C 13

```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     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  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
_$2

Sy
---
NT$
NT$
NT$
NT$
NT$
NT$



NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$



NT$
NT$
NT$
NT$
NT$
NT$
PI
```

```
**FILE**ID**RMOCACHE                                    C 14





RRRRRRRR    MM      MM    000000      CCCCCCCC    AAAAAA      CCCCCCCC  HH      HH  EEEEEEEEEE
RRRRRRRR    MM      MM    000000      CCCCCCCC    AAAAAA      CCCCCCCC  HH      HH  EEEEEEEEEE
RR      RR  MMMM  MMMM  00      00  CC          AA      AA  CC          HH      HH  EE
RR      RR  MMMM  MMMM  00      00  CC          AA      AA  CC          HH      HH  EE
RR      RR  MM  MM  MM  00    0000  CC          AA      AA  CC          HH      HH  EE
RR      RR  MM  MM  MM  00    0000  CC          AA      AA  CC          HH      HH  EE
RRRRRRRR    MM      MM  00  00  00  CC          AA      AA  CC          HHHHHHHHHH  EEEEEEEE
RRRRRRRR    MM      MM  00  00  00  CC          AA      AA  CC          HHHHHHHHHH  EEEEEEEE
RR  RR      MM      MM  0000    00  CC          AAAAAAAAAA  CC          HH      HH  EE
RR  RR      MM      MM  0000    00  CC          AAAAAAAAAA  CC          HH      HH  EE
RR    RR    MM      MM  00      00  CC          AA      AA  CC          HH      HH  EE              ....
RR    RR    MM      MM  00      00  CC          AA      AA  CC          HH      HH  EE              ....
RR      RR  MM      MM    000000      CCCCCCCC  AA      AA    CCCCCCCC  HH      HH  EEEEEEEEEE      ....
RR      RR  MM      MM    000000      CCCCCCCC  AA      AA    CCCCCCCC  HH      HH  EEEEEEEEEE      ....

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

```
Table of contents
```

```
0000          $BEGIN  RMOCACHE,000,RM$RMS0,<IO CACHE ROUTINE>
0000
0000    ;
0000    ;*****************************************************************************
0000    ;*                                                                           *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    ;*  ALL RIGHTS RESERVED.                                                     *
0000    ;*                                                                           *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    ;*  TRANSFERRED.                                                             *
0000    ;*                                                                           *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    ;*  CORPORATION.                                                             *
0000    ;*                                                                           *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    ;*                                                                           *
0000    ;*                                                                           *
0000    ;*****************************************************************************
0000    ;
```

```
0000  ;++
0000  ;
0000  ; Facility: rms32
0000  ;
0000  ; Abstract:
0000  ;
0000  ;       This module provides a block cache and access control
0000  ;       to the buckets of the relative and indexed file organizations
0000  ;
0000  ; Environment:
0000  ;       VAX/VMS
0000  ;
0000  ; Author: E.H. Marison  15-SEP-1977
0000  ;
0000  ; Modified By:
0000  ;
0000  ;       V03-023 JEJ0044         J E Johnson             21-Jun-1984
0000  ;               Tweak the instructions a little for a performance boost.
0000  ;
0000  ;       V03-022 SHZ0011         Stephen H. Zalewski,    30-Apr-1984
0000  ;               If we stall in CACHE, do not set the event flag unless it
0000  ;               is nonzero.  In async I/O case it will be zero.
0000  ;
0000  ;       V03-021 JEJ0007         J E Johnson             16-Mar-1984
0000  ;               Add global buffer quota accounting to limit the number of
0000  ;               system-wide locks taken out by the users.
0000  ;
0000  ;       V03-020 SHZ0010         Stephen H. Zalewski,    13-Mar-1984
0000  ;               Make sure we are record locking before we try to scan
0000  ;               the BLB list in RM$FREE_LCL.  Thus, in the case of
0000  ;               the local buffer count being wrong, we will not
0000  ;               try to scan the BLB list to try and free up a BDB.
0000  ;
0000  ;       V03-019 JWT0160         Jim Teague              29-Feb-1984
0000  ;               Remove call to RM$DEALLEFN.
0000  ;
0000  ;       V03-018 SHZ0009         Stephen H. Zalewski     26-Jan-1984
0000  ;               If we must stall waiting for a writeback to occur on
0000  ;               a blb we want to throw out of cache, we must set the
0000  ;               event flag after the stall, or we wait forever on the flag.
0000  ;
0000  ;       V03-017 SHZ0008         Stephen H. Zalewski     5-Dec-1983
0000  ;               If we stall waiting for a writeback to occur, set the
0000  ;               event flag after the stall has completed, or we will
0000  ;               wait forever on the event flag to be set.
0000  ;
0000  ;       V03-016 KPL0001         Peter Lieberwirth       28-Oct-1983
0000  ;               Fix problem with BI journaling.  Before-image copy of
0000  ;               the buffer was only made if the buffer was read in with
0000  ;               intent to write.  However, if the buffer was cached for
0000  ;               read, and is found in the cache to write out, no before
0000  ;               image copy was made.
0000  ;
0000  ;               This fix unconditionally copies the before-image of the
0000  ;               bucket to the before-image buffer - read or write.  While
0000  ;               this is good enough for FT1, it should be changed for FT2
0000  ;               such that the copy is only made when the bucket is to be
```

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

```
0000    ;
0000    ;       V03-004 CDS0028         C Saether               13-Apr-1982
0000    ;               Modify lock_bkt routine to attempt to toss something
0000    ;               out of the local cache when the enq results in
0000    ;               either an exenqlm or nolockid error.  Hopefully this
0000    ;               frees up a lock so the operation can continue.
0000    ;
0000    ;       V03-003 CDS0027         C Saether               30-Mar-1982
0000    ;               Correct problem when LOCK, NOREAD, NOBUFFER is
0000    ;               specified for a bucket which is already accessed
0000    ;               with a buffer.  This was causing the new flags to
0000    ;               overwrite those from the initial access and not
0000    ;               store the value block on release.
0000    ;
0000    ;               Set default error in R1 before call to MAPERR.
0000    ;               Increment use count in GBD only after GBPB accessed.
0000    ;
0000    ;               Check if BDB was present before looking at it.
0000    ;
0000    ;       V03-002 CDS0026         C Saether               22-Mar-1982
0000    ;               Modify SCAN_GBL routine to avoid end test and speed
0000    ;               up by looping in line.
0000    ;
0000    ;       V03-001 CDS0025         C Saether               1-Mar-1982
0000    ;               Count hits and misses for global buffers.
0000    ;               Fix incorrect register use in SCAN_LOCKS checking
0000    ;               for GBPB.
0000    ;                                                       26-Feb-1982
0000    ;               Don't bother trying to get read locks on buckets in
0000    ;               a compatible mode.  It causes conversion deadlock
0000    ;               problems when other streams (processes) are attempting
0000    ;               to modify the same bucket simultaneously.  This is
0000    ;               the same problem only partially corrected in V02-036.
0000    ;               Also have LOCK_BKT routine call RM$MAPERR to map
0000    ;               enqueue failures to RMS errors if appropriate.
0000    ;                                                       16-Feb-1982
0000    ;               Changes to allow modification of global buffers.
0000    ;               FREE_LCL and GET_LCL_BUFF become RM$FREE_LCL and
0000    ;               RM$GET_LCL_BUFF.
0000    ;               SCAN_LOCKS changed to not return GBPB address unless
0000    ;               caller already owns it (don't want more than one
0000    ;               accessor on a GBPB at once).
0000    ;               Don't try to use global buffers if stream didn't
0000    ;               connect for them.
0000    ;                                                       7-Feb-1982
0000    ;               Shorten scan_local_cache to scan_lcl_cache.
0000    ;
0000    ;--
0000    ;
0000
```

```
                    0000          .SBTTL  DECLARATIONS
                    0000
                    0000    ;
                    0000    ; Include Files:
                    0000    ;
                    0000
                    0000          $BDBDEF                             ; bucket descriptor blocks
                    C000          $BKTDEF                             ; bucket definitions
                    0000          $BLBDEF                             ; bucket lock blocks
                    0000          $CSHDEF                             ; rm$cache flag definitions
                    0000          $ENQDEF                             ; enq sys service definitions
                    0000          $GBDDEF                             ; global buffer descriptor defs
                    0000          $GBHDEF                             ; global buffer header
                    0000          $GBPBDEF                            ; global buffer pointer block defs
                    0000          $IFBDEF                             ; ifab defs
                    0000          $IMPDEF                             ; impure area definitions
                    0000          $IRBDEF                             ; irab defs
                    0000          $LCKDEF                             ; lock manager flags, constant defs
                    0000          $PIODEF                             ; process i/o definitions
                    0000          $RABDEF                             ; record access block
                    0000          $RJRDEF                             ; RMS Journalling Record definitions
                    0000          $RLSDEF                             ; rm$release flags
                    0000          $RMSDEF                             ; rms error code defs
                    0000          $SFSBDEF                            ; shared file lock block defs
                    0000          $SSDEF                              ; System service error codes
                    0000
                    0000    ;
                    0000    ; Macros:
                    0000    ;
                    0000    ;
                    0000    ; Equated Symbols:
                    0000    ;
                    0000
          0000000F  0000    CSH_MASK_ALL    =       CSH$M_LOCK!CSH$M_NOWAIT!CSH$M_NOREAD!CSH$M_NOBUFFER
                    0000
                    0000    ;
                    0000    ; Own Storage:
                    0000    ;
                    0000
```

```
0000          .SBTTL  RM$CACHE ROUTINE
0000
0000    ;++
0000    ;
0000    ; RM$CACHE - access and read bucket if necessary
0000    ;
0000    ;     1. obtains access to requested block/bucket and waits for the access
0000    ;        unless csh$v_nowait is set in the control flags.
0000    ;
0000    ;     2. obtains a buffer for the block unless csh$v_nobuffer
0000    ;        is set in the control flags.
0000    ;
0000    ;     3. if there is a buffer read the block into it if required, and the
0000    ;        csh$v_noread bit is off in the control flags.
0000    ;
0000    ;     4. waits for io completion
0000    ;
0000    ;     5. if the csh$v_lock bit is set in the flags then exclusive access to the
0000    ;        block is obtained.
0000    ;
0000    ; Calling sequence:
0000    ;
0000    ;       BSBW    RM$CACHE
0000    ;
0000    ; Input Parameters:
0000    ;
0000    ;       r11     impure area address
0000    ;       r10     ifab address
0000    ;       r9      irab/ifab address
0000    ;       r8      rab/fab address
0000    ;       r3      cache control flags
0000    ;       r2      transfer/buffer size in bytes
0000    ;       r1      requested vbn
0000    ;
0000    ; Output Parameters:
0000    ;
0000    ;       r0      internal rms status code
0000    ;       r4      bdb address
0000    ;       r5      buffer address unless a nobuffer call
0000    ;               in which case r5 is destroyed
0000    ;       r1,r2,r3,ap destroyed
0000    ;
0000    ; ****  bdb and buffer not accessed on errors
0000    ;
0000    ; Completion Codes:
0000    ;
0000    ;       standard internal rms, including:
0000    ;
0000    ;               suc     normal success
0000    ;               rlk     block was accessed or locked and nowait
0000    ;               dme     could not get a buffer
0000    ;               exenqlm the enq limit for this process was exceeded
0000    ;                       while attempting to lock a bucket.
0000    ;               various errors writing a deferred write buffer or reading
0000    ;               in the new buffer.
0000    ;
0000    ; 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   ; on VBN 1 when VBN 1 is already accessed with a buffer.  In that case,
                    0045   ; the owner will be the current stream.  The address of the BDB (as opposed
                    0045   ; to the BLB) must be returned in that case because the routine compares
                    0045   ; the original BDB address from the first call (saved in IRB$L_LOCK_BDB)
                    0045   ; with the return from this call to determine whether to release the lock.
                    0045   ; Pretty tacky, but that's the way it is.  No checks are made to determine
                    0045   ; if the bucket is LOCKed when one is found, as it is not believed any routines
                    0045   ; would do that and follow it with a LOCK, NOBUFFER call.
                    0045   ;
                    0045
 59   10 A4    D1   0045                  CMPL    BLB$L_OWNER(R4), R9       ; This stream already have it accessed?
         05    12   0049                  BNEQ    25$                       ; NEQ, then go access it normally.
   54    55    D0   004B                  MOVL    R5, R4                    ; Get BDB addr into R4.
         E0    11   004E                  BRB     15$                       ; And exit with success.
   53    0C    CA   0050   25$:           BICL2   #CSH$M_NOBUFFER!CSH$M_NOREAD, R3 ; There already is a buffer.
       00B9    31   0053                  BRW     LOCK_IT                   ; Raise mode if req'd.
                    0056
       01EC    30   0056   30$:           BSBW    GET_BLB                   ; Get a BLB for this lock.
       0331    30   0059                  BSBW    LOCK_BKT                  ; Go lock it.
               05   005C                  RSB                               ; Return.
                    005D
                    005D   ;
                    005D   ; Got the bucket locked, but no buffer with it.
                    005D   ;
                    005D
                    005D   NEED_BUFFONLY:
    0084 CA    B7   005D                  DECW    IFB$W_AVLCL(R10)          ; Decrement available count.
         6C    18   0061                  BGEQ    GET_BUFF                  ; Enough buffers, go get BDB.
       020D    30   0063                  BSBW    RM$FREE_LCL               ; Free up a local buffer.
      66 50    E8   0066                  BLBS    R0, GET_BUFF              ; Branch and go use it on success.
    0084 CA    B6   0069                  INCW    IFB$W_AVLCL(R10)          ; Restore count.
               05   006D                  RSB                               ; Return with error in R0.
```

```
                 006E  NEED_BUFFER:
     6A   33  E0 006E          BBS     #IFB$V_NORECLK, (R10),- ; Branch if no locking.
          6E     0071                  NOLOCKING
                 0072
                 0072  ;
                 0072  ; Locking is being done.  Scan list of buffer lock blocks (BLB's) to
                 0072  ; determine if bucket already has lock.  Normally locating a bucket in
                 0072  ; the BLB list means that either a NL or PW lock is held on a buffer
                 0072  ; currently present in the local cache.
                 0072  ;
                 0072  ; Under some conditions a request is made for a lock with buffer on a
                 0072  ; bucket which was previously locked with the NOBUFFER flag, in which
                 0072  ; case a lock will be found with no BDB.
                 0072  ;
                 0072  ; Lastly, a BDB for the desired bucket may be found, but no BLB.
                 0072  ; This will occur when multi-streaming and another stream has the
                 0072  ; desired bucket accessed.  Only a BLB must be acquired in this case.
                 0072  ;
                 0072
                 0072  CHECK_LOCKS:
        0164  30 0072          BSBW    SCAN_LOCKS                  ; See if lock already held.
     07   50  E9 0075          BLBC    R0, NEED_BLB                ; No - go to get BLB.
          55  D5 0078          TSTL    R5                          ; Was there a BDB also with the BLB?
          E1  13 007A          BEQL    NEED_BUFFONLY               ; Go get a buffer for the BLB.
        0087  31 007C          BRW     CHKWBK                      ; Have BDB, BLB, so access them.
                 007F  NEED_BLB:
     0088 CA  D5 007F          TSTL    IFB$L_GBH_PTR(R10)          ; Global buffer cache present?
          2B  13 0083          BEQL    LOCAL                       ; EQL then there is none.
                 0085
                 0085  ;
                 0085  ; Global buffer cache is present.  If a BDB has already been found, though,
                 0085  ; always use it.  It didn't have a BLB if here, meaning another stream
                 0085  ; must currently have it accessed, therefore it has a much better chance
                 0085  ; of being valid.
                 0085  ;
                 0085
          55  D5 0085          TSTL    R5                          ; Is BDB already present?
          27  12 0087          BNEQ    LOCAL                       ; NEQ then use it.
                 0089
                 0089          ASSUME  IRB$B_BID         EQ        IFB$B_BID
                 0089          ASSUME  <IRB$C_BID & 1> EQ          0
                 0089          ASSUME  <IFB$C_BID & 1> EQ          1
                 0089
  23 08 A9    E8 0089          BLBS    IFB$B_BID(R9), LOCAL        ; Use local if ifab operation.
1F 69     36  E1 008D          BBC     #IRB$V_GBLBUFF, (R9), LOCAL ; Use local if stream did not
                 0091                                              ; want global buffs when connecting.
                 0091
                 0091  ;
                 0091  ; Search global cache, if failure the gbsb lock is not released since it may
                 0091  ; be needed shortly in find_free_gbl.  If success the lock is released.
                 0091  ;
                 0091
        03C6  30 0091          BSBW    SCAN_GBL                    ; Search global cache
     0A   50  E8 0094          BLBS    R0, TO$                     ; Branch if got a match and use it.
                 0097
                 0097  ;
                 0097  ; Did not find the desired bucket in the global cache.
                 0097  ; If a lock is not requested, attempt to get a global buffer and use it.
                 0097  ; If a lock is requested, use a local buffer.  The belief is that if
                 0097  ; the bucket wasn't already in the global cache, this must be a new
                 0097  ; insert, therefore the chance of another process potentially having
                 0097  ; an interest in it is very low.  In addition, if deferred write is
                 0097  ; enabled, modified global buffers must be copied to a local buffer
                 0097  ; when they are released.  The extra cpu overhead to do that would
                 0097  ; outweigh the rare instances where an i/o would be saved because
                 0097  ; another process was interested in the same bucket.
                 0097  ;
                 0097
                 0097                  ASSUME  CSH$V_LOCK      EQ      0
                 0097
     07 53   E9  0097                  BLBC    R3, 10$                         ; Br to use gbl if not locking.
     FF63'   30  009A                  BSBW    RM$LOWER_GBS_LOCK               ; Release lock on gbsb (taken in scan_gbl)
        55   D4  009D                  CLRL    R5                              ; Note that no buffer is present.
        0F   11  009F                  BRB     LOCAL                           ; Go use local buffer.
                 00A1
                 00A1  ;
                 00A1  ; We wish to use global buffers.  R0 contains the status from the global
                 00A1  ; cache scan above.   Whether or not the requested bucket was found, we
                 00A1  ; will need a blb.  In the rare case where a global buffer cannot be freed
                 00A1  ; when the desired bucket was not located, reset the owner and vbn fields of
                 00A1  ; the blb just obtained, and drop through to use a local buffer instead.
                 00A1  ;
                 00A1
      01A1   30  00A1  10$:            BSBW    GET_BLB                         ; Get a free BLB for the lock.
     33 50   E8  00A4                  BLBS    R0, GOT_BUFF                    ; Branch if match found in gbl cache -
                 00A7                                                          ; R0 is the result from SCAN_GBL here.
      0453   30  00A7                  BSBW    FIND_FREE_GBL                   ; Attempt to find a free global buffer.
     29 50   E8  00AA                  BLBS    R0, NEED_READ                   ; Br to force read if one found.
                 00AD                  ASSUME  <BLB$L_OWNER + 4> EQ BLB$L_VBN
     10 A4   7C  00AD                  CLRQ    BLB$L_OWNER(R4)                 ; Free up BLB.  Drop thru to use local.
                 00B0
                 00B0  ;
                 00B0  ; A local buffer is to be used.
                 00B0  ; If R5 is non-zero, it contains the address of the BDB for the requested
                 00B0  ; bucket even though a BLB must be obtained.
                 00B0  ;
                 00B0
                 00B0  LOCAL:
   0084 CA   B7  00B0                  DECW    IFB$W_AVLCL(R10)                ; Decrement available count.
        12   18  00B4                  BGEQ    10$                             ; Got enough - go get BLB.
   0084 CA   B6  00B6                  INCW    IFB$W_AVLCL(R10)                ; Put count back - will go round again.
      01B6   30  00BA                  BSBW    RM$FREE_LCL                     ; Free up a buffer.
     38 50   E9  00BD                  BLBC    R0, EXBR                        ; Exit on error.
     6A 31   E0  00C0                  BBS     #IFB$V_MSE, (R10),-             ; If multi-streaming, need to scan
        AE       00C3                          CHECK_LOCKS                     ; locks again (may have changed).
   0084 CA   B7  00C4                  DECW    IFB$W_AVLCL(R10)                ; Dec count.  One is available now.
      017A   30  00C8  10$:            BSBW    GET_BLB                         ; Get a free BLB.
        55   D5  00CB                  TSTL    R5                              ; Is there a BDB already?
        0B   12  00CD                  BNEQ    GOT_BUFF                        ; NEQ already have one.
                 00CF  GET_BUFF:
      027C   30  00CF                  BSBW    RM$GET_LCL_BUFF                 ; Get a free BDB.
     6A 33   E0  00D2                  BBS     #IFB$V_NORECLK, (R10),-         ; Branch if not locking.
        1D       00D5                          READ_NOLOCKING
                 00D6  NEED_READ:
```

```
        10  88  00D6          BISB2   #BLB$M_IOLOCK,-         ; Know that I/O will be req'd.
     0A A4      00D8                  BLB$B_BLBFLGS(R4)
                00DA  GOT_BUFF:
  0C A4  55  D0  00DA          MOVL    R5, BLB$L_BDB_ADDR(R4)  ; Store BDB address in BLB.
        2F  11  00DE          BRB     LOCK_IT                 ; Go to lock code.
                00E0
                00E0  ;
                00E0  ; Branch here when using local buffers in exclusive mode, i.e., no locking
                00E0  ; is being performed.
                00E0  ;
                00E0
                00E0  NOLOCKING:
      024E  30  00E0          BSBW    SCAN_LCL_CACHE          ; Look in local cache.
     54  55  D0  00E3          MOVL    R5, R4                  ; Anticipate suc - load R4 with BDB.
        03  12  00E6          BNEQ    10$                     ; NEQ we have a buffer.
      FF72  31  00E8          BRW     NEED_BUFFONLY           ; Need to go get a buffer.
                00EB
                00EB  ;
                00EB  ; We have found the desired bucket in the local cache when no locking
                00EB  ; is required.  The bucket must be valid, therefore simply return
                00EB  ; with success.
                00EB  ;
                00EB
                00EB  10$:    RMSSUC                          ; Set success.
     0C A4  B6  00EE          INCW    BDB$W_USERS(R4)         ; Note in use.
        6F  11  00F1          BRB     SETR5                   ; Branch to set R5 and exit.  .
                00F3
                00F3  ;
                00F3  ; Branch here after getting buffer when no locking.
                00F3  ;
                00F3
                00F3  READ_NOLOCKING:
     0C A5  B6  00F3          INCW    BDB$W_USERS(R5)         ; Note in use.
        5C  11  00F6          BRB     READ_BKT                ; Go read bucket - no lock req'd.
                00F8
      009F  31  00F8  EXBR:   BRW     EXIT
```

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

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

```
                     01D9          .SBTTL  SCAN_LOCKS Search BLB list for BLB.
                     01D9     ;++
                     01D9     ;
                     01D9     ;        SCAN_LOCKS
                     01D9     ;
                     01D9     ;        Scan BLB list for match on desired vbn.  If BDB for that
                     01D9     ;        vbn is found, note BDB address.  If BLB free or this stream
                     01D9     ;        owns it, then note that also.
                     01D9     ;        GBPB's are only reported if this stream already has it
                     01D9     ;        accessed.  (Only 1 GBPB per accessor, ever).
                     01D9     ;
                     01D9     ; Calling sequence:
                     01D9     ;
                     01D9     ;        BSBW    SCAN_LOCKS
                     01D9     ;
                     01D9     ; Input Parameters:
                     01D9     ;
                     01D9     ;        R1 - VBN of bucket desired
                     01D9     ;        R3 - CSH flags.
                     01D9     ;        R9 - IFAB/IRAB address
                     01D9     ;        R10 - IFAB
                     01D9     ;          IFB$L_BLBFLNK - listhead for BLB list
                     01D9     ;
                     01D9     ; Output Parameters:
                     01D9     ;
                     01D9     ;        R0 - success if non-accessed BLB or BLB owned by this
                     01D9     ;                stream is found.
                     01D9     ;           - failure otherwise.
                     01D9     ;
                     01D9     ;        R4 - BLB address if R0 success, undefined otherwise.
                     01D9     ;
                     01D9     ;        R5 - BDB address of BDB for bucket R1 if any present, regardless
                     01D9     ;                of R0 status.  Zero if no BDB for bucket R1 at all.
                     01D9     ;             GBPB address only if this stream currently has it accessed.
                     01D9     ;
                     01D9     ; Side effects:
                     01D9     ;
                     01D9     ;        If existing deferred writeback buffer accessed for a locker,
                     01D9     ;        then the DFW flag in the BLB is cleared.
                     01D9     ;        AP destroyed.
                     01D9     ;--
                     01D9
                     01D9     SCAN_LOCKS:
          55   D4    01D9          CLRL    R5                      ; Init BDB return.
50   0098 CA   DE    01DB          MOVAL   IFB$L_BLBFLNK(R10), R0  ; Save for end test
     54   50   D0    01E0          MOVL    R0, R4                  ; Get start of list
                     01E3
                     01E3          ASSUME  BLB$L_FLNK      EQ      0
                     01E3
     54   64   D0    01E3     10$: MOVL    (R4), R4                ; Get next BLB
     50   54   D1    01E6          CMPL    R4, R0                  ; At end of list?
          57   13    01E9          BEQL    50$                     ; EQL get a free one
51     14 A4   D1    01EB          CMPL    BLB$L_VBN(R4), R1       ; Is this the one?
          F2   12    01EF          BNEQ    10$                     ; NEQ then try next one
                     01F1
                     01F1     ;
                     01F1     ; A BLB for the requested bucket has been located.
```

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

```
                               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  ;++
                    0245  ;
                    0245  ;               GET_BLB
                    0245  ;
                    0245  ;               Look for free BLB from the end of list to the front.
                    0245  ;               A 'free BLB' has a zero vbn field.
                    0245  ;
                    0245  ; Calling sequence:
                    0245  ;
                    0245  ;               BSBW    GET_BLB
                    0245  ;
                    0245  ; Input parameters:
                    0245  ;
                    0245  ;               R10 -   ifab address
                    0245  ;                       BLB$L_FLNK - BLB listhead forward link
                    0245  ;                       BLB$L_BLNK - BLB listhead backward link
                    0245  ;
                    0245  ;               R9  -   structure which will own lock (ifab/irab)
                    0245  ;
                    0245  ;               R1  -   VBN to be accessed by bucket lock
                    0245  ;
                    0245  ; Output parameters:
                    0245  ;
                    0245  ;               R4  -   address of free BLB
                    0245  ;
                    0245  ; Side effects:
                    0245  ;
                    0245  ;               BLB returned in R4 is moved to head of BLB chain.
                    0245  ;               AP destroyed.
                    0245  ;               Bugcheck if no BLB is available.
                    0245  ;
                    0245  ;--
                    0245
                    0245  GET_BLB:
                    0245
                    0245          ASSUME  BLB$L_BLNK        EQ      <BLB$L_FLNK + 4>
                    0245          ASSUME  IFB$L_BLBBLNK     EQ      <IFB$L_BLBFLNK + 4>
                    0245
54   0098 CA   DE   0245          MOVAL   IFB$L_BLBFLNK(R10), R4  ; Get list head.
       5C 54   D0   024A          MOVL    R4, AP                  ; Save for end test.
  54   04 A4   D0   024D  100$:   MOVL    4(R4), R4               ; Get back link.
       5C 54   D1   0251          CMPL    R4, AP                  ; Back at list head?
          16   13   0254          BEQL    110$                    ; If so, then bugcheck
       14 A4   D5   0256          TSTL    BLB$L_VBN(R4)           ; This one free?
          F2   12   0259          BNEQ    100$                    ; No, move on to next one.
  10 A4   59   D0   025B          MOVL    R9, BLB$L_OWNER(R4)     ; Note owner.
  14 A4   51   D0   025F          MOVL    R1, BLB$L_VBN(R4)       ; Note resource.
     54   64   0F   0263          REMQUE  (R4), R4                ; Remove from current place in chain.
0098 CA   64   0E   0266          INSQUE  (R4), IFB$L_BLBFLNK(R10) ; Put in front to find quick.
               05   026B          RSB                             ; Return.
                    026C  110$:
                    026C          RMSPBUG FTL$_NOBLB              ; Should always find one.
```

```
                         0273   ;++
                         0273   ;
                         0273   ;       RM$FREE_LCL
                         0273   ;
                         0273   ; Toss the oldest least valuable buffer out of the cache.  This
                         0273   ; routine is called when the AVLCL count is less than zero, meaning
                         0273   ; that all BLB's are being used either for caching or access of a
                         0273   ; bucket.  It does not necessarily mean that there are no BDB's free
                         0273   ; because multiple BLB's may be tied up referencing a single BDB.
                         0273   ; Nonetheless, a BDB must be tossed out of the cache to free up its
                         0273   ; associated BLB.
                         0273   ;
                         0273   ; Calling sequence:
                         0273   ;
                         0273   ;       BSB     RM$FREE_LCL
                         0273   ;
                         0273   ; Input Parameters:
                         0273   ;
                         0273   ;       R10 -   Ifab address
                         0273   ;           IFB$L_BDB_BLNK - back link of BDB listhead.
                         0273   ;       R9 - Thread to stall (if necessary)
                         0273   ;
                         0273   ; Output Parameters:
                         0273   ;
                         0273   ;       R0  -   status value from call to RM$RELEASE
                         0273   ;               DME - all valid BDB's were in use.
                         0273   ;
                         0273   ; Side effects:
                         0273   ;
                         0273   ;       AP destroyed.
                         0273   ;
                         0273   ;--
                         0273
                         0273  RM$FREE_LCL::
            1E     BB    0273           PUSHR   #^M<R1,R2,R3,R4>        ; Save registers.
            54     D4    0275           CLRL    R4                      ; Init last BDB seen.
      51    01     CE    0277           MNEGL   #1, R1                  ; Init last cache value.
                         027A                                           ; This should always cause the first
                         027A                                           ; BDB possible to be selected.
                         027A           ASSUME  BDB$L_FLINK     EQ      0
                         027A           ASSUME  BDB$L_BLINK     EQ      <BDB$L_FLINK + 4>
                         027A
   50    40 AA     DE    027A           MOVAL   IFB$L_BDB_FLNK(R10), R0 ; Get list head into R0.
      5C    50     D0    027E           MOVL    R0, AP                  ; Save for end test.
                         0281  10$:
   50    04 A0     D0    0281           MOVL    4(R0), R0               ; Get next BDB.
      5C    50     D1    0285           CMPL    R0, AP                  ; End of chain?
            1F     13    0288           BEQL    20$                     ; EQL means at end.
                         028A
                         028A           ASSUME  BDB$B_BID       EQ      GBPB$B_BID
                         028A           ASSUME  <BDB$C_BID & 1> EQ      0
                         028A           ASSUME  <GBPB$C_BID & 1> EQ     1
                         028A
   F3 08 A0        E8    028A           BLBS    BDB$B_BID(R0), 10$      ; Continue scan if gbpb.
      0C A0        B5    028F           TSTW    BDB$W_USERS(R0)         ; Is this one accessed?
         EE        12    0291           BNEQ    10$                     ; NEQ it is - keep going.
      1C A0        D5    0293           TSTL    BDB$L_VBN(R0)           ; See if valid.
```

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

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

```
                     0331  ;++
                     0331  ;
                     0331  ;       SCAN_LCL_CACHE
                     0331  ;
                     0331  ;       Search list of all BDB's for match on the VBN field.
                     0331  ;
                     0331  ; Calling sequence:
                     0331  ;
                     0331  ;       BSB     SCAN_LCL_CACHE
                     0331  ;
                     0331  ; Input Parameters:
                     0331  ;
                     0331  ;       R10 - ifab address
                     0331  ;           IFB$L_BDB_FLNK - forward link of BDB listhead.
                     0331  ;
                     0331  ;       R1 - VBN of bucket desired
                     0331  ;
                     0331  ; Output Parameters:
                     0331  ;
                     0331  ;       R5 - address of BDB that matches desired VBN or 0 if not found
                     0331  ;--
                     0331
                     0331  SCAN_LCL_CACHE:
55    40 AA     DE   0331          MOVAL   IFB$L_BDB_FLNK(R10), R5 ; Address of BDB listhead.
   5C    55     D0   0335          MOVL    R5, AP                  ; Save for end test.
                     0338
                     0338          ASSUME  BDB$L_FLINK     EQ      0
                     0338
   55    65     D0   0338  10$:    MOVL    (R5), R5                ; Get next BDB.
   5C    55     D1   033B          CMPL    R5, AP                  ; At end of list yet?
         0B     13   033E          BEQL    50$                     ; EQL yes, quit.
51    1C A5     D1   0340          CMPL    BDB$L_VBN(R5), R1       ; Is this the right one?
         F2     12   0344          BNEQ    10$                     ; No, try next one.
                     0346
                     0346          ASSUME  <BDB$C_BID & 1> EQ      0
                     0346          ASSUME  <GBPB$C_BID & 1> EQ     1
                     0346
   EE 08 A5     E8   0346          BLBS    BDB$B_BID(R5), 10$      ; Not a BDB.  Continue search.
                05   034A  20$:    RSB                             ; Return.
         55     D4   034B  50$:    CLRL    R5                      ; Note failure.
                05   034D          RSB                             ; Return.
```

```
                    034E  ;++
                    034E  ;
                    034E  ;      RM$GET_LCL_BUFF
                    034E  ;
                    034E  ;         Search list of BDB's from the end of list to the front for
                    034E  ;         one with VBN=0, USERS=0, and SIZE large enough.
                    034E  ;
                    034E  ; Calling sequence:
                    034E  ;
                    034E  ;         BSBW    RM$GET_LCL_BUFF
                    034E  ;
                    034E  ; Input Parameters:
                    034E  ;
                    034E  ;         R1 - VBN desired
                    034E  ;         R2 - Size of bucket
                    034E  ;         R10 - Ifab address
                    034E  ;           BDB_BLNK - Back BDB link
                    034E  ;
                    034E  ; Output Parameters:
                    034E  ;
                    034E  ;         R5 - BDB address
                    034E  ;           VBN set to R1
                    034E  ;           NUMB set to R2
                    034E  ;
                    034E  ; Side effects:
                    034E  ;
                    034E  ;         NOLCLBUF bugcheck if no BDB is found.
                    034E  ;
                    034E  ;--
                    034E
                    034E  RM$GET_LCL_BUFF::
                    034E
                    034E          ASSUME  BDB$L_FLINK       EQ      0
                    034E          ASSUME  BDB$L_BLINK       EQ      <BDB$L_FLINK + 4>
                    034E          ASSUME  IFB$L_BDB_BLNK    EQ      <IFB$L_BDB_FLNK + 4>
                    034E
55    40 AA    DE   034E          MOVAL   IFB$L_BDB_FLNK(R10), R5 ; Get BDB list head.
   5C    55    D0   0352          MOVL    R5, AP                  ; Save for end test.
                    0355  10$:
55    04 A5    D0   0355          MOVL    4(R5), R5               ; Get next BDB.
   5C    55    D1   0359          CMPL    R5, AP                  ; At end of list?
         28    13   035C          BEQL    100$                    ; Bug if none found in list.
      1C A5    D5   035E          TSTL    BDB$L_VBN(R5)           ; This buffer free?
         F2    12   0361          BNEQ    10$                     ; NEQ it's in use.  Continue search.
      0C A5    B5   0363          TSTW    BDB$W_USERS(R5)         ; In use?
         ED    12   0366          BNEQ    10$                     ; NEQ in use, try another.
                    0368
                    0368          ASSUME  <BDB$C_BID & 1> EQ      0
                    0368          ASSUME  <GBPB$C_BID & 1> EQ     1
                    0368
  E9 08 A5     E8   0368          BLBS    BDB$B_BID(R5), 10$      ; It's not a BDB.  Continue search.
52    16 A5    B1   036C          CMPW    BDB$W_SIZE(R5), R2      ; Is this buffer large enough?
         E3    1F   0370          BLSSU   10$                     ; LSSU not big enough - try another.
                    0372
      10 A5    D4   0372          CLRL    BDB$L_BLB_PTR(R5)       ; Make sure this is 0.
1C A5    51    D0   0375          MOVL    R1, BDB$L_VBN(R5)       ; Found one- store VBN.
14 A5    52    B0   0379          MOVW    R2, BDB$W_NUMB(R5)      ; Store size desired.
```

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

```
038D  ;++
038D  ;
038D  ;       LOCK_BKT
038D  ;
038D  ;       This routine is called to obtain a lock for the requested bucket
038D  ;       for either read only or modify.  In addition, it may also be known
038D  ;       that the lock must be of a high enough mode to interlock an i/o
038D  ;       operation.  The following lock manager modes are used:
038D  ;
038D  ;       LCK$K_PWMODE if holding modified buffer.  No conversion will
038D  ;               be done if this is not a LOCK request to cache.
038D  ;       LCK$K_EXMODE for all locks.
038D  ;
038D  ; Calling sequence:
038D  ;
038D  ;       BSBW    LOCK_BKT
038D  ;
038D  ; Input Parameters:
038D  ;
038D  ;       R10 -   ifab address
038D  ;           IFB$L_SFSB_PTR - pointer to shared file synchronization block
038D  ;               SFSB$L_LOCK_ID - lock id of shared file lock
038D  ;
038D  ;       R9  -   ifab/irab address
038D  ;           IRB$B_EFN - event flag to use.
038D  ;
038D  ;       R4  -   BLB address
038D  ;           BLB$B_MODEHELD - mode of lock currently held.
038D  ;           BLB$L_LOCK_ID - ID of lock if one already held
038D  ;           BLB$L_RESDSC - resource name descriptor (should point to BLB$L_VBN)
038D  ;
038D  ;       R3  -   Cache flags (same as input to RM$CACHE).
038D  ;
038D  ; Output Parameters:
038D  ;
038D  ;       R0  -   status value of $ENQ service after call to RM$MAPERR.
038D  ;
038D  ; Side effects:
038D  ;
038D  ;       BLB$L_LKSTS field contains value of ENQ service.
038D  ;       BLB$L_VALBLK contains value block for lock requested.
038D  ;       BLB$L_OWNER is set to R9.
038D  ;       BLB$B_MODEHELD contains mode of lock obtained.
038D  ;       BLB$B_BLBFLGS contain the cache flags also.
038D  ;
038D  ;       R1 is always destroyed.
038D  ;       If lock not granted synchonously, will return at AST level with
038D  ;       event flag (IRB$B_EFN) set and AP, and R2 will be destroyed.
038D  ;
038D  ;--
038D
038D  LOCK_BKT:
038D
038D          ASSUME  CSH$V_LOCK      EQ      BLB$V_LOCK
038D          ASSUME  CSH$V_NOWAIT    EQ      BLB$V_NOWAIT
038D          ASSUME  CSH$V_NOREAD    EQ      BLB$V_NOREAD
038D          ASSUME  CSH$V_NOBUFFER  EQ      BLB$V_NOBUFFER
```

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

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

```
                        045A  ;++
                        045A  ;
                        045A  ; SCAN_GBL
                        045A  ;
                        045A  ; Scan global cache for match on desired bucket.
                        045A  ;
                        045A  ; Calling sequence:
                        045A  ;
                        045A  ;       BSBW    SCAN_GBL
                        045A  ;
                        045A  ; Input parameters:
                        045A  ;
                        045A  ;       R10 - ifab address
                        045A  ;          GBH_PTR - pointer to global buffer header area
                        045A  ;
                        045A  ;       R1 - vbn of desired bucket
                        045A  ;
                        045A  ; Output parameters:
                        045A  ;
                        045A  ;       R0 - success
                        045A  ;       R5 - GBPB address
                        045A  ; else
                        045A  ;       R0 - failure (0)
                        045A  ;       R5 - addr of next lower valued GBD or zero if no GBPB found.
                        045A  ;
                        045A  ; Side effects:
                        045A  ;       AP destroyed.
                        045A  ;       EXclusive lock on global buffer section is obtained and kept.
                        045A  ;
                        045A  ;--
                        045A
                        045A  SCAN_GBL:
                        045A          ASSUME  GBH$L_GBD_FLNK  EQ      0
                        045A          ASSUME  GBH$L_GBD_BLNK  EQ      4
                        045A          ASSUME  GBD$L_FLINK     EQ      0
                        045A          ASSUME  GBD$L_BLINK     EQ      4
                        045A          ASSUME  GBD$L_VBN       EQ      GBH$L_HI_VBN
                        045A
           FBA3'  30    045A          BSBW    RM$RAISE_GBS_LOCK       ; Get EX lock on GBS to search cache.
  50   0088 CA    D0    045D          MOVL    IFB$L_GBH_PTR(R10), R0  ; Get pointer to gbl header.
       5C   50    D0    0462          MOVL    R0, AP                  ; Save for later.
                        0465
       50   60    C0    0465  10$:    ADDL2   (R0), R0                ; Get address of next GBD element.
  51     0C A0    D1    0468          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
            2D    1E    046C          BGEQU   20$                     ; GEQU either found it or not here.
       50   60    C0    046E          ADDL2   (R0), R0                ; Get address of next GBD element.
  51     0C A0    D1    0471          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
            24    1E    0475          BGEQU   20$                     ; GEQU either found it or not here.
       50   60    C0    0477          ADDL2   (R0), R0                ; Get address of next GBD element.
  51     0C A0    D1    047A          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
            1B    1E    047E          BGEQU   20$                     ; GEQU either found it or not here.
       50   60    C0    0480          ADDL2   (R0), R0                ; Get address of next GBD element.
  51     0C A0    D1    0483          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
            12    1E    0487          BGEQU   20$                     ; GEQU either found it or not here.
       50   60    C0    0489          ADDL2   (R0), R0                ; Get address of next GBD element.
  51     0C A0    D1    048C          CMPL    GBD$L_VBN(R0), R1       ; Is this one desired bucket?
            09    1E    0490          BGEQU   20$                     ; 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  ;++
                      04AB  ;
                      04AB  ; GET_GBPB
                      04AB  ;
                      04AB  ; Branch or drop through to this point after locating GBD to get a
                      04AB  ; GBPB and point it to GBD.
                      04AB  ;
                      04AB  ; Input parameters:
                      04AB  ;           R10 - ifab
                      04AB  ;              BDB_FLNK - BDB (and GBPB) listhead.
                      04AB  ;           R0 - GBD address
                      04AB  ;
                      04AB  ; Output parameters:
                      04AB  ; If        R0 - success
                      04AB  ; then      R5 - GBPB address
                      04AB  ; else      R5 = 0 if GBPB not found.
                      04AB  ;
                      04AB  ;--
                      04AB  GET_GBPB:
       20 A0    B6    04AB          INCW    GBD$W_USECNT(R0)          ; Bump use count in GBD (we are assuming
                      04AE                                            ;  that a GBPB will be found).
          50    DD    04AE          PUSHL   R0                        ; Save GBD around lock manager call.
       FB4D'    30    04B0          BSBW    RM$LOWER_GBS_LOCK         ; Lower lock on GBS to NL.
       50 8ED0        04B3          POPL    R0                        ; Restore GBD.
                      04B6
   55   40 AA   DE    04B6          MOVAL   IFB$L_BDB_FLNK(R10), R5   ; BDB/GBPB listhead.
     5C    55   D0    04BA          MOVL    R5, AP                    ; Save for end test.
                      04BD  10$:
                      04BD          ASSUME  BDB$L_FLINK     EQ      0
                      04BD          ASSUME  BDB$L_BLINK     EQ      4
                      04BD
   55   04 A5   D0    04BD          MOVL    4(R5), R5                 ; Get next element.(going backwards)
     5C    55   D1    04C1          CMPL    R5, AP                    ; Back to listhead yet?
           2F   13    04C4          BEQL    100$                      ; That's a bug.
                      04C6
                      04C6          ASSUME  BDB$B_BID       EQ      GBPB$B_BID
                      04C6          ASSUME  <BDB$C_BID & 1> EQ      0
                      04C6          ASSUME  <GBPB$C_BID & 1> EQ     1
                      04C6
     F3 08 A5   E9    04C6          BLBC    BDB$B_BID(R5), 10$        ; Branch if BDB and keep looking.
        0C A5   B5    04CA          TSTW    GBPB$W_USERS(R5)          ; Make sure it's not in use.
           EE   12    04CD          BNEQ    10$                       ; NEQ it is, so keep looking.
                      04CF  ;
                      04CF  ; Found the GBPB.  Fill in relevant data from GBD.
                      04CF  ;
   24 A5   50   D0    04CF          MOVL    R0, GBPB$L_GBD_PTR(R5)    ; Pointer to GBD.
 1C A5   0C A0  D0    04D3          MOVL    GBD$L_VBN(R0), GBPB$L_VBN(R5) ; Bucket vbn.
        10 A0   D0    04D8          MOVL    GBD$L_VBNSEQNUM(R0),-     ; Move sequence number from GBD to GBPB.
   20 A5              04DB                  GBPB$L_VBNSEQNO(R5)
                      04DD
                      04DD          ASSUME  <GBD$W_NUMB + 2>          EQ      GBD$W_SIZE
                      04DD          ASSUME  <GBPB$W_NUMB + 2>         EQ      GBPB$W_SIZE
                      04DD
 14 A5   18 A0  D0    04DD          MOVL    GBD$W_NUMB(R0), GBPB$W_NUMB(R5) ; Numb and size fields.
           11   90    04E2          MOVB    #BDB$M_VAL!BDB$M_NOLOCATE,- ; Init flags.
     0A A5            04E4                  GBPB$B_FLGS(R5)           ;
 5C    0088 CA  D0    04E6          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  ;++
                        04FD  ;
                        04FD  ; FIND_FREE_GBL
                        04FD  ;
                        04FD  ; This routine is called when the desired bucket was not in the
                        04FD  ; global cache.  Either find a free global buffer or toss one out
                        04FD  ; of the cache to use.
                        04FD  ;
                        04FD  ; We already have an exclusive lock on the GBS from calling SCAN_GBL.
                        04FD  ;
                        04FD  ; Input parameters:
                        04FD  ;         R5 - Address of preceding GBD.
                        04FD  ;
                        04FD  ; Output parameters:
                        04FD  ;
                        04FD  ; R0 - success, failure
                        04FD  ; R5 - GBPB addr if success.
                        04FD  ;      - 0 if failure.
                        04FD  ;
                        04FD  ;--
                        04FD  FIND_FREE_GBL:
                3C   BB 04FD          PUSHR   #^M<R2,R3,R4,R5>
                53   7C 04FF          CLRQ    R3                          ; Init lowest val, gbpb found.
                53   D7 0501          DECL    R3                          ; Make lowval -1.
      5C   0088 CA   D0 0503          MOVL    IFB$L_GBH_PTR(R10), AP      ; Get gb' header.
   50    5C  04 AC   C1 0508          ADDL3   GBH$L_GBD_BLNK(AP), AP, R0  ; Get last GBD in list.
         53  0C A0   D1 050D          CMPL    GBD$L_VBN(R0), R3           ; Is the vbn -1?
                6F   13 0511          BEQL    GOT_GBD                     ; EQL then just use this one.
                        0513  ;
                        0513  ; There are no available GBD's at the end of the list.
                        0513  ; Will have to select GBD to toss out of the cache.
                        0513  ;
                7E   D4 0513          CLRL    -(SP)                       ; Note first pass.
   55    5C  2C AC   C1 0515          ADDL3   GBH$L_GBD_END(AP), AP, R5   ; End addr of scan.
   50    5C  30 AC   C1 051A          ADDL3   GBH$L_GBD_NEXT(AP), AP, R0  ; Starting point of scan.
             28 AC   D1 051F          CMPL    GBH$L_GBD_START(AP),-       ; Is NEXT pointer the
             30 AC      0522                  GBH$L_GBD_NEXT(AP)          ; same as the START of GBD's?
                02   12 0524          BNEQ    10$                         ; NEQ then make two passes.
                6E   D6 0526          INCL    (SP)                        ; Else only make one.
         52  34 AC   D0 0528  10$:    MOVL    GBH$L_SCAN_NUM(AP), R2      ; Number of GBD's to search.
                        052C  LTOP:
             20 A0   B5 052C          TSTW    GBD$W_USECNT(R0)            ; This one in use?
                13   12 052F          BNEQ    LTST                        ; NEQ it is. Br to loop test.
         53  0B A0   91 0531          CMPB    GBD$B_CACHE_VAL(R0), R3     ; Is this lowest cache value seen?
                09   1E 0535          BGEQU   SCANTST                     ; GEQU lower has been seen.  Ignore it.
             54 50   D0 0537          MOVL    R0, R4                      ; Save this GBD addr.
         53  0B A0   9A 053A          MOVZBL  GBD$B_CACHE_VAL(R0), R3     ; Save this cache value.
                1D   13 053E          BEQL    USE_GBD                     ; Use first zero value one found.
                        0540  SCANTST:
                52   D7 0540          DECL    R2                          ; Keep scanning if counter not run out.
                19   15 0542          BLEQ    USE_GBD                     ; Use what's been found.
                        0544  LTST:
FFE2 50   28    55   F1 0544          ACBL    R5, #GBD$C_BLN, R0, LTOP    ; Keep going if limit not hit yet.
                        054A
       0F 6E    00   E2 054A          BBSS    #0, (SP), USE_GBD           ; Br if second pass.
   50    5C  28 AC   C1 054E          ADDL3   GBH$L_GBD_START(AP), AP, R0 ; Start at beginning this time.
   55    5C  30 AC   C1 0553          ADDL3   GBH$L_GBD_NEXT(AP), AP, R5  ; End with current next ptr.
```

```
              55    28  C2 0558  1454          SUBL2   #GBD$C_BLN, R5                ; Back up one - did it first pass.
                    CF  11 055B  1455          BRB     LTOP                          ; Branch into loop.
                           055D  1456 USE_GBD:
              5E    04  C0 055D  1457          ADDL2   #4, SP                        ; Pop pass counter off stack.
        50    5C 30 AC  C1 0560  1458          ADDL3   GBH$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
   3C   BA  05CA  ERXIT:   POPR     #^M<R2,R3,R4,R5>          ; Restore registers.
FA31'   30  05CC           BSBW     RM$LOWER_GBS_LOCK         ; Release our EX lock on the GBS.
   55   D4  05CF           CLRL     R5                        ; Note no GBPB.
   50   D4  05D1           CLRL     R0                        ; Note failure in R0.
        05  05D3           RSB                                ; Return.
            05D4
            05D4           .END
```

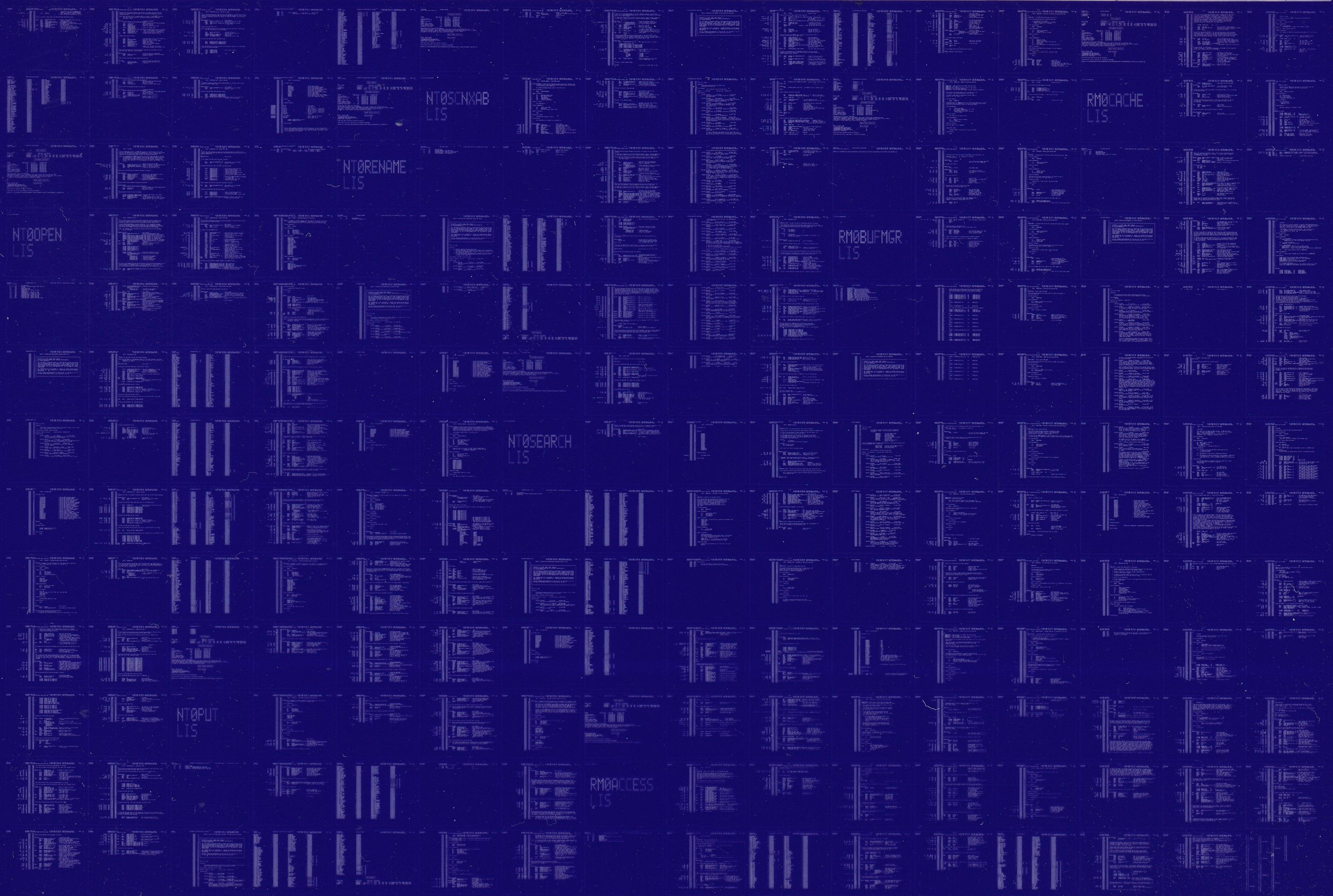
```
$$.PSECT_EP                   = 00000000              CSH$V_NOWAIT                  = 00000001
$$ARGS                        = 0000000B              CSH_MASK_ALL                  = 0000000F
$$RMSTEST                     = 0000001A              ENQ$_ACMODE                   = 00000028
$$RMS_PBUGCHK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

```
GET_BUFF                         000000CF R      01    RM$STALL                     ********   X  01
GET_GBPB                         000004AB R      01    RM$STALLAST                  ********   X  01
GOT_BUFF                         000000DA R      01    RMS$GW_GBLBUFQUO             ********   X  01
GOT_GBD                          00000582 R      01    RMS$_DME                   = 000184D4
IFB$B_BID                      = 00000008              RMS$_ENQ                   = 0001C134
IFB$B_JNLFLG                   = 000000A0              RMS$_RLK                   = 000182AA
IFB$B_ORGCASE                  = 00000023              RMS$_WER                   = 0001C114
IFB$C_BID                      = 0000000B              SCANTST                      00000540 R    01
IFB$C_IDX                      = 00000002              SCAN_GBL                     0000045A R    01
IFB$L_BDB_BLNK                 = 00000044              SCAN_LCL_CACHE               00000331 R    01
IFB$L_BDB_FLNK                 = 00000040              SCAN_LOCKS                   000001D9 R    01
IFB$L_BLBBLNK                  = 0000009C              SETR5                        00000162 R    01
IFB$L_BLBFLNK                  = 00000098              SS$_DEADLOCK               = 00000E0A
IFB$L_GBH_PTR                  = 00000088              SS$_EXENQLM                = 00002A44
IFB$L_PAR_LOCK_ID              = 00000080              SS$_NOLOCKID               = 00000E12
IFB$V_BI                       = 00000002              SS$_SYNCH                  = 00000689
IFB$V_MSE                      = 00000031              SS$_VALNOTVALID            = 000009F0
IFB$V_NORECLK                  = 00000033              SYS$DEQ                      ********  GX  01
IFB$W_AVLCL                    = 00000084              SYS$ENQ                      ********  GX  01
IRB$B_BID                      = 00000008              SYS$SETEF                    ********  GX  01
IRB$B_EFN                      = 0000000B              TRACE                        00000000 R    01
IRB$C_BID                      = 0000000A              USE_GBD                      0000055D R    01
IRB$V_GBLBUFF                  = 00000036              WBKERR                       000000FB R    01
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
RM$BUG                           ********   X    01
RM$CACHE                         00000012 RG     01
RM$CACH_IN                       ********   X    01
RM$CACH_OUT                      ********   X    01
RM$FREE_LCL                      00000273 RG     01
RM$GET_LCL_BUFF                  0000034E RG     01
RM$LOWER_GBS_LOCK                ********   X    01
RM$MAPERR                        ********   X    01
RM$RAISE_GBS_LOCK                ********   X    01
RM$RDBUFWT                       ********   X    01
RM$RELEASE                       ********   X    01
RM$SETEFN                        ********   X    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
RMS$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
------------------                       --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                   22
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                    1
_$255$DUA28:[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
```

0317 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0318 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

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