

```

RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSS
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
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSS

```

53

Syn

NTS
NTS

NTS

NTS
NTS

NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTSNTS
NTS

NTS

NTS

NTS
NTSNTS
NTSNTS
NTS

NTS

NTS
NTSNTS
NTSNTS
NTSNTS
NTS

NTS

NT
NT

NTS

NT
NT

NT

PI

```
RRRRRRRR  MM      MM      000000  RRRRRRRR  EEEEEEEEE  LL      EEEEEEEEE  AAAAAA  SSSSSSSS
RRRRRRRR  MM      MM      000000  RRRRRRRR  EEEEEEEEE  LL      EEEEEEEEE  AAAAAA  SSSSSSSS
RR      RR  MMMM  MMMM  00      00  RR      RR  EE      LL      EE      AA      AA  SS
RR      RR  MMMM  MMMM  00      00  RR      RR  EE      LL      EE      AA      AA  SS
RR      RR  MM      MM  00      00  RR      RR  EE      LL      EE      AA      AA  SS
RRRRRRRR  MM      MM  00      00  RRRRRRRR  EEEEEEEEE  LL      EEEEEEEEE  AA      AA  SSSSSS
RRRRRRRR  MM      MM  00      00  RRRRRRRR  EEEEEEEEE  LL      EEEEEEEEE  AA      AA  SSSSSS
RR      RR  MM      MM  0000      00  RR      RR  EE      LL      EE      AAAAAAAAAA  SS
RR      RR  MM      MM  0000      00  RR      RR  EE      LL      EE      AAAAAAAAAA  SS
RR      RR  MM      MM  00      00  RR      RR  EE      LL      EE      AA      AA  SS
RR      RR  MM      MM  00      00  RR      RR  EE      LL      EE      AA      AA  SS
RR      RR  MM      MM  000000  RR      RR  EEEEEEEEE  LLLLLLLLLL  EEEEEEEEE  AA      AA  SSSSSSSS
RR      RR  MM      MM  000000  RR      RR  EEEEEEEEE  LLLLLLLLLL  EEEEEEEEE  AA      AA  SSSSSSSS
                                     ....
                                     ....
                                     ....
                                     ....

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
```


(2) 159
(3) 196
(5) 358
(6) 436

DECLARATIONS
RMSRELEASE - RELEASE BUFFER FOR ALL FILE ORGS
SUPPORT ROUTINES AND INFREQUENT PATHS
SHARED FILE SUPPORT (RELATIVE AND INDEXED)

```
0000 1      $BEGIN  RMORELEAS,000,RM$RMS0,<RELEASE BUFFER ROUTINE>
0000 2
0000 3
0000 4      :*****
0000 5      :*
0000 6      :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7      :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8      :*  ALL RIGHTS RESERVED.
0000 9      :*
0000 10     :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11     :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12     :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13     :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14     :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15     :*  TRANSFERRED.
0000 16     :*
0000 17     :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18     :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19     :*  CORPORATION.
0000 20     :*
0000 21     :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22     :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23     :*
0000 24     :*
0000 25     :*****
0000 26
0000 27     :++
0000 28     : Facility: rms32
0000 29
0000 30     : Abstract:
0000 31     :         this module releases a buffer for all file
0000 32     :         organizations, handling writing if dirty
0000 33     :         and sharing.
0000 34
0000 35     : Environment:
0000 36     :         star processor running starlet exec.
0000 37
0000 38     : Author: L F Laverdure,          creation date: 30-MAR-1977
0000 39
0000 40     : Modified By:
0000 41
0000 42     :         V03-022 JEJ0044          J E Johnson          21-Jun-1984
0000 43     :         Tweak the instructions for a performance boost.
0000 44
0000 45     :         V03-021 JEJ0037          J E Johnson          14-May-1984
0000 46     :         Have BLKASTFIN save all the registers that it uses.
0000 47
0000 48     :         V03-020 JEJ0036          J E Johnson          11-May-1984
0000 49     :         Fix bug in INVGBL that was counting on AP being preserved
0000 50     :         across a possible stall.
0000 51
0000 52     :         V03-019 JEJ0033          J E Johnson          02-May-1984
0000 53     :         Fix bug in BLOCK_AST so that RMS will bugcheck if it
0000 54     :         runs out of AST quota in a state with no path available
0000 55     :         back to the user.
0000 56
0000 57     :         V03-018 SHZ0007          Stephen H. Zalewski,    30-Apr-1984
```


0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :
0000 80 :
0000 81 :
0000 82 :
0000 83 :
0000 84 :
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :

Fix bug in BLKASTFIN that was clearing the owner field
of the BLB when it wasn't supposed to. This was causing 2
streams to \$ENQ for a bucket using the same lock id.

Also, fix bug where 2 streams would convert their bucket locks
to system locks for the same bucket, thus causing one system
lock to be lost.

V03-017 JEJ0008 J E Johnson 16-Mar-1984
Add in global buffer quota accounting.

V03-016 JWT0160 Jim Teague 29-Feb-1984
Remove calls to RMS\$DEALLEFN.

V03-015 SHZ0006 Stephen H. Zalewski 01-Dec-1983
Clear BLB flags before doing the conversion to a PW lock
instead of after because if a blocking ast arrived between the
conversion and the clearing of the flags, the wrong
flags would be cleared.

V03-014 SHZ0005 Stephen H. Zalewski 20-Oct-1983
When a global buffer is invalidated, the user count on the
GBD must be greater than one for it to remain in the cache.

V03-013 SHZ0004 Stephen H. Zalewski 28-Jul-1983
Add support for cluster global buffers.

V03-012 KPL0001 Peter Lieberwirth 21-Jun-1983
Change use of JNLFLG to JNLFLG2.

V03-011 TMK0001 Todd M. Katz 20-Feb-1983
Add support for AI and BI Journalling of ISAM files. Whenever
RMS\$RELEASE is called with a dirty buffer and the file's
organization is index, then the original bucket contents are
journalled if BI Journalling is enabled, and the modified
bucket's contents are journalled if AI Journalling is enabled
before the bucket is actually released.

V03-010 JWH0175 Jeffrey W. Horn 31-Jan-1983
Save some more registers around call to RMS\$FRCJNL.

V03-009 SHZ0003 Stephen H. Zalewski, 18-Jan-1983 15:40
Added code to bugcheck if \$ENQ or \$DCLAST system services
fail.

V03-008 SHZ0002 Stephen H. Zalewski, 22-Sep-1982 13:53
Take EXCLUSIVE lock on Global buffer cache when invalidating
a buffer, or updating the position of a buffer in the cache.

V03-007 SHZ0001 Stephen H. Zalewski, 6-Sep-1982 20:21
Use interlocked self-relative queue instructions when
placing and removing GBDs.

V03-006 KBT0214 Keith B. Thompson 23-Aug-1982
Reorganize psects

V03-005 JWH0001 Jeffrey W. Horn 28-Jun-1982


```
0000 115 : Perform journal force using high water mark(s) stored
0000 116 : in BDB$_JNLSEQ before writing out a buffer.
0000 117 :
0000 118 : V03-004 CDS0019 C D Saether 13-Apr-1982
0000 119 : If write back qio (in blocking ast routine) fails,
0000 120 : note error, reset DFW flag and dismiss AST without
0000 121 : releasing lock, and setting WRITEBACK.
0000 122 : If a reader finds DFW and WRITEBACK set, the buffer
0000 123 : is written on release (a locker also writes
0000 124 : it back if WRITEBACK is set).
0000 125 : Invalidate global buffer if GBPB released with
0000 126 : VAL flag cleared.
0000 127 :
0000 128 : V03-003 CDS0018 C D Saether 5-Apr-1982
0000 129 : Force total release of buffer when released and invalid.
0000 130 :
0000 131 : V03-002 CDS0017 C D Saether 31-Mar-1982
0000 132 : Fixup BLB pointer from BDB when doing KEEPLOCK release
0000 133 : and other accessors are present.
0000 134 :
0000 135 : V03-001 CDS0016 C D Saether 1-Mar-1982
0000 136 : Decrement GBD use count in code that clears out
0000 137 : GBPB as all GBPB's are dequeued. Increment use
0000 138 : count on GBPB before calling routine to free local
0000 139 : buffer in case stall occurs so that other streams
0000 140 : will not select it.
0000 141 : Only clear out BDB on keeplock release if use count
0000 142 : is zero.
0000 143 : Bugcheck if a reader (didn't say LOCK initially)
0000 144 : has modified a buffer and the write was deferred.
0000 145 : If this happens we get into trouble further on.
0000 146 : 26-Feb-1982
0000 147 : Reflect fact that all accesses except read lock on
0000 148 : DFW buffer hold lock in EX mode. Situation previously
0000 149 : being checked where CR lock held on release of dirty
0000 150 : buffer no longer possible.
0000 151 : 17-Feb-1982
0000 152 : Changes to allow for release of modified global
0000 153 : buffer.
0000 154 :
0000 155 : --
0000 156 :
0000 157 :
```



```

0000 159      .SBTTL  DECLARATIONS
0000 160
0000 161  :
0000 162  : Include Files:
0000 163  :
0000 164  :
0000 165  :
0000 166  : Macros:
0000 167  :
0000 168
0000 169      $BDBDEF
0000 170      $BLBDEF
0000 171      $CJFDEF
0000 172      $CSHDEF
0000 173      $DEQDEF
0000 174      $ENQDEF
0000 175      $FABDEF
0000 176      $GBDDEF
0000 177      $GBHDEF
0000 178      $GBPBDEF
0000 179      $IFBDEF
0000 180      $IMPDEF
0000 181      $IODEF
0000 182      $IRBDEF
0000 183      $LCKDEF
0000 184      $RLSDEF
0000 185      $RMSDEF
0000 186
0000 187  :
0000 188  : Equated Symbols:
0000 189  :
0000 190
0000 191  :
0000 192  : Own Storage:
0000 193  :
0000 194

```



```
0000 196      .SBTTL RM$RELEASE - RELEASE BUFFER FOR ALL FILE ORGS
0000 197
0000 198 :++
0000 199 : RM$RELEASE
0000 200 :
0000 201 : For all file organizations:
0000 202 :   1) if the buffer valid and dirty and
0000 203 :       any of the flags rls$wrt_thru, rls$keep_lock* or
0000 204 :       rls$return set, writes the buffer out,
0000 205 :       possibly writing behind
0000 206 :       This routine is not prepared for anyone to call it with
0000 207 :       rls$return set and R4 pointing to a GBP, however.
0000 208 :   2) if the flag rls$return set, waits for
0000 209 :       read-ahead or write-behind to complete,
0000 210 :       if any, and returns the bdb and
0000 211 :       buffer to the free memory lists
0000 212 :   3) if the rls$keep_lock* bit off release access
0000 213 :       to the block; otherwise, retain access to block
0000 214 :   4) if entry rm$rlnerr is invoked clear all error
0000 215 :       producing bdb flags, clear release control flags
0000 216 :       and fall thru to rm$release
0000 217 :   5) if an error occurs and rls$keep_acc is not set in flags,
0000 218 :       call rm$rlnerr before returning (i.e., deaccess bdb)
0000 219 :
0000 220 :   * note: rls$keep_lock is only input for the
0000 221 :       relative and indexed file organizations.
0000 222 :
0000 223 : Calling sequence:
0000 224 :
0000 225 :     bsbw    rm$release
0000 226 :     bsbw    rm$rlnerr      - alternate entry to merely free the bdb
0000 227 :
0000 228 : Input Parameters:
0000 229 :
0000 230 :
0000 231 :     r11     impure area address
0000 232 :     r10     ifab address
0000 233 :     r9      irab/ifab address
0000 234 :     r8      rab/fab address
0000 235 :     r4      bdb address
0000 236 :     r3      release control flags (not an input for entry at rm$rlnerr)
0000 237 :
0000 238 : Implicit Inputs:
0000 239 :
0000 240 :     ifb$b_caseorg
0000 241 :     bdb$b_flg
0000 242 :     if block written, all inputs to rm$seqwtbd
0000 243 :
0000 244 : Output Parameters:
0000 245 :
0000 246 :     r0      status code
0000 247 :     r1,r2,ap destroyed
0000 248 :     r3      zeroed if entry at rm$rlnerr otherwise unchanged
0000 249 :     r4,r5   destroyed if rls$return set
0000 250 :
0000 251 : Implicit Outputs:
0000 252 :
```



```

0000 253 :      bdb$u_drt - cleared if not deferred write
0000 254 :      bdb$u_val and bdb$u_prm cleared on entry at rm$rlnerr
0000 255 :      the various outputs from rm$seqwtbd
0000 256 :
0000 257 : Completion Codes:
0000 258 :
0000 259 :      standard rms.
0000 260 :
0000 261 : Side Effects:
0000 262 :
0000 263 :      Journalling of the modified and/or un-modified contents of the bucket
0000 264 :      being released may have been done if the file is an ISAM file marked
0000 265 :      for AI or BI Journalling.
0000 266 :      May have switched to running at ast level.
0000 267 :--
0000 268 :

```

```
00000000'EF 16 0000 270 TRACE:
00000000'EF 16 0000 271 JSB RMSRLS_IN
00000000'EF 16 0006 272 BSBB RLS
00000000'EF 16 0008 273 JSB RMSRLS_OUT
05 000E 274 RSB
000F 275
000F 276
000F 277
000F 278 : entry for unconditional release
000F 279 : clear all possible error producing flags and fall thru
000F 280
000F 281
000F 282 RMSRLNERR::
0A A4 0B 8A 000F 283 BICB2 #BDB$M_VAL!BDB$M_DRT!BDB$M_PRM,BDB$B_FLGS(R4)
53 08 D0 0013 284 RMSRLNER1::
0016 285 MOVL #RLS$M_DEQ, R3 ; Release entirely.
0016 286
0016 287 : normal entry
0016 288
0016 289
0016 290
0016 291 RMSRELEASE::
0016 292 $TSTPT RELEASE
01 001C 293 NOP ; Patch for tracing.
01 001D 294 NOP
001E 295
53 DD 001E 296 RLS: PUSHL R3 ; save flags
54 D5 0020 297 TSTL R4 ; Is there a structure?
2A 13 0022 298 BEQL SUCXIT ; Assume this was lock blb and no locks.
0024 299 ASSUME BLB$B-BID EQ BDB$B-BID
0024 300 ASSUME BDB$B-BID EQ GBP$B-BID
08 A4 10 91 0024 301 CMPB #BLB$C-BID, BDB$B-BID(R4) ; Is this a BLB?
33 13 0028 302 BEQL ISBLB ; This is lock BLB.
002A 303 ASSUME BDB$V_VAL EQ 0
07 0A A4 E8 002A 304 BLBS BDB$B_FLGS(R4),10$ ; branch if buffer valid
0A A4 02 8A 002E 305 BICB2 #BDB$M_DRT,BDB$B_FLGS(R4); make sure dirty not set
6E 08 C8 0032 306 BISL2 #RLS$M_DEQ, (SP) ; force total release if invalid.
0035 307 10$:
0035 308 ASSUME FAB$C_SEQ EQ 0
23 AA 95 0035 309 TSTB IFB$B_ORGCASE(R10) ; sequential file org?
03 13 0038 310 BEQL 15$
00C3 31 003A 311 BRW RLS_SHARED ; branch if not
003D 312
003D 313 : release for sequential file org (non-shared)
003D 314
003D 315 : write the buffer if dirty
003D 316
003D 317
003D 318
06 0A A4 01 E1 003D 319 15$: BBC #BDB$V_DRT,BDB$B_FLGS(R4),20$; branch if not dirty
FFBB' 30 0042 320 BSBW RMS$SEQWTD ; write the buffer
0045 321
0045 322 :
0045 323 : note: this code not executed if unit record
0045 324
0045 325
0D 50 E9 0045 326 BLBC R0,ERRXFR ; branch if errors
```



```
0048 327  
0048 328 ;  
0048 329 ; return bdb and buffer if requested  
0048 330 ;  
0048 331 ;  
03 6E E9 0048 332 20$: ASSUME RLS$V RETURN EQ 0  
0048 333 EXIT: BLBC (SP),SUCXIT ; branch if return not requested  
004B 334 ;  
004B 335 ;  
004B 336 ;!!!! if write behind, must await i/o done !!!!!  
004B 337 ;  
004B 338 ;  
FFB2' 30 004B 339 BSBW RM$RETBDB ; return bdb & buffer  
004E 340 SUCXIT: RMSSUC  
0051 341 ;  
0051 342 ;  
0051 343 ; clean up stack and return  
0051 344 ;  
0051 345 ;  
53 8E D0 0051 346 RLSXIT: MOVL (SP)+,R3 ; Restore release flags.  
05 05 0054 347 RSB  
0055 348 ;  
0055 349 ;  
0055 350 ; error occurred - do rlnerr call  
0055 351 ;  
0055 352 ;  
50 DD 0055 353 ERRXFR: PUSHL R0 ; save status  
B6 10 0057 354 BSBB RM$RLNERR ; release access  
01 BA 0059 355 POPR #^M<R0> ; restore status code  
F4 11 005B 356 BRB RLSXIT ; and return
```

```
005D 358 .SBTTL SUPPORT ROUTINES AND INFREQUENT PATHS
005D 359
52 54 D0 005D 360 ISBLB: MOVL R4,R2 ; Get BLB into right register.
54 D4 0060 361 CLRL R4 ; Note no BDB/GBPB.
01FA 31 0062 362 BRW DEQ ; Dequeue the lock.
0065 363
0065 364 NOLOCK:
08 6E 03 E0 0065 365 BBS #RLSSV_DEQ, (SP), DQBDB ; Make BDB free if DEQ spec'd.
DB 6E 02 E1 0069 366 BBC #RLSSV_KEEF_LOCK, (SP), EXIT ; If not keep lock, exit.
54 D4 006D 367 CLRL R4 ; So we know there is no BLB when this
006F 368 ; non-existent lock is released.
DD 11 006F 369 BRB SUCXIT ; And return with success.
0071 370 DQBDB: ASSUME RLSSV_RETURN EQ 0
0071 371 BLBS (SP),EXIT ; Exit if 'return' spec'd.
0074 372 RMSSUC ; Load R0 with success.
0228 31 0077 373 BRW DEQBDB ; Branch to free up BDB.
007A 374
007A 375 ;
007A 376 ; This is a lock on a global buffer.
007A 377 ; Deq the GBPB entirely. If the buffer was marked invalid,
007A 378 ; mark the GBD invalid also. Also, if first accessor, then convert lock
007A 379 ; to system owned null lock in order to keep buffer cached. If not first
007A 380 ; accessor, then just $DEQ our lock on the buffer.
007A 381 ;
007A 382 ; R1 = GBD Address
007A 383 ; R2 = BLB Address
007A 384 ; R4 = GBPB Address
007A 385
6E 08 C8 007A 386 GBPB: BISL2 #RLSSM_DEQ, (SP) ; Force deq of gbpb.
51 24 A4 D0 007D 387 MOVL GBPB$L_GBD_PTR(R4),R1 ; Get address of GBD.
20 A4 D0 0081 388 MOVL GBPB$L_VBNSEQNO(R4),- ; Move sequence number from gbpb to gbd.
10 A1 0084 389 GBD$L_VBNSEQNUM(R1)
0086 390 ASSUME BDBSV_VAL EQ 0
06 0A A4 E8 0086 391 BLBS GBPB$B_FLGS(R4), VALID ; Br if valid.
0275 30 008A 392 BADGBL: BSBW INVGBL ; Invalidate GBD.
01CF 31 008D 393 DQBR: BRW DEQ ; And go release lock.
0090 394
0090 395 ;
0090 396 ; At this time we must look at the global buffer quota and the global buffer
0090 397 ; usage to determine if we will be able to convert the enq to a system lock.
0090 398 ; If we are out of quota, we must dump the buffer.
0090 399 ;
0090 400 ; Note: This piece of code has non-closeable timing hole in that the process
0090 401 ; can be deleted after we have taken the quota for the lock, but before we
0090 402 ; have actually converted the lock. This can result in the permanent loss of
0090 403 ; a global buffer from the quota. Also the reverse can occur in RMSOCLOSE.
0090 404 ;
0090 405 ;
14 A1 D5 0090 406 VALID: TSTL GBD$L_LOCK_ID(R1) ; Has buffer been cached?
F8 12 0093 407 BNEQ DQBR ; Yes, go release lock.
50 00000000'9F B0 0095 408 MOVW @#SYSS$GW_GBLBUFQUO,R0 ; Get the global buffer quota
00000000'9F FFFF 8F 58 009C 409 ADAWI #-1,@#RMS$GW_GBLBUFQUO ; Now try to get the quota for this lock
50 00000000'9F A0 00A5 410 ADDW2 @#RMS$GW_GBLBUFQUO,R0 ; Can we really take the quota?
37 19 00AC 411 BLSS NOCSH ; Branch if out of quota
24 A2 D0 00AE 412 MOVL BLB$L_LOCK_ID(R2),- ; Save lock id of cached buffer
14 A1 00B1 413 GBD$L_LOCK_ID(R1) ; in GBD.
51 DD 00B3 414 PUSHL R1 ; Save GBD address around $ENQ.
```



```
FF48' 30 00B5 415 BSBW RMS$SETEFN ; Get efn
50 8ED0 00B8 416 POPL R0 ; Put efn in R0.
00BB 417 $ENQ_S ; Convert lock to system owned NULL lock.
00BB 418 LKMODE = R0,- ;
00BB 419 LKSB = BLB$W [KSTS(R2),- ;
00BB 420 FLAGS = #LCK$M_VALBLK!LCK$M_CONVERT!LCK$M_CVTSYS!LCK$M_SYSTEM!LCK$ ;
06 51 8ED0 00D9 421 POPL R1 ; Restore GBD address.
24 50 E9 00DC 422 BLBC R0,NOC SH ; If conversion failed trash buffer.
24 A2 D4 00DF 423 CLRL BLB$L_LOCK_ID(R2) ; Don't $DEQ lock
017A 31 00E2 424 BRW DEQ ; Go release structures.
00000000'9F 01 58 00E5 425
00E5 426 NOC SH: ADAWI #1,a#RMS$GW_GBLBUFQUO ; Unable to convert the lock, so give
00EC 427 ; back the quota
00EC 428 CLRL GBD$L_LOCK_ID(R1) ; Conversion failed, clear lock_id in GBD.
5C 14 A1 D4 00EF 429 MOVL IFB$L-GBH PTR(R10), AP ; Get global buffer header.
0088 CA D0 00F4 430 INCL GBH$L-OUTBUFQUO(AP) ; Count this one as missed,
50 AC D6 00F7 431 BRB BADGBL ; and invalidate the global buffer.
91 11 00F9 432
00F9 433 NOTLOCKED:
00F9 434 RMSPBUG FTL$_NOTLOCKED ; Not locked correctly.
```

```
.SBTTL  SHARED FILE SUPPORT (RELATIVE AND INDEXED)

0100 436
0100 437
0100 438 RLS_SHARED:
OC A4 B5 0100 439 TSTW BDB$W_USERS(R4) ; Check use count.
F4 13 0103 440 BEQL NOTLOCKED ; Bug if zero.
OA A4 01 E1 0105 441 BBC #BDB$V_DRT, BDB$B_FLGS(R4),- ; Branch if not dirty.
64 0109 442 CHECK_LOCK
010A 443
00A0 OC 93 010A 444 BITB #IFB$M_AI!IFB$M_BI,- ; If neither AI nor BI Journaling is
CA 010C 445 IFB$B_JNLFLG(R10) ; enabled, then skip the journaling
31 13 010F 446 BEQL 5$ ; of the bucket about to be released
0111 447
02 91 0111 448 CMPB #IFB$C_IDX,- ; AI and BI Journaling on a bucket
23 AA 0113 449 IFB$B_ORGCASE(R10) ; basis is only performed if the
2B 12 0115 450 BNEQ 5$ ; file's organization is index
0117 451
03 E1 0117 452 BBC #IFB$V_AI,- ; if this file is to be AI Journalled
11 00A0 CA 0119 453 IFB$B_JNLFLG(R10),2$ ; then write out a journaling entry
03 DD 011D 454 PUSHL #CJFS_AI ; to the AI Journal containing the
00000000'EF 16 011F 455 JSB RMSAI_AND_BI_3 ; modified bucket before releasing
5E 04 C0 0125 456 ADDL2 #4,SP ; the dirty bucket
03 50 E8 0128 457 BLBS R0,2$
FF27 31 012B 458 1$: BRW ERRXFR
012E 459
02 E1 012E 460 2$: BBC #IFB$V_BI,- ; if this file is to be BI Journalled
OE 00A0 CA 0130 461 IFB$B_JNLFLG(R10),5$ ; then write out a journaling entry
02 DD 0134 462 PUSHL #CJFS_BI ; to the BI Journal containing the
00000000'EF 16 0136 463 JSB RMSAI_AND_BI_3 ; un-modified bucket before releasing
5E 04 C0 013C 464 ADDL2 #4,SP ; the dirty bucket
E9 50 E9 013F 465 BLBC R0,1$
0142 466
OC 53 03 E0 0142 467 5$: BBS #RLS$V_DEQ, R3, WRITE ; If DEQ just write (seqnum correct).
D3 0146 468 BITL #RLS$M_RETURN - ; Check for options that
0147 469 ! RL$M_WRT_THRU ! RL$M_KEEP_LOCK,- ; disable deferred write.
53 07 0147 470 R3
04 12 0149 471 BNEQ 10$ ; If any set, then write it.
1F 6A 2C E0 014B 472 BBS #IFB$V_DFW, (R10), CHECK_LOCK ; Don't write if deferred write.
014F 473
014F 474 ;
014F 475 ; The sequence number is only bumped here if the buffer is actually written.
014F 476 ; This prevents invalidating cached buffers when readers have accessed
014F 477 ; a dirty buffer. The sequence number is also bumped later when a locker
014F 478 ; converts EX to PW.
014F 479 ;
014F 480
014F 481
20 A4 D6 014F 482 10$: ASSUME BDB$L_VBNSEQNO EQ GBPBS$L_VBNSEQNO
0152 483 INCL BDB$L_VBNSEQNO(R4) ; Bump count, inv gbl if necessary.
0D 00A2 CA 01 E1 0152 484 WRITE: BBC #IFB$V_JNL,IFB$B_JNLFLG2(R10),10$ ; Branch if not journaling.
2C BB 0158 485 PUSHF #^M<R2,R3,R5> ; Save registers.
00000000'EF 16 015A 486 JSB RMSFRCJNL ; Perform journal force(s).
2C BA 0160 487 POPR #^M<R2,R3,R5> ; Restore registers.
06 50 E9 0162 488 BLBC R0,20$ ; Branch on error.
FE98' 30 0165 489 10$: BSBW RM$SEQWTBD ; Write the buffer.
03 50 E8 0168 490 BLBS R0,CHECK_LOCK ; If success, continue, else ...
FEE7 31 016B 491 20$: BRW ERRXFR ; stop if error detected
016E 492
```



```
016E 493 CHECK_LOCK:
OC A4 B7 016E 494 DECW BDB$W_USERS(R4) ; Decrement use count.
1C A4 D5 0171 495 TSTL BDB$L_VBN(R4) ; Was this buffer only (vbn 0)?
03 12 0174 496 BNEQ 5$ ; If so, then make buffer free.
FEF8 31 0176 497 BRW DQBDB
52 10 A4 D0 0179 498 5$: MOVL BDB$L_BLB_PTR(R4), R2 ; Get pointer to BLB.
03 12 017D 499 BNEQ 10$ ; continue unless there is no
FEE3 31 017F 500 BRW NOLOCK ; BLB structure
0182 501
5C 10 A2 D0 0182 502 10$: MOVL BLB$L_OWNER(R2), AP ; Get owner of this BLB.
16 13 0186 503 BEQL HAVE_BLB ; EQL is NL lock.
5C 59 D1 0188 504 CMPL R9, AP ; Check owner of this BLB.
11 13 018B 505 BEQL HAVE_BLB ; EQL then it's ours.
018D 506 ASSUME BLB$L_FLNK EQ 0
52 52 62 D0 018D 507 MOVL (R2), R2 ; Get next BLB.
54 0C A2 D1 0190 508 CMPL BLB$L_BDB_ADDR(R2), R4 ; Still in the chain?
EC 13 0194 509 BEQL 10$ ; Check the owner.
FF60 30 0196 510 BSBW NOTLOCKED ; Something is wrong.
0199 511
0199 512
0199 513 ; This is a locker who wanted to defer write back of the buffer, but
0199 514 ; a blocking AST has already arrived so we must write it now.
0199 515
0199 516
0199 517 WB:
OC A4 B6 0199 518 INCW BDB$W_USERS(R4) ; Bump user count back up
B4 11 019C 519 BRB WRITE ; and write the buffer back.
019E 520 HAVE_BLB:
019E 521
019E 522
019E 523 ; Assumption is being made that DFW can only be set at this time
019E 524 ; if this lock is being held by a reader. If this is a write lock (LOCK)
019E 525 ; then DFW was cleared when the lock was granted, even if it was DFW
019E 526 ; prior to that time. The exception is when rm$release has been called
019E 527 ; from rm$cache to free a DFW buffer - which was just written above -
019E 528 ; with the rls$v_deq flag set. In this case, the blb$v lock flag is
019E 529 ; still set from the original cache call that made the buffer dirty.
019E 530 ; At any rate, if this is really a reader, and this is a DFW buffer,
019E 531 ; then the lock is simply retained and no further action taken.
019E 532
019E 533
019E 534 ASSUME BDB$L_VBNSEQNO EQ GBPBL_VBNSEQNO
28 A2 20 A4 D0 019E 535 MOVL BDB$L_VBNSEQNO(R4), BLB$L_VALSEQNO(R2) ; Update value block.
03 6E 02 E1 01A3 536 BBC #RLS$v_KEEP_LOCK, (SP), 25$ ; Continue if not keeplock.
0122 31 01A7 537 BRW KEELOCK ; Branch if keeplock.
01AA 538 25$:
25 0A A4 01 E0 01AA 539 BBS #BDB$v_DRT, BDB$b_FLGS(R4), 35$ ; Branch if dirty.
0A A2 20 8A 01AF 540 BICB2 #BLB$m_DFW, BLB$b_BLBFLGS(R2) ; Can't be DFW if not dirty.
01B3 541 ASSUME <BDB$c_BID & 1> EQ 0
01B3 542 ASSUME <GBPBL_BID & 1> EQ 1
01B3 543 ASSUME BDB$b_BID EQ GBPBL_BID
03 08 A4 E9 01B3 544 BLBC BDB$b_BID(R4), 28$
FEC0 31 01B7 545 BRW GBPBL ; Br to DEQ GBPBL.
01BA 546
OC A4 B5 01BA 547 28$: TSTW BDB$W_USERS(R4) ; Any other accessors?
03 13 01BD 548 BEQL 30$ ; EQL then branch to convert the lock.
0093 31 01BF 549 BRW DEQ1 ; If other accessors, dq blb entirely.
```

```
03 6E 03 E1 01C2 550 30$: BBC #RLSV_DEQ, (SP), 33$ ; Continue if no dequeue requested.
      0096 31 01C6 551 BRW DEQ ; Otherwise Branch
      10 A2 D4 01C9 552 33$: CLRL BLBSL_OWNER(R2) ; No owner for a NL lock.
      7E D4 01CC 553 ASSUME LCK$K_NLMODE EQ 0
      7E 7C 01CC 554 CLRL -(SP) ; Want to convert to NL.
      7E 7C 01CE 555 CLRQ -(SP) ; Acmode and protection.
      28 11 01D0 556 CLRQ -(SP) ; No block AST or parameter.
      01D2 557 BRB FINISH_ARGS ; Rejoin main line.
      01D4 558
      01D4 559 35$:
      01D4 560
      01D4 561
      01D4 562 : At this point we have a dirty buffer. This can only happen if
      01D4 563 : deferred write was enabled, as otherwise the dirty flag got
      01D4 564 : cleared when the buffer was written above.
      01D4 565
      01D4 566
      01D4 567 ASSUME BLBSV_LOCK EQ 0
      6C 0A A2 E9 01D4 568 BLBC BLBSB_BLBFLGS(R2), DFW_RD ; Branch if a reader.
      01D8 569
      01D8 570
      01D8 571 : This is a locker and deferred write is desired.
      01D8 572 : If this is a global buffer, branch off to copy it to a local buffer
      01D8 573 : so that a blocking ast will write out the local copy. This simplifies
      01D8 574 : the global cache replacement algorithm so that it doesn't have to
      01D8 575 : deal with the problem of forcing another process to write a modified
      01D8 576 : buffer from the global cache.
      01D8 577 : The COPY_GBL code will return with R4 pointing to the BDB clone if
      01D8 578 : it succeeds at the DFW_FIN label. If it fails, it branches back to
      01D8 579 : the WRITE label above to flush the buffer to disk, and it won't come
      01D8 580 : here again.
      01D8 581
      01D8 582
      01D8 583 ASSUME BDB$S_VBNSEQNO EQ GBPBSL_VBNSEQNO
      20 A4 D6 01D8 584 INCL BDB$S_VBNSEQNO(R4) ; Bump seq num, inv gbl if local.
      28 A2 D6 01DB 585 INCL BLBSL_VALUEQNO(R2) ; Do value block also.
      01DE 586 ASSUME <BDB$C_BID & 1> EQ 0
      01DE 587 ASSUME <GBPBSL_BID & 1> EQ 1
      01DE 588 ASSUME BDB$B_BID EQ GBPBSB_BID
      03 08 A4 E9 01DE 589 BLBC BDB$B_BID(R4), CHKWB ; Branch if this is a BDB.
      016E 31 01E2 590 BRW COPY_GBL ; Else branch to copy buffer.
      01E5 591
      01E5 592
      01E5 593 : If the WRITEBACK flag is set in the BLB we must go back and write through
      01E5 594 : the buffer because a blocking AST arrived after we selected the BLB.
      01E5 595
      01E5 596
      01E5 597 CHKWB: BBSC #BLBSV_WRITEBACK, - ; Need to write back?
      AF 0A A2 06 E4 01E5 598 BLBSB_BLBFLGS(R2), WB ; Branch if yes.
      01E7 599 DFW_FIN:
      10 A2 5A D0 01EA 600 MOVL R10, BLBSL_OWNER(R2) ; Ifab owns a dfw lock.
      04 DD 01EE 601 PUSHL #LCK$K_PWMODE ; Convert to PW mode.
      7E 7C 01F0 602 CLRQ -(SP) ; Acmode and protection.
      03A8'CF DF 01F2 603 PUSHAL W^BLOCK_AST ; Want block ast for DFW.
      52 DD 01F6 604 PUSHL R2 ; BLB is AST parameter.
      0A A2 20 88 01F8 605 BISB2 #BLBSM_DFW, BLBSB_BLBFLGS(R2) ; Note this is DFW lock.
      01FC 606 FINISH_ARGS:
```



```

      7E 7C 01FC 607      CLRQ      -(SP)      ; No comp AST or parid.
      7E D4 01FE 608      CLRL      -(SP)      ; No resource name.
      1A DD 0200 609      PUSHL     #LCK$M SYSTEM ! LCK$M_SYNCSTS - ; lock options.
                                ! LCK$M CONVERT
      03 E0 0202 611      BBS       #BLB$V NOBUFFER,-      ; Check if originally nobuffer req.
09 0A A2 0204 612      BITB       BLB$B_BLBFLGS(R2), 50$ ; and don't store value if so.
      21 93 0207 613      BITB       #BLB$M LOCK ! BLB$M_DFW,- ; Check if value block needs
0A A2 0209 614      BEQL         BLB$B_BLBFLGS(R2) ; to be stored.
      03 13 020B 615      50$      ; Readers don't update VALBLK.
      6E 01 C8 020D 616      BLSL2   #LCK$M VALBLK, (SP) ; Store value block.
      20 A2 DF 0210 617 50$:      PUSHAL BLB$W_CKSTS(R2) ; Lock status block.
      20 A2 D4 0213 618      CLRL     BLB$W_LKSTS(R2) ; Init to zero.
      24 AE DD 0216 619      PUSHL     9*4(SP) ; Lock mode requested.
0A A2 DF 8F 8A 0219 620      BICB2    #^CBLB$M DFW, BLB$B_BLBFLGS(R2) ; Clear all blbflgs except dfw
      FDDF' 30 021E 621      BSBW     RM$SETEFN ; Get EFN to use.
00000000'9F 0B FB 0221 622      CALLS   #11, @#SY$SENG ; Do the conversion.
      53 8ED0 0228 623      POPL      R3 ; Pop requested mode off stack.
      0B 50 E9 022B 624      BLBC     R0, 70$ ; Exit on error.
      50 20 A2 3C 022E 625      MOVZWL BLB$W_LKSTS(R2), R0 ; Get status.
      04 50 E9 0232 626      BLBC     R0, 70$ ; Branch on error.
      0B A2 53 90 0235 627      MOVB   R3, BLB$B_MODEHELD(R2) ; Store mode granted in blb.
      FE15 31 0239 628 70$:      BRW    RLSXIT ; And exit.
      023C 629
      023C 630
      023C 631 ; Out of line check to make sure that WRITEBACK is in progress if DFW
      023C 632 ; was clear. Normally expect either DFW is set, or buffer wasn't dirty
      023C 633 ; to begin with and we wouldn't have gotten here.
      023C 634
      023C 635
      06 E0 023C 636 L1:      BBS      #BLB$V WRITEBACK,- ; Writeback is the only reason DFW
0D 0A A2 023E 637      BSBW     BLB$B_BLBFLGS(R2), L2 ; should be clear.
      FEB5 30 0241 638      BSBW     NOTLOCKED ; BSBW because we won't come back
      0244 639 ; from the bugcheck and we can tell
      0244 640 ; where it came from.
      0244 641 DFW_RD:
      0244 642
      0244 643 ; This is a reader, yet the buffer is dirty.
      0244 644 ; Make sure that this is a deferred write lock as otherwise it means
      0244 645 ; the caller did not lock the bucket when RM$SCACHE was called.
      0244 646 ; If WRITEBACK is also set, there was a problem on the blocking AST qio
      0244 647 ; and the buffer should be written back now and released.
      0244 648 ; The BLB we are using is the PW lock.
      0244 649
      0244 650
      F3 0A A2 05 E1 0244 651      BBC      #BLB$V DFW, BLB$B_BLBFLGS(R2), L1 ; If not dfw, check further.
      06 E0 0249 652      BBS       #BLB$V WRITEBACK,- ; Br if buffer must be written back.
      97 0A A2 024B 653      BLB$B_BLBFLGS(R2), CHKWB ; Only occurs if block ast qio failed.
      10 A2 5A D0 024E 654 L2:      MOVL   R10, BLB$W_OWNER(R2) ; Ifab is owner of DFW locks.
      FDF9 31 0252 655      BRW      SUCXIT ; and exit.
      0255 656 DEQ1:
      0255 657
      0255 658
      0255 659 ; There are other streams que'd for the same lock, so DEQ this
      0255 660 ; lock and make the BLB available. Also must get BDB$W_BLB_PTR
      0255 661 ; pointing to the next BLB if it was pointing to this one.
      0255 662 ; A separate GBP is associated with every access to a GBD so
      0255 663 ; therefore the structure through this path will always be a BDB.
```



```
0255 664 :
0255 665 :
6E 08 CA 0255 666 BICL2 #RLSSM_DEQ, (SP) ; Clear DEQ so BDB isn't released.
0091 30 0258 667 BSBW FIX BLBPTR ; Fixup BLB pointer.
0084 CA B6 025B 668 INCW IFBSW_AVLCL(R10) ; Note BLB is free.
025F 669 DEQ:
51 24 A2 D0 025F 670 MOVL BLB$L_LOCK_ID(R2), R1 ; Is lock id 0?
1E 13 0263 671 BEQL 15$ ; Yes it is, don't do DEQ.
50 D4 0265 672 CLRL R0 ; Assume no value block.
03 E0 0267 673 BBS #BLB$V_NOBUFFER, - ; Check if originally nobuffer req.
0A 0A A2 0269 674 BLB$B_BLBFLGS(R2), 10$ ; and don't update value if so.
21 93 026C 675 BITB #BLB$M_LOCK ! BLB$M_DFW, - ; Check if value block needs
0A A2 026E 676 BLB$B_BLBFLGS(R2) ; to be stored.
04 13 0270 677 BEQL 10$ ; EQL then don't store.
50 28 A2 DE 0272 678 MOVAL BLB$L_VALBLK(R2), R0 ; Note value block.
0276 679 10$: $DEQ_S LKID=R1 VALBLK=(R0) ; Dequeue the lock.
0283 680 15$:
0A A2 B4 0283 681 ASSUME BLB$B_MODEHELD EQ <BLB$B_BLBFLGS + 1>
OC A2 D4 0286 682 CLRW BLB$B_BLBFLGS(R2) ; Clear out all flags.
0289 683 CLRL BLB$L_BDB_ADDR(R2) ; Disassociate from BDB.
10 A2 7C 0289 684 ASSUME BLB$L_VBN_EQ <BLB$L_OWNER + 4>
20 A2 7C 028C 685 CLRQ BLB$L_OWNER(R2) ; Clear owner and vbn.
52 62 0F 028C 686 CLRQ BLB$W_LKSTS(R2) ; Clear lock status block.
009C DA 62 0E 028F 687 REMQUE (R2), R2 ; Remove from chain.
54 D5 0292 688 INSQUE (R2), @IFB$L_BLBBLNK(R10) ; Insert at end.
03 12 0297 689 TSTL R4 ; Is there a BDB?
FDB3 31 0299 690 BNEQ 30$ ; Continue if BDB present.
F9 6E 03 E1 029B 691 BRW RLSXIT ; Otherwise done now.
029E 692 30$: BBC #RLS$V_DEQ, (SP), 20$ ; Return if deq flag not set.
02A2 693 ; (came from deq1 label above).
53 8ED0 02A2 694 DEQBDB: POPL R3 ; Restore release flags.
02A5 695 DQBDB1:
02A5 696 ASSUME <BDB$C_BID & 1> EQ 0
02A5 697 ASSUME <GBP$C_BID & 1> EQ 1
02A5 698 ASSUME BDB$B_BID EQ GBP$B_BID
OE 08 A4 E9 02A5 700 BLBC BDB$B_BID(R4), 10$ ; Br if this is a BDB.
5C 24 A4 D0 02A9 701 MOVL GBP$C_GBD_PTR(R4), AP ; Get pointer to GBD.
20 AC B7 02AD 702 DECW GBD$W_OSECNT(AP) ; Reduce use count on GBD.
0B A4 90 02B0 703 MOVW GBP$B_CACHE_VAL(R4), - ; Store cache value in GBD.
0B AC 02B3 704 GBD$B_CACHE_VAL(AP)
04 11 02B5 705 BRB 20$ ; Br to finish up.
0084 CA B6 02B7 706 10$: INCW IFBSW_AVLCL(R10) ; Note another buffer available.
02BB 707 ASSUME BDB$B_CACHE_VAL EQ <BDB$B_FLGS + 1>
0A A4 B4 02BB 708 20$: CLRW BDB$B_FLGS(R4) ; Clear cache_val and flgs.
10 A4 D4 02BE 709 CLRL BDB$L_BLB_PTR(R4) ; Remove pointer to BLB.
02C1 710 ASSUME BDB$L_VBNSEQNO EQ <BDB$L_VBN + 4>
1C A4 7C 02C1 711 CLRQ BDB$L_VBN(R4) ; Clear vbn and seq num.
54 64 0F 02C4 712 REMQUE (R4), R4 ; Remove from chain.
44 BA 64 0E 02C7 713 INSQUE (R4), @IFB$L_BDB_BLNK(R10) ; Insert at end.
05 02CB 714 RSB ; And return.
02CC 715
02CC 716 KEELOCK:
OC A4 B5 02CC 717 TSTW BDB$W_USERS(R4) ; Is use count zero?
0B 12 02CF 718 BNEQ 20$ ; NEQ others are queued for BDB.
D2 10 02D1 719 BSBB DQBDB1 ; Free up buffer.
OC A2 D4 02D3 720 10$: CLRL BLB$L_BDB_ADDR(R2) ; No BDB assoc. with BDB now.
```


54	52	D0	02D6	721	MOVL	R2, R4	; Return BLB address in R4.
	FD72	31	02D9	722	BRW	SUCXIT	; And return.
			02DC	723			
	0E	10	02DC	724	20\$: BSBB	FIX_BLBPTR	; Fixup BLB pointer.
52	62	0F	02DE	725	REMQUE	(R2), R2	; Remove from chain.
009C DA	62	0E	02E1	726	INSQUE	(R2), @IFB\$L_BLBBLNK(R10)	; Insert at end.
0084 CA	B6	02E6	727	INCW	IFB\$W_AVLCL(R10)		; Note one less accessor on buffer.
	E7	11	02EA	728	BRB	10\$	; Clear pointer and exit.
			02EC	729			

```
02EC 731 :++
02EC 732 : FIX_BLBPTR
02EC 733 :
02EC 734 : Routine to point BLB_PTR from BDB to next BLB when a BLB is being removed.
02EC 735 :
02EC 736 : Input:
02EC 737 :
02EC 738 : R4 = BDB
02EC 739 : R2 = BLB
02EC 740 :
02EC 741 : --
02EC 742 : FIX_BLBPTR:
52 10 A4 D1 02EC 743 : CMPL BDB$L_BLB_PTR(R4), R2 ; Is BDB pointing to this BLB?
      OF 12 02F0 744 : BNEQ 20$ ; NEQ it's not, so continue.
      50 62 D0 02F2 745 : ASSUME BLB$L_FLNK EQ 0
54 0C A0 D1 02F2 746 : MOVL (R2), R0 ; Get next BLB.
      02 13 02F5 747 : CMPL BLB$L_BDB_ADDR(R0), R4 ; Does it point to this BDB?
      50 D4 02F9 748 : ; This assumes this offset in the
      50 D4 02F9 749 : ; ifab from the listhead fails always.
10 A4 50 D0 02F9 750 : BEQL 10$ ; EQL points to same BDB.
      50 D4 02FB 751 : CLRL R0 ; No other BLB's point to this BDB.
      05 D0 02FD 752 10$: MOVL R0, BDB$L_BLB_PTR(R4) ; Set or clear as appropriate.
      05 0301 753 20$: RSB ; Return.
      0302 754
```



```
0302 756 :++
0302 757 : INVGBL
0302 758 :
0302 759 : This routine invalidates a buffer in the global buffer cache. It is done
0302 760 : as follows. If any other accessors are queued waiting for their lock
0302 761 : on this bucket to be granted, simply force a read by stuffing the sequence
0302 762 : number. If no other accessors are present, move the GBD to the end of the
0302 763 : list to identify it as a free buffer.
0302 764 :
0302 765 : Inputs:
0302 766 :
0302 767 : R1 - Address of GBD
0302 768 : R4 - Address of associated with BLB, if any.
0302 769 : R10 - Address of IFAB.
0302 770 :
0302 771 : Destroys R0, R1, R3, AP.
0302 772 :--
0302 773 : INVGBL:
0302 774 : ASSUME GBD$$_FLINK EQ 0
0302 775 : ASSUME GBD$$_BLINK EQ 4
0302 776 : BSBW RM$RAISE_GBS_LOCK ; Get EX lock on GBS.
0302 777 : MOVL IFB$$_GBR_PTR(R10), AP ; Get global buffer header.
0302 778 : CMPW #1,GBD$$_USECNT(R1) ; Anyone que'd for this buffer?
0302 779 : BEQL 10$ ; EQL no, so put at end of queue.
0302 780 : MNEGL #1,GBD$$_VBNSEQNUM(R1) ; Invalidate seq number to force read.
0302 781 : BRB 20$ ; Return.
0302 782 :
0302 783 :
0302 784 : Remove from current position in list.
0302 785 :
0302 786 :
0302 787 10$: ADDL3 (R1), R1, R3 ; R3 = successor.
0302 788 : REMQTI (R3), R1 ; Remove from queue.
0302 789 :
0302 790 : Set VBN to -1. Zero out cache_val, number bytes in use, flags.
0302 791 :
0302 792 : ASSUME <GBD$$_FLAGS + 1> EQ GBD$$_CACHE_VAL
0302 793 : CLRW GBD$$_FLAGS(R1) ; Clear flags, cache_val.
0302 794 : CLRW GBD$$_NUMB(R1) ; Clear numb bytes used.
0302 795 : MNEGL #1, GBD$$_VBN(R1) ; Set invalid VBN.
0302 796 : CLRL GBD$$_VBNSEQNUM(R1) ; Clear sequence number.
0302 797 : TSTL GBD$$_LOCK_ID(R1) ; Is there a system lock to drop?
0302 798 : BEQL 15$ ; Branch if not.
0302 799 : PUSHL R1 ; Save address of GBD
0302 800 : ADAWI #1,@#RM$$_GW_GBLBUFQUO ; Count the quota back
0302 801 : SDEQ_S LKID=GBD$$_LOCK_ID(R1) ; Remove buffer from cache.
0302 802 : POPL R1 ; Restore GBD address
0302 803 : CLRL GBD$$_LOCK_ID(R1) ; Zero the old cached buffer lock id.
0302 804 :
0302 805 :
0302 806 : Now put this GBD at the end of the list.
0302 807 : AP = list head.
0302 808 :
0302 809 :
0302 810 15$: INSQTI (R1), (AP) ; Insert GBD at tail of queue.
0302 811 :
0302 812 20$: BSBW RM$$_LOWER_GBS_LOCK ; Release EX lock on GBS.
```

5C 0088 CA D0 30 0302 776
20 A1 01 B1 0302 777
06 13 0302 778
10 A1 01 CE 0302 779
39 11 0302 780
0302 781
0302 782
0302 783
0302 784
0302 785
0302 786
53 51 61 C1 0302 787
51 63 5F 0302 788
0302 789
0302 790
0302 791
0302 792
0A A1 B4 0302 793
18 A1 B4 0302 794
OC A1 01 CE 0302 795
10 A1 D4 0302 796
14 A1 D5 0302 797
1D 13 0302 798
51 DD 0302 799
00000000'9F 01 58 0302 800
0302 801
51 8ED0 0302 802
14 A1 D4 0302 803
0302 804
0302 805
0302 806
0302 807
0302 808
0302 809
6C 61 5D 0302 810
0302 811
FCAE' 30 0302 812

RMORELEAS
V04-000

RELEASE BUFFER ROUTINE
SHARED FILE SUPPORT (RELATIVE AND INDEXE

F 14

16-SEP-1984 00:33:33 VAX/VMS Macro V04-00
5-SEP-1984 16:22:20 [RMS.SRC]RMORELEAS.MAR;1

Page 19
(8)

05 0352 813
0353 814

RSB

; All done.


```
0353 816 :++
0353 817 : COPY_GBL
0353 818 :
0353 819 : Copy the global buffer to a local buffer if deferred write is
0353 820 : desired to avoid the problems of deferred write from the global
0353 821 : cache.
0353 822 :
0353 823 : Input:
0353 824 : R4 - GBPb address.
0353 825 :
0353 826 : Output:
0353 827 : R4 - BDB copy addr if branch to DFW_FIN
0353 828 : else same as input GBPb if branch to WRITE.
0353 829 :
0353 830 : Destroys R0, R1, R3, AP.
0353 831 :--
0353 832 : COPY_GBL:
0353 833 :
0353 834 :
0353 835 : First try to get a local buffer to copy to
0353 836 :
0353 837 :
0084 CA B7 0353 838 : DECW IFB$W_AVLCL(R10) ; Reduce local available count.
OC 18 0357 839 : BGEQ 10$ ; Branch if one is already free.
OC A4 B6 0359 840 : INCW GBPb$W_USERS(R4) ; Bump use count so other streams
035C 841 : ; won't take it if free lcl stalls.
FCA1' 30 035C 842 : BSBW RMS$FREE_LCL ; Else call routine to free one.
3F 50 E9 035F 843 : BLBC R0, NO_BUFF ; Branch if that fails.
OC A4 B7 0362 844 : DECW GBPb$W_USERS(R4) ; Put use count back.
24 BB 0365 845 10$: PUSHR #^M<R2,R5> ; Save registers.
51 1C A4 D0 0367 846 : MOVL GBPb$L_VBN(R4), R1 ; VBN of bucket to copy.
52 14 A4 3C 036B 847 : MOVZWL GBPb$W_NUMB(R4), R2 ; Size of bucket to copy.
FC8E' 30 036F 848 : BSBW RMS$GET_LCL_BUFF ; Get the local buffer.
0372 849 : ; Returns BDB addr in R5.
OB A4 90 0372 850 : MOVb GBPb$B_CACHE_VL(R4),- ; Copy cache value.
OB A5 0375 851 : BDB$B_CACHE_VAL(R5)
20 A4 D0 0377 852 : MOVL GBPb$C_VBNSEQNO(R4),- ; Copy sequence number from GBPb.
20 A5 037A 853 : BDB$L_VBNSEQNO(R5)
5C 6E D0 037C 854 : MOVL (SP), AP ; Pickup saved BLB address.
10 A5 5C D0 037F 855 : MOVL AP, BDB$L_BLB_PTR(R5) ; Point BDB to BLB.
OC AC 55 D0 0383 856 : MOVL R5, BLB$L_BDB_ADDR(AP) ; Point BLB to BDB.
51 18 A4 D0 0387 857 : MOVL GBPb$L_ADDR(R4), R1 ; Get source addr for copy.
FF17 30 038B 858 : BSBW DQBDB1 ; Reset the fields in the GBPb.
55 DD 038E 859 : PUSHL R5 ; Save the BDB address.
18 B5 61 52 28 0390 860 : MOVC3 R2, (R1), @BDB$L_ADDR(R5) ; Copy the buffer.
54 8ED0 0395 861 : POPL R4 ; Get BDB addr into R4.
03 88 0398 862 : BISB2 #BDB$M_VAL!BDB$M_DRT,- ; Note valid and dirty.
OA A4 039A 863 : BDB$B_FLGS(R4)
24 BA 039C 864 : POPR #^M<R2,R5> ; Restore original values.
FE49 31 039E 865 : BRW DFW_FIN ; Jump back into mainline.
03A1 866 : NO_BUFF:
03A1 867 :
03A1 868 :
03A1 869 : Couldn't get a local buffer to copy to.
03A1 870 : Fix local available count and branch back to force write of the
03A1 871 : global buffer.
03A1 872 :
```

RMORELEAS
V04-000

RELEASE BUFFER ROUTINE
SHARED FILE SUPPORT (RELATIVE AND INDEXE

H 14

16-SEP-1984 00:33:33 VAX/VMS Macro V04-00
5-SEP-1984 16:22:20 [RMS.SRC]RMORELEAS.MAR;1

Page 21
(9)

0084 CA 03A1 873
FDAA B6 03A1 874
31 03A5 875

INCW IFB\$W_AVLCL(R10)
BRW WRITE

; Put count back.
; Go write it through then.


```
03A8 877 :++
03A8 878 : BLOCK_AST
03A8 879 :
03A8 880 : This is the deferred buffer write back routine which is specified
03A8 881 : as the blocking AST when a dirty buffer is held in the cache with
03A8 882 : a PW lock.
03A8 883 :
03A8 884 :--
03A8 885 BLOCK_AST:
03A8 886 .WORD ^M<R4,R5,R10>
54 04 AC 0430 03AA 887 MOVL 4(AP), R4 ; AST parameter is BLB address.
52 10 03AE 888 BSBB SETUP ; Setup R4, R5, and R10.
03B0 889
03B0 890 :
03B0 891 : Note that the fact the BDB is checked out prior to checking the DFW
03B0 892 : flag is assuming that those pointers in the BLB are not being mucked
03B0 893 : with after CACHE clears the DFW flag and prior to the time the lock
03B0 894 : is converted or DEQ'd in RELEASE.
03B0 895 :
03B0 896
03B0 897 BBCC #BLB$V DFW,- ; Check if BLB is being accessed
0A 05 E5 03B2 898 BLB$B_BLBFLGS(R4),- ; already for lock access, and if
40 0A A4 03B4 899 SETWRTBCK ; so, simply set writeback flag.
03B5 900 ASSUME IFB$B_BID EQ IRB$B_BID
0B 08 AA 91 03B5 901 CMPB IFB$B_BID(R10), #IFB$C_BID ; Is this an ifab?
09 13 03B9 902 BEQL WRTBCK
0A 08 AA 91 03BB 903 CMPB IRB$B_BID(R10), #IRB$C_BID ; Then this better be an irab.
68 12 03BF 904 BNEQ BADOWN ; If not, then complain.
SA 6A D0 03C1 905 ASSUME IRB$B_IFAB_LNK EQ 0
50 20 AA 3C 03C1 906 MOVL (R10), R10 ; Get ifab address into R10 then.
51 14 A5 3C 03C4 907 WRTBCK: MOVZWL IFB$W_CHNL(R10), R0 ; Get channel into R0.
5A 18 A5 D0 03C8 909 MOVZWL BDB$W_NUMB(R5), R1 ; Get size of buffer into R1.
03CC 910 MOVL BDB$B_ADDR(R5), R10 ; Address of buffer.
03D0 911
03D0 912 $QIO_S EFN = #IMP$C_ASYQIOEFN,- ; Initiate write to disk.
03D0 913 CHAN = R0,-
03D0 914 FUNC = #IOS$ WRITEVBLK,-
03D0 915 IOSB = BDB$C_IOSB(R5),-
03D0 916 ASTADR = B^BCKASTFIN,-
03D0 917 ASTPRM = R4,-
03D0 918 P1 = (R10),-
03D0 919 P2 = R1,-
03D0 920 P3 = BDB$B_VBN(R5)
06 50 E9 03F2 921 BLBC R0, WBQIOERR ; Branch if qio failed.
03F5 922
03F5 923 SETWRTBCK:
40 8F 88 03F5 924 BISB2 #BLB$M_WRITEBACK,- ; Note buffer writeback necessary,
0A A4 03F8 925 BLB$B_BLBFLGS(R4) ; in progress, or attempted.
04 03FA 926 RET ; Exit from the blocking AST.
03FB 927
03FB 928 ; An error has occurred attempting to write back the dirty buffer.
03FB 929 ; Expected problem here is lack of AST quota.
03FB 930
03FB 931 WBQIOERR:
03FB 932 RMSPBUG FTL$_CANTDOAST ; bugcheck with the likely error.
```

```
0402 934 :++
0402 935 : SETUP
0402 936 :
0402 937 : Setup registers and verify structures.
0402 938 :
0402 939 : Input:
0402 940 :     R4 - BLB address
0402 941 :         BLB$L_BDB_ADDR
0402 942 :         BLB$L_OWNER
0402 943 :
0402 944 : Output:
0402 945 :     R4 - BLB
0402 946 :     R5 - BDB
0402 947 :     R10 - owner
0402 948 :
0402 949 : Bugcheck if BLB not a BLB or BDB not a BDB.  Owner not checked.
0402 950 :
0402 951 :--
0402 952 :
0402 953 : SETUP:
0402 954 :     ASSUME <BLB$B_BID + 1> EQ BLB$B_BLN
0402 955 :     CMPW  BLB$B_BID(R4),- ; Verify this is really a BLB
0402 956 :     #<BLB$C_BID +<BLB$C_BLN/4a8>> ; by checking BID and BLN fields.
0402 957 :     BNEQ  10$ ; Bugcheck if no good.
0402 958 :     MOVL  BLB$L_BDB_ADDR(R4), R5 ; Get BDB address.
0402 959 :     ASSUME <BDB$B_BID + 1> EQ BDB$B_BLN
0402 960 :     CMPW  BDB$B_BID(R5),- ; Verify this is a BDB
0402 961 :     #<BDB$C_BID +<BDB$C_BLN/4a8>> ; by checking BID and BLN fields.
0402 962 :     BNEQ  20$ ; NEQ then BDB not right.
0402 963 :     MOVL  BLB$L_OWNER(R4), R10 ; Assume owner is the ifab.
0402 964 :     RSB ; Return.
0402 965 : 10$:
0402 966 :     RMSPBUG FTL$_BADBLB ; BLB is bad.
0402 967 : 20$:
0402 968 :     RMSPBUG FTL$_BADBDB ; BDB is bad.
0402 969 : BADOWN:
0402 970 :     RMSPBUG FTL$_BADOWNER ; Owner field no good.
0402 971 :
```

08 A4 B1 0402 955
OE10 8F 12 0405 956
11 12 0408 957
55 0C A4 D0 040A 958
08 A5 B1 040E 959
140C 8F 12 0411 961
0C 12 0414 962
5A 10 A4 D0 0416 963
05 041A 964


```
0430 973 :++
0430 974 : BLKASTFIN
0430 975 :
0430 976 : This is the completion routine for the deferred write back qio.
0430 977 :
0430 978 :--
0430 979 :
0430 980 BLKASTFIN:
0430 981 .WORD ^M<R2,R3,R4,R5,R9,R10>
0432 982 BSBW RMSBLKFINCHK ; Check for AST's inhibited.
0435 983 MOVL R9, R4 ; Want BLB addr into R4.
0438 984 BSBB SETUP ; Setup R4, R5, and R10.
043A 985 BICB2 #BDB$M_DRT, BDB$B_FLGS(R5) ; Clear dirty.
043E 986 BICB2 #BLB$M_DFW, BLB$B_BLBFLGS(R4) ; Not dfw anymore.
0442 987 :
0442 988 :
0442 989 : The LOCK flag is being used here to indicate that a thread has
0442 990 : stalled after finding the DFW flag clear on a DFW BLB.
0442 991 :
0442 992 :
0442 993 BBSC #BLB$V_LOCK,- ; Br if necessary to start thread.
0444 994 BLB$B_BLBFLGS(R4), STARTTHREAD
0447 995 IFB$B_BID EQ IRB$B_BID
0447 996 IRB$B_BID(R10), #IRB$C_BID ; Is this an irab?
0448 997 BEQL CLRWRTBCK ; Don't convert lock if so.
044D 998 CMPB IFB$B_BID(R10), #IFB$C_BID ; It should be an ifab then.
0451 999 BNEQ BADOWN ; Bugcheck if not an ifab.
0453 1000 TSTW BDB$W_USERS(R5) ; Any other streams have this accessed?
0456 1001 BNEQ DQ ; Branch if so.
0458 1002 CLRL BLB$L_OWNER(R4) ; No owner anymore.
045B 1003 CONVN:
045B 1004 $ENQ_S EFN = #IMP$C_ASYQIOEFN,- ; Convert the lock to NL.
045B 1005 LKMODE = #LCK$K_NLMODE,-
045B 1006 LKSB = BLB$W_LKSTS(R4),-
045B 1007 FLAGS = #LCK$M_SYSTEM!LCK$M_CONVERT!LCK$M_SYNCSTS
0475 1008 BLBC RO,ENQBUG ; BUGCHECK if failure....
0478 1009 ASSUME LCK$K_NLMODE EQ 0
0478 1010 CLRB BLB$B_MODEHELD(R4) ; NL lock held now.
047B 1011 CLRWRTBCK:
047B 1012 BICB2 #BLB$M_WRITEBACK,- ; Clear writeback flag.
047E 1013 BLB$B_BLBFLGS(R4)
0480 1014 SETASTDCL:
0480 1015 BISB2 #BDB$M_AST_DCL,- ; Note writeback has occurred.
0483 1016 BDB$B_FLGS(R5)
0485 1017 RET ; And exit.
0486 1018 :
0486 1019 ENQBUG:
0486 1020 RMSPBUG FTL$_ENQDEQFAIL ; $ENQ failed.....
048D 1021 :
048D 1022 : Starting a stalled thread.
048D 1023 : Declare the AST, then convert the lock.
048D 1024 :
048D 1025 :
048D 1026 STARTTHREAD:
048D 1027 $DCLAST_S ASTADR = RMS$STALLAST,- ; Declare AST to start stalled thread.
048D 1028 ASTPRM = BLB$L_OWNER(R4)
049F 1029 BLBS RO,CONVN ; And exit on success.
```

```

04A2 1030      RMSPBUG FTL$_ASTDECERR      ; AST failed, BUGCHECK.....
04A9 1031
04A9 1032 :
04A9 1033 : Other streams have this buffer accessed. Setup registers correctly and
04A9 1034 : branch into release code that will dequeue the lock entirely.
04A9 1035 :
04A9 1036
52 54 D0 04A9 1037 DQ:  MOVL  R4, R2      ; BLB expected in R2.
54 55 D0 04AC 1038      MOVL  R5, R4      ; BDB expected in R4.
CE AF 9F 04AF 1039      PUSHAB SETASTDCL  ; Release will RSB back.
7E D4 04B2 1040      CLRL  -(SP)         ; No release flags.
FD9E 31 04B4 1041      BRW   DEQ1         ; Go do it.
04B7 1042      .END

```


RMORELEAS
Symbol table

RELEASE BUFFER ROUTINE

M 14

16-SEP-1984 00:33:33 VAX/VMS Macro V04-00
5-SEP-1984 16:22:20 [RMS.SRC]RMORELEAS.MAR;1

Page 26
(12)

\$\$PSECT_EP	= 00000000		
\$\$ARGS	= 0000000B		
\$\$RMSTEST	= 0000001A		
\$\$RMS_PBUGCHK	= 00000010		
\$\$RMS_TBUGCHK	= 00000008		
\$\$RMS_UMODE	= 00000004		
\$\$T1	= 00000000		
BADGBL	0000008A	R	01
BADOWN	00000429	R	01
BDB\$B_BID	= 00000008		
BDB\$B_BLN	= 00000009		
BDB\$B_CACHE_VAL	= 0000000B		
BDB\$B_FLGS	= 0000000A		
BDB\$C_BID	= 0000000C		
BDB\$C_BLN	= 00000050		
BDB\$S_ADDR	= 00000018		
BDB\$S_BLB_PTR	= 00000010		
BDB\$S_IOSB	= 00000048		
BDB\$S_VBN	= 0000001C		
BDB\$S_VBNSEQNO	= 00000020		
BDB\$M_AST_DCL	= 00000040		
BDB\$M_DRT	= 00000002		
BDB\$M_PRM	= 00000008		
BDB\$M_VAL	= 00000001		
BDB\$V_DRT	= 00000001		
BDB\$V_VAL	= 00000000		
BDB\$W_NUMB	= 00000014		
BDB\$W_USERS	= 0000000C		
BLB\$B_BID	= 00000008		
BLB\$B_BLBFLGS	= 0000000A		
BLB\$B_BLN	= 00000009		
BLB\$B_MODEHELD	= 0000000B		
BLB\$C_BID	= 00000010		
BLB\$C_BLN	= 00000038		
BLB\$S_BDB_ADDR	= 0000000C		
BLB\$S_FLNR	= 00000000		
BLB\$S_LOCK_ID	= 00000024		
BLB\$S_OWNER	= 00000010		
BLB\$S_VALBLK	= 00000028		
BLB\$S_VALSEQNO	= 00000028		
BLB\$S_VBN	= 00000014		
BLB\$M_DFW	= 00000020		
BLB\$M_LOCK	= 00000001		
BLB\$M_WRITEBACK	= 00000040		
BLB\$V_DFW	= 00000005		
BLB\$V_LOCK	= 00000000		
BLB\$V_NOBUFFER	= 00000003		
BLB\$V_WRITEBACK	= 00000006		
BLB\$W_LKSTS	= 00000020		
BLKASTFIN	00000430	R	01
BLOCK_AST	000003A8	R	01
CHECK_LOCK	0000016E	R	01
CHKWB	000001E5	R	01
CJFS_AI	= 00000003		
CJFS_BI	= 00000002		
CLRWRBCK	0000047B	R	01
CONVNL	0000045B	R	01

COPY_GBL	00000353	R	01
DEQ	0000025F	R	01
DEQ\$_ACMODE	= 0000000C		
DEQ\$_FLAGS	= 00000010		
DEQ\$_LKID	= 00000004		
DEQ\$_NARGS	= 00000004		
DEQ\$_VALBLK	= 00000008		
DEQ1	00000255	R	01
DEQBDB	000002A2	R	01
DFW_FIN	000001EA	R	01
DFW_RD	00000244	R	01
DQ	000004A9	R	01
DQBDB	00000071	R	01
DQBDB1	000002A5	R	01
DQBR	0000008D	R	01
ENQ\$_ACMODE	= 00000028		
ENQ\$_ASTADR	= 0000001C		
ENQ\$_ASTPRM	= 00000020		
ENQ\$_BLKAST	= 00000024		
ENQ\$_EFN	= 00000004		
ENQ\$_FLAGS	= 00000010		
ENQ\$_LKMODE	= 00000008		
ENQ\$_LKSB	= 0000000C		
ENQ\$_NARGS	= 0000000B		
ENQ\$_PARID	= 00000018		
ENQ\$_PROT	= 0000002C		
ENQ\$_RESNAM	= 00000014		
ENQB0G	00000486	R	01
ERRXFR	00000055	R	01
EXIT	00000048	R	01
FAB\$C_SEQ	= 00000000		
FINISH_ARGS	000001FC	R	01
FIX_BLBPTR	000002EC	R	01
FTL\$_ASTDECERR	= FFFFFFFE5		
FTL\$_BADBDB	= FFFFFFFFA		
FTL\$_BADBLB	= FFFFFFFD7		
FTL\$_BADOWNER	= FFFFFFFD6		
FTL\$_CANTDOAST	= FFFFFFFF7		
FTL\$_ENQDEQFAIL	= FFFFFFFF2		
FTL\$_NOTLOCKED	= FFFFFFFEC		
GBD\$B_CACHE_VAL	= 0000000B		
GBD\$B_FLAGS	= 0000000A		
GBD\$S_BLINK	= 00000004		
GBD\$S_FLINK	= 00000000		
GBD\$S_LOCK_ID	= 00000014		
GBD\$S_VBN	= 0000000C		
GBD\$S_VBNSEQNUM	= 00000010		
GBD\$W_NUMB	= 00000018		
GBD\$W_USECNT	= 00000020		
GBH\$S_OUTBUFQUO	= 00000050		
GBPB	0000007A	R	01
GBPB\$B_BID	= 00000008		
GBPB\$B_CACHE_VL	= 0000000B		
GBPB\$B_FLGS	= 0000000A		
GBPB\$C_BID	= 00000015		
GBPB\$S_ADDR	= 00000018		
GBPB\$S_GBD_PTR	= 00000024		

RMORELEAS
Symbol table

RELEASE BUFFER ROUTINE

N 14

16-SEP-1984 00:33:33 VAX/VMS Macro V04-00
5-SEP-1984 16:22:20 [RMS.SRC]RMORELEAS.MAR;1

Page 27
(12)

GBPB\$\$_VBN	= 0000001C		
GBPB\$\$_VBNSEQNO	= 00000020		
GBPB\$\$_NUMB	= 00000014		
GBPB\$\$_USERS	= 0000000C		
HAVE_BCB	= 0000019E	R	01
IFB\$\$_BID	= 00000008		
IFB\$\$_JNLFLG	= 000000A0		
IFB\$\$_JNLFLG2	= 000000A2		
IFB\$\$_ORGCASE	= 00000023		
IFB\$\$_BID	= 0000000B		
IFB\$\$_IDX	= 00000002		
IFB\$\$_BDB_BLNK	= 00000044		
IFB\$\$_BLBBLNK	= 0000009C		
IFB\$\$_GBH_PTR	= 00000088		
IFB\$\$_AI	= 00000008		
IFB\$\$_BI	= 00000004		
IFB\$\$_AI	= 00000003		
IFB\$\$_BI	= 00000002		
IFB\$\$_DFW	= 0000002C		
IFB\$\$_JNL	= 00000001		
IFB\$\$_AVLCL	= 00000084		
IFB\$\$_CHNL	= 00000020		
IMP\$\$_ASYQIOEFN	= 0000001F		
INVGBL	= 00000302	R	01
IO\$\$_WRITEVBLK	= 00000030		
IRB\$\$_BID	= 00000008		
IRB\$\$_BID	= 0000000A		
IRB\$\$_IFAB_LNK	= 00000000		
ISBLB	= 0000005D	R	01
KEEPLOCK	= 000002CC	R	01
L1	= 0000023C	R	01
L2	= 0000024E	R	01
LCK\$\$_NLMODE	= 00000000		
LCK\$\$_PWMODE	= 00000004		
LCK\$\$_CONVERT	= 00000002		
LCK\$\$_CVTSYS	= 00000040		
LCK\$\$_SYNCSTS	= 00000008		
LCK\$\$_SYSTEM	= 00000010		
LCK\$\$_VALBLK	= 00000001		
NOC\$\$_SH	= 000000E5	R	01
NOLOCK	= 00000065	R	01
NOTLOCKED	= 000000F9	R	01
NO_BUFF	= 000003A1	R	01
PIOSA_TRACE	= *****	X	01
RLS	= 0000001E	R	01
RLS\$\$_DEQ	= 00000008		
RLS\$\$_KEEP_LOCK	= 00000004		
RLS\$\$_RETURN	= 00000001		
RLS\$\$_WRT_THRU	= 00000002		
RLS\$\$_DEQ	= 00000003		
RLS\$\$_KEEP_LOCK	= 00000002		
RLS\$\$_RETURN	= 00000000		
RLSXIT	= 00000051	R	01
RLS_SHARED	= 00000100	R	01
RMS\$\$_AI AND BI_3	= *****	X	01
RMS\$\$_BLRFINCHK	= *****	X	01
RMS\$\$_BUG	= *****	X	01

RMS\$\$_FRCJNL	*****	X	01
RMS\$\$_FREE_LCL	*****	X	01
RMS\$\$_GET_CCL_BUFF	*****	X	01
RMS\$\$_LOWER_GBS_LOCK	*****	X	01
RMS\$\$_RAISE_GBS_LOCK	*****	X	01
RMS\$\$_RELEASE	00000016	RG	01
RMS\$\$_RETBDB	*****	X	01
RMS\$\$_RLNER1	00000013	RG	01
RMS\$\$_RLNERR	0000000F	RG	01
RMS\$\$_RLS_IN	*****	X	01
RMS\$\$_RLS_OUT	*****	X	01
RMS\$\$_SEQWTD	*****	X	01
RMS\$\$_SETEFN	*****	X	01
RMS\$\$_STALLAST	*****	X	01
RMS\$\$_SGW_GBLBUFQUO	*****	X	01
SETASTDCL	00000480	R	01
SETUP	00000402	R	01
SETWRTBCK	000003F5	R	01
STARTTHREAD	0000048D	R	01
SUCXIT	0000004E	R	01
SYSS\$\$_DCLAST	*****	GX	01
SYSS\$\$_DEQ	*****	GX	01
SYSS\$\$_ENQ	*****	GX	01
SYSS\$\$_GW_GBLBUFQUO	*****	X	01
SYSS\$\$_QIO	*****	GX	01
TPT\$\$_RELEASE	*****	X	01
TRACE	00000000	R	01
VALID	00000090	R	01
WB	00000199	R	01
WBQIOERR	000003FB	R	01
WRITE	00000152	R	01
WRTBCK	000003C4	R	01

RMC
V04

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM\$RMSO	000004B7 (1207.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
\$AB\$\$	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	35	00:00:00.10	00:00:00.76
Command processing	127	00:00:00.78	00:00:04.88
Pass 1	436	00:00:17.83	00:00:52.91
Symbol table sort	0	00:00:02.37	00:00:04.35
Pass 2	193	00:00:03.95	00:00:11.84
Symbol table output	23	00:00:00.19	00:00:00.55
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	818	00:00:25.26	00:01:15.34

The working set limit was 1950 pages.
98292 bytes (192 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1665 non-local and 34 local symbols.
1042 source lines were read in Pass 1, producing 16 object records in Pass 2.
41 pages of virtual memory were used to define 40 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	17
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	18
TOTALS (all libraries)	36

1844 GETS were required to define 36 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:RMORELEAS/OBJ=OBJ\$:RMORELEAS MSRC\$:RMORELEAS/UPDATE=(ENH\$:RMORELEAS)+EXECMLS/LIB+LIB\$:RMS/LIB

0319 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

