.
0000   175 ;               Also have LOCK_BKT routine call RM$MAPERR to map
0000   176 ;               enqueue failures to RMS errors if appropriate.
0000   177 ;                                                        16-Feb-1982
0000   178 ;               Changes to allow modification of global buffers.
0000   179 ;               FREE_LCL and GET_LCL_BUFF become RM$FREE_LCL and
0000   180 ;               RM$GET_LCL_BUFF.
0000   181 ;               SCAN_LOCKS changed to not return GBPB address unless
0000   182 ;               caller already owns it (don't want more than one
0000   183 ;               accessor on a GBPB at once).
0000   184 ;               Don't try to use global buffers if stream didn't
0000   185 ;               connect for them.
0000   186 ;                                                         7-Feb-1982
0000   187 ;               Shorten scan_local_cache to scan_lcl_cache.
0000   188 ;
0000   189 ;--
0000   190 ;
0000   191
```

```
                     0000   193          .SBTTL  DECLARATIONS
                     0000   194
                     0000   195 ;
                     0000   196 ; Include Files:
                     0000   197 ;
                     0000   198
                     0000   199          $BDBDEF                         ; bucket descriptor blocks
                     0000   200          $BKTDEF                         ; bucket definitions
                     0000   201          $BLBDEF                         ; bucket lock blocks
                     0000   202          $CSHDEF                         ; rm$cache flag definitions
                     0000   203          $ENQDEF                         ; enq sys service definitions
                     0000   204          $GBDDEF                         ; global buffer descriptor defs
                     0000   205          $GBHDEF                         ; global buffer header
                     0000   206          $GBPBDEF                        ; global buffer pointer block defs
                     0000   207          $IFBDEF                         ; ifab defs
                     0000   208          $IMPDEF                         ; impure area definitions
                     0000   209          $IRBDEF                         ; irab defs
                     0000   210          $LCKDEF                         ; lock manager flags, constant defs
                     0000   211          $PIODEF                         ; process i/o definitions
                     0000   212          $RABDEF                         ; record access block
                     0000   213          $RJRDEF                         ; RMS Journalling Record definitions
                     0000   214          $RLSDEF                         ; rm$release flags
                     0000   215          $RMSDEF                         ; rms error code defs
                     0000   216          $SFSBDEF                        ; shared file lock block defs
                     0000   217          $SSDEF                          ; System service error codes
                     0000   218
                     0000   219 ;
                     0000   220 ; Macros:
                     0000   221 ;
                     0000   222 ;
                     0000   223 ; Equated Symbols:
                     0000   224 ;
                     0000   225
0000000F             0000   226 CSH_MASK_ALL     =       CSH$M_LOCK!CSH$M_NOWAIT!CSH$M_NOREAD!CSH$M_NOBUFFER
                     0000   227
                     0000   228 ;
                     0000   229 ; Own Storage:
                     0000   230 ;
                     0000   231
```

```
0000   233          .SBTTL  RM$CACHE ROUTINE
0000   234
0000   235 ;++
0000   236 ;
0000   237 ; RM$CACHE - access and read bucket if necessary
0000   238 ;
0000   239 ;    1. obtains access to requested block/bucket and waits for the access
0000   240 ;       unless csh$v_nowait is set in the control flags.
0000   241 ;
0000   242 ;    2. obtains a buffer for the block unless csh$v_nobuffer
0000   243 ;       is set in the control flags.
0000   244 ;
0000   245 ;    3. if there is a buffer read the block into it if required, and the
0000   246 ;       csh$v_noread bit is off in the control flags.
0000   247 ;
0000   248 ;    4. waits for io completion
0000   249 ;
0000   250 ;    5. if the csh$v_lock bit is set in the flags then exclusive access to the
0000   251 ;       block is obtained.
0000   252 ;
0000   253 ; Calling sequence:
0000   254 ;
0000   255 ;       BSBW    RM$CACHE
0000   256 ;
0000   257 ; Input Parameters:
0000   258 ;
0000   259 ;       r11     impure area address
0000   260 ;       r10     ifab address
0000   261 ;       r9      irab/ifab address
0000   262 ;       r8      rab/fab address
0000   263 ;       r3      cache control flags
0000   264 ;       r2      transfer/buffer size in bytes
0000   265 ;       r1      requested vbn
0000   266 ;
0000   267 ; Output Parameters:
0000   268 ;
0000   269 ;       r0      internal rms status code
0000   270 ;       r4      bdb address
0000   271 ;       r5      buffer address unless a nobuffer call
0000   272 ;               in which case r5 is destroyed
0000   273 ;       r1,r2,r3,ap destroyed
0000   274 ;
0000   275 ; ****  bdb and buffer not accessed on errors
0000   276 ;
0000   277 ; Completion Codes:
0000   278 ;
0000   279 ;       standard internal rms, including:
0000   280 ;
0000   281 ;               suc     normal success
0000   282 ;               rlk     block was accessed or locked and nowait
0000   283 ;               dme     could not get a buffer
0000   284 ;               exenqlm the enq limit for this process was exceeded
0000   285 ;                       while attempting to lock a bucket.
0000   286 ;               various errors writing a deferred write buffer or reading
0000   287 ;               in the new buffer.
0000   288 ;
0000   289 ; Side Effects:
```

```
                0000   290 ;
                0000   291 ;       May have switched to running at ast level if not already there as
                0000   292 ;       a result of stalling for i/o to complete or waiting for a
                0000   293 ;       resource to be granted.
                0000   294 ;
                0000   295 ;--
```

```
                     0000   297 TRACE:
00000000'EF   16     0000   298         JSB     RM$CACH_IN
          0C 10      0006   299         BSBB    CACHE
00000000'EF   16     0008   300         JSB     RM$CACH_OUT
              05     000E   301         RSB
                     000F   302
                     000F   303 BUFF_ONLY_BR:
        01A8  31     000F   304         BRW     BUFF_ONLY               ; No lock - only want scratch buffer.
                     0012   305
                     0012   306 RM$CACHE::
              01     0012   307         NOP                             ; Patch this for tracing.
              01     0013   308         NOP
                     0014   309
          51  D5     0014   310 CACHE:  TSTL    R1                      ; Is this VBN 0 call?
          F7  13     0016   311         BEQL    BUFF_ONLY_BR            ; EQL then only want buffer, no lock.
                     0018   312
    52 53 03  E1     0018   313         BBC     #CSH$V_NOBUFFER, R3,-   ; Branch if buffer is desired.
                     001C   314                 NEED_BUFFER
    14 53 02  E1     001C   315         BBC     #CSH$V_NOREAD, R3, 17$  ; Expectation is that NOREAD is set.
                     0020   316
                     0020   317
                     0020   318 ;
                     0020   319 ; If here, this is a nobuffer request, meaning that a lock only on the
                     0020   320 ; bucket is desired.  This type of lock will be requested to interlock
                     0020   321 ; an extend type operation with other processes sharing the file.  If
                     0020   322 ; the file is open for exclusive access, nothing further is required.
                     0020   323 ;
                     0020   324
 17 6A   33  E1      0020   325 10$:    BBC     #IFB$V_NORECLK, (R10), 20$ ; Branch if locking req'd.
        030A 30      0024   326         BSBW    SCAN_LCL_CACHE          ; Is it already in cache?
     54  55  D0      0027   327         MOVL    R5, R4                  ; See if it is.
         04  13      002A   328         BEQL    15$                     ; Nope.  Just return.
 0C A4   01  B0      002C   329         MOVW    #1, BDB$W_USERS(R4)     ; Make it look accessed.
                     0030   330 15$:    RMSSUC                          ; Success.
              05     0033   331         RSB                             ; Return.
                     0034   332
                     0034   333 17$:    RMSPBUG FTL$_NORDNOTSET         ; NOREAD is not set and no buff wanted.
                     003B   334
        019B 30      003B   335 20$:    BSBW    SCAN_LOCKS              ; See if lock is already held.
      15 50  E9      003E   336         BLBC    R0, 30$                 ; Branch if lock not found.
         55  D5      0041   337         TSTL    R5                      ; Is there a BDB also?
         EB  13      0043   338         BEQL    15$                     ; Already have lock.  All done then.
                     0045   339
                     0045   340 ;
                     0045   341 ; Note that finding a buffer in the cache on a nobuffer call and simply
                     0045   342 ; accessing the buffer along with the lock violates the assumptions about
                     0045   343 ; the worst case condition where two buffers plus a lock only on another
                     0045   344 ; bucket are required.  However, given the current use of the nobuffer call
                     0045   345 ; for vbn 1, and the sequence of operations used, it would not be possible
                     0045   346 ; for vbn 1 to be in the cache at the time of the nobuffer call because
                     0045   347 ; the two buffers would be occupied by the bucket being split and the area
                     0045   348 ; descriptor already.  With two streams, it is difficult to construct a
                     0045   349 ; sequence of events where block 1 could happen to be in the cache at the
                     0045   350 ; time two streams were extending separate areas during separate splits
                     0045   351 ; such that the problem would actually arise.
                     0045   352 ;
                     0045   353 ; The logic which initializes an index will call CACHE for a lock, nobuffer
```

```
                     0045   354 ; on VBN 1 when VBN 1 is already accessed with a buffer.  In that case,
                     0045   355 ; the owner will be the current stream.  The address of the BDB (as opposed
                     0045   356 ; to the BLB) must be returned in that case because the routine compares
                     0045   357 ; the original BDB address from the first call (saved in IRB$L_LOCK_BDB)
                     0045   358 ; with the return from this call to determine whether to release the lock.
                     0045   359 ; Pretty tacky, but that's the way it is.  No checks are made to determine
                     0045   360 ; if the bucket is LOCKed when one is found, as it is not believed any routines
                     0045   361 ; would do that and follow it with a LOCK, NOBUFFER call.
                     0045   362 ;
                     0045   363
59   10 A4    D1     0045   364         CMPL    BLB$L_OWNER(R4), R9     ; This stream already have it accessed?
       05     12     0049   365         BNEQ    25$                     ; NEQ, then go access it normally.
   54  55     D0     004B   366         MOVL    R5, R4                  ; Get BDB addr into R4.
       E0     11     004E   367         BRB     15$                     ; And exit with success.
   53  0C     CA     0050   368 25$:     BICL2   #CSH$M_NOBUFFER!CSH$M_NOREAD, R3 ; There already is a buffer.
     00B9     31     0053   369         BRW     LOCK_IT                 ; Raise mode if req'd.
                     0056   370
     01EC     30     0056   371 30$:     BSBW    GET_BLB                 ; Get a BLB for this lock.
     0331     30     0059   372         BSBW    LOCK_BKT                ; Go lock it.
              05     005C   373         RSB                             ; Return.
                     005D   374
                     005D   375 ;
                     005D   376 ; Got the bucket locked, but no buffer with it.
                     005D   377 ;
                     005D   378
                     005D   379 NEED_BUFFONLY:
 0084 CA     B7     005D   380         DECW    IFB$W_AVLCL(R10)        ; Decrement available count.
       6C    18     0061   381         BGEQ    GET_BUFF                ; Enough buffers, go get BDB.
     020D    30     0063   382         BSBW    RM$FREE_LCL             ; Free up a local buffer.
   66 50     E8     0066   383         BLBS    R0, GET_BUFF            ; Branch and go use it on success.
 0084 CA     B6     0069   384         INCW    IFB$W_AVLCL(R10)        ; Restore count.
              05     006D   385         RSB                             ; Return with error in R0.
```

```
                     006E    387 NEED_BUFFER:
6A   33   E0        006E    388         BBS     #IFB$V_NORECLK, (R10),- ; Branch if no locking.
     6E             0071    389                 NOLOCKING
                    0072    390
                    0072    391 ;
                    0072    392 ; Locking is being done.  Scan list of buffer lock blocks (BLB's) to
                    0072    393 ; determine if bucket already has lock.  Normally locating a bucket in
                    0072    394 ; the BLB list means that either a NL or PW lock is held on a buffer
                    0072    395 ; currently present in the local cache.
                    0072    396 ;
                    0072    397 ; Under some conditions a request is made for a lock with buffer on a
                    0072    398 ; bucket which was previously locked with the NOBUFFER flag, in which
                    0072    399 ; case a lock will be found with no BDB.
                    0072    400 ;
                    0072    401 ; Lastly, a BDB for the desired bucket may be found, but no BLB.
                    0072    402 ; This will occur when multi-streaming and another stream has the
                    0072    403 ; desired bucket accessed.  Only a BLB must be acquired in this case.
                    0072    404 ;
                    0072    405
                    0072    406 CHECK_LOCKS:
  0164   30        0072    407         BSBW    SCAN_LOCKS              ; See if lock already held.
07 50    E9        0075    408         BLBC    R0, NEED_BLB            ; No - go to get BLB.
   55    D5        0078    409         TSTL    R5                      ; Was there a BDB also with the BLB?
   E1    13        007A    410         BEQL    NEED_BUFFONLY           ; Go get a buffer for the BLB.
 0087    31        007C    411         BRW     CHKWBK                  ; Have BDB, BLB, so access them.
                    007F    412 NEED_BLB:
0088 CA  D5        007F    413         TSTL    IFB$L_GBH_PTR(R10)      ; Global buffer cache present?
   2B    13        0083    414         BEQL    LOCAL                   ; EQL then there is none.
                    0085    415
                    0085    416 ;
                    0085    417 ; Global buffer cache is present.  If a BDB has already been found, though,
                    0085    418 ; always use it.  It didn't have a BLB if here, meaning another stream
                    0085    419 ; must currently have it accessed, therefore it has a much better chance
                    0085    420 ; of being valid.
                    0085    421 ;
                    0085    422
   55    D5        0085    423         TSTL    R5                      ; Is BDB already present?
   27    12        0087    424         BNEQ    LOCAL                   ; NEQ then use it.
                    0089    425
                    0089    426         ASSUME  IRB$B_BID       EQ      IFB$B_BID
                    0089    427         ASSUME  <IRB$C_BID & 1> EQ      0
                    0089    428         ASSUME  <IFB$C_BID & 1> EQ      1
                    0089    429
23 08 A9  E8        0089    430         BLBS    IFB$B_BID(R9), LOCAL    ; Use local if ifab operation.
1F 69  36  E1       008D    431         BBC     #IRB$V_GBLBUFF, (R9), LOCAL ; Use local if stream did not
                    0091    432                                         ; want global buffs when connecting.
                    0091    433
                    0091    434 ;
                    0091    435 ; Search global cache, if failure the gbsb lock is not released since it may
                    0091    436 ; be needed shortly in find_free_gbl.  If success the lock is released.
                    0091    437 ;
                    0091    438
  03C6   30        0091    439         BSBW    SCAN_GBL                ; Search global cache
0A 50    E8        0094    440         BLBS    R0, TO$                 ; Branch if got a match and use it.
                    0097    441
                    0097    442 ;
                    0097    443 ; Did not find the desired bucket in the global cache.
```

```
                     0097   444 ; If a lock is not requested, attempt to get a global buffer and use it.
                     0097   445 ; If a lock is requested, use a local buffer.  The belief is that if
                     0097   446 ; the bucket wasn't already in the global cache, this must be a new
                     0097   447 ; insert, therefore the chance of another process potentially having
                     0097   448 ; an interest in it is very low.  In addition, if deferred write is
                     0097   449 ; enabled, modified global buffers must be copied to a local buffer
                     0097   450 ; when they are released.  The extra cpu overhead to do that would
                     0097   451 ; outweigh the rare instances where an i/o would be saved because
                     0097   452 ; another process was interested in the same bucket.
                     0097   453 ;
                     0097   454
                     0097   455         ASSUME  CSH$V_LOCK      EQ      0
                     0097   456
07 53   E9            0097   457         BLBC    R3, 10$                 ; Br to use gbl if not locking.
 FF63'  30            009A   458         BSBW    RM$LOWER_GBS_LOCK       ; Release lock on gbsb (taken in scan_gbl)
   55   D4            009D   459         CLRL    R5                      ; Note that no buffer is present.
   0F   11            009F   460         BRB     LOCAL                   ; Go use local buffer.
                      00A1   461
                      00A1   462 ;
                      00A1   463 ; We wish to use global buffers.  R0 contains the status from the global
                      00A1   464 ; cache scan above.   Whether or not the requested bucket was found, we
                      00A1   465 ; will need a blb.  In the rare case where a global buffer cannot be freed
                      00A1   466 ; when the desired bucket was not located, reset the owner and vbn fields of
                      00A1   467 ; the blb just obtained, and drop through to use a local buffer instead.
                      00A1   468 ;
                      00A1   469
 01A1   30            00A1   470 10$.     BSBW    GET_BLB                 ; Get a free BLB for the lock.
33 50   E8            00A4   471         BLBS    R0, GOT_BUFF            ; Branch if match found in gbl cache -
                      00A7   472                                         ; R0 is the result from SCAN_GBL here.
 0453   30            00A7   473         BSBW    FIND_FREE_GBL           ; Attempt to find a free global buffer.
29 50   E8            00AA   474         BLBS    R0, NEED_READ           ; Br to force read if one found.
                      00AD   475         ASSUME  <BLB$L_OWNER + 4> EQ BLB$L_VBN
10 A4   7C            00AD   476         CLRQ    BLB$L_OWNER(R4)         ; Free up BLB.  Drop thru to use local.
                      00B0   477
                      00B0   478 ;
                      00B0   479 ; A local buffer is to be used.
                      00B0   480 ; if R5 is non-zero, it contains the address of the BDB for the requested
                      00B0   481 ; bucket even though a BLB must be obtained.
                      00B0   482 ;
                      00B0   483
                      00B0   484 LOCAL:
0084 CA   B7          00B0   485         DECW    IFB$W_AVLCL(R10)        ; Decrement available count.
     12   18          00B4   486         BGEQ    10$                     ; Got enough - go get BLB.
0084 CA   B6          00B6   487         INCW    IFB$W_AVLCL(R10)        ; Put count back - will go round again.
  01B6   30           00BA   488         BSBW    RM$FREE_LCL             ; Free up a buffer.
  38 50  E9           00BD   489         BLBC    R0, EXBR                ; Exit on error.
6A   31  E0           00C0   490         BBS     #IFB$V_MSE, (R10),-     ; If multi-streaming, need to scan
     AE                00C3   491                 CHECK_LOCKS             ; locks again (may have changed).
0084 CA   B7          00C4   492         DECW    IFB$W_AVLCL(R10)        ; Dec count.  One is available now.
  017A   30           00C8   493 10$:     BSBW    GET_BLB                 ; Get a free BLB.
     55   D5          00CB   494         TSTL    R5                      ; Is there a BDB already?
     0B   12          00CD   495         BNEQ    GOT_BUFF                ; NEQ already have one.
                      00CF   496 GET_BUFF:
  027C   30           00CF   497         BSBW    RM$GET_LCL_BUFF         ; Get a free BDB.
6A   33  E0           00D2   498         BBS     #IFB$V_NORECLK, (R10),- ; Branch if not locking.
     1D                00D5   499                 READ_NOLOCKING
                      00D6   500 NEED_READ:
```

```
                10   88  00D6   501          BISB2   #BLB$M_IOLOCK,-          ; Know that I/O will be req'd.
             0A A4       00D8   502                  BLB$B_BLBFLGS(R4)
                         00DA   503 GOT_BUFF:
          0C A4   55   D0  00DA   504          MOVL    R5, BLB$L_BDB_ADDR(R4)  ; Store BDB address in BLB.
                2F   11  00DE   505          BRB     LOCK_IT                 ; Go to lock code.
                         00E0   506
                         00E0   507 ;
                         00E0   508 ; Branch here when using local buffers in exclusive mode, i.e., no locking
                         00E0   509 ; is being performed.
                         00E0   510 ;
                         00F0   511
                         00E0   512 NOLOCKING:
              024E   30  00E0   513          BSBW    SCAN_LCL_CACHE          ; Look in local cache.
           54   55   D0  00E3   514          MOVL    R5, R4                  ; Anticipate suc - load R4 with BDB.
                03   12  00E6   515          BNEQ    10$                     ; NEQ we have a buffer.
                FF72   31  00E8   516          BRW     NEED_BUFFONLY           ; Need to go get a buffer.
                         00EB   517
                         00EB   518 ;
                         00EB   519 ; We have found the desired bucket in the local cache when no locking
                         00EB   520 ; is required.  The bucket must be valid, therefore simply return
                         00EB   521 ; with success.
                         00EB   522 ;
                         00EB   523
                         00EB   524 10$:    RMSSUC                          ; Set success.
             0C A4   B6  00EE   525          INCW    BDB$W_USERS(R4)         ; Note in use.
                6F   11  00F1   526          BRB     SETR5                   ; Branch to set R5 and exit.  .
                         00F3   527
                         00F3   528 ;
                         00F3   529 ; Branch here after getting buffer when no locking.
                         00F3   530 ;
                         00F3   531
                         00F3   532 READ_NOLOCKING:
          0C A5   B6  00F3   533          INCW    BDB$W_USERS(R5)         ; Note in use.
                5C   11  00F6   534          BRB     READ_BKT                ; Go read bucket - no lock req'd.
                         00F8   535
                009F   31  00F8   536 EXBR:    BRW     EXIT
```

```
                        00FB    538 ;
                        00FB    539 ; Check for possible writeback errors.  This is only branched to after
                        00FB    540 ; finding a lock with bdb (i.e., cached).  A similar check is made in
                        00FB    541 ; the RM$FREE_LCL routine.
                        00FB    542 ;
                        00FB    543
0C A8   48 A5   3C      00FB    544 WBKERR: MOVZWL  BDB$L_IOSB(R5), RAB$L_STV(R8) ; Store i/o error code.
                        0100    545         RMSERR  WER                         ; Note error with RMS code.
                     05 0105    546         RSB                                 ; And return.
                06   E5 0106    547 CHKWBK: BBCC    #BDB$V_AST_DCL,-            ; Branch if no writeback has been done.
          04 0A A5           0108    548                 BDB$B_FLGS(R5), LOCK_IT
          EC 48 A5   E9      010B    549         BLBC    BDB$L_IOSB(R5), WBKERR  ; Branch if error occured.
                             010F    550
                             010F    551 ;
                             010F    552 ; At this point:
                             010F    553 ;       R3 = CSH flags
                             010F    554 ;       R4 = BLB
                             010F    555 ;       R5 = BDB
                             010F    556 ;
                             010F    557
                             010F    558 LOCK_IT:
             0C A5   B6      010F    559         INCW    BDB$W_USERS(R5)         ; Bump user count.
       50    10 A5   D0      0112    560         MOVL    BDB$L_BLB_PTR(R5), R0   ; Other BLB's already?
                06   12      0116    561         BNEQ    10$                     ; NEQ there are others.
       10 A5    54   D0      0118    562         MOVL    R4, BDB$L_BLB_PTR(R5)   ; Point from BDB to our BLB.
                0B   11      011C    563         BRB     20$                     ; Branch to lock bucket.
          54    50   D1      011E    564 10$:    CMPL    R0, R4                  ; Is this us?
                06   13      0121    565         BEQL    20$                     ; EQL then yes it is.
          54    64   0F      0123    566         REMQUE  (R4), R4                ; Remove from current position in list.
          60    64   0E      0126    567         INSQUE  (R4), (R0)              ; Insert after BLB pointed to.
             0261   30      0129    568 20$:    BSBW    LOCK_BKT                ; Acquire bucket lock
            7B 50   E9      012C    569         BLBC    R0, ERREX1              ; Exit on error
                            012F    570
07 0A A4    02      E1      012F    571         BBC     #BLB$V_NOREAD,BLB$B_BLBFLGS(R4),30$ ; All done if read not req'd
          28 A4     D0      0134    572         MOVL    BLB$L_VALSEQNO(R4),-    ; It is valid, so update sequence
          20 A5             0137    573                 BDB$L_VBNSEQNO(R5)      ; number from value block.
             0C     11      0139    574         BRB     40$                     ; And branch to exit with success.
                            013B    575
             04     E4      013B    576 30$:    BBSC    #BLB$V_IOLOCK,-         ; Know bucket must be read in if set.
       0F 0A A4             013D    577                 BLB$B_BLBFLGS(R4), 50$  ; Clear so it doesn't remain set.
          28 A4     D1      0140    578         CMPL    BLB$L_VALSEQNO(R4),-    ; Compare lock value number with
          20 A5             0143    579                 BDB$L_VBNSEQNO(R5)      ; BDB sequence number.
             08     12      0145    580         BNEQ    50$                     ; NEQ cached copy is invalid.
                            0147    581 40$:
       54    55     D0      0147    582         MOVL    R5, R4                  ; Set address of BDB into R4.
                            014A    583         RMSSUC                          ; Note success
             13     11      014D    584         BRB     SETR5                   ; Branch to exit.
                            014F    585
                            014F    586 ;
                            014F    587 ; Bucket must be read because sequence numbers don't match, meaning the
                            014F    588 ; cached copy is invalid, or because the iolock bit is set, meaning this
                            014F    589 ; bucket is just being faulted into the cache.
                            014F    590 ;
                            014F    591
          28 A4     D0      014F    592 50$:    MOVL    BLB$L_VALSEQNO(R4),-    ; Update BDB copy of sequence num
          20 A5             0152    593                 BDB$L_VBNSEQNO(R5)      ; assuming success.  BDB will be deq'd
                            0154    594                                         ; on errors.
```

```
                     0154   595 READ_BKT:
    54    55    D0   0154   596          MOVL    R5, R4                      ; Get BDB/GBPB addr into R4 for read.
          53    DD   0157   597          PUSHL   R3                          ; save cache flags over call
       FEA4'    30   0159   598          BSBW    RM$RDBUFWT                  ; Read in the bucket.
          53 8ED0   015C   599          POPL    R3                          ; restore cache flags over call
    39    50    E9   015F   600          BLBC    R0, ERREX                   ; Branch on error.
                     0162   601 SETR5:
55    18 A4    D0   0162   602          MOVL    BDB$L_ADDR(R4), R5          ; Buffer address into R5.
08 A4    0C    91   0166   603          CMPB    #BDB$C_BID,BDB$B_BID(R4)    ; If this is not a BDB
         2E    12   016A   604          BNEQ    EXIT                        ;  then return.
                     016C   605
                     016C   606 ;
                     016C   607 ; If this is an ISAM file marked for BI Journaling, and if an EXclusive lock
                     016C   608 ; has been requested for the bucket that is about to be returned, then move
                     016C   609 ; the contents of the bucket (before they are potentially modified) into the
                     016C   610 ; buffer controlled by the BI BDB associated with the BDB that is about to be
                     016C   611 ; returned.
                     016C   612 ;
                     016C   613 ; ** Actually make the copy whether EX or not.  If bucket accessed for read
                     016C   614 ; ** and later upgraded to write, journaling failed because no before-image
                     016C   615 ; ** was in the buffer.  Improve performance here for FT2 as in note in
                     016C   616 ; ** revision history.
                     016C   617
                     016C   618 10$:
                     016C   619 ;        blbc    r3,20$                      ; branch if EX lock not requested
         02    E1   016C   620          BBC     #IFB$V_BI,-                 ; branch if file is not marked for
21 00A0 CA           016E   621                  IFB$B_JNLFLG(R10),20$       ; BI Journaled
         02    91   0172   622          CMPB    #IFB$C_IDX,-                ; branch if file is not an index file
      23 AA          0174   623                  IFB$B_ORGCASE(R10)          ; otherwise set up to save bucket before
         1B    12   0176   624          BNEQ    20$                         ; it is modified
                     0178   625
         3F    BB   0178   626          PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; save registers over move
   50    14 A4    3C   017A   627          MOVZWL  BDB$W_NUMB(R4),R0           ; move the entire bucket
   54    30 A4    D0   017E   628          MOVL    BDB$L_BI_BDB(R4),R4         ; retrieve address of BI Journaling BDB
         0D    13   0182   629          beql    15$                         ; skip if none, too early
 00000044 8F    C1   0184   630          ADDL3   #RJR$C_BKTLEN,-             ; position within the BI Journaling
   54    18 A4         018A   631                  BDB$L_ADDR(R4),R4           ; record to where the saved bucket goes
64    65    50   28   018D   632          MOVC3   R0,(R5),(R4)                ; save the un-modified bucket
         3F    BA   0191   633 15$:     POPR    #^M<R0,R1,R2,R3,R4,R5>      ; restore the saved registers
                     0193   634
      54    64    0F   0193   635 20$:     REMQUE  (R4), R4                    ; Take out of current position.
   40 AA    64    0E   0196   636          INSQUE  (R4), IFB$L_BDB_FLNK(R10)   ; And stick it up front.
                     019A   637 EXIT:
                05   019A   638          RSB                                 ; Done
                     019B   639 ERREX:
      53    08    D0   019B   640          MOVL    #RLS$M_DEQ, R3              ; Force complete release of buffer.
            50    DD   019E   641 ERRX:    PUSHL   R0                          ; Save the error code.
      54    55    D0   01A0   642          MOVL    R5, R4                      ; BDB/GBPB addr into R4 for release.
         FE5A'    30   01A3   643          BSBW    RM$RELEASE                  ; And release the buffer.
            50 8ED0   01A6   644          POPL    R0                          ; Restore error code.
                05   01A9   645          RSB                                 ; And Return.
                     01AA   646
50    82AA 8F    B1   01AA   647 ERREX1:  CMPW    #RMS$_RLK&^XFFFF, R0        ; Was it not queued? (nowait was set)
         EA    12   01AF   648          BNEQ    ERREX                       ; NEQ, it was something else.
      24 A4    D5   01B1   649          TSTL    BLB$L_LOCK_ID(R4)           ; Was this already locked?
         E5    13   01B4   650          BEQL    ERREX                       ; EQL no so DEQ entirely.
         53    D4   01B6   651          CLRL    R3                          ; No need to DEQ then.
```

```
E4  11  01B8  652        BRB     ERRX                    ; Br to release it.
```

```
                     01BA   654          .SBTTL  BUFF_ONLY path.
                     01BA   655 BUFF_ONLY:
                     01BA   656
                     01BA   657 ;
                     01BA   658 ; Always use local buffer for scratch buffer.
                     01BA   659 ;
                     01BA   660
0084 CA   B7         01BA   661          DECW    IFB$W_AVLCL(R10)        ; Note use of local buffer.
     0B   18         01BE   662          BGEQ    10$                     ; GEQ if have enough.
   00B0   30         01C0   663          BSBW    RM$FREE_LCL             ; Free up a buffer.
 05 50    E8         01C3   664          BLBS    R0, 10$                 ; Branch if success.
0084 CA   B6         01C6   665          INCW    IFB$W_AVLCL(R10)        ; Restore count.
          05         01CA   666          RSB                             ; Return with error in R0.
                     01CB   667 10$:
   0180   30         01CB   668          BSBW    RM$GET_LCL_BUFF         ; Go get a free buffer.
 0C A5    B6         01CE   669          INCW    BDB$W_USERS(R5)         ; Note in use.
54   55    D0         01D1   670          MOVL    R5, R4                  ; Return BDB addr in R4.
                     01D4   671          RMSSUC                          ; Note success.
     89    11         01D7   672          BRB     SETR5                   ; Set R5 and return.
```

```
                     674          .SBTTL  SCAN_LOCKS Search BLB list for BLB.
                     675 ;++
                     676 ;
                     677 ;       SCAN_LOCKS
                     678 ;
                     679 ;       Scan BLB list for match on desired vbn.  If BDB for that
                     680 ;       vbn is found, note BDB address.  If BLB free or this stream
                     681 ;       owns it, then note that also.
                     682 ;       GBPB's are only reported if this stream already has it
                     683 ;       accessed.  (Only 1 GBPB per accessor, ever).
                     684 ;
                     685 ; Calling sequence:
                     686 ;
                     687 ;       BSBW    SCAN_LOCKS
                     688 ;
                     689 ; Input Parameters:
                     690 ;
                     691 ;       R1 - VBN of bucket desired
                     692 ;       R3 - CSH flags.
                     693 ;       R9 - IFAB/IRAB address
                     694 ;       R10 - IFAB
                     695 ;         IFB$L_BLBFLNK - listhead for BLB list
                     696 ;
                     697 ; Output Parameters:
                     698 ;
                     699 ;       R0 - success if non-accessed BLB or BLB owned by this
                     700 ;               stream is found.
                     701 ;          - failure otherwise.
                     702 ;
                     703 ;       R4 - BLB address if R0 success, undefined otherwise.
                     704 ;
                     705 ;       R5 - BDB address of BDB for bucket R1 if any present, regardless
                     706 ;               of R0 status.  Zero if no BDB for bucket R1 at all.
                     707 ;            GBPB address only if this stream currently has it accessed.
                     708 ;
                     709 ; Side effects:
                     710 ;
                     711 ;       If existing deferred writeback buffer accessed for a locker,
                     712 ;       then the DFW flag in the BLB is cleared.
                     713 ;       AP destroyed.
                     714 ;--
                     715
                     716 SCAN_LOCKS:
          55    D4   717          CLRL    R5                              ; Init BDB return.
50  0098 CA    DE   718          MOVAL   IFB$L_BLBFLNK(R10), R0          ; Save for end test
     54    50   D0   719          MOVL    R0, R4                          ; Get start of list
                     720
                     721          ASSUME  BLB$L_FLNK      EQ      0
                     722
     54    64   D0   723 10$:     MOVL    (R4), R4                        ; Get next BLB
     50    54   D1   724          CMPL    R4, R0                          ; At end of list?
          57    13   725          BEQL    50$                             ; EQL get a free one
  51    14 A4   D1   726          CMPL    BLB$L_VBN(R4), R1               ; Is this the one?
          F2    12   727          BNEQ    10$                             ; NEQ then try next one
                     728
                     729 ;
                     730 ; A BLB for the requested bucket has been located.
```

```
                     01F1    731 ; If this stream already owns it, return with success.
                     01F1    732 ;
                     01F1    733
5C   0C A4    D0    01F1    734          MOVL    BLB$L_BDB_ADDR(R4), AP  ; Pick up BDB/GBPB address.
59   10 A4    D1    01F5    735          CMPL    BLB$L_OWNER(R4), R9     ; Does this stream own it?
           42  13    01F9    736          BEQL    30$                     ; Br if so.
           5C  D5    01FB    737          TSTL    AP                      ; Was there a BDB/GBPB?
           E4  13    01FD    738          BEQL    10$                     ; No, keep looking.
                     01FF    739
                     01FF    740          ASSUME  BDB$B_BID       EQ      GBPB$B_BID
                     01FF    741          ASSUME  <BDB$C_BID & 1> EQ      0
                     01FF    742          ASSUME  <GBPB$C_BID & 1> EQ     1
                     01FF    743
  E0 08 AC    E8    01FF    744          BLBS    GBPB$B_BID(AP), 10$     ; Ignore all other GBPB's.
    55    5C  D0    0203    745          MOVL    AP, R5                  ; Note BDB address.
5C    10 A4    D0    0206    746          MOVL    BLB$L_OWNER(R4), AP     ; Pick up owner, if any.
           2D  13    020A    747          BEQL    20$                     ; No owner, then use it.
     5C    5A  D1    020C    748          CMPL    R10, AP                 ; Is the ifab the owner?
           D2  12    020F    749          BNEQ    10$                     ; If not, keep looking.
                     0211    750
                     0211    751 ;
                     0211    752 ; This is a BLB for the desired bucket, with the ifab as the owner.
                     0211    753 ; It is a deferred write buffer (DFW).  If this stream only wants read
                     0211    754 ; access to the buffer, then simply use this BLB.  No conversion will
                     0211    755 ; be done in that case, which means a blocking AST to write back the
                     0211    756 ; buffer can occur at any time.  This is not a problem because readers
                     0211    757 ; don't modify the buffer.
                     0211    758 ; If the bucket is to be locked, the DFW flag is used to interlock access
                     0211    759 ; to this BLB.  By clearing the flag, the blocking AST is inhibited from
                     0211    760 ; writing back the buffer while it is being modified.  The buffer will
                     0211    761 ; be written back when this access is complete.
                     0211    762 ; If the DFW flag is already clear, it indicates that a write back is
                     0211    763 ; already in progress.  In that case, this thread must be stalled
                     0211    764 ; until the writeback is complete.
                     0211    765 ; This avoids the need to send a blocking AST in the normal multistream case.
                     0211    766 ; Dirty buffers are therefore passed from stream to stream, although
                     0211    767 ; they are not passed from process to process.
                     0211    768 ;
                     0211    769
                     0211    770          ASSUME  CSH$V_LOCK      EQ      0
                     0211    771
     25 53    E9    0211    772          BLBC    R3, 20$                 ; If only a reader, take the buffer.
        05    E4    0214    773          BBSC    #BLB$V_DFW,-            ; Branch unless writeback is
  20 0A A4          0216    774                  BLB$B_BLBFLGS(R4), 20$  ; already in progress.
10 A4    59    D0    0219    775          MOVL    R9, BLB$L_OWNER(R4)     ; Note thread that is stalling.
0A A4    01    88    021D    776          BISB2   #BLB$M_LOCK, BLB$B_BLBFLGS(R4) ; Note that thread is stalling.
   7E    52    7D    0221    777          MOVQ    R2, -(SP)               ; Save registers.
      FDD9'    30    0224    778          BSBW    RM$STALL                ; Wait for writeback to complete.
     0B A9     95    0227    779          TSTB    IRB$B_EFN(R9)           ; DO NOT set efn if zero.
        0A     13    022A    780          BEQL    18$
                     022C    781          $SETEF_S IRB$B_EFN(R9)          ; Set event flag.
   52    8E    7D    0236    782 18$:     MOVQ    (SP)+, R2               ; Restore registers.
                     0239    783 20$:
                     0239    784          RMSSUC                          ; Note success.
              05     023C    785          RSB                             ; Return.
                     023D    786
                     023D    787 ;
```

```
                     023D   788 ; BLB was found which we own.  Note the BDB/GBPB address and return success.
                     023D   789 ;
                     023D   790
55   5C   D0   023D   791 30$:    MOVL    AP, R5                  ; Note BDB/GBPB address.
     F7   11   0240   792          BRB     20$                     ; Return success.
     50   D4   0242   793 50$:    CLRL    R0                      ; Note failure.
          05   0244   794          RSB                             ; Return.
```

```
                     0245   796 ;++
                     0245   797 ;
                     0245   798 ;       GET_BLB
                     0245   799 ;
                     0245   800 ;       Look for free BLB from the end of list to the front.
                     0245   801 ;       A 'free BLB' has a zero vbn field.
                     0245   802 ;
                     0245   803 ; Calling sequence:
                     0245   804 ;
                     0245   805 ;       BSBW    GET_BLB
                     0245   806 ;
                     0245   807 ; Input parameters:
                     0245   808 ;
                     0245   809 ;       R10 -   ifab address
                     0245   810 ;               BLB$L_FLNK - BLB listhead forward link
                     0245   811 ;               BLB$L_BLNK - BLB listhead backward link
                     0245   812 ;
                     0245   813 ;       R9  -   structure which will own lock (ifab/irab)
                     0245   814 ;
                     0245   815 ;       R1  -   VBN to be accessed by bucket lock
                     0245   816 ;
                     0245   817 ; Output parameters:
                     0245   818 ;
                     0245   819 ;       R4  -   address of free BLB
                     0245   820 ;
                     0245   821 ; Side effects:
                     0245   822 ;
                     0245   823 ;       BLB returned in R4 is moved to head of BLB chain.
                     0245   824 ;       AP destroyed.
                     0245   825 ;       Bugcheck if no BLB is available.
                     0245   826 ;
                     0245   827 ;--
                     0245   828
                     0245   829 GET_BLB:
                     0245   830
                     0245   831          ASSUME  BLB$L_BLNK      EQ      <BLB$L_FLNK + 4>
                     0245   832          ASSUME  IFB$L_BLBBLNK   EQ      <IFB$L_BLBFLNK + 4>
                     0245   833
54   0098 CA   DE   0245   834          MOVAL   IFB$L_BLBFLNK(R10), R4  ; Get list head.
     5C   54   D0   024A   835          MOVL    R4, AP                  ; Save for end test.
  54   04 A4   D0   024D   836 100$:    MOVL    4(R4), R4               ; Get back link.
     5C   54   D1   0251   837          CMPL    R4, AP                  ; Back at list head?
          16   13   0254   838          BEQL    110$                    ; If so, then bugcheck
       14 A4   D5   0256   839          TSTL    BLB$L_VBN(R4)           ; This one free?
          F2   12   0259   840          BNEQ    100$                    ; No, move on to next one.
  10 A4   59   D0   025B   841          MOVL    R9, BLB$L_OWNER(R4)     ; Note owner.
  14 A4   51   D0   025F   842          MOVL    R1, BLB$L_VBN(R4)       ; Note resource.
     54   64   0F   0263   843          REMQUE  (R4), R4                ; Remove from current place in chain.
0098 CA   64   0E   0266   844          INSQUE  (R4), IFB$L_BLBFLNK(R10) ; Put in front to find quick.
               05   026B   845          RSB                             ; Return.
                    026C   846 110$:
                    026C   847          RMSPBUG FTL$_NOBLB              ; Should always find one.
```

```
                     0273   849 ;++
                     0273   850 ;
                     0273   851 ;       RM$FREE_LCL
                     0273   852 ;
                     0273   853 ; Toss the oldest least valuable buffer out of the cache.  This
                     0273   854 ; routine is called when the AVLCL count is less than zero, meaning
                     0273   855 ; that all BLB's are being used either for caching or access of a
                     0273   856 ; bucket.  It does not necessarily mean that there are no BDB's free
                     0273   857 ; because multiple BLB's may be tied up referencing a single BDB.
                     0273   858 ; Nonetheless, a BDB must be tossed out of the cache to free up its
                     0273   859 ; associated BLB.
                     0273   860 ;
                     0273   861 ; Calling sequence:
                     0273   862 ;
                     0273   863 ;       BSB     RM$FREE_LCL
                     0273   864 ;
                     0273   865 ; Input Parameters:
                     0273   866 ;
                     0273   867 ;       R10 -   Ifab address
                     0273   868 ;           IFB$L_BDB_BLNK - back link of BDB listhead.
                     0273   869 ;       R9 - Thread to stall (if necessary)
                     0273   870 ;
                     0273   871 ; Output Parameters:
                     0273   872 ;
                     0273   873 ;       R0  -   status value from call to RM$RELEASE
                     0273   874 ;               DME - all valid BDB's were in use.
                     0273   875 ;
                     0273   876 ; Side effects:
                     0273   877 ;
                     0273   878 ;       AP destroyed.
                     0273   879 ;
                     0273   880 ;--
                     0273   881
                     0273   882 RM$FREE_LCL::
        1E   BB      0273   883         PUSHR   #^M<R1,R2,R3,R4>        ; Save registers.
        54   D4      0275   884         CLRL    R4                      ; Init last BDB seen.
   51   01   CE      0277   885         MNEGL   #1, R1                  ; Init last cache value.
                     027A   886                                         ; This should always cause the first
                     027A   887                                         ; BDB possible to be selected.
                     027A   888         ASSUME  BDB$L_FLINK     EQ      0
                     027A   889         ASSUME  BDB$L_BLINK     EQ      <BDB$L_FLINK + 4>
                     027A   890
50   40   AA   DE      027A   891         MOVAL   IFB$L_BDB_FLNK(R10), R0 ; Get list head into R0.
   5C   50   D0      027E   892         MOVL    R0, AP                  ; Save for end test.
                     0281   893 10$:
50   04   A0   D0      0281   894         MOVL    4(R0), R0               ; Get next BDB.
   5C   50   D1      0285   895         CMPL    R0, AP                  ; End of chain?
        1F   13      0288   896         BEQL    20$                     ; EQL means at end.
                     028A   897
                     028A   898         ASSUME  BDB$B_BID       EQ      GBPB$B_BID
                     028A   899         ASSUME  <BDB$B_BID & 1> EQ      0
                     028A   900         ASSUME  <GBPB$B_BID & 1> EQ     1
                     028A   901
F3 08   A0   E8      028A   902         BLBS    BDB$B_BID(R0), 10$      ; Continue scan if gbpb.
   0C   A0   B5      028F   903         TSTW    BDB$W_USERS(R0)         ; Is this one accessed?
        EE   12      0291   904         BNEQ    10$                     ; NEQ it is - keep going.
   1C   A0   D5      0293   905         TSTL    BDB$L_VBN(R0)           ; See if valid.
```

```
                E9  13  0296   906          BEQL    10$                             ; Continue search for valid BDB.
          51    0B A0   91  0298   907          CMPB    BDB$B_CACHE_VAL(R0), R1 ; Less valuable than what has already
                        029C   908                                                  ; been seen?
                E3  1E  029C   909          BGEQU   10$                             ; GEQU, then just keep looking.
             54 50  D0  029E   910          MOVL    R0, R4                          ; Save this BDB address.
          51    0B A0   9A  02A1   911          MOVZBL  BDB$B_CACHE_VAL(R0), R1 ; Save this as lowest seen.
                DA  12  02A5   912          BNEQ    10$                             ; If non-zero, keep trying.
                3D  11  02A7   913          BRB     50$                             ; Use the first BDB with zero value.
                        02A9   914
                        02A9   915 ;
                        02A9   916 ; At end of list.  If anything was noted, then use it.
                        02A9   917 ; If no BDB's can be tossed out, scan the BLB list for a deferred write
                        02A9   918 ; BLB, and DEQ that lock.
                        02A9   919 ;
                        02A9   920
                54  D5  02A9   921 20$:     TSTL    R4                              ; Did we find a BDB?
                39  12  02AB   922          BNEQ    50$                             ; Go free it, if one found.
          07 6A 33  E1  02AD   923          BBC     #IFB$V_NORECLK,(R10),25$; None found, check BLBs if locking.
                        02B1   924          RMSPBUG FTL$_NOLCLBUF                   ; We should have found a BDB, The
                        02B8   925                                                  ; AVLCL count is probably wrong.
                        02B8   926
                        02B8   927          ASSUME  BLB$L_FLNK      EQ      0
                        02B8   928          ASSUME  BLB$L_BLNK      EQ      <BLB$L_FLNK + 4>
                        02B8   929          ASSUME  IFB$L_BLBBLNK   EQ      <IFB$L_BLBFLNK + 4>
                        02B8   930
       52   0098 CA  DE  02B8   931 25$:     MOVAL   IFB$L_BLBFLNK(R10), R2  ; Address of listhead.
             5C 52  D0  02BD   932          MOVL    R2, AP                          ; Save for end test.
          52    04 A2  D0  02C0   933 30$:     MOVL    4(R2), R2               ; Get next BLB (going backwards).
             5C 52  D1  02C4   934          CMPL    R2, AP                          ; Done yet?
                16  13  02C7   935          BEQL    40$                             ; EQL then at end of list.
       5A    10 A2  D1  02C9   936          CMPL    BLB$L_OWNER(R2), R10    ; Does ifab own this lock?
                F1  12  02CD   937          BNEQ    30$                             ; No, check the next one.
                05  E0  02CF   938          BBS     #BLB$V_DFW,-                    ; Found one, now make sure this
          05 0A A2        02D1   939                  BLB$B_BLBFLGS(R2),35$   ; is really a deferred write lock
                06  E1  02D4   940          BBC     #BLB$V_WRITEBACK,-      ; and not the IFAB from a $OPEN
          E7 0A A2        02D6   941                  BLB$B_BLBFLGS(R2),30$   ; command.
                        02D9   942
                        02D9   943
                        02D9   944 ;
                        02D9   945 ; Have found a lock with the ifab as the owner.  This is a deferred
                        02D9   946 ; write lock.  Get the BDB address into R4 and branch into code to
                        02D9   947 ; release this lock and free the BLB.
                        02D9   948 ;
                        02D9   949
       54   0C A2  D0  02D9   950 35$:     MOVL    BLB$L_BDB_ADDR(R2), R4  ; Want BDB address in R4.
                12  11  02DD   951          BRB     55$                             ; And go release it.
                        02DF   952 40$:
                        02DF   953          RMSERR  DME                             ; Couldn't find anything to release.
                3C  11  02E4   954          BRB     80$                             ; Branch to exit.
                        02E6   955
                        02E6   956 50$:
                        02E6   957
                        02E6   958 ;
                        02E6   959 ; A BDB has been selected to toss out of the cache.  Release it, forcing
                        02E6   960 ; write-thru if dirty.  The RLS$M_DEQ flag causes the buffer to be
                        02E6   961 ; completely released and made available.
                        02E6   962 ;
```

```
                        02E6   963 ; Check now to see if a writeback is already in progress on this
                        02E6   964 ; BLB.  If so, then stall until it is complete.
                        02E6   965 ; At this point, we know there are no other accessors to this bucket,
                        02E6   966 ; and therefore the BLB_PTR must point to the deferred write
                        02E6   967 ; BLB if there is one.
                        02E6   968 ;
                        02E6   969
      52   10 A4   D0   02E6   970          MOVL    BDB$L_BLB_PTR(R4), R2   ; Get the BLB address.
             24   13   02EA   971          BEQL    60$                     ; No locking, just dequeue then.
           10 A2   D5   02EC   972          TSTL    BLB$L_OWNER(R2)         ; Any owner for this lock?
             1F   13   02EF   973          BEQL    60$                     ; EQL then it's not dirty.
   10 A2     59   D0   02F1   974 55$:     MOVL    R9, BLB$L_OWNER(R2)     ; Note thread that is stalling.
             05   E4   02F5   975          BBSC    #BLB$V_DFQ,-            ; Branch if writeback has not
     16 0A A2           02F7   976                  BLB$B_BLBFLGS(R2), 60$  ; started and claim this BLB.
   0A A2     01   88   02FA   977          BISB2   #BLB$M_LOCK, BLB$B_BLBFLGS(R2) ; Note this thread is stalled.
         FCFF'  30   02FE   978          BSBW    RM$STALL                ; Stall until writeback complete.
        0B A9   95   0301   979          TSTB    IRB$B_EFN(R9)           ; DO NOT set efn if zero.
             0A   13   0304   980          BEQL    60$
                       0306   981          $SETEF_S IRB$B_EFN(R9)          ; Set event flag.
                       0310   982 60$:
             06   E5   0310   983          BBCC    #BDB$V_AST_DCL,-        ; Branch if no writeback has been done.
     04 0A A4           0312   984                  BDB$B_FLGS(R4), 70$
     0C 48 A4   E9   0315   985          BLBC    BDB$L_IOSB(R4), 90$     ; Branch if an error has occured.
      53   08   D0   0319   986 70$:     MOVL    #RLS$M_DEQ, R3          ; Init release flags to release lock.
                       031C   987                                          ; DEQ flag implies write thru if dirty.
        0C A4   B6   031C   988          INCW    BDB$W_USERS(R4)         ; Make it look accessed.
         FCDE'  30   031F   989          BSBW    RM$RELEASE              ; And release it.
             1E   BA   0322   990 80$:     POPR    #^M<R1,R2,R3,R4>        ; Restore registers.
             05   0324   991          RSB                             ; Return.
                       0325   992 90$:
OC A8   48 A4   3C   0325   993          MOVZWL  BDB$L_IOSB(R4), RAB$L_STV(R8) ; Store i/o error code.
                       032A   994          RMSERR  WER                     ; Note error with RMS code.
         F1   11   032F   995          BRB     80$                     ; and return.
```

```
                     0331   997 ;++
                     0331   998 ;
                     0331   999 ;       SCAN_LCL_CACHE
                     0331  1000 ;
                     0331  1001 ;       Search list of all BDB's for match on the VBN field.
                     0331  1002 ;
                     0331  1003 ; Calling sequence:
                     0331  1004 ;
                     0331  1005 ;       BSB     SCAN_LCL_CACHE
                     0331  1006 ;
                     0331  1007 ; Input Parameters:
                     0331  1008 ;
                     0331  1009 ;       R10 - ifab address
                     0331  1010 ;           IFB$L_BDB_FLNK - forward link of BDB listhead.
                     0331  1011 ;
                     0331  1012 ;       R1 - VBN of bucket desired
                     0331  1013 ;
                     0331  1014 ; Output Parameters:
                     0331  1015 ;
                     0331  1016 ;       R5 - address of BDB that matches desired VBN or 0 if not found
                     0331  1017 ;--
                     0331  1018
                     0331  1019 SCAN_LCL_CACHE:
55   40 AA   DE      0331  1020         MOVAL   IFB$L_BDB_FLNK(R10), R5 ; Address of BDB listhead.
   5C  55    D0      0335  1021         MOVL    R5, AP                  ; Save for end test.
                     0338  1022
                     0338  1023         ASSUME  BDB$L_FLINK     EQ      0
                     0338  1024
   55  65    D0      0338  1025 10$:    MOVL    (R5), R5                ; Get next BDB.
   5C  55    D1      033B  1026         CMPL    R5, AP                  ; At end of list yet?
       0B    13      033E  1027         BEQL    50$                     ; EQL yes, quit.
51   1C A5   D1      0340  1028         CMPL    BDB$L_VBN(R5), R1       ; Is this the right one?
       F2    12      0344  1029         BNEQ    10$                     ; No, try next one.
                     0346  1030
                     0346  1031         ASSUME  <BDB$C_BID & 1> EQ      0
                     0346  1032         ASSUME  <GBPB$C_BID & 1> EQ     1
                     0346  1033
EE 08 A5   E8        0346  1034         BLBS    BDB$B_BID(R5), 10$      ; Not a BDB.  Continue search.
       05    034A  1035 20$:    RSB                                     ; Return.
   55  D4    034B  1036 50$:    CLRL    R5                              ; Note failure.
       05    034D  1037         RSB                                     ; Return.
```

```
                     034E  1039 ;++
                     034E  1040 ;
                     034E  1041 ;     RM$GET_LCL_BUFF
                     034E  1042 ;
                     034E  1043 ;       Search list of BDB's from the end of list to the front for
                     034E  1044 ;       one with VBN=0, USERS=0, and SIZE large enough.
                     034E  1045 ;
                     034E  1046 ; Calling sequence:
                     034E  1047 ;
                     034E  1048 ;       BSBW    RM$GET_LCL_BUFF
                     034E  1049 ;
                     034E  1050 ; Input Parameters:
                     034E  1051 ;
                     034E  1052 ;       R1 - VBN desired
                     034E  1053 ;       R2 - Size of bucket
                     034E  1054 ;       R.0 - Ifab address
                     034E  1055 ;         BDB_BLNK - Back BDB link
                     034E  1056 ;
                     034E  1057 ; Output Parameters:
                     034E  1058 ;
                     034E  1059 ;       R5 - BDB address
                     034E  1060 ;         VBN set to R1
                     034E  1061 ;         NUMB set to R2
                     034E  1062 ;
                     034E  1063 ; Side effects:
                     034E  1064 ;
                     034E  1065 ;       NOLCLBUF bugcheck if no BDB is found.
                     034E  1066 ;
                     034E  1067 ;--
                     034E  1068
                     034E  1069 RM$GET_LCL_BUFF::
                     034E  1070
                     034E  1071          ASSUME  BDB$L_FLINK     EQ      0
                     034E  1072          ASSUME  BDB$L_BLINK     EQ      <BDB$L_FLINK + 4>
                     034E  1073          ASSUME  IFB$L_BDB_BLNK  EQ      <IFB$L_BDB_FLNK + 4>
                     034E  1074
55   40 AA    DE     034E  1075          MOVAL   IFB$L_BDB_FLNK(R10), R5 ; Get BDB list head.
   5C   55    D0     0352  1076          MOVL    R5, AP                  ; Save for end test.
                     0355  1077 10$:
55   04 A5    D0     0355  1078          MOVL    4(R5), R5               ; Get next BDB.
   5C   55    D1     0359  1079          CMPL    R5, AP                  ; At end of list?
        28    13     035C  1080          BEQL    100$                    ; Bug if none found in list.
     1C A5    D5     035E  1081          TSTL    BDB$L_VBN(R5)           ; This buffer free?
        F2    12     0361  1082          BNEQ    10$                     ; NEQ it's in use.  Continue search.
     0C A5    B5     0363  1083          TSTW    BDB$W_USERS(R5)         ; In use?
        ED    12     0366  1084          BNEQ    10$                     ; NEQ in use, try another.
                     0368  1085
                     0368  1086          ASSUME  <BDB$C_BID & 1> EQ      0
                     0368  1087          ASSUME  <GBPB$C_BID & 1> EQ     1
                     0368  1088
  E9 08 A5    E8     0368  1089          BLBS    BDB$B_BID(R5), 10$      ; It's not a BDB.  Continue search.
52   16 A5    B1     036C  1090          CMPW    BDB$W_SIZE(R5), R2      ; Is this buffer large enough?
        E3    1F     0370  1091          BLSSU   10$                     ; LSSU not big enough - try another.
                     0372  1092
     10 A5    D4     0372  1093          CLRL    BDB$L_BLB_PTR(R5)       ; Make sure this is 0.
1C A5    51    D0     0375  1094          MOVL    R1, BDB$L_VBN(R5)       ; Found one- store VBN.
14 A5    52    B0     0379  1095          MOVW    R2, BDB$W_NUMB(R5)      ; Store size desired.
```

```
04 6A   31   E1  037D  1096          BBC     #IFB$V_MSE, (R10), 20$  ; Branch if not multistreaming.
0A A5   10   88  0381  1097          BISB2   #BDB$M_NOLOCATE, BDB$B_FLGS(R5) ; Don't allow locate mode.
             05  0385  1098 20$:      RSB
                 0386  1099
                 0386  1100 100$:     RMSPBUG FTL$_NOLCLBUF           ; Should always find one.
```

```
038D  1102 ;++
038D  1103 ;
038D  1104 ;       LOCK_BKT
038D  1105 ;
038D  1106 ;       This routine is called to obtain a lock for the requested bucket
038D  1107 ;       for either read only or modify.  In addition, it may also be known
038D  1108 ;       that the lock must be of a high enough mode to interlock an i/o
038D  1109 ;       operation.  The following lock manager modes are used:
038D  1110 ;
038D  1111 ;       LCK$K_PWMODE if holding modified buffer.  No conversion will
038D  1112 ;            be done if this is not a LOCK request to cache.
038D  1113 ;       LCK$K_EXMODE for all locks.
038D  1114 ;
038D  1115 ; Calling sequence:
038D  1116 ;
038D  1117 ;       BSBW    LOCK_BKT
038D  1118 ;
038D  1119 ; Input Parameters:
038D  1120 ;
038D  1121 ;       R10 -   ifab address
038D  1122 ;           IFB$L_SFSB_PTR - pointer to shared file synchronization block
038D  1123 ;               SFSB$L_LOCK_ID - lock id of shared file lock
038D  1124 ;
038D  1125 ;       R9  -   ifab/irab address
038D  1126 ;           IRB$B_EFN - event flag to use.
038D  1127 ;
038D  1128 ;       R4  -   BLB address
038D  1129 ;           BLB$B_MODEHELD - mode of lock currently held.
038D  1130 ;           BLB$L_LOCK_ID - ID of lock if one already held
038D  1131 ;           BLB$L_RESDSC - resource name descriptor (should point to BLB$L_VBN)
038D  1132 ;
038D  1133 ;       R3  -   Cache flags (same as input to RM$CACHE).
038D  1134 ;
038D  1135 ; Output Parameters:
038D  1136 ;
038D  1137 ;       R0  -   status value of $ENQ service after call to RM$MAPERR.
038D  1138 ;
038D  1139 ; Side effects:
038D  1140 ;
038D  1141 ;       BLB$L_LKSTS field contains value of ENQ service.
038D  1142 ;       BLB$L_VALBLK contains value block for lock requested.
038D  1143 ;       BLB$L_OWNER is set to R9.
038D  1144 ;       BLB$B_MODEHELD contains mode of lock obtained.
038D  1145 ;       BLB$B_BLBFLGS contain the cache flags also.
038D  1146 ;
038D  1147 ;       R1 is always destroyed.
038D  1148 ;       If lock not granted synchonously, will return at AST level with
038D  1149 ;       event flag (IRB$B_EFN) set and AP, and R2 will be destroyed.
038D  1150 ;
038D  1151 ;--
038D  1152
038D  1153 LOCK_BKT:
038D  1154
038D  1155          ASSUME  CSH$V_LOCK      EQ      BLB$V_LOCK
038D  1156          ASSUME  CSH$V_NOWAIT    EQ      BLB$V_NOWAIT
038D  1157          ASSUME  CSH$V_NOREAD    EQ      BLB$V_NOREAD
038D  1158          ASSUME  CSH$V_NOBUFFFR  EQ      BLB$V_NOBUFFER
```

```
                     038D  1159
10 A4    59    D0    038D  1160          MOVL    R9, BLB$L_OWNER(R4)      ; Note owning stream.
53   F0 8F     8A    0391  1161          BICB2   #^CCSH_MASK_ALL, R3      ; Clear out all but csh flags
0A A4    0F    8A    0395  1162          BICB2   #CSH_MASK_ALL, BLB$B_BLBFLGS(R4) ; Clear out csh flags
0A A4    53    88    0399  1163          BISB2   R3, BLB$B_BLBFLGS(R4)    ; Store csh flags in blb flags field.
                     039D  1164
                     039D  1165 ;
                     039D  1166 ; The lock may currently be held in NL, PW or EX mode.
                     039D  1167 ; NL mode is used to cache local buffers.
                     039D  1168 ; PW mode is used when a modified buffer is held locally.
                     039D  1169 ; EX mode is normally used when a bucket is accessed.
                     039D  1170 ; The exception is that if a PW lock is already held and only a read
                     039D  1171 ; lock on the bucket is desired, no conversion is necessary.
                     039D  1172 ; PW is converted to EX for modify access to eliminate the possibility of
                     039D  1173 ; a blocking AST arriving during the access.
                     039D  1174 ;
                     039D  1175          ASSUME  CSH$V_LOCK EQ 0
                     039D  1176
  51    04    D0     039D  1177          MOVL    #LCK$K_PWMODE, R1        ; Assume read lock desired.
03 0A A4    E9     03A0  1178          BLBC    BLB$B_BLBFLGS(R4), 10$   ; Branch if write lock not wanted.
  51    05    D0     03A4  1179          MOVL    #LCK$K_EXMODE, R1        ; Get exclusive lock.
51   0B A4    91     03A7  1180 10$:     CMPB    BLB$B_MODEHELD(R4), R1   ; Is high enough mode held already?
        04    1F     03AB  1181          BLSSU   30$                      ; LSSU then continue
                     03AD  1182          RMSSUC                           ; return success.
              05     03B0  1183          RSB
                     03B1  1184 ;
                     03B1  1185 ; Build FLAG list for ENQ.
                     03B1  1186 ;
        53    DD     03B1  1187 30$:     PUSHL   R3                       ; save cache flags
     53   19    D0     03B3  1188          MOVL    #LCK$M_SYSTEM!LCK$M_SYNCSTS!LCK$M_VALBLK,R3 ; Always use these flags
       24 A4    D5     03B6  1189          TSTL    BLB$L_LOCK_ID(R4)        ; Is lock already held?
        03    13     03B9  1190          BEQL    31$                      ; No, go on
     53   02    C8     03BB  1191          BISL2   #LCK$M_CONVERT,R3        ; Add conversion flag to our list.
03 0A A4    01    E1     03BE  1192 31$:     BBC     #BLB$V_NOWAIT,BLB$B_BLBFLGS(R4),32$ ; Br if wait desired.
     53   04    C8     03C3  1193          BISL2   #LCK$M_NOQUEUE,R3        ; Do not wait for this lock.
                     03C6  1194
     0080 CA    D5     03C6  1195 32$:     TSTL    IFB$L_PAR_LOCK_ID(R10)   ; Make sure parent lock is present.
        07    12     03CA  1196          BNEQ    35$
                     03CC  1197          RMSPBUG FTL$_NOSFSB              ; No, we are in trouble!
                     03D3  1198
                     03D3  1199 ;
                     03D3  1200 ; Do the ENQ for the bucket.
                     03D3  1201 ;
                     03D3  1202
        FC2A'  30    03D3  1203 35$:     BSBW    RM$SETEFN                ; Get an event flag
          50 8ED0    03D6  1204          POPL    R0                       ; Put it in R0.
                     03D9  1205          $ENQ_S  EFN     = R0,-
                     03D9  1206                  LKMODE  = #LCK$K_EXMODE,-
                     03D9  1207                  LKSB    = BLB$W_LKSTS(R4),-
                     03D9  1208                  FLAGS   = R3,-
                     03D9  1209                  RESNAM  = BLB$L_RESDSC(R4),-
                     03D9  1210                  PARID   = IFB$L_PAR_LOCK_ID(R10),-
                     03D9  1211                  ASTADR  = W^RM$STALLAST,-
                     03D9  1212                  ASTPRM  = R9
                     03FA  1213
       1E 50    E9    03FA  1214          BLBC    R0, 110$                 ; Exit on error.
50   0689 8F    B1    03FD  1215          CMPW    #SS$_SYNCH, R0           ; Need to stall?
```

```
                08   13  0402  1216          BEQL    70$                             ; EQL all done.
                53   DD  0404  1217 50$:     PUSHL   R3                              ; Save ENQ flags around stall.
              FBF7'  30  0406  1218          BSBW    RMS$STALL                       ; Stall for lock.
                53 8ED0  0409  1219          POPL    R3                              ; Restore ENQ flags.
          50    20 A4    3C  040C  1220 70$:     MOVZWL  BLB$W_LKSTS(R4), R0             ; Get completion status into R0.
                08 50    E9  0410  1221          BLBC    R0, 1T0$                        ; Branch on error.
          0B A4    05    90  0413  1222 80$:     MOVB    #LCK$K_EXMODE, BLB$B_MODEHELD(R4); Store lock mode in blb.
                53 8ED0  0417  1223 90$:     POPL    R3                              ; restore cache flags
                   05    041A  1224          RSB                                     ; Return.
                         041B  1225
                         041B  1226 ;
                         041B  1227 ; An error occured on the ENQ service.  Deadlock errors may occur
                         041B  1228 ; normally because of the deferred write mechanism and are simply
                         041B  1229 ; requeued.
                         041B  1230 ;
                         041B  1231
          50    0E0A 8F    B1  041B  1232 110$:    CMPW    #SS$_DEADLOCK, R0               ; Was it deadlock?
                   B1    13  0420  1233          BEQL    35$                             ; Try it again if it was.
          50    2A44 8F    B1  0422  1234          CMPW    #SS$_EXENQLM, R0                ; Exceed our lock quota?
                   23    13  0427  1235          BEQL    150$                            ; Br if yes.
          50    0E12 8F    B1  0429  1236          CMPW    #SS$_NOLOCKID, R0               ; Lock id table full?
                   1C    13  042E  1237          BEQL    150$                            ; Br if yes.
          50    09F0 8F    B1  0430  1238          CMPW    #SS$_VALNOTVALID,R0             ; Has lock manager returned old value block?
                   0B    12  0435  1239          BNEQ    120$                            ; No, report the error.
                28 A4    D6  0437  1240          INCL    BLB$L_VALSEQNO(R4)              ; Bump sequence number on bucket to get new
                         043A  1241          RMSSUC                                  ; Consider this ENQ successful.
                   D4    11  043D  1242          BRB     80$                             ; Branch to finish up.
                   50 8ED0  043F  1243 115$:    POPL    R0                              ; Return with original error.
                         0442  1244 120$:    RMSERR  ENQ, R1                         ; Default error code.
                FBB6'  30  0447  1245          BSBW    RMS$MAPERR                      ; Try to map error.
                   CB    11  044A  1246          BRB     90$                             ; Return.
                         044C  1247
                         044C  1248 ;
                         044C  1249 ; This is either a case where we've exceeded our lock quota or the system
                         044C  1250 ; lock id table is full.  It may be the case that something can be tossed
                         044C  1251 ; out of our local cache and thereby free up a lock to continue with this
                         044C  1252 ; operation.  If that fails, return with the original error.  Otherwise,
                         044C  1253 ; requeue the lock request.
                         044C  1254 ;
                         044C  1255
            50    DD     044C  1256 150$:    PUSHL   R0                              ; Save this error code.
          FE22    30     044E  1257          BSBW    RMS$FREE_LCL                    ; Try to free a buffer.
        EB 50     E9     0451  1258          BLBC    R0, 115$                        ; Branch on failure.
            50 8ED0      0454  1259          POPL    R0                              ; Pop error off stack.
          FF79    31     0457  1260          BRW     35$                             ; Go try request again.
```

```
                     045A  1262 ;++
                     045A  1263 ;
                     045A  1264 ; SCAN_GBL
                     045A  1265 ;
                     045A  1266 ; Scan global cache for match on desired bucket.
                     045A  1267 ;
                     045A  1268 ; Calling sequence:
                     045A  1269 ;
                     045A  1270 ;       BSBW    SCAN_GBL
                     045A  1271 ;
                     045A  1272 ; Input parameters:
                     045A  1273 ;
                     045A  1274 ;       R10 - ifab address
                     045A  1275 ;         GBH_PTR - pointer to global buffer header area
                     045A  1276 ;
                     045A  1277 ;       R1 - vbn of desired bucket
                     045A  1278 ;
                     045A  1279 ; Output parameters:
                     045A  1280 ;
                     045A  1281 ;       R0 - success
                     045A  1282 ;       R5 - GBPB address
                     045A  1283 ; else
                     045A  1284 ;       R0 - failure (0)
                     045A  1285 ;       R5 - addr of next lower valued GBD or zero if no GBPB found.
                     045A  1286 ;
                     045A  1287 ; Side effects:
                     045A  1288 ;       AP destroyed.
                     045A  1289 ;       EXclusive lock on global buffer section is obtained and kept.
                     045A  1290 ;
                     045A  1291 ;--
                     045A  1292
                     045A  1293 SCAN_GBL:
                     045A  1294          ASSUME  GBH$L_GBD_FLNK  EQ      0
                     045A  1295          ASSUME  GBH$L_GBD_BLNK  EQ      4
                     045A  1296          ASSUME  GBD$L_FLINK     EQ      0
                     045A  1297          ASSUME  GBD$L_BLINK     EQ      4
                     045A  1298          ASSUME  GBD$L_VBN       EQ      GBH$L_HI_VBN
                     045A  1299
        FBA3'  30    045A  1300          BSBW    RM$RAISE_GBS_LOCK       ; Get EX lock on GBS to search cache.
50  0088 CA  D0      045D  1301          MOVL    IFB$L_GBH_PTR(R10), R0  ; Get pointer to gbl header.
    5C   50  D0      0462  1302          MOVL    R0, AP                  ; Save for later.
                     0465  1303
    50   60  C0      0465  1304 10$:     ADDL2   (R0), R0                ; Get address of next GBD element.
  51   0C  A0  D1    0468  1305          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
           2D  1E    046C  1306          BGEQU   20$                     ; GEQU either found it or not here.
    50   60  C0      046E  1307          ADDL2   (R0), R0                ; Get address of next GBD element.
  51   0C  A0  D1    0471  1308          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
           24  1E    0475  1309          BGEQU   20$                     ; GEQU either found it or not here.
    50   60  C0      0477  1310          ADDL2   (R0), R0                ; Get address of next GBD element.
  51   0C  A0  D1    047A  1311          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
           1B  1E    047E  1312          BGEQU   20$                     ; GEQU either found it or not here.
    50   60  C0      0480  1313          ADDL2   (R0), R0                ; Get address of next GBD element.
  51   0C  A0  D1    0483  1314          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
           12  1E    0487  1315          BGEQU   20$                     ; GEQU either found it or not here.
    50   60  C0      0489  1316          ADDL2   (R0), R0                ; Get address of next GBD element.
  51   0C  A0  D1    048C  1317          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
           09  1E    0490  1318          BGEQU   20$                     ; GEQU either found it or not here.
```

```
                50   60    C0  0492  1319          ADDL2   (R0), R0                    ; Get address of next GBD element.
              51  0C A0    D1  0495  1320          CMPL    GBD$L_VBN(R0), R1           ; Is this one desired bucket?
                  CA       1F  0499  1321          BLSSU   10$                         ; LSSU then keep scanning.
                  0B       13  049B  1322 20$:     BEQL    30$                         ; EQL we found it - finish up.
                               049D  1323
                  3C AC    D6  049D  1324          INCL    GBH$L_MISS(AP)              ; Note cache miss.
           55  50  04 A0    C1  04A0  1325          ADDL3   4(R0), R0, R5               ; Get addr of previous element.
                  50       D4  04A5  1326          CLRL    R0                          ; Note failure.
                           05  04A7  1327          RSB                                 ;
                               04A8  1328 30$:
                  38 AC    D6  04A8  1329          INCL    GBH$L_HIT(AP)               ; Note cache hit and drop through.
```

```
                        04AB  1331 ;++
                        04AB  1332 ;
                        04AB  1333 ; GET_GBPB
                        04AB  1334 ;
                        04AB  1335 ; Branch or drop through to this point after locating GBD to get a
                        04AB  1336 ; GBPB and point it to GBD.
                        04AB  1337 ;
                        04AB  1338 ; Input parameters:
                        04AB  1339 ;       R10 - ifab
                        04AB  1340 ;         BDB_FLNK - BDB (and GBPB) listhead.
                        04AB  1341 ;       R0 - GBD address
                        04AB  1342 ;
                        04AB  1343 ; Output parameters:
                        04AB  1344 ; If    R0 - success
                        04AB  1345 ; then  R5 - GBPB address
                        04AB  1346 ; else  R5 = 0 if GBPB not found.
                        04AB  1347 ;
                        04AB  1348 ;--
                        04AB  1349 GET_GBPB:
          20 A0   B6    04AB  1350          INCW    GBD$W_USECNT(R0)          ; Bump use count in GBD (we are assuming
                        04AE  1351                                            ;  that a GBPB will be found).
             50   DD    04AE  1352          PUSHL   R0                        ; Save GBD around lock manager call.
           FB4D'  30    04B0  1353          BSBW    RM$LOWER_GBS_LOCK         ; Lower lock on GBS to NL.
             50 8ED0    04B3  1354          POPL    R0                        ; Restore GBD.
                        04B6  1355
       55   40 AA   DE  04B6  1356          MOVAL   IFB$L_BDB_FLNK(R10), R5   ; BDB/GBPB listhead.
          5C   55   D0  04BA  1357          MOVL    R5, AP                    ; Save for end test.
                        04BD  1358 10$:
                        04BD  1359          ASSUME  BDB$L_FLINK     EQ      0
                        04BD  1360          ASSUME  BDB$L_BLINK     EQ      4
                        04BD  1361
          55   04 A5   D0  04BD  1362          MOVL    4(R5), R5                 ; Get next element.(going backwards)
             5C   55   D1  04C1  1363          CMPL    R5, AP                    ; Back to listhead yet?
                  2F   13  04C4  1364          BEQL    100$                      ; That's a bug.
                        04C6  1365
                        04C6  1366          ASSUME  BDB$B_BID       EQ      GBPB$B_BID
                        04C6  1367          ASSUME  <BDB$B_BID & 1> EQ      0
                        04C6  1368          ASSUME  <GBPB$B_BID & 1> EQ     1
                        04C6  1369
            F3 08 A5    E9  04C6  1370          BLBC    BDB$B_BID(R5), 10$        ; Branch if BDB and keep looking.
               0C A5    B5  04CA  1371          TSTW    GBPB$W_USERS(R5)          ; Make sure it's not in use.
                  EE    12  04CD  1372          BNEQ    10$                       ; NEQ it is, so keep looking.
                        04CF  1373 ;
                        04CF  1374 ; Found the GBPB.  Fill in relevant data from GBD.
                        04CF  1375 ;
             24 A5   50   D0  04CF  1376          MOVL    R0, GBPB$L_GBD_PTR(R5)    ; Pointer to GBD.
          1C A5    0C A0   D0  04D3  1377          MOVL    GBD$L_VBN(R0), GBPB$L_VBN(R5) ; Bucket vbn.
                   10 A0   D0  04D8  1378          MOVL    GBD$L_VBNSEQNUM(R0),-     ; Move sequence number from GBD to GBPB.
                   20 A5       04DB  1379                  GBPB$L_VBNSEQNO(R5)
                               04DD  1380
                               04DD  1381          ASSUME  <GBD$W_NUMB + 2>        EQ      GBD$W_SIZE
                               04DD  1382          ASSUME  <GBPB$W_NUMB + 2>       EQ      GBPB$W_SIZE
                               04DD  1383
          14 A5    18 A0   D0  04DD  1384          MOVL    GBD$W_NUMB(R0), GBPB$W_NUMB(R5) ; Numb and size fields.
                      11   90  04E2  1385          MOVB    #BDB$M_VAL!BDB$M_NOLOCATE,- ; Init flags.
                   0A A5       04E4  1386                  GBPB$B_FLGS(R5)           ;
           5C   0088 CA   D0  04E6  1387          MOVL    IFB$L_GBH_PTR(R10), AP    ; Get pointer to gbl header.
```

```
                18 A5   5C   1C A0   C1   04EB  1388          ADDL3   GBD$L_REL_ADDR(R0), AP, GBPB$L_ADDR(R5) ; Addr of buffer.
                                      04F1  1389          RMSSUC                                  ; Note success.
                                 05   04F4  1390          RSB                                     ;
                                      04F5  1391 100$:
                           20 A0   B7   04F5  1392          DECW    GBD$W_USECNT(R0)                ; Decrement use count for GBD.
                              50   D4   04F8  1393          CLRL    R0                              ; Note failure.
                              55   D4   04FA  1394          CLRL    R5                              ; Note no gbpb.
                                   05   04FC  1395          RSB                                     ;
```

```
                     04FD  1397 ;++
                     04FD  1398 ;
                     04FD  1399 ; FIND_FREE_GBL
                     04FD  1400 ;
                     04FD  1401 ; This routine is called when the desired bucket was not in the
                     04FD  1402 ; global cache.  Either find a free global buffer or toss one out
                     04FD  1403 ; of the cache to use.
                     04FD  1404 ;
                     04FD  1405 ; We already have an exclusive lock on the GBS from calling SCAN_GBL.
                     04FD  1406 ;
                     04FD  1407 ; Input parameters:
                     04FD  1408 ;       R5 - Address of preceding GBD.
                     04FD  1409 ;
                     04FD  1410 ; Output parameters:
                     04FD  1411 ;
                     04FD  1412 ; R0 - success, failure
                     04FD  1413 ; R5 - GBPB addr if success.
                     04FD  1414 ;    - 0 if failure.
                     04FD  1415 ;
                     04FD  1416 ;--
                     04FD  1417 FIND_FREE_GBL:
           3C   BB   04FD  1418         PUSHR   #^M<R2,R3,R4,R5>
           53   7C   04FF  1419         CLRQ    R3                      ; Init lowest val, gbpb found.
           53   D7   0501  1420         DECL    R3                      ; Make lowval -1.
      5C 0088 CA   D0   0503  1421         MOVL    IFB$L_GBH_PTR(R10), AP  ; Get gb' header.
   50 5C   04 AC   C1   0508  1422         ADDL3   GBH$L_GBD_BLNK(AP), AP, R0 ; Get last GBD in list.
      53   0C A0   D1   050D  1423         CMPL    GBD$L_VBN(R0), R3       ; Is the vbn -1?
           6F   13   0511  1424         BEQL    GOT_GBD                 ; EQL then just use this one.
                     0513  1425 ;
                     0513  1426 ; There are no available GBD's at the end of the list.
                     0513  1427 ; Will have to select GBD to toss out of the cache.
                     0513  1428 ;
           7E   D4   0513  1429         CLRL    -(SP)                   ; Note first pass.
   55 5C   2C AC   C1   0515  1430         ADDL3   GBH$L_GBD_END(AP), AP, R5 ; End addr of scan.
   50 5C   30 AC   C1   051A  1431         ADDL3   GBH$L_GBD_NEXT(AP), AP, R0 ; Starting point of scan.
           28 AC   D1   051F  1432         CMPL    GBH$L_GBD_START(AP),-   ; Is NEXT pointer the
           30 AC   0522  1433                 GBH$L_GBD_NEXT(AP)      ; same as the START of GBD's?
           02   12   0524  1434         BNEQ    10$                     ; NEQ then make two passes.
           6E   D6   0526  1435         INCL    (SP)                    ; Else only make one.
      52   34 AC   D0   0528  1436 10$:    MOVL    GBH$L_SCAN_NUM(AP), R2  ; Number of GBD's to search.
                     052C  1437 LTOP:
           20 A0   B5   052C  1438         TSTW    GBD$W_USECNT(R0)        ; This one in use?
           13   12   052F  1439         BNEQ    LTST                    ; NEQ it is. Br to loop test.
      53   0B A0   91   0531  1440         CMPB    GBD$B_CACHE_VAL(R0), R3 ; Is this lowest cache value seen?
           09   1E   0535  1441         BGEQU   SCANTST                 ; GEQU lower has been seen.  Ignore it.
      54   50   D0   0537  1442         MOVL    R0, R4                  ; Save this GBD addr.
      53   0B A0   9A   053A  1443         MOVZBL  GBD$B_CACHE_VAL(R0), R3 ; Save this cache value.
           1D   13   053E  1444         BEQL    USE_GBD                 ; Use first zero value one found.
                     0540  1445 SCANTST:
           52   D7   0540  1446         DECL    R2                      ; Keep scanning if counter not run out.
           19   15   0542  1447         BLEQ    USE_GBD                 ; Use what's been found.
                     0544  1448 LTST:
FFE2 50    28   55   F1   0544  1449         ACBL    R5, #GBD$C_BLN, R0, LTOP ; Keep going if limit not hit yet.
                     054A  1450
      0F 6E    00   E2   054A  1451         BBSS    #0, (SP), USE_GBD       ; Br if second pass.
   50 5C   28 AC   C1   054E  1452         ADDL3   GBH$L_GBD_START(AP), AP, R0 ; Start at beginning this time.
   55 5C   30 AC   C1   0553  1453         ADDL3   GBH$L_GBD_NEXT(AP), AP, R5 ; End with current next ptr.
```

```
                55   28   C2  0558  1454          SUBL2   #GBD$C_BLN, R5          ; Back up one - did it first pass.
                     CF   11  055B  1455          BRB     LTOP                    ; Branch into loop.
                          055D  1456 USE_GBD:
                5E   04   C0  055D  1457          ADDL2   #4, SP                  ; Pop pass courter off stack.
          50    5C   30 AC   C1  0560  1458          ADDL3   GBH$L_GBD_NEXT(AP), AP, R0 ; Get addr of this 'victim'
                     0B A0   95  0565  1459          TSTB    GBD$B_CACHE_VAL(R0)     ; Is it already zero?
                     03      13  0568  1460          BEQL    10$                     ; EQL then can't age anymore.
                     0B A0   97  056A  1461          DECB    GBD$B_CACHE_VAL(R0)     ; Age him a unit.
                30 AC   28   C0  056D  1462 10$:     ADDL2   #GBD$C_BLN, GBH$L_GBD_NEXT(AP) ; Move on to next GBD next time.
             2C AC   30 AC   D1  0571  1463          CMPL    GBH$L_GBD_NEXT(AP), GBH$L_GBD_END(AP) ; Is this past GBD's?
                     05      1B  0576  1464          BLEQU   20$                     ; LEQU it's still in range.
             30 AC   28 AC   D0  0578  1465          MOVL    GBH$L_GBD_START(AP), GBH$L_GBD_NEXT(AP) ; Else reset to start.
                50   54      D0  057D  1466 20$:     MOVL    R4, R0                  ; Get addr of GBD to use.
                     48      13  0580  1467          BEQL    ERXIT                   ; EQL didn't find one.
                             0582  1468
                             0582  1469 GOT_GBD:
                     3C      BA  0582  1470          POPR    #^M<R2,R3,R4,R5>        ; Restore registers.
                             0584  1471 ;
                             0584  1472 ; R0 = GBD to use
                             0584  1473 ; R1 = vbn desired
                             0584  1474 ; R2 = size of bucket
                             0584  1475 ; R5 = previous GBD from search scan.
                             0584  1476 ;
                             0584  1477
                             0584  1478 ;
                             0584  1479 ; Remove this GBD from current position in queue.
                             0584  1480 ;
                             0584  1481          ASSUME  GBD$L_FLINK     EQ      0
                             0584  1482          ASSUME  GBD$L_BLINK     EQ      4
                             0584  1483
             5C   50   60    C1  0584  1484          ADDL3   (R0), R0, AP            ; Addr of successor
                  50   6C    5F  0588  1485          REMQTI  (AP), R0                ; Remove GBD from queue.
                             058B  1486 ;
                             058B  1487 ; Init this GBD.
                             058B  1488 ;
                             058B  1489
                             058B  1490          ASSUME  <GBD$B_FLAGS + 1>       EQ      GBD$B_CACHE_VAL
                             058B  1491
                     0A A0   B4  058B  1492          CLRW    GBD$B_FLAGS(R0)         ; Init flags, cache value.
                     14 A0   D5  058E  1493          TSTL    GBD$L_LOCK_ID(R0)       ; Is there a system lock out?
                         1C  13  0591  1494          BEQL    5$                      ; Branch around this if not.
                         03  BB  0593  1495          PUSHR   #^M<R0,R1>
00000000'9F              01  58  0595  1496          ADAWI   #1,@#RMS$GW_GBLBUFQUO   ; Count this buffer back in.
                             059C  1497          $DEQ_S  LKID = GBD$L_LOCK_ID(R0); Remove old buffer from cache.
                         03  BA  05AA  1498          POPR    #^M<R0,R1>
                     14 A0   D4  05AC  1499          CLRL    GBD$L_LOCK_ID(R0)       ; Clear system lock id from gbd.
                10 A0    01  CE  05AF  1500 5$:      MNEGL   #1,GBD$L_VBNSEQNUM(R0)  ; INVALIDATE buffer.
                0C A0    51  D0  05B3  1501          MOVL    R1, GBD$L_VBN(R0)       ; Fill in vbn of bucket.
                18 A0    52  B0  05B7  1502          MOVW    R2, GBD$W_NUMB(R0)      ; Note size of bucket.
                             05BB  1503 ;
                             05BB  1504 ; Place GBD back into queue.
                             05BB  1505 ;
                   55    50   D1  05BB  1506          CMPL    R0, R5                  ; Did we select the previous GBD?
                         04   12  05BE  1507          BNEQ    10$                     ; NEQ no.
                55   04 A5   C0  05C0  1508          ADDL2   4(R5), R5               ; Get the previous one then.
                   65    60   5C  05C4  1509 10$:     INSQHI  (R0), (R5)              ; Insert GBD in queue.
                      FEE1    31  05C7  1510          BRW     GET_GBPB                ; Fill in GBPB.
```

```
                05CA  1511
      3C    BA  05CA  1512 ERXIT:  POPR    #^M<R2,R3,R4,R5>        ; Restore registers.
  FA31'     30  05CC  1513         BSBW    RM$LOWER_GBS_LOCK       ; Release our EX lock on the GBS.
      55    D4  05CF  1514         CLRL    R5                      ; Note no GBPB.
      50    D4  05D1  1515         CLRL    R0                      ; Note failure in R0.
            05  05D3  1516         RSB                             ; Return.
                05D4  1517
                05D4  1518         .END
```

```
$$.PSECT_EP                     = 00000000
$$ARGS                          = 0000000B
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
$$T1                            = 00000001
BDB$B_BID                       = 00000008
BDB$B_CACHE_VAL                 = 0000000B
BDB$B_FLGS                      = 0000000A
BDB$C_BID                       = 0000000C
BDB$L_ADDR                      = 00000018
BDB$L_BI_BDB                    = 00000030
BDB$L_BLB_PTR                   = 00000010
BDB$L_BLINK                     = 00000004
BDB$L_FLINK                     = 00000000
BDB$L_IOSB                      = 00000048
BDB$L_VBN                       = 0000001C
BDB$L_VBNSEQNO                  = 00000020
BDB$M_NOLOCATE                  = 00000010
BDB$M_VAL                       = 00000001
BDB$V_AST_DCL                   = 00000006
BDB$W_NUMB                      = 00000014
BDB$W_SIZE                      = 00000016
BDB$W_USERS                     = 0000000C
BLB$B_BLBFLGS                   = 0000000A
BLB$B_MODEHELD                  = 00000009
BLB$L_BDB_ADDR                  = 0000000C
BLB$L_BLNK                      = 00000004
BLB$L_FLNK                      = 00000000
BLB$L_LOCK_ID                   = 00000024
BLB$L_OWNER                     = 00000010
BLB$L_RESDSC                    = 00000018
BLB$L_VALSEQNO                  = 00000028
BLB$L_VBN                       = 00000014
BLB$M_IOLOCK                    = 00000010
BLB$M_LOCK                      = 00000001
BLB$V_DFW                       = 00000005
BLB$V_IOLOCK                    = 00000004
BLB$V_LOCK                      = 00000000
BLB$V_NOBUFFER                  = 00000003
BLB$V_NOREAD                    = 00000002
BLB$V_NOWAIT                    = 00000001
BLB$V_WRITEBACK                 = 00000006
BLB$W_STS                       = 00000020
BUFF_ONLY                       000001BA R      01
BUFF_ONLY_BR                    0000000F R      01
CACHE                           00000014 R      01
CHECK_LOCKS                     00000072 R      01
CHKWBR                          00000106 R      01
CSH$M_LOCK                      = 00000001
CSH$M_NOBUFFER                  = 00000008
CSH$M_NOREAD                    = 00000004
CSH$M_NOWAIT                    = 00000002
CSH$V_LOCK                      = 00000000
CSH$V_NOBUFFER                  = 00000003
CSH$V_NOREAD                    = 00000002
CSH$V_NOWAIT                    = 00000001
CSH_MASK_ALL                    = 0000000F
ENQ$_ACMODE                     = 00000028
ENQ$_ASTADR                     = 0000001C
ENQ$_ASTPRM                     = 00000020
ENQ$_BLKAST                     = 00000024
ENQ$_EFN                        = 00000004
ENQ$_FLAGS                      = 00000010
ENQ$_LKMODE                     = 00000008
ENQ$_LKSB                       = 0000000C
ENQ$_NARGS                      = 0000000B
ENQ$_PARID                      = 00000018
ENQ$_PROT                       = 0000002C
ENQ$_RESNAM                     = 00000014
ERREX                           0000019B R      01
ERREX1                          000001AA R      01
ERRX                            0000019E R      01
ERXIT                           000005CA R      01
EXBR                            000000F8 R      01
EXIT                            0000019A R      01
FIND_FREE_GBL                   000004FD R      01
FTL$_NOBLB                      = FFFFFFDF
FTL$_NOLCLBUF                   = FFFFFFDD
FTL$_NORDNOTSET                 = FFFFFFDC
FTL$_NOSFSB                     = FFFFFFDA
GBD$B_CACHE_VAL                 = 0000000B
GBD$B_FLAGS                     = 0000000A
GBD$C_BLN                       = 00000028
GBD$L_BLINK                     = 00000004
GBD$L_FLINK                     = 00000000
GBD$L_LOCK_ID                   = 00000014
GBD$L_REL_ADDR                  = 0000001C
GBD$L_VBN                       = 0000000C
GBD$L_VBNSEQNUM                 = 00000010
GBD$W_NUMB                      = 00000018
GBD$W_SIZE                      = 0000001A
GBD$W_USECNT                    = 00000020
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_HIT                       = 00000038
GBH$L_HI_VBN                    = 0000000C
GBH$L_MISS                      = 0000003C
GBH$L_SCAN_NUM                  = 00000034
GBPB$B_BID                      = 00000008
GBPB$B_FLGS                     = 0000000A
GBPB$C_BID                      = 00000015
GBPB$L_ADDR                     = 00000018
GBPB$L_GBD_PTR                  = 00000024
GBPB$L_VBN                      = 0000001C
GBPB$L_VBNSEQNO                 = 00000020
GBPB$W_NUMB                     = 00000014
GBPB$W_SIZE                     = 00000016
GBPB$W_USERS                    = 0000000C
GET_BLB                         00000245 R      01
```

```
GET_BUFF                        000000CF R      01
GET_GBPB                        000004AB R      01
GOT_BUFF                        000000DA R      01
GOT_GBD                         00000582 R      01
IFB$B_BID                   =   00000008
IFB$B_JNLFLG                =   000000A0
IFB$B_ORGCASE               =   00000023
IFB$C_BID                   =   0000000B
IFB$C_IDX                   =   00000002
IFB$L_BDB_BLNK              =   00000044
IFB$L_BDB_FLNK              =   00000040
IFB$L_BLBBLNK               =   0000009C
IFB$L_BLBFLNK               =   00000098
IFB$L_GBH_PTR               =   00000088
IFB$L_PAR_LOCK_ID           =   00000080
IFB$V_BI                    =   00000002
IFB$V_MSE                   =   00000031
IFB$V_NORECLK               =   00000033
IFB$W_AVLCL                 =   00000084
IRB$B_BID                   =   00000008
IRB$B_EFN                   =   0000000B
IRB$C_BID                   =   0000000A
IRB$V_GBLBUFF               =   00000036
LCK$K_EXMODE                =   00000005
LCK$K_PWMODE                =   00000004
LCK$M_CONVERT               =   00000002
LCK$M_NOQUEUE               =   00000004
LCK$M_SYNCSTS               =   00000008
LCK$M_SYSTEM                =   00000010
LCK$M_VALBLK                =   00000001
LOCAL                           000000B0 R      01
LOCK_BKT                        0000038D R      01
LOCK_IT                         0000010F R      01
LTOP                            0000052C R      01
LTST                            00000544 R      01
NEED_BLB                        0000007F R      01
NEED_BUFFER                     0000006E R      01
NEED_BUFFONLY                   0000005D R      01
NEED_READ                       000000D6 R      01
NOLOCKING                       000000E0 R      01
RAB$L_STV                   =   0000000C
READ_BKT                        00000154 R      01
READ_NOLOCKING                  000000F3 R      01
RJR$C_BKTLEN                =   00000044
RLS$M_DEQ                   =   00000008
RMSBUG                          ********   X    01
RMSCACHE                        00000012 RG     01
RMSCACH_IN                      ********   X    01
RMSCACH_OUT                     ********   X    01
RMSFREE_LCL                     00000273 RG     01
RMSGET_LCL_BUFF                 0000034E RG     01
RMSLOWER_GBS_LOCK               ********   X    01
RMSMAPERR                       ********   X    01
RMSRAISE_GBS_LOCK               ********   X    01
RMSRDBUFQT                      ********   X    01
RMSRELEASE                      ********   X    01
RMSSETEFN                       ********   X    01
RMS$STALL                       ********   X    01
RMS$STALLAST                    ********   X    01
RMS$GW_GBLBUFQUO                ********   X    01
RMS$_DME                    =   000184D4
RMS$_ENQ                    =   0001C134
RMS$_RLK                    =   000182AA
RMS$_WER                    =   0001C114
SCANTST                         00000540 R      01
SCAN_GBL                        0000045A R      01
SCAN_LCL_CACHE                  00000331 R      01
SCAN_LOCKS                      000001D9 R      01
SETR5                           00000162 R      01
SS$_DEADLOCK                =   00000E0A
SS$_EXENQLM                 =   00002A44
SS$_NOLOCKID                =   00000E12
SS$_SYNCH                   =   00000689
SS$_VALNOTVALID             =   000009F0
SYS$DEQ                         ********  GX    01
SYS$ENQ                         ********  GX    01
SYS$SETEF                       ********  GX    01
TRACE                           00000000 R      01
USE_GBD                         0000055D R      01
WBKERR                          000000FB R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM$RMS0	000005D4 (1492.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.07	00:00:00.92
Command processing	118	00:00:00.76	00:00:04.42
Pass 1	480	00:00:19.66	00:00:49.10
Symbol table sort	0	00:00:02.54	00:00:04.82
Pass 2	275	00:00:05.03	00:00:14.64
Symbol table output	22	00:00:00.18	00:00:00.30
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	930	00:00:28.27	00:01:14.24

The working set limit was 2100 pages.
108548 bytes (213 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1733 non-local and 66 local symbols.
1518 source lines were read in Pass 1, producing 17 object records in Pass 2.
43 pages of virtual memory were used to define 42 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	22
_255DUA28:[SYS.OBJ]LIB.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	15
TOTALS (all libraries)	38

1938 GETS were required to define 38 macros.

There were no errors, warnings or information messages.

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

NT0SCNXAB
LIS

RM0CACHE
LIS

NT0RENAME
LIS

NT0OPEN
LIS

RM0BUFMGR
LIS

NT0SEARCH
LIS

NT0PUT
LIS

RM0ACCESS
LIS

RM0EXTRMS
LIS

RM0DIRSCN
LIS

RM0CRECOM
LIS

RM0FABCHK
LIS

RM0CHKSUM
LIS

RM0FWASET
LIS

RM0EXTEND
LIS

RM0IFISI
LIS

RM0JOURNL
LIS

RM0FSETI
LIS

RM0FSET
LIS

RM0COMCLN
LIS

RM0DUMMY
LIS

RM0FILFNC
LIS