


```

RRRRRRR      DDDDDDD      BBBB      LL      000000      KK      KK
RRRRRRR      DDDDDDD      BBBB      LL      000000      KK      KK
RR      RR    DD      DD    BB      BB    LL      00      00      KK      KK
RR      RR    DD      DD    BB      BB    LL      00      00      KK      KK
RR      RR    DD      DD    BB      BB    LL      00      00      KK      KK
RR      RR    DD      DD    BB      BB    LL      00      00      KK      KK
RRRRRRR      DD      DD    BBBB      LL      00      00      KKKKKK
RRRRRRR      DD      DD    BBBB      LL      00      00      KKKKKK
RR      RR    DD      DD    BB      BB    LL      00      00      KK      KK
RR      RR    DD      DD    BB      BB    LL      00      00      KK      KK
RR      RR    DD      DD    BB      BB    LL      00      00      KK      KK
RR      RR    DD      DD    BB      BB    LL      00      00      KK      KK
RR      RR    DDDDDDD      BBBB      LLLLLLLLLL  000000?  KK      KK
RR      RR    DDDDDDD      BBBB      LLLLLLLLLL  000000   KK      KK

```

```

LL      11111      SSSSSSS
LL      11111      SSSSSSS
LL      11      SS
LL      11      SS
LL      11      SS
LL      11      SS
LL      11      SSSSSS
LL      11      SSSSSS
LL      11      SS
LL      11      SS
LL      11      SS
LL      11      SS
LLLLLLLLLL  11111      SSSSSSS
LLLLLLLLLL  11111      SSSSSSS

```



```

: 1 0001 0 MODULE RDBLOK (
: 2 0002 0
:001 :CDS0020 0003 0 LANGUAGE (BLISS32),
: 4-1 0004 0 IDENT = 'V04-005'
: 5 0005 1 BEGIN
: 6 0006 1
: 7 0007 1
: 8 0008 1
: 9 0009 1
:10 0010 1 *****
:11 0011 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
:12 0012 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
:13 0013 1 * ALL RIGHTS RESERVED. *
:14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
:15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
:16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
:17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
:18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
:19 0019 1 * TRANSFERRED. *
:20 0020 1
:21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
:22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
:23 0023 1 * CORPORATION. *
:24 0024 1
:25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
:26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
:27 0027 1
:28 0028 1
:29 0029 1 *****
:30 0030 1
:31 0031 1 ++
:32 0032 1
:33 0033 1 FACILITY: F11ACP Structure Level 2
:34 0034 1
:35 0035 1 ABSTRACT:
:36 0036 1
:37 0037 1 This module contains routines for basic block I/O, as well
:38 0038 1 as the buffer management mechanism.
:39 0039 1
:40 0040 1 ENVIRONMENT:
:41 0041 1
:42 0042 1 STARLET operating system, including privileged system services
:43 0043 1 and internal exec routines.
:44 0044 1
:45 0045 1 --
:46 0046 1
:47 0047 1
:48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 13-Dec-1976 22:48
:49 0049 1
:50 0050 1 MODIFIED BY:
:51 0051 1
:001 :CDS0020 0052 1 V04-005 CDS0020 Christian D. Saether 30-Dec-1984
:002 :CDS0020 0053 1 Modify FREE_ONE to return status so we can tell whether
:003 :CDS0020 0054 1 the cache interlock was released or not. If so, a
:004 :CDS0020 0055 1 found buffer must be re-searched for.
:005 :CDS0020 0056 1
:006 :CDS0020 0057 1 When freeing a buffer in FREE_ONE, unhook and return bfrd

```

```

:007 :CDS0020 0058 1 |
:008 :CDS0020 0059 1 |
:009 :CDS0020 0060 1 |
:010 :CDS0020 0061 1 |
:011 :CDS0019 0062 1 |
:012 :CDS0019 0063 1 |
:013 :CDS0019 0064 1 |
:014 :CDS0018 0065 1 |
:015 :CDS0018 0066 1 |
:016 :CDS0018 0067 1 |
:017 :CDS0018 0068 1 |
:018 :CDS0018 0069 1 |
:019 :CDS0017 0070 1 |
:020 :CDS0017 0071 1 |
:021 :CDS0017 0072 1 |
:022 :CDS0017 0073 1 |
:023 :CDS0016 0074 1 |
:024 :CDS0016 0075 1 |
:025 :CDS0016 0076 1 |
:026 :CDS0016 0077 1 |
:027 :CDS0016 0078 1 |
:028 :CDS0016 0079 1 |
:029 :CDS0016 0080 1 |
52 0081 1 |
53 0082 1 |
54 0083 1 |
55 0084 1 |
56 0085 1 |
57 0086 1 |
58 0087 1 |
59 0088 1 |
60 0089 1 |
61 0090 1 |
62 0091 1 |
63 0092 1 |
64 0093 1 |
65 0094 1 |
66 0095 1 |
67 0096 1 |
68 0097 1 |
69 0098 1 |
70 0099 1 |
71 0100 1 |
72 0101 1 |
73 0102 1 |
74 0103 1 |
75 0104 1 |
76 0105 1 |
77 0106 1 |
78 0107 1 |
79 0108 1 |
80 0109 1 |
81 0110 1 |
82 0111 1 |
83 0112 1 |
84 0113 1 |
85 0114 1 |

```

if there is no BFRL link yet.

Remove CDS0018 bugtrap as the bug has been found.

V04-004 CDS0019 Christian D. Saether 27-Nov-1984
Check buffer credits when considering multi-block reads.

V04-003 CDS0018 Christian D. Saether 15-Nov-1984
Limit the number of links followed to the buffer count when following a hash chain in UNHOOK_BFRD and bugcheck if it is exceeded to avoid infinite looping.

V04-002 CDS0017 Christian D. Saether 14-Nov-1984
Modify KILL_BUFFERS to deal with directory index blocks also.

V04-001 CDS0016 Christian D. Saether 12-Sep-1984
Deal with the situation where a buffer is currently accessed by another process.
Also delay removing the bfrd from the process list in FREE_ONE because of potential double remque if the buffer is dirty and an error occurs writing it.

V03-032 CDS0015 Christian D. Saether 30-Aug-1984
Allow for multi-header directory files.

V03-031 CDS0014 Christian D. Saether 24-Aug-1984
Removing an entry from the directory index cache free list had one too many levels of indirection, occasionally messing up the queue.

V03-030 CDS0013 Christian D. Saether 7-Aug-1984
Add KILL_DINDX routine. Remove tests for dir_fcb and primary_fcb in unhook_bfrd - the fcb\$V_dir flag handles those conditions correctly.

V03-029 CDS0012 Christian D. Saether 6-Aug-1984
Fix bug in CDS0011 causing infinite loop when mounting disk /nocache, or while marked for dismount. Use L_NORM linkage for UNHOOK_BFRD.

V03-028 CDS0011 Christian D. Saether 15-Jul-1984
Add support for maintaining directory index blocks. Add MAKE_DIRINDX routine.
Remove intermediate blocks from CDS0010. The problem was in the form of the bind_common declarations and was resolved there.

V03-027 CDS0010 Christian D. Saether 7-Jul-1984
Break up READ_BLOCK and FIND_BUFFER into smaller blocks to stop the absurd amount of cse that the compiler generates.

V03-026 CDS0009 Christian D. Saether 2-Jul- 984
Remove CACHELOCK consistency check so that cell can be used for STS_DISKREAD flag.
Raise minimum file header requirements to 3 buffers

```

86 0115 1 from 2 buffers.
87 0116 1
88 0117 1 V03-025 CDS0008 Christian D. Saether 24-Jun-1984
89 0118 1 Restore multi-block read capability.
90 0119 1
91 0120 1 V03-024 CDS0007 Christian D. Saether 20-Jun-1984
92 0121 1 Raise/lower process diocnt around qio so that file
93 0122 1 system i/o is not blocked for lack of quota. Do
94 0123 1 same for ASTCMT so it does not fail for lack of
95 0124 1 ast quota either.
96 0125 1
97 0126 1 V03-023 CDS0006 Christian D. Saether 25-May-1984
98 0127 1 When not cluster accessible, do not bump
99 0128 1 sequence number on modifies.
100 0129 1 Add routine to scan appropriate pool and kick
101 0130 1 out buffers of a given type.
102 0131 1
103 0132 1 V03-022 Expand test for clusterness to test whether we are
104 0133 1 are a cluster at all.
105 0134 1
106 0135 1 V03-021 CDS0004 Christian D. Saether 8-May-1984
107 0136 1 Do not leave lock on buffer if not cluster device.
108 0137 1 Modify buffer validation for bitmap, index, and quota
109 0138 1 type buffers.
110 0139 1
111 0140 1 V03-020 CDS0003 Christian D. Saether 10-Apr-1984
112 0141 1 Bump appropriate pool hit/miss counters.
113 0142 1 Don't set VALID, DIRTY in bfrd in CREATE_BLOCK if
114 0143 1 for special lbn -1. Set VALID, DIRTY in bfrd in
115 0144 1 RESET_LBN routine.
116 0145 1
117 0146 1 V03-019 CDS0002 Christian D. Saether 4-Apr-1984
118 0147 1 On lockbasis mismatch, when tolerated at all (for
119 0148 1 blocks in the data or headers), toss the buffer out
120 0149 1 to get the lock dissociated because it's the wrong
121 0150 1 one.
122 0151 1
123 0152 1 V03-018 CDS0001 Christian D. Saether 20-Mar-1984
124 0153 1 Establish BASE as AST parameter in SERIAL_CACHE routine.
125 0154 1
126 0155 1 V03-017 ACG0408 Andrew C. Goldstein, 23-Mar-1984 14:47
127 0156 1 Add AST parameter so that impure storage is fully based
128 0157 1
129 0158 1 V03-016 ACG0403 Andrew C. Goldstein, 15-Mar-1984 17:35
130 0159 1 Correct test for new buffer in WRONG_LOCKBASIS;
131 0160 1 fix buffer address computation in TOSS_CACHE_DATA.
132 0161 1
133 0162 1 V03-015 CDS Christian D. Saether 9-Mar-1984
134 0163 1 Rewrite cache routines for shared paged pool caching.
135 0164 1
136 0165 1 **
137 0166 1
138 0167 1
139 0168 1 LIBRARY 'SYS$LIBRARY:LIB.L32';
140 0169 1 REQUIRE 'SRCS:FCPDEF.B32';
141 1160 1
142 1161 1 FORWARD ROUTINE

```

```

:001 CDS0016 1162 1 RESOLVE_AMBIGUITY : L_NORM NOVALUE,
:143 1163 1 GET_READ_BFR_CREDITS : L_NORM NOVALUE,
:144 1164 1 RETURN_BFRD : L_JSB TARG NOVALUE,
:145 1165 1 FIND_BUFFER : L_NORM,
:146 1166 1 INVACIDATE : L_NORM NOVALUE,
:147 1167 1 UNHOOK_BFRD : L_NORM NOVALUE,
:148 1168 1 WRITE_BLOCK : L_NORM NOVALUE,
:001 CDS0020 1169 1 FREE_ONE : L_NORM,
:150-1 1170 1 WAKE_WAITER : NOVALUE,
:151 1171 1 RELEASE_CACHE : L_JSB NOVALUE,
:152 1172 1 SERIAL_CACHE : L_JSB NOVALUE;
:153 1173 1
:154 1174 1 EXTERNAL ROUTINE
:155 1175 1 WAIT_FOR_AST : NOVALUE, ! exit thread until completion ast
:156 1176 1 CONTINUE_THREAD : NOVALUE; ! completion AST to resume thread
:157 1177 1
:158 1178 1 EXTERNAL
:159 1179 1 CTL$GL_PHD : REF BBLOCK ADDRESSING_MODE (GENERAL),
:160 1180 1 CTL$GL_PCB : REF BBLOCK ADDRESSING_MODE (GENERAL);
:161 1181 1
:162 1182 1 MACRO
:163 1183 1 QFLNK = 0,0,32,0 %,
:164 1184 1 QBLNK = 4,0,32,0 %,
:165 1185 1
:166 M 1186 1 LOOKUP_LBN (LBN) =
:167 M 1187 1 BEGIN
:168 M 1188 1 BIND
:169 M 1189 1 HSHTBL = .CACHE_HDR [F11BC$L_LBNHSHBAS] : VECTOR [,WORD];
:170 M 1190 1
:171 M 1191 1 HSHTBL [ABS (LBN MOD .CACHE_HDR [F11BC$W_LBNHSHCNT])]
:172 M 1192 1 END %,
:173 M 1193 1
:174 M 1194 1
:175 M 1195 1 LOOKUP_LOCK (LOCKBASIS, PARLKID) =
:176 M 1196 1 BEGIN
:177 M 1197 1 BIND
:178 M 1198 1 HSHTBL = .CACHE_HDR [F11BC$L_BLHSHBAS] : VECTOR [,WORD];
:179 M 1199 1
:180 M 1200 1 HSHTBL [ABS ((LOCKBASIS + PARLKID) MOD .CACHE_HDR [F11BC$W_BLHSHCNT])]
:181 M 1201 1 END %,
:182 M 1202 1
:183 M 1203 1
:184 M 1204 1 ! Calculate address of given descriptor within 1-based descriptor blockvectors.
:185 M 1205 1 !
:186 M 1206 1
:187 M 1207 1 BFRD_ADDR (BAS1INDX) =
:188 M 1208 1 .CACHE_HDR [F11BC$L_BFRDBAS] + (BAS1INDX - 1)*BFRD$$_BFRDDEF %,
:189 M 1209 1
:190 M 1210 1 BFRD_ADDR (BAS1INDX) =
:191 M 1211 1 .CACHE_HDR [F11BC$L_BFRDDBAS] + (BAS1INDX - 1)*BFRD$$_BFRDDEF %;
:192 M 1212 1
:193 M 1213 1 BIND
:194 M 1214 1 POOL_TABLE = UPLIT BYTE ( 2, ! file headers
:195 M 1215 1 0, ! storage map
:196 M 1216 1 1, ! directories
:197 M 1217 1 2, ! index file blocks
:198 M 1218 1 1, ! random data blocks

```

RDBLOK
V04-005

M 14
8-Jan-1985 18:20:35 VAX-11 Bliss-32 V4.0-742
2-Oct-1984 12:43:36 [F11X.BUGSRC]RDBLOK.B32;1

Page 5
(1)

```
: 199      1219  1
: 200      1220  1
: 201      1221  1
```

```
1.      : quota file blocks
3.      : directory index blocks
) : VECTOR [ ,BYTE];
```

RE
VC

.....
00

```
203 1222 1 GLOBAL ROUTINE MAKE_DIRINDX (FCB) : L_NORM =
204 1223 1
205 1224 1 ++
206 1225 1
207 1226 1 FUNCTIONAL DESCRIPTION:
208 1227 1
209 1228 1 This routine validates the directory index pointed to from the
210 1229 1 given FCB, under the cache interlock.
211 1230 1
212 1231 1 If the given FCB does not have a directory index, attempt to
213 1232 1 attach one to it.
214 1233 1
215 1234 1 ROUTINE VALUE:
216 1235 1
217 1236 1 0 - if the given FCB is invalid
218 1237 1 1 - if the given FCB is valid
219 1238 1
220 1239 1 --
221 1240 1
222 1241 2 BEGIN
223 1242 2
224 1243 2 MAP
225 1244 2 FCB : REF BBLOCK;
226 1245 2
227 1246 2 BIND_COMMON;
228 1247 2
229 1248 2 EXTERNAL
230 1249 2 PMSSGL_DIRHIT : ADDRESSING_MODE (GENERAL),
231 1250 2 PMSSGL_DIRMISS : ADDRESSING_MODE (GENERAL);
232 1251 2 LOCAL
233 1252 2 DIRINDX : REF BBLOCK FIELD (DIRC),
234 1253 2 DNDX_BFRD : REF BBLOCK,
235 1254 2 FCBVALID : INITIAL (0);
236 1255 2
237 1256 2 IF .LB_HDRSEQ [.DIR_LCKINDX] EQL 0
238 1257 2 THEN
239 1258 2 LB_HDRSEQ [.DIR_LCKINDX] = .LB_HDRSEQ [.DIR_LCKINDX] + 1;
240 1259 2
241 1260 2 IF .LB_DATASEQ [.DIR_LCKINDX] EQL 0
242 1261 2 THEN
243 1262 2 LB_DATASEQ [.DIR_LCKINDX] = .LB_DATASEQ [.DIR_LCKINDX] + 1;
244 1263 2
245 1264 2 SERIAL_CACHE ();
246 1265 2
247 1266 2 IF (DIRINDX = .FCB [FCB$L_DIRINDX]) EQL 0
248 1267 2 THEN
249 1268 3 BEGIN
250 1269 3 LOCAL
251 1270 3 INDX0,
252 1271 3 POOL_LRU : REF BBLOCK;
253 1272 3
254 1273 3 POOL_LRU = CACHE_HDR [F11BC$Q_POOL_LRU] + 3*8;
255 1274 3
256 1275 3 REMQUE (.POOL_LRU [QFLNK], DNDX_BFRD);
257 1276 3
258 1277 3 INDX0 = ((.DNDX_BFRD - .CACHE_HDR [F11BC$L_BFRDBAS])/BFRD$$_BFRDDEF);
259 1278 3
```



```

: 260      1279  3      UNHOOK_BFRD (.INDXO + 1, .DNDX_BFRD);
: 261      1280  3
: 262      1281  3      DIRINDX = (.INDXO*512) + .CACHE_HDR [F11BC$L_BUFBASE];
: 263      1282  3
: 264      1283  3      DNDX_BFRD [BFRD$L_LBN] = .FCB;
: 265      1284  3      DNDX_BFRD [BFRD$B_BTYPE] = DIRINDX_TYPE;
: 266      1285  3      DNDX_BFRD [BFRD$L_LOCKBASIS] = .LB_BASIS [.DIR_LCKINDX];
: 267      1286  3      DNDX_BFRD [BFRD$L_UCB] = .CURRENT_UCB;
: 268      1287  3      FCB [FCB$L_DIRINDX] = .DIRINDX;
: 269      1288  3      DNDX_BFRD [BFRD$V_VALID] = 1;
: 270      1289  3      DIRINDX [DIRC$W_INUSE] = 0;
: 271      1290  3      PM$GL_DIRMISS = .PM$GL_DIRMISS + 1;
: 272      1291  3      END
: 273      1292  3  ELSE
: 274      1293  3      BEGIN
: 275      1294  3
: 276      1295  4      DNDX_BFRD = (((.DIRINDX - .CACHE_HDR [F11BC$L_BUFBASE])/512)*BFRD$S_BFRDDEF)
: 277      1296  3          + .CACHE_HDR [F11BC$L_BFRDBAS];
: 278      1297  3
: 279      1298  3      REMQUE (.DNDX_BFRD, DNDX_BFRD);
: 280      1299  3
: 281      1300  3      IF .DNDX_BFRD [BFRD$L_SEQNUM] EQL .LB_HDRSEQ [.DIR_LCKINDX]
: 282      1301  3      THEN
: 283      1302  4          BEGIN
: 284      1303  4              FCBVALID = 1;
: 285      1304  4
: 286      1305  4              IF .DIRINDX [DIRC$L_DATASEQ] NEQ .LB_DATASEQ [.DIR_LCKINDX]
: 287      1306  4              THEN
: 288      1307  5                  BEGIN
: 289      1308  5                      DIRINDX [DIRC$W_INUSE] = 0;
: 290      1309  5                      PM$GL_DIRMISS = .PM$GL_DIRMISS + 1;
: 291      1310  5                  END
: 292      1311  4              ELSE
: 293      1312  4                  PM$GL_DIRHIT = .PM$GL_DIRHIT + 1;
: 294      1313  4
: 295      1314  4              END
: 296      1315  3          ELSE
: 297      1316  4              BEGIN
: 298      1317  4                  DIRINDX [DIRC$W_INUSE] = 0;
: 299      1318  4                  PM$GL_DIRMISS = .PM$GL_DIRMISS + 1;
: 300      1319  3              END;
: 301      1320  3      END;
: 302      1321  2
: 303      1322  2  IF .BFRS_USED [3] NEQ 0
: 304      1323  2  THEN
: 305      1324  2      BUG_CHECK (XQPERR, 'should not have any in use')
: 306      1325  3  ELSE
: 307      1326  2      BFRS_USED [3] = 1;
: 308      1327  2
: 309      1328  2  IF .DNDX_BFRD [BFRD$W_CURPID] NEQ 0
: 310      1329  2  THEN
: 311      1330  2      BUG_CHECK (XQPERR, 'directory index buffer should not be in use')
: 312      1331  3  ELSE
: 313      1332  2      DNDX_BFRD [BFRD$W_CURPID] = .CTL$GL_PCB [PCB$L_PID];
: 314      1333  2
: 315      1334  2
: 316      1335  2  INSQUE (.DNDX_BFRD, .BFR_LIST [3, QBLNK]);
```

```

: 317      1336 2
: 318      1337 2 RELEASE_CACHE ();
: 319      1338 2
: 320      1339 2 .FCBVALID
: 321      1340 2
: 322      1341 1 END;
! of routine MAKE_DIRINDX

```

```

.TITLE RDBLOK
.IDENT \V04-005\
.PSECT $CODE$,NOWRT,2
03 01 01 02 01 00 02 0000 P.AAA: .BYTE 2, 0, 1, 2, 1, 1, 3
POOL_TABLE= P.AAA
.EXTRN WAIT_FOR_AST, CONTINUE_THREAD
.EXTRN CTLSGL_PFD, CTLSGL_PCB
.EXTRN PMSSGL_DIRHIT, PMSSGL_DIRMISS
.EXTRN BUGS_XOPERR

.ENTRY MAKE_DIRINDX, Save R2,R3,R4,R5,R6,R7,R8 : 1222
58      CC      AA      9E      00002      MOVAB      -52(BASE), R8 : 1244
55      F4      AA      9E      00006      MOVAB      -12(BASE), R5
56      00D4    CA      9E      0000A      MOVAB      212(BASE), R6
50      66      D4      0000F      CLRL      FCBVALID
50      0094    CA40    DE      00C14      MOVL      (R6), R0 : 1256
50      60      D5      0001A      MOVAL     148(BASE)[R0], R0
50      02      12      0001C      TSTL      (R0)
50      60      D6      0001E      BNEQ      1$
50      00A8    CA40    DE      00020      1$: MOVL      (R6), R0 : 1258
50      66      D0      00020      MOVAL     168(BASE)[R0], R0 : 1260
50      60      D5      00029      TSTL      (R0)
50      02      12      0002B      BNEQ      2$
50      60      D5      0002D      INCL      (R0) : 1262
50      0000V  30      0002F      2$: BSBW     SERIAL_CACHE : 1264
54      04      AC      D0      00032      MOVL      FCB, R0 : 1266
54      00B0    C0      D0      00036      MOVL      176(R0), DIRINDX
50      54      12      0003B      BNEQ      3$
50      FC      AA      00000040 8F      C1      0003D      ADDL3     #64, -4(BASE), POOL_LRU : 1273
52      00      B0      0F      00046      REMQIJE  @0(PPOOL_LRU), DNDX_BFRD : 1275
50      FC      AA      D0      0004A      MOVL      -4(BASE), R0 : 1277
50      52      18      A0      C3      0004E      SUBL3     24(R0), DNDX_BFRD, R0
53      50      20      C7      00053      DIVL3     #32, R0, INDX0
50      52      DD      00057      PUSHL     DNDX_BFRD : 1279
50      01      A3      9F      00059      PUSHAB    1(INDX0)
53      0000V  CF      02      FB      0005C      CALLS     #2, UNHOOK_BFRD : 1281
54      53      09      78      00061      ASHL      #9, R3, R3
50      FC      BA      C1      00065      ADDL3     @-4(BASE), R3, DIRINDX
50      08      A2      04      AC      D0      0006A      MOVL      FCB, 8(DNDX_BFRD) : 1283
50      19      A2      06      90      0006F      MOVAB     #6, 25(DNDX_BFRD) : 1284
50      66      D0      00073      MOVL      (R6), R0 : 1285
50      10      A2      00B0    CA40    D0      00076      MOVL      128(BASE)[R0], 16(DNDX_BFRD)
50      0C      A2      94      AA      D0      0007D      MOVL      -108(BASE), 12(DNDX_BFRD) : 1286
50      04      AC      D0      00082      MOVL      FCB, R0 : 1287
50      00B0    C0      54      D0      00086      MOVL      DIRINDX, 176(R0)

```

18	A2		08	88	0008B	BISB2	#8, 24(DNDX_BFRD)	1288	
			3D	11	0008F	BRB	4\$	1289	
51	50	FC	AA	D0	00091	3\$:	MOVL	-4(BASE), R0	1295
	54		60	C3	00095		SUBL3	(R0), DIRINDX, R1	
	51	00000200	8F	C6	00099		DIVL2	#512, R1	
	51		20	C4	000A0		MULL2	#32, R1	
52	51	18	A0	C1	000A3		ADDL3	24(R0), R1, DNDX_BFRD	1296
	52		62	0F	000A8		REMQUE	(DNDX_BFRD), DNDX_BFRD	1298
	50		66	D0	000AB		MOVL	(R6), R0	1300
	0094	CA40	14	A2	D1	000AE	CPL	20(DNDX_BFRD), 148(BASE)[R0]	
			17	12	000B5		BNEQ	4\$	
	57		01	D0	000B7		MOVL	#1, FCBVALID	1303
	50		66	D0	000BA		MOVL	(R6), R0	1305
	00AB	CA40	08	A4	D1	000BD	CPL	8(DIRINDX), 168(BASE)[R0]	
			08	12	000C4		BNEQ	4\$	
		00000000G	00	D6	000C6		INCL	PMS\$GL_DIRHIT	1312
			08	11	000CC		BRB	5\$	1300
		00000000G	64	B4	000CE	4\$:	CLRW	(DIRINDX)	1317
		06	00	D6	000D0		INCL	PMS\$GL_DIRMISS	1318
			A5	B5	000D6	5\$:	TSTW	6(R5)	1325
			06	13	000D9		BEQL	6\$	
					FEFF 000DB		BUGW		1325
					0000* 000DD		.WORD	<BUGS_XQPERR!4>	
	06	A5	04	11	000DF		BRB	7\$	
			01	B0	000E1	6\$:	MOVW	#1, 6(R5)	1327
		1C	A2	B5	000E5	7\$:	TSTW	28(DNDX_BFRD)	1329
			06	13	000E8		BEQL	8\$	
					FEFF 000EA		BUGW		1331
					0000* 000EC		.WORD	<BUGS_XQPERR!4>	
			0C	11	000EE		BRB	9\$	
	1C	50	00	D0	000F0	8\$:	MOVL	CTL\$GL_PCB, R0	1333
		60	A0	B0	000F7		MOVW	96(R0), 28(DNDX_BFRD)	
	1C	B8	62	0E	000FC	9\$:	INSQUE	(DNDX_BFRD), @28(R8)	1335
					0000V 30 00100		BSBW	RELEASE CACHE	1337
		50	57	D0	00103		MOVL	FCBVALID, R0	1341
					04 00106		RET		

; Routine Size: 263 bytes, Routine Base: \$CODE\$ + 0007

```
.. 324      1342  1 GLOBAL ROUTINE KILL_DINDX (FCB) : L_NORM NOVALUE =
.. 325      1343  1
.. 326      1344  1  ++
.. 327      1345  1
.. 328      1346  1  FUNCTIONAL DESCRIPTION:
.. 329      1347  1
.. 330      1348  1      Invalidate the directory index block pointed to by
.. 331      1349  1      the given fcb, if any. This is done under the cache
.. 332      1350  1      serialization lock.
.. 333      1351  1  --
.. 334      1352  1
.. 335      1353  1
.. 336      1354  2 BEGIN
.. 337      1355  2
.. 338      1356  2 MAP
.. 339      1357  2     FCB      : REF BBLOCK;
.. 340      1358  2
.. 341      1359  2 BIND_COMMON;
.. 342      1360  2
.. 343      1361  2 LOCAL
.. 344      1362  2     INDX0,
.. 345      1363  2     BFRD   : REF BBLOCK,
.. 346      1364  2     DIRINDX : REF BBLOCK,
.. 347      1365  2     PIDINDX : WORD;
.. 348      1366  2
.. 349      1367  2 SERIAL_CACHE ();
.. 350      1368  2
.. 351      1369  2 IF (DIRINDX = .FCB [FCBSL_DIRINDX]) NEQ 0
.. 352      1370  2 THEN
.. 353      1371  2     BEGIN
.. 354      1372  2     INDX0 = (.DIRINDX - .CACHE_HDR [F11BCSL_BUFBASE])/512;
.. 355      1373  2     BFRD = .CACHE_HDR [F11BCSL_BFRDBAS] + BFRD$$_BFRDDEF*(.INDX0);
.. 356      1374  2
.. 357      1375  2     PIDINDX = .CTL$GL_PCB [PCBSL_PID];
.. 358      1376  2
.. 359      1377  2     IF .BFRD [BFRD$W_CURPID] NEQ 0
.. 360      1378  2     THEN
.. 361      1379  2         BEGIN
.. 362      1380  2             IF .BFRD [BFRD$W_CURPID] NEQ .PIDINDX
.. 363      1381  2             THEN
.. 364      1382  2                 BUG_CHECK (XQPERR, 'should not belong to anyone else')
.. 365      1383  2             ELSE
.. 366      1384  2                 BEGIN
.. 367      1385  2
.. 368      1386  2 ! If this one is in-process, simply clear valid to cause it to
.. 369      1387  2 ! be cleaned up when locks are run down. Also clear the pointer
.. 370      1388  2 ! to the fcb in the bfrd.
.. 371      1389  2
.. 372      1390  2
.. 373      1391  2         BFRD [BFRD$V_VALID] = 0;
.. 374      1392  2         BFRD [BFRD$L_LBN] = 0;
.. 375      1393  2         END;
.. 376      1394  2     END
.. 377      1395  2 ELSE
.. 378      1396  2     BEGIN
.. 379      1397  2     UNHOOK_BFRD (.INDX0+1, .BFRD);
.. 380      1398  2     RETURN_BFRD (.BFRD);
```

```

: 381      1399      3      END;
: 382      1400
: 383      1401      FCB [FCB$L_DIRINDX] = 0;
: 384      1402
: 385      1403      END;
: 386      1404
: 387      1405      RELEASE_CACHE ();
: 388      1406
: 389      1407      1 END;          ! of routine KILL_DINDX

```

52

```

                                000C 00000
                                0000V 30 00002
50      04      AC      D0 00005      .ENTRY KILL_DINDX, Save R2,R3
50      0080    C0      D0 00009      BSBW SERIAL_CACHE
                                54 13 0000E      MOVL FCB, R0
                                BA C2 00010      MOVL 176(R0), DIRINDX
50      FC      BA      C2 00010      BEQL 4$
50      00000200 8F C6 00014      SUBL2 @-4(BASE), R0
51      FC      AA      D0 0001B      DIVL2 #512, INDX0
50      18      A1      C0 00023      MOVL -4(BASE), R1
52      00000000G 00 D0 00027      ASHL #5, INDX0, R2
53      60      A1      B0 0002E      ADDL2 24(R1), BFRD
                                15 13 00035      MOVL CTL$GL_PCB, R1
53      1C      A2      B1 00037      MOVW 96(R1), PIDINDX
                                06 13 0003B      TSTW 28(BFRD)
                                FEFF 0003D      BEQL 2$
                                0000* 0003F      CMPW 28(BFRD), PIDINDX
18      A2      08      8A 00043 1$:  BUGW 1$
                                10 11 0004A      .WORD <BUGS_XQPERR!4>
                                52 DD 0004C 2$: BRB 3$
0000V CF      01      A0 9F 0004E      BICB2 #8, 24(BFRD)
50      52      D0 00056      CLRL 8(BFRD)
                                0000V 30 00059      BRB 3$
50      04      AC      D0 0005C 3$: PUSHL BFRD
                                0080 C0 D4 00060      PUSHAB 1(INDX0)
                                0000V 30 00064 4$: MOVL #2, UNHOOK_BFRD
                                04 00067      MOVL BFRD, R0
                                04 00067      BSBW RETURN_BFRD
                                04 00067      MOVL FCB, R0
                                04 00067      CLRL 176(R0)
                                04 00067      BSBW RELEASE_CACHE
                                04 00067      RET

```

: Routine Size: 104 bytes, Routine Base: \$CODE\$ + 010E

```
391 1408 1 GLOBAL ROUTINE READ_BLOCK (LBN, COUNT, TYPE) : L_NORM =
392 1409 1
393 1410 1 :++
394 1411 1
395 1412 1 FUNCTIONAL DESCRIPTION:
396 1413 1
397 1414 1 This routine reads the desired block(s) from the disk.
398 1415 1 Blocks are categorized by type to aid buffer management.
399 1416 1 Note that the caller assumes only one block is ever read; multiple
400 1417 1 blocks read ahead are acquired through cache hits on subsequent calls.
401 1418 1 CALLING SEQUENCE:
402 1419 1 READ_BLOCK (ARG1, ARG2, ARG3)
403 1420 1
404 1421 1 INPUT PARAMETERS:
405 1422 1 ARG1: LBN of block(s)
406 1423 1 ARG2: number of blocks to read
407 1424 1 ARG3: block type code
408 1425 1
409 1426 1 IMPLICIT INPUTS:
410 1427 1
411 1428 1 OUTPUT PARAMETERS:
412 1429 1 NONE
413 1430 1
414 1431 1 IMPLICIT OUTPUTS:
415 1432 1 IO_STATUS receives status of I/O transfer
416 1433 1
417 1434 1 ROUTINE VALUE:
418 1435 1 address of buffer containing block
419 1436 1
420 1437 1 SIDE EFFECTS:
421 1438 1 BLOCK READ
422 1439 1
423 1440 1 --
424 1441 1
425 1442 2 BEGIN
426 1443 2
427 1444 2 LOCAL
428 1445 2
429 1446 2 I, ! index of buffer used
430 1447 2 BFRD : REF BBLOCK, ! pointer to buffer descriptor
431 1448 2 STATUS, ! QIO service status
432 1449 2 FOUND_COUNT; ! count of buffers gotten
433 1450 2
434 1451 2 EXTERNAL
435 1452 2 ACPSGB_DATACHK : BITVECTOR ADDRESSING MODE (ABSOLUTE);
436 1453 2 ! ACP datacheck enable flags
437 1454 2
438 1455 2 BIND_COMMON;
439 1456 2
440 1457 2 EXTERNAL LITERAL
441 1458 2 ACPSV_READCHK : UNSIGNED (6); ! read check enable flag
442 1459 2
443 1460 2 ! Find a suitable block buffer. If it does not already contain the block,
444 1461 2 ! read it.
445 1462 2
446 1463 2
447 1464 2 STSFLGS [STS_DISKREAD] = 0;
```

```
448 1465 I = FIND_BUFFER (.LBN, .TYPE, .COUNT, FOUND_COUNT);
449 1466
450 1467 BFRD = .CACHE_HDR [F11BCSL_BFRDBAS] + (.I*BFRDSS_BFRDDEF);
451 1468
452 1469
453 1470 IF .BFRD [BFRD$V_VALID]
454 1471 THEN
455 1472     RETURN .CACHE_HDR [F11BCSL_BUFBASE] + (.I*512);
456 1473
457 1474 BEGIN
458 1475 LOCAL
459 1476     PTR : REF BBLOCK,
460 1477     SAVE_PRIV : VECTOR [4];
461 1478
462 1479 STSFLGS [STS_DISKREAD] = 1;
463 1480
464 1481 PTR = .CTL$GL_PCB;
465 1482 PTR [PCBSW_DIOCNT] = .PTR [PCBSW_DIOCNT] + 1;
466 1483 PTR [PCBSW_ASTCNT] = .PTR [PCBSW_ASTCNT] + 1;
467 1484 SAVE_PRIV [0] = .(PTR [PCBSQ_PRIV]);
468 1485 SAVE_PRIV [1] = .(PTR [PCBSQ_PRIV]+4);
469 1486 BBLOCK [PTR [PCBSQ_PRIV], PRVSV_LOG IO] = 1;
470 1487 BBLOCK [PTR [PCBSQ_PRIV], PRVSV_BYPASS] = 1;
471 1488 PTR = .CTL$GL_PHD;
472 1489 SAVE_PRIV [2] = .(PTR [PHDSQ_PRIVMSK]);
473 1490 SAVE_PRIV [3] = .(PTR [PHDSQ_PRIVMSK]+4);
474 1491 BBLOCK [PTR [PHDSQ_PRIVMSK], PRVSV_LOG IO] = 1;
475 1492 BBLOCK [PTR [PHDSQ_PRIVMSK], PRVSV_BYPASS] = 1;
476 1493
477 1494 PMS_TOT_READ = .PMS_TOT_READ + 1;
478 1495 STATUS = $QIO (
479 1496     EFN = EFN,
480 1497     ASTADR = CONTINUE_THREAD,
481 1498     ASTPRM = .BASE,
482 1499     CHAN = .IO_CHANNEL,
483 1500     FUNC = (IOS_READBLK
484 1501             OR .ACPSGB_DATACHK[ACPSV_READCHK]
485 1502             ^ $BITPOSITION (IOSV_DATACHECK)),
486 1503     IOSB = IO_STATUS,
487 1504     P1 = .CACHE_HDR [F11BCSL_BUFBASE] + (.I*512),
488 1505     P2 = .FOUND_COUNT*512,
489 1506     P3 = .LBN
490 1507 );
491 1508
492 1509 (PTR [PHDSQ_PRIVMSK]) = .SAVE_PRIV [2];
493 1510 (PTR [PHDSQ_PRIVMSK]+4) = .SAVE_PRIV [3];
494 1511 PTR = .CTL$GL_PCB;
495 1512 PTR [PCBSW_DIOCNT] = .PTR [PCBSW_DIOCNT] - 1;
496 1513 PTR [PCBSW_ASTCNT] = .PTR [PCBSW_ASTCNT] - 1;
497 1514 (PTR [PCBSQ_PRIV]) = .SAVE_PRIV [0];
498 1515 (PTR [PCBSQ_PRIV]+4) = .SAVE_PRIV [1];
499 1516
500 1517 IF NOT .STATUS
501 1518 THEN IO_STATUS = .STATUS
502 1519 ELSE WAIT_FOR_AST();
503 1520
504 1521 IF NOT .IO_STATUS
```

```

: 505      1522  3      THEN
: 506      1523  4      BEGIN
: 507      1524  4      INCR J FROM 0 TO .FOUND_COUNT-1
: 508      1525  4      DO
: 509      1526  4      INVALDATE (.CACHE_HDR [F11BC$$_BUFBASE] + (.I+.J)*512);
: 510      1527  4      ERR_EXIT (.IO_STATUS[0]);
: 511      1528  3      END;
: 512      1529  3
: 513      1530  3
: 514      1531  3      INCR J FROM 0 TO .FOUND_COUNT - 1
: 515      1532  4      DO
: 516      1533  4      BEGIN
: 517      1534  4      BFRD [BFRDSV_VALID] = 1;
: 518      1535  4      BFRD = .BFRD + BFRD$$_BFRDDEF;
: 519      1536  3      END;
: 520      1537  3      END;
: 521      1538  3
: 522      1539  3      RETURN .CACHE_HDR [F11BC$$_BUFBASE] + (.I*512)
: 523      1540  3
: 524      1541  1      END;

```

! end of routine READ_BLOCK

					.EXTRN	ACPSGB DATACHK, ACPSV_READCHK	
					.EXTRN	SYSSQID	
					.ENTRY	READ_BLOCK, Save R2,R3,R4,R5,R6,R7	1408
					MOVAB	CTL\$GL PCB, R7	
					SUBL2	#20, SP	
					MOVAB	-4(BASE), R4	1452
					BICB2	#1, -90(BASE)	1464
					PUSHL	SP	1466
					PUSHL	COUNT	
					PUSHL	TYPE	
					PUSHL	LBN	
					CALLS	#4, FIND_BUFFER	
					MOVL	R0, I	1468
					MOVL	(R4), R1	
					ASHL	#5, I, R0	
					ADDL3	24(R1), R0, BFRD	
					BBC	#3, 24(BFRD), 1\$	1470
					BRW	9\$	
					BISB2	#1, -90(BASE)	1479
					MOVL	CTL\$GL PCB, PTR	1481
					INCW	62(PTR)	1482
					INCW	56(PTR)	1483
					MOVQ	132(PTR), SAVE_PRIV	1484
					BISL2	#536871040, 132(PTR)	1487
					MOVL	CTL\$GL_PHD, PTR	1488
					MOVQ	(PTR) - SAVE_PRIV+8	1489
					BISL2	#536871040, -(PTR)	1492
					INCL	2280(BASE)	1494
					CLRQ	-(SP)	1507
					CLRL	-(SP)	
					PUSHL	LBN	
					ASHL	#9, FOUND_COUNT, -(SP)	
					MOVL	@0(R4), RT	

50	56	09 78 0007C	ASHL	#9, I, R0	:	
		6041 9F 00081	PUSHAB	(R0)[R1]	:	
		5A DD 00084	PUSHL	BASE	:	
	0000G	CF 9F 00086	PUSHAB	CONTINUE THREAD	:	
	88	AA 9F 0008A	PUSHAB	-120(BASE)	:	
50 00 J00000G	9F 01	00G EF 0008D	EXTZV	S^ACPSV_READCHK, #1, @#ACPSGB_DATACHK, R0	:	
	50	0E 78 00096	ASHL	#14, R0, R0	:	
	7E 50	21 C9 0009A	BISL3	#33, R0, -(SP)	:	
		FF78 CA DD 0009E	PUSHL	-136(BASE)	:	
		1E DD 000A2	PUSHL	#30	:	
00000000G	00	0C FB 000A4	CALLS	#12, SYSSQIO	:	
	62	0C AE 7D 000AB	MOVQ	SAVE PRIV+8, (PTR)	:	1509
	52	67 DO 000AF	MOVL	CTL\$GL_PCB, PTR	:	1511
		3E A2 B7 000B2	DECW	62(PTR)	:	1512
		38 A2 B7 000B5	DECW	56(PTR)	:	1513
0084	C2	04 AE 7D 000B8	MOVQ	SAVE PRIV, 132(PTR)	:	1514
	06	50 E8 000BE	BLBS	STATUS, 2\$:	1517
88	AA	50 DO 000C1	MOVL	STATUS, -120(BASE)	:	1518
		05 11 000C5	BRB	3\$:	
0000G	CF	00 FB 000C7	CALLS	#0, WAIT FOR AST	:	1519
	24	88 AA E8 000CC	BLBS	-120(BASE), 8\$:	1521
	55	6E DO 000D0	MOVL	FOUND_COUNT, R5	:	1524
	52	01 CE 000D3	MNEGL	#1, J	:	
		14 11 000D6	BRB	5\$:	
	51	00 B4 DO 000D8	MOVL	@0(R4), R1	:	1526
50	56	52 C1 000DC	ADDL3	J, I, R0	:	
50	50	09 78 000E0	ASHL	#9, R0, R0	:	
		6041 9F 000E4	PUSHAB	(R0)[R1]	:	
	0000V	CF 01 FB 000E7	CALLS	#1, INVALIDATE	:	
E8	52	88 55 F2 000EC	AOBLSS	R5, J, 4\$:	
		04 00CF3	CHMU	-120(BASE)	:	1527
		6E DO 000F4	RET		:	
	51	01 CE 000F7	MOVL	FOUND_COUNT, R1	:	1530
	50	07 11 000FA	MNEGL	#1, J	:	
18	A3	08 88 000FC	BRB	8\$:	
	53	20 C0 00100	BISB2	#8, 24(BFRD)	:	1533
		51 F2 00103	ADD 2	#32, BFRD	:	1534
F5	50	09 78 00107	AOBLSS	R1, J, 7\$:	1530
50	56	00 B4 C0 0010B	ASHL	#9, I, R0	:	1539
	50	04 0010F	ADDL2	@0(R4), R0	:	
			RET		:	1541

; Routine Size: 272 bytes, Routine Base: \$CODE\$ + 0176

```

: 526      1542 1 POUTINE SERIAL_CACHE : L_JSB NOVALUE =
: 527      1543 1
: 528      1544 1 :++
: 529      1545 1
: 530      1546 1 : FUNCTIONAL DESCRIPTION:
: 531      1547 1
: 532      1548 1 : Serialize cache processing by queuing the CDRP part of our
: 533      1549 1 : IO packet onto the AQB queue. Go to sleep if someone else
: 534      1550 1 : is already there.
: 535      1551 1
: 536      1552 1 :--
: 537      1553 1
: 538      1554 2 BEGIN
: 539      1555 2
: 540      1556 2 BIND_COMMON;
: 541      1557 2
: 542      1558 2 EXTERNAL
: 543      1559 2     PMS$GL_XQPCACHEWAIT      : ADDRESSING_MODE (GENERAL);
: 544      1560 2
: 545      1561 2 BUILTIN
: 546      1562 2     TESTBITSS,
: 547      1563 2     INSQUE;
: 548      1564 2
: 549      1565 2 LOCAL
: 550      1566 2     AQB      : REF BBLOCK,
: 551      1567 2     ACB      : REF BBLOCK;
: 552      1568 2
: 553      1569 2 AQB = .CURRENT_VCB [VCBSL_AQB];
: 554      1570 2
: 555      1571 2 IF (ACB = .ACB_ADDR) EQL 0
: 556      1572 2 THEN
: 557      1573 3     BEGIN
: 558      1574 3     ACB_ADDR = (ACB = .IO_PACKET + IRPSC_CDRP);
: 559      1575 3     ACB [ACBSL_PID] = .CT[$GL_PCB [PCBSL_PID]];
: 560      1576 3     ACB [ACBSL_AST] = CONTINUE_THREAD;
: 561      1577 3     ACB [ACBSL_ASTPRM] = .BASE;
: 562      1578 3     ACB [ACBSB_RMOD] = PLSL_KERNEL + ACBSM_NODELETE;
: 563      1579 3     ACB [ACBSB_TYPE] = DYN$C_ACB;
: 564      1580 3     ACB [ACBSW_SIZE] = 0;
: 565      1581 3     END;
: 566      1582 2
: 567      1583 2 IF INSQUE (.IO_PACKET, .AQB [AQB$S_ACPOBL])
: 568      1584 2 THEN
: 569      1585 2     RETURN
: 570      1586 2 ELSE
: 571      1587 3     BEGIN
: 572      1588 3     PMS$GL_XQPCACHEWAIT = .PMS$GL_XQPCACHEWAIT + 1;
: 573      1589 3     WAIT_FOR_AST ();
: 574      1590 2     END;
: 575      1591 2
: 576      1592 1 END;
```

.EXTRN PMS\$GL_XQPCACHEWAIT

52 DD 0000 SERIAL_CACHE:

		50	98	AA	D0	00002		PUSHL	R2	:	1542
		52	10	AO	D0	00006		MOVL	-104(BASE), R0	:	1569
		50	C8	AA	D0	0000A		MOVL	16(R0), AQB	:	
				2B	12	0000E		MOVL	-56(BASE), ACB	:	1571
50	90	AA	00000060	8F	C1	00010		BNEQ	1\$:	
	C8	AA		50	D0	00019		ADDL3	#96, -112(BASE), ACB	:	1574
		5	00000000G	00	D0	0001D		MOVL	ACB, -56(BASE)	:	
	0C	A0	60	A1	D0	00024		MOVL	CTL\$GL_PCB, R1	:	1575
	10	A0	0000G	CF	9E	00029		MOVL	96(R1), 12(ACB)	:	
	14	A0		5A	D0	0002F		MOVAB	CONTINUE_THREAD, 16(ACB)	:	1576
	08	A0	20020000	8F	D0	00033		MOVL	BASE, 20(ACB)	:	1577
	04	B2	90	BA	0E	0003B	1\$:	MOVL	#537001984, 8(ACB)	:	1580
				0B	13	00040		INSQUE	@-112(BASE), @4(AQB)	:	1583
				00	D6	00042		BEQL	2\$:	
	0000G	CF	00000000G	00	FB	00048		INCL	PMSSGL_XOPCACHEWAIT	:	1588
				00	FB	00048		CALLS	#0, WAIT_FOR_AST	:	1589
				04	BA	0004D	2\$:	POPR	#^M<R2>	:	1592
				05	0004F			RSB		:	

; Routine Size: 80 bytes, Routine Base: \$CODE\$ + 0286

```

: 578 1593 1 GLOBAL ROUTINE FIND_BUFFER (LBN, TYPE, COUNT, FOUND_COUNT) : L_NORM =
: 579 1594 1
: 580 1595 1 !++
: 581 1596 1
: 582 1597 1 FUNCTIONAL DESCRIPTION:
: 583 1598 1
: 584 1599 1
: 585 1600 1 CALLING SEQUENCE:
: 586 1601 1
: 587 1602 1 INPUT PARAMETERS:
: 588 1603 1
: 589 1604 1 IMPLICIT INPUTS:
: 590 1605 1
: 591 1606 1 OUTPUT PARAMETERS:
: 592 1607 1
: 593 1608 1 IMPLICIT OUTPUTS:
: 594 1609 1
: 595 1610 1 ROUTINE VALUE:
: 596 1611 1     index of first buffer found
: 597 1612 1
: 598 1613 1 SIDE EFFECTS:
: 599 1614 1     LRU list relinked, buffers may be written
: 600 1615 1
: 601 1616 1 !--
: 602 1617 1
: 603 1618 2 BEGIN
: 604 1619 2
: 605 1620 2 MAP
: 606 1621 2     TYPE      : BYTE;
: 607 1622 2
: 608 1623 2 BIND_COMMON:
: 609 1624 2
: 610 1625 2 EXTERNAL
: 611 1626 2     ACP$GB_MAXREAD      : BYTE ADDRESSING_MODE (ABSOLUTE),
: 612 1627 2     PM$SGL_FILHDR_HIT   : ADDRESSING_MODE (ABSOLUTE),
: 613 1628 2     PM$SGL_FILHDR_MISS  : ADDRESSING_MODE (ABSOLUTE),
: 614 1629 2     PM$SGL_DIRDATA_HIT  : ADDRESSING_MODE (ABSOLUTE),
: 615 1630 2     PM$SGL_DIRDATA_MISS : ADDRESSING_MODE (ABSOLUTE),
: 616 1631 2     PM$SGL_STORAGMAP_HIT : ADDRESSING_MODE (ABSOLUTE),
: 617 1632 2     PM$SGL_STORAGMAP_MISS : ADDRESSING_MODE (ABSOLUTE);
: 618 1633 2
: 619 1634 2 LOCAL
:001 :CDS0016 1635 2     LOOKUP_AGAIN,
: 620 1636 2     FND_CNT,
: 621 1637 2     INDX,
: 622 1638 2     POOL,
: 623 1639 2     INDX_ADDR,
: 624 1640 2     LOCKBASIS,
: 625 1641 2     SEQNUM,
: 626 1642 2     PIDINDX      : WORD,
: 627 1643 2     BFRD         : REF BBLOCK,
: 628 1644 2     BFRL         : REF BBLOCK,
: 629 1645 2     POOL_LRU    : REF BBLOCK;
: 630 1646 2
: 631 1647 2 LABEL
:001 :CDS0016 1648 2     GOT_ONE,
:002 :CDS0C16 1649 2     LOOKUP;

```



```

: 727      1707      3
: 728      1708      3
: 729      1709      3
: 730      1710      2
: 731      1711      2
: 732      1712      3
: 733      1713      3
: 734      1714      3
: 735      1715      3
: 736      1716      3
: 737      1717      2
: 738      1718      2
: 739      1719      2
: 740      1720      2
: 741      1721      3
: 742      1722      3
: 743      1723      3
: 744      1724      3
: 745      1725      3
: 746      1726      3
: 747      1727      2
: 748      1728      2
: 749      1729      2
: 750      1730      2
: 751      1731      2
: 752      1732      2
: 753      1733      2
: 754      1734      2
: 755      1735      2
: 756      1736      3
: 757      1737      3
: 758      1738      3
: 759      1739      3
: 760      1740      3
: 761      1741      3
: 762      1742      2
: 763      1743      2
: 764      1744      3
: 765      1745      3
: 766      1746      3
: 767      1747      3
: 768      1748      3
: 769      1749      3
: 770      1750      3
: 771      1751      2
: 772      1752      2
: 773      1753      2
: 774      1754      3
: 775      1755      3
: 776      1756      3
: 777      1757      3
: 778      1758      3
: 779      1759      4
: 780      1760      4
: 781      1761      4
: 782      1762      3
: 783      1763      3

      SEQNUM = .LB_HDRSEQ [.CURR_LCKINDX];
      END;

[DIRECTORY_TYPE]:
      BEGIN
      IF .LB_DATASEQ [.DIR_LCKINDX] EQL 0
      THEN
          LB_DATASEQ [.DIR_LCKINDX] = 1;

      SEQNUM = .LB_DATASEQ [.DIR_LCKINDX];
      END;

[DATA_TYPE]:
      BEGIN
      IF .LB_DATASEQ [.CURR_LCKINDX] EQL 0
      THEN
          LB_DATASEQ [.CURR_LCKINDX] = 1;

      SEQNUM = .LB_DATASEQ [.CURR_LCKINDX];
      END;

! The storage bitmap, index file bitmap, and quota file data blocks
! get their sequence numbers from the volume lock value block.
! Check with the code in allocation_lock and allocation_unlock before
! changing any of this.

[BITMAP_TYPE]:
      BEGIN
      IF .(LB_DATASEQ [0])<0,16,0> EQL 0
      THEN
          (LB_DATASEQ [0])<0,16,0> = 1;

      SEQNUM = .(LB_DATASEQ [0])<0,16,0>;
      END;

[INDEX_TYPE]:
      BEGIN
      IF .(LB_DATASEQ [0])<16,16,0> EQL 0
      THEN
          (LB_DATASEQ [0])<16,16,0> = 1;

      SEQNUM = .(LB_DATASEQ [0])<16,16,0>;
      END;

[QUOTA_TYPE]:
      BEGIN
      SEQNUM = .SAVE_VC_FLAGS<1,15,0>;

      IF .SEQNUM EQL 0
      THEN
          BEGIN
          SEQNUM = .SEQNUM + 1;
          SAVE_VC_FLAGS<1,15,0> = .SEQNUM;
          END;
      END;

```

```

: 784      1764 2      END;
: 785      1765 2
: 786      1766 2      TES;
: 787      1767 2
:001 CDS0016 1768 2 : Search the buffer descriptors for the desired LdN. This is in
:002 CDS0016 1769 2 : a large loop because if the buffer is currently in use by another
:003 CDS0016 1770 2 : process, we have a lockbasis mismatch and will need to go around
:004 CDS0016 1771 2 : again.
:005 CDS0016 1772 2 : Serialize on the cache interlock at this point as we are now searching
:006 CDS0016 1773 2 : and modifying shared structures.
:007 CDS0016 1774 2
:008 CDS0016 1775 2
:009 CDS0016 1776 2 SERIAL_CACHE ();
:010 CDS0016 1777 2
:011 CDS0016 1778 2 DO
:012 CDS0016 1779 2 LOOKUP.
:013 CDS0016 1780 2 BEGIN
:014 CDS0016 1781 2
:015 CDS0016 1782 3 LOOKUP_AGAIN = 0;
:016 CDS0016 1783 3
:017 CDS0016 1784 3 : If this is for LBN = -1, we just want a buffer. Don't look for it,
:018 CDS0016 1785 3 : because other processes may be doing this also, and we don't want
:019 CDS0016 1786 3 : to find theirs.
:020 CDS0016 1787 3
:021 CDS0016 1788 3
:022 CDS0016 1789 3 IF (.LBN+1) EQL 0
:023 CDS0016 1790 3 THEN
:024 CDS0016 1791 3     INDX = 0
:025 CDS0016 1792 3 ELSE
:026 CDS0016 1793 4     BEGIN
:027 CDS0016 1794 4
:028 CDS0016 1795 4 : Get initial index by hashing. Follow links, if any, until we
:029 CDS0016 1796 4 : get match on both LBN and UCB, or we run out of links to follow.
:030 CDS0016 1797 4
:031 CDS0016 1798 4
:032 CDS0016 1799 4     INDX = .LOOKUP_LBN (.LBN)<0,16>;
:033 CDS0016 1800 4
:034 CDS0016 1801 4     WHILE .INDX NEQ 0
:035 CDS0016 1802 4     DO
:036 CDS0016 1803 5         BEGIN
:037 CDS0016 1804 5
:038 CDS0016 1805 5         BFRD = BFRD_ADDR (.INDX);
:039 CDS0016 1806 5
:040 CDS0016 1807 5 : Determine if this in fact matches on both LBN and UCB.
:041 CDS0016 1808 5
:042 CDS0016 1809 5
:043 CDS0016 1810 5         IF .BFRD [BFRD$L_LBN] EQL .LBN
:044 CDS0016 1811 5         AND .BFRD [BFRD$L_UCB] EQL .CURRENT_UCB
:045 CDS0016 1812 5         THEN
:046 CDS0016 1813 5             EXITLOOP
:047 CDS0016 1814 5         ELSE
:048 CDS0016 1815 5             INDX = .BFRD [BFRD$W_NXTBFRD];
:049 CDS0016 1816 4         END;
:050 CDS0016 1817 3     END;
:051 CDS0016 1818 3
: 788      1819 3 IF .INDX NEQ 0
: 789      1820 3 THEN
```

```

790 1821 3
791 1822 3
792 1823 3
793 1824 3
794 1825 3
795 1826 3
796 1827 3 GOT_ONE:
797 1828 4 -BEGIN
798 1829 4
799 1830 4 IF .BFRD [BFRD$W_CURPID] NEQ 0
800 1831 4 THEN
801 1832 4 IF .BFRD [BFRD$W_CURPID] EQL .PIDINDX
802 1833 4 THEN
803 1834 4
804 1835 4 ! This is a buffer we've already put onto our in-process queue.
805 1836 4 ! Move to the head of the LRU list and return.
806 1837 4
807 1838 4
808 1839 5 BEGIN
809 1840 5 REMQUE (.BFRD, BFRD);
810 1841 5 INSQUE (.BFRD, .BFR_LIST [.POOL, QBLNK]);
811 1842 5 RELEASE_CACHE ();
812 1843 5 RETURN .INDX - 1
813 1844 5 END
814 1845 5
815 1846 4 ELSE
001 CDS0016 1847 5 BEGIN
002 CDS0016 1848 5
003 CDS0016 1849 5 ! Set flag to cause us to go around this block again.
004 CDS0016 1850 5 ! Note that the resolve_ambiguity routine will release the cache serialization
005 CDS0016 1851 5 ! lock while we are stalled. It is reacquired when we are awakened.
006 CDS0016 1852 5
007 CDS0016 1853 5
008 CDS0016 1854 5 LOOKUP_AGAIN = 1;
009 CDS0016 1855 5 RESOLVE_AMBIGUITY ();
010 CDS0016 1856 5 LEAVE_LOOKUP;
011 CDS0016 1857 4 END;
818-2 1858 4
819 1859 4
820 1860 4 ! Verify lock basis and that things are in the right pool.
821 1861 4
822 1862 4 ! Blocks in the data block pool may legitimately have the wrong
823 1863 4 ! lockbasis as a result of having been deallocated to free storage
824 1864 4 ! and reallocated to another file.
825 1865 4
826 1866 4 ! File headers may legitimately have the wrong lockbasis if they
827 1867 4 ! have been deleted and are being treated as primary headers when
828 1868 4 ! they used to be extension headers or vice-versa. This will
829 1869 4 ! force a read from disk. Further checks will be made in read_header
830 1870 4 ! against the actual header to see if things really make sense.
831 1871 4
832 1872 4 ! Blocks can also cross pools (rare in practice) if directory or quota
833 1873 4 ! file data blocks, for example, are deallocated to free storage, and
834 1874 4 ! the index file is then extended, causing them to become file headers.
835 1875 4
836 1876 4 ! In all cases, unhook and return the bfrd to destroy the association
837 1877 4 ! of this buffer with the lock basis and lock that is backing it.
```



```

: 838 1878 4 :
: 839 1879 4 :
: 840 1880 4 IF .BFRD [BFRD$L_LOCKBASIS] NEQ .LOCKBASIS
: 841 1881 4 OR .BFRD [BFRD$V_POOL] NEQ .POOL
: 842 1882 4 THEN
: 843 1883 4 IF .POOL EQL 1
: 844 1884 4 OR .TYPE EQL HEADER_TYPE
: 845 1885 4 OR .BFRD [BFRD$V_POOL] EQL 1
: 846 1886 4 THEN
: 847 1887 5 BEGIN
: 848 1888 5 UNHOOK_BFRD (
: 849 1889 5 ((.BFRD - .CACHE_HDR [F11BC$L_BFRDBAS])/BFRD$$_BFRDDEF) + 1,
: 850 1890 5 .BFRD);
: 851 1891 5 RETURN_BFRD (.BFRD);
: 852 1892 5 LEAVE GOT_ONE;
: 853 1893 5 END
: 854 1894 5 ELSE
: 855 1895 5 BUG_CHECK (XQPERR, 'invalid lock basis');
: 856 1896 5
: 857 1897 4 : This was not on our in-process queue. We'll need to account for the
: 858 1898 4 : fact that we are taking another buffer out of general circulation for
: 859 1899 4 : this operation.
: 860 1900 4 :
: 861 1901 4 :
: 862 1902 4 :
: 863 1903 4 :
: 864 1904 4 :
: 865 1905 4 IF .BFRS_USED [.POOL] EQL .BFR_CREDITS [.POOL]
: 866 1906 4 THEN
: 867 1907 5 BEGIN
: 868 1908 5 BIND POOLAVAIL = CACHE_HDR [F11BC$L_POOLAVAIL] + .POOL*4;
: 869 1909 5
: 001 :CDS0020 1910 5 IF .POOLAVAIL GTR 6
: 871-1 1911 5 THEN
: 872 1912 6 BEGIN
: 873 1913 6 POOLAVAIL = .POOLAVAIL - 1;
: 874 1914 6 BFR_CREDITS [.POOL] = .BFR_CREDITS [.POOL] + 1;
: 875 1915 6 END
: 876 1916 5 ELSE
: 001 :CDS0020 1917 5 : FREE_ONE returns true if the cache interlock was released (to write
: 002 :CDS0020 1918 5 : a dirty buffer). In that case, the bfrd we located may not be free,
: 003 :CDS0020 1919 5 : or even present, anymore. Loop back to look it up all over again.
: 004 :CDS0020 1920 5 :
: 005 :CDS0020 1921 5 :
: 006 :CDS0020 1922 5 :
: 007 :CDS0020 1923 5 IF FREE_ONE (.POOL)
: 008 :CDS0020 1924 5 THEN
: 009 :CDS0020 1925 6 BEGIN
: 010 :CDS0020 1926 6 LOOKUP_AGAIN = 1;
: 011 :CDS0020 1927 6 LEAVE [LOOKUP;
: 012 :CDS0020 1928 5 END;
: 013 :CDS0020 1929 4 END;
: 014 :CDS0020 1930 4 : Claim this bfrd for our process.
: 015 :CDS0020 1931 4 :
: 016 :CDS0020 1932 4 :
: 017 :CDS0020 1933 4 :
: 018 :CDS0020 1934 4 BFRD [BFRD$W_CURPID] = .PIDINDX;
```

```
:019 !CDS0020 1935 4
:020 !CDS0020 1936 4
:021 !CDS0020 1937 4
880-3 1938 4
881 1939 4
882 1940 4
883 1941 4
884 1942 4
885 1943 4
886 1944 4
887 1945 4
888 1946 4
889 1947 4
898-8 1948 4
899 1949 4
900 1950 4
901 1951 4
902 1952 4
903 1953 4
904 1954 4
905 1955 4
906 1956 4
907 1957 5
908 1958 5
909 1959 5
910 1960 5
911 1961 5
912 1962 5
913 1963 5
914 1964 5
915 1965 5
916 1966 4
917 1967 4
918 1968 4
919 1969 4
920 1970 4
921 1971 4
922 1972 4
923 1973 4
924 1974 4
925 1975 3
926 1976 3
:001 !CDSU020 1977 3
:002 !CDS0020 1978 2
927 1979 2
928 1980 2
929 1981 2
930 1982 2
931 1983 2
932 1984 2
933 1985 2
934 1986 2
935 1987 2
936 1988 3
937 1989 3
938 1990 3
:001 !CDS0020 1991 3

REMQUE (.BFRD, BFRD);
INSQUE (.BFRD, .BFR_LIST [.POOL, QBLNK]);
BFRS_USED [.POOL] = .BFRS_USED [.POOL] + 1;

IF .BFRD [BFRD$S_SEQNUM] NEQ .SEQNUM
THEN
    BFRD [BFRD$V_VALID] = 0;

! The sequence number will be stored in the BFRD when the buffer is
! released back to the cache, otherwise it would be stored here.

RELEASE_CACHE ();

! Count finding a valid buffer as a hit, an invalid one as a miss.
! We are deliberately not counting hits on buffers already on our
! in-process list to get a more meaningful hit ratio.

IF .BFRD [BFRD$V_VALID]
THEN
    BEGIN
        PMS_TOT_CACHE = .PMS_TOT_CACHE + 1;
        CASE .POOL FROM 0 TO 2 OF
        SET
        [0]: PMS$GL_STORAGMAP_HIT = .PMS$GL_STORAGMAP_HIT + 1;
        [1]: PMS$GL_DIRDATA_HIT = .PMS$GL_DIRDATA_HIT + 1;
        [2]: PMS$GL_FILHDR_HIT = .PMS$GL_FILHDR_HIT + 1;
        TES;
        END
    ELSE
        CASE .POOL FROM 0 TO 2 OF
        SET
        [0]: PMS$GL_STORAGMAP_MISS = .PMS$GL_STORAGMAP_MISS + 1;
        [1]: PMS$GL_DIRDATA_MISS = .PMS$GL_DIRDATA_MISS + 1;
        [2]: PMS$GL_FILHDR_MISS = .PMS$GL_FILHDR_MISS + 1;
        TES;

RETURN .INDX - 1;
END;
! of block GOT_ONE

END
! of block LOOKUP

WHILE .LOOKUP_AGAIN;

! Failed to find a buffer matching desired LBN and UCB in the cache.
! Account for us using another buffer from the cache.

FND CNT = 1;

IF .BFRS_USED [.POOL] EQL .BFR_CREDITS [.POOL]
THEN
    BEGIN
        BIND POOLAVAIL = CACHE_HDR [F11BC$S_POOLAVAIL] + .POOL*4;

IF .POOLAVAIL GTR 6
```

```

: 940-1      1992 3      THEN
: 941      1993 4      BEGIN
: 942      1994 4      POOLAVAIL = .POOLAVAIL - 1;
: 943      1995 4      BFR_CREDITS [.POOL] = .BFR_CREDITS [.POOL] + 1;
: 944      1996 4      END
: 945      1997 4      ELSE
:001 :CDS0020 1998 3      ! We don't need to check for potential cache interlock release here
:002 :CDS0020 1999 3      ! because we haven't got a bfrd in hand yet.
:003 :CDS0020 2000 3      !
:004 :CDS0020 2001 3      !
:005 :CDS0020 2002 3      !
: 946      2003 3      FREE_ONE (.POOL);
: 947      2004 2      END;
: 948      2005 2      POOL_LRU = CACHE_HDR [F11BC$Q_POOL_LRU] + .POOL*8;
: 949      2006 2      REMQUE (.POOL_LRU [QFLNK], BFRD);
: 950      2007 2      BFRS_USED [.POOL] = .BFRS_USED [.POOL] + 1;
: 951      2008 2      IF .COUNT GTRU 1
: 952      2009 2      AND .POOL EQL 1
:001 :CDS0019 2010 2      THEN
:002 :CDS0019 2011 2      BEGIN
: 953      2012 2      LABEL CHK_BFRDS;
: 954      2013 2      BIND POOLAVAIL = CACHE_HDR [F11BC$L_POOLAVAIL] + 4; ! Pool 1
: 955      2014 2      BIND POOLCNT = CACHE_HDR [F11BC$W_POOLCNT] : VECTOR [,WORD];
: 956      2015 2      LOCAL
: 957      2016 2      DOWN,
: 958      2017 2      TRY_COUNT,
: 959      2018 2      LO_BFRD,
: 960      2019 2      HI_BFRD,
: 961      2020 2      CUR_BFRD : REF BBLOCK;
: 962      2021 2      TRY_COUNT = .ACPSGB_MAXREAD;
: 963      2022 2      IF .COUNT LSSU .TRY_COUNT
: 964      2023 2      THEN
: 965      2024 2      TRY_COUNT = .COUNT;
: 966      2025 2      LO_BFRD = .CACHE_HDR [F11BC$L_BFRDBAS] + (.POOLCNT [0])*BFRD$$_BFRDDEF;
: 967      2026 2      HI_BFRD = .LO_BFRD + (.POOLCNT [1])*BFRD$$_BFRDDEF;
: 968      2027 2      DOWN = 0;
: 969      2028 2      CUR_BFRD = .BFRD;
: 970      2029 2      WHILE .FNDCNT LSSU .TRY_COUNT
: 971      2030 2      DO
: 972      2031 2      CHK_BFRDS:
: 973      2032 2      BEGIN
: 974      2033 2      IF NOT .DOWN
: 975      2034 2      THEN
: 976      2035 2      BEGIN
: 977      2036 2
: 978      2037 2
: 979      2038 2
: 980      2039 2
: 981      2040 2
: 982      2041 2
: 983      2042 2
: 984      2043 2
: 985      2044 2
: 986      2045 2
: 987      2046 2
: 988      2047 2
: 989      2048 2
```

```

: 990      2049  5      CUR_BFRD = .CUR_BFRD + BFRD$$_BFRDDEF;
: 991      2050  5
: 992      2051  5      IF .CUR_BFRD GEQA .HI_BFRD
: 993      2052  5      THEN
: 994      2053  6          BEGIN
: 995      2054  6              CUR_BFRD = .BFRD;
: 996      2055  6              DOWN = 1;
: 997      2056  6              LEAVE CHK_BFRDS;
: 998      2057  5              END;
: 999      2058  5
: 1000     2059  5      END
: 1001     2060  4      ELSE
: 1002     2061  5          BEGIN
: 1003     2062  5
: 1004     2063  5              CUR_BFRD = .CUR_BFRD - BFRD$$_BFRDDEF;
: 1005     2064  5
: 1006     2065  5              IF .CUR_BFRD LSSA .LO_BFRD
: 1007     2066  5              THEN
: 1008     2067  5                  EXITLOOP;
: 1009     2068  5
: 1010     2069  4              END;
: 1011     2070  4
: 1012     2071  4              IF .CUR_BFRD [BFRD$$_CURPID] NEQ 0
: 1013     2072  4              THEN
: 1014     2073  4                  EXITLOOP;
: 1015     2074  4
: 1016     2075  4      ! NOTE: FNDCNT has not been bumped yet.
: 1017     2076  4
: 1018     2077  4              INDX = .LOOKUP_LBN ((.LBN + .FNDCNT))<0,16>;
: 1019     2078  4
: 1020     2079  4              WHILE .INDX NEQ 0
: 1021     2080  4              DO
: 1022     2081  5                  BEGIN
: 1023     2082  5
: 1024     2083  5                      LOCAL TMPBFRD : REF BBLOCK;
: 1025     2084  5                      TMPBFRD = BFRD_ADDR (.INDX);
: 1026     2085  5
: 1027     2086  5                      IF .TMPBFRD [BFRD$$L_LBN] EQL (.LBN + .FNDCNT)
: 1028     2087  6                          AND .TMPBFRD [BFRD$$L_UCB] EQL .CURRENT_UCB
: 1029     2088  5                          THEN
: 1030     2089  5                              EXITLOOP
: 1031     2090  5                          ELSE
: 1032     2091  5                              INDX = .TMPBFRD [BFRD$$_NXTBFRD];
: 1033     2092  5                          END;
: 1034     2093  4
: 1035     2094  4
: 1036     2095  4              IF .INDX NEQ 0
: 1037     2096  4              THEN
: 1038     2097  4                  EXITLOOP;
: 1039     2098  4
: 001 :CDS0019 2099  4      IF .BFRS_USED [.POOL] EQL .BFR_CREDITS [.POOL]
: 002 :CDS0019 2100  4      THEN
: 003 :CDS0019 2101  4          IF .POOLAVAIL GTR 4
: 004 :CDS0019 2102  4          THEN
: 005 :CDS0019 2103  5              BEGIN
: 006 :CDS0019 2104  5                  POOLAVAIL = .POOLAVAIL - 1;
: 007 :CDS0019 2105  5                  BFR_CREDITS [1] = .BFR_CREDITS [1] + 1;
```

```
:008 :CDS0019 2106 5
:009 :CDS0019 2107 4
:010 :CDS0019 2108 4
:011 :CDS0019 2109 4
:012 :CDS0019 2110 4
:013 :CDS0019 2111 4 ! This one is ok. Yank off LRU and count it.
:014 :CDS0019 2112 4 !
:015 :CDS0019 2113 4
:016 :CDS0019 2114 4
1052 -12 2115 4 BFRS USED [.POOL] = .BFRS USED [.POOL] + 1;
1053 2116 4 REMOVE (.CUR_BFRD, CUR_BFRD);
1054 2117 4 FNDCNT = .FNDCNT + 1;
1055 2118 4 IF .CUR_BFRD LSSA .BFRD
1056 2119 4 THEN
1057 2120 4 BFRD = .CUR_BFRD;
1058 2121 4
1059 2122 4 END; ! of block CHK_BFRDS
1060 2123 3
1061 2124 3 .FOUND_COUNT = .FNDCNT;
1062 2125 3
1063 2126 3 ! Set BFRD to the highest one in the range found.
1064 2127 3 !
1065 2128 3
1066 2129 3 BFRD = .BFRD + (.FNDCNT - 1)*BFRD$$_BFRDDEF;
1067 2130 3
1068 2131 2 END; ! of consider multi-block read
1069 2132 2
1072 -2 2133 2 DECR I FROM (.FNDCNT - 1) TO 0
1073 2134 2 DO
1074 2135 3 BEGIN
1075 2136 3
1076 2137 3 INDX = ((.BFRD - .CACHE_HDR [F11BC$L_BFRDBAS])/BFRD$$_BFRDDEF) + 1;
1077 2138 3
1078 2139 3 UNHOOK_BFRD (.INDX, .BFRD);
1079 2140 3
1080 2141 3 ! Insert the new buffer into the hash list using the new LBN.
1081 2142 3 !
1082 2143 3
1083 2144 3 INDX_ADDR = LOOKUP_LBN ((.LBN + .I));
1084 2145 3 BFRD [BFRD$$_NXTBFRD] = (.INDX_ADDR)<0,16>;
1085 2146 3 (.INDX_ADDR)<0,16> = .INDX;
1086 2147 3
1087 2148 3 ! Fill in our new or recycled BFRD, as the case may be, and
1088 2149 3 ! insert it onto the appropriate in-process queue.
1089 2150 3 !
1090 2151 3
1091 2152 3 BFRD [BFRD$$_LBN] = .LBN + .I;
1092 2153 3 BFRD [BFRD$$_UCB] = .CURRENT_UCB;
1093 2154 3 BFRD [BFRD$$_CURPID] = .PIDINDX;
1094 2155 3
1095 2156 3 BFRD [BFRD$$_LOCKBASIS] = .LOCKBASIS;
1096 2157 3 BFRD [BFRD$$_SEQNUM] = .SEQNUM;
1097 2158 3 BFRD [BFRD$$_BTYPE] = .TYPE;
1098 2159 3
1099 2160 3 INSQUE (.BFRD, .BFR_LIST [.POOL, QBLNK]);
1100 2161 3
1101 2162 3 BFRD = .BFRD - BFRD$$_BFRDDEF;
```


				0000*	00059				.WORD	<BUG\$_XQPERR!4>		
		50		63	D0	0005B	3\$:		MOVL	(R3), -R0	1671	
				0B	11	0005E			BRB	6\$		
				65	D5	00060	4\$:		TSTL	(R5)	1676	
				04	12	00062			BNEQ	5\$		
				FEFF		00064			BUGW		1678	
				0000*	00066				.WORD	<BUG\$_XQPERR!4>		
		50		65	D0	00068	5\$:		MOVL	(R5), -R0	1680	
1C		AE		6640	D0	0006B	6\$:		MOVL	(R6)[R0], LOCKBASIS		
				0A	11	00070			BRB	8\$	1662	
1C		AE		66	D0	00072	7\$:		MOVL	(R6), LOCKBASIS	1684	
				04	12	00076			BNEQ	8\$		
				FEFF		00078			BUGW		1686	
				0000*	0007A				.WORD	<BUG\$_XQPERR!4>		
		00		08	AC	8F	8\$:		CASEB	TYPE, #0, #5	1700	
005C		05		000C		00081	9\$:		.WORD	10\$-9\$,-		
		0028		0039		00089				17\$-9\$,-		
										12\$-9\$,-		
										19\$-9\$,-		
										14\$-9\$,-		
										21\$-9\$		
		50		0094	63	D0	0008D	10\$:	MOVL	(R3), R0	1704	
					CA40	D5	00090		TSTL	148(BASE)[R0]		
					06	12	00095		BNEQ	11\$		
		0094	CA40		01	D0	00097		MOVL	#1, 148(BASE)[R0]	1706	
			50		63	D0	0009D	11\$:	MOVL	(R3), R0	1708	
		14	AE	0094	CA40	D0	000A0		MOVL	148(BASE)[R0], SEQNUM		
					57	11	000A7		BRB	22\$	1700	
		50			65	D0	000A9	12\$:	MOVL	(R5), R0	1713	
					6240	D5	000AC		TSTL	(R2)[R0]		
					04	12	000AF		BNEQ	13\$		
		6240			01	D0	000B1		MOVL	#1, (R2)[R0]	1715	
		50			65	D0	000B5	13\$:	MOVL	(R5), R0	1717	
					0F	11	000B8		BRB	16\$		
		50			63	D0	000BA	14\$:	MOVL	(R3), R0	1722	
					6240	D5	000BD		TSTL	(R2)[R0]		
					04	12	000C0		BNEQ	15\$		
		6240			01	D0	000C2		MOVL	#1, (R2)[R0]	1724	
		50			63	D0	000C6	15\$:	MOVL	(R3), R0	1726	
		14	AE		6240	D0	000C9	16\$:	MOVL	(R2)[R0], SEQNUM		
					30	11	000CE		BRB	22\$	1700	
					62	B5	000D0	17\$:	TSTW	(R2)	1737	
					03	12	000D2		BNEQ	18\$		
		62			01	B0	000D4		MOVW	#1, (R2)	1739	
		14	AE		62	3C	000D7	18\$:	MOVZWL	(R2), SEQNUM	1741	
					23	11	000DB		BRB	22\$	1700	
					02	A2	B5	000DD	19\$:	TSTW	2(R2)	1746
					04	12	000E0		BNEQ	20\$		
		02	A2		01	B0	000E2		MOVW	#1, 2(R2)	1748	
		14	AE		02	A2	3C	000E6	20\$:	MOVZWL	2(R2), SEQNUM	1750
					13	11	000EB		BRB	22\$	1700	
14	AE	A4	AA		01	EF	000ED	21\$:	EXTZV	#1, #15, -92(BASE), SEQNUM	1755	
					0A	12	000F4		BNEQ	22\$	1757	
					14	AE	D6	000F6		INCL	SEQNUM	1760
A4	AA		OF		14	AE	F0	000F9		INSV	SEQNUM, #1, #15, -92(BASE)	1761
					FEAD	30	00100	22\$:	BSBW	SERIAL_CACHE	1776	
					52	D4	00103	23\$:	CLRL	LOOKUP_AGAIN	1782	

			0000*	001BE		.WORD	<BUG\$ XQPERR!4>	
	6744		6844	B1 001C0	35\$:	CMPW	(R11)[POOL], (R7)[POOL]	1905
				24 12 001C5		BNEQ	38\$	
	50		00 8844	DE 001C7		MOVAL	@0(R8)[POOL], R0	1908
	50		68	A0 9E 001CC		MOVAB	104(R0), R0	
	06			60 D1 001D0		CMP	(R0), #6	1910
				07 15 001D3		BLEQ	36\$	
				60 D7 001D5		DECL	(R0)	1913
				6744 B6 001D7		INCW	(R7)[POOL]	1914
				0F 11 001DA		BRB	38\$	1910
				54 DD 001DC	36\$:	PUSHL	POOL	1923
0000V	CF			01 FB 001DE		CALLS	#1, FREE_ONE	
	05			50 E9 001E3		BLBC	R0, 38\$	
	52			01 D0 001E6		MOVL	#1, LOOKUP_AGAIN	1926
				5A 11 001E9	37\$:	BRB	47\$	1927
1C	A3			6E B0 001EB	38\$:	MOVW	PIDINDX, 28(BFRD)	1934
	53			63 OF 001EF		REMQUE	(BFRD), BFRD	1936
	50		D0	AA44 7E 001F2		MOVAQ	-48(BASE)[POOL], R0	1937
	00			63 OE 001F7		INSQUE	(BFRD), @0(R0)	
	14			6844 B6 001FB		INCW	(R11)[POOL]	1938
	AE		14	A3 D1 001FE		CMP	20(BFRD), SEQNUM	1940
				04 13 00203		BEQL	39\$	
	18			08 8A 00205		BICB2	#8, 24(BFRD)	1942
				0000V 30 00209	39\$:	BSBW	RELEASE CACHE	1948
27	18			03 E1 0020C		BBC	#3, 24(BFRD), 45\$	1955
			08F0	CA D6 00211		INCL	2288(BASE)	1958
02	00			54 CF 00215		CASEL	POOL, #0, #2	1959
0016	000E			0006 00219	40\$:	.WORD	41\$-40\$,-	
							42\$-40\$,-	
							43\$-40\$	
			00000000G	9F D6 0021F	41\$:	INCL	@#PMSSGL_STORAGMAP_HIT	1961
				0E 11 00225		BRB	44\$	
			00000000G	9F D6 00227	42\$:	INCL	@#PMSSGL_DIRDATA_HIT	1962
				06 11 0022D		BRB	44\$	
			00000000G	9F D6 0022F	43\$:	INCL	@#PMSSGL_FILHDR_HIT	1963
				01D8 31 00235	44\$:	BRW	72\$	1955
02	00			54 CF 00238	45\$:	CASEL	POOL, #0, #2	1967
01CE	01C6			01BE 0023C	46\$:	.WORD	69\$-46\$,-	
							70\$-46\$,-	
							71\$-46\$	
				01B5 31 00242		BRW	69\$	1969
	03			52 E9 00245	47\$:	BLBC	LOOKUP_AGAIN, 48\$	1978
				FEB8 31 00248		BRW	23\$	
	52			01 D0 0024B	48\$:	MOVL	#1, FNDCNT	1984
6744				6844 B1 0024E		CMPW	(R11)[POOL], (R7)[POOL]	1984
				1C 12 00253		BNEQ	50\$	
	50		00 8844	DE 00255		MOVAL	@0(R8)[POOL], R0	1989
	50		68	A0 9E 0025A		MOVAB	104(R0), R0	
	06			60 D1 0025E		CMP	(R0), #6	1991
				07 15 00261		BLEQ	49\$	
				60 D7 00263		DECL	(R0)	1994
				6744 B6 00265		INCW	(R7)[POOL]	1995
				07 11 00268		BRB	50\$	1991
				54 DD 0026A	49\$:	PUSHL	POOL	2003
0000V	CF			01 FB 0026C		CALLS	#1, FREE_ONE	
	50		00 8844	7E 00271	50\$:	MOVAQ	@0(R8)[POOL], POOL_LRU	2006
	50			28 C0 00276		ADDL2	#40, POOL_LRU	

7E
51

59
5'

50

53	00	B0	0F	00279	REMQUE	@0(P00L LRU), BFRD	2008
		6B44	B6	0027D	INCW	(R11)[P00L]	2010
01	0C	AC	D1	00280	CMPL	COUNT, #1	2012
		03	1A	00284	BGTRU	52\$	
		0161	3i	00286	BRW	67\$	
01		54	D1	00289	CMPL	POOL, #1	2013
		FB	12	0028C	BNEQ	51\$	
68	0000006C	8F	C1	0028E	ADDL3	#108, (R8), R9	2018
68	00000078	8F	C1	00296	ADDL3	#120, (R8), R1	2019
0C	AE	00000000G	9F	9A	MOVZBL	@#ACP\$GB MAXREAD, TRY_COUNT	2028
0C	AE	0C	AC	D1	CMPL	COUNT, TRY_COUNT	2030
			05	1E	BGEQU	53\$	
0C	AE	0C	AC	D0	MOVL	COUNT, TRY_COUNT	2032
50			68	D0	MOVL	(R8), R0	2034
55			61	3C	MOVZWL	(R1), R5	
55			20	C4	MULL2	#32, R5	
06	AE	18	B045	9E	MOVAB	@24(R0)LR5], LO_BFRD	
51		02	A1	3C	MOVZWL	2(R1), R1	2035
51			20	C4	MULL2	#32, R1	
18	AE	08	BE41	9E	MOVAB	@LO BFRD[R1], HI_BFRD	
		10	AE	D4	CLRL	DOWN	2037
55			53	D0	MOVL	BFRD, CUR_BFRD	2038
0C	AE		52	D1	CMPL	FNDCNT, TRY_COUNT	2040
			03	1F	BLSSU	55\$	
			0093	31	BRW	64\$	
12		10	AE	E8	BLBS	DOWN, 57\$	2045
55			20	C0	ADDL2	#32, CUR_BFRD	2049
18	AE		55	D1	CMPL	CUR_BFRD, HI_BFRD	2051
			12	1F	RLSSU	58\$	
55			53	D0	MOVL	BFRD, CUR_BFRD	2054
10	AE		01	D0	MOVL	#1, DOWN	2055
			E1	11	BRB	54\$	2056
55			20	C2	SUBL2	#32, CUR_BFRD	2063
08	AE		55	D1	CMPL	CUR_BFRD, LO_BFRD	2065
			74	1F	BLSSU	64\$	
			1C	A5	TSTW	28(CUR_BFRD)	2071
			6F	12	BNEQ	64\$	
50			68	DC	MOVL	(R8), R0	2077
04	AE	04	BC42	9E	MOVAB	@LBN[FNDCNT], 4(SP)	
51		14	A0	3C	MOVZWL	20(R0), R1	
04	AE		01	7A	EMUL	#1, 4(SP), #0, -(SP)	
8E			51	7B	EDIV	R1, (SP)+, R1, R1	
			51	D5	TSTL	R1	
			03	18	BGEQ	59\$	
51			51	CE	MNEGL	R1, R1	
56		10	B041	3C	MOVZWL	@16(R0)[R1], INDX	
			22	13	BEQL	62\$	2079
51			68	D0	MOVL	(R8), R1	2085
56			05	78	ASHL	#5, INDX, RC	
50		18	A1	C0	ADCL2	24(R1), R0	
50			20	C2	SUBL2	#32, TMPBFRD	
04	AE	08	A0	D1	CMPL	8(TMPBFRD), 4(SP)	2087
			07	12	BNEQ	61\$	
94	AA	0C	A0	D1	CMPL	12(TMPBFRD), -108(BASE)	2088
			06	13	BEQL	62\$	
56		1E	A0	3C	MOVZWL	30(TMPBFRD), INDX	2092
			DC	11	BRB	60\$	2079

		56	D5	00349	62\$:	TSTL	INDX	2095	
		23	12	0034B		BNEQ	64\$		
	6744	6B44	B1	0034D		CMPW	(R11)[POOL], (R7)[POOL]	2099	
		0A	12	00352		BNEQ	63\$		
	04	69	D1	00354		CMP	(R9), #4	2101	
		17	15	00357		BLEQ	64\$		
		69	D7	00359		DECL	(R9)	2104	
		A7	B6	0035B		INCW	2(R7)	2105	
		6B44	B6	0035E	63\$:	INCW	(R11)[POOL]	2114	
	55	65	OF	00361		REMQUE	(CUR_BFRD), CUR_BFRD	2115	
		52	D6	00364		INCL	FNDCNT	2116	
	53	55	D1	00366		CMP	CUR_BFRD, BFRD	2118	
		86	1E	00369		BGEQU	56\$		
	53	55	D0	0036B		MOVL	CUR_BFRD, BFRD	2120	
		E1	11	0036E		BRB	56\$	2040	
	50	10	BC	52	D0	00370	64\$:	2124	
		52		05	78	00374		2129	
		53	E0	A043	9E	00378			
				6B	11	0037D		2133	
		50		68	D0	0037F	65\$:	2137	
	50	53	18	A0	C3	00382			
		50		20	C6	00387			
		56	01	A0	9E	0038A			
				53	DD	0038E		2139	
				56	DD	00390			
		0000V	CF	02	FB	00392			
				68	D0	00397		2144	
	50	52	04	AC	C1	0039A			
		55	14	A1	3C	0039F			
	7E	00		01	7A	003A3			
	50			55	7B	003A8			
				50	D5	003AD			
				03	18	003AF			
		50		50	CE	003B1			
		57	10	B140	3E	003B4	66\$:		
	1E	A3		67	B0	003B9		2145	
		67		56	B0	003BD		2146	
	08	A3	04	BC42	9E	003C0		2152	
	0C	A3	94	AA	D0	003C6		2153	
	1C	A3		6E	B0	003CB		2154	
	10	A3	1C	AE	D0	003CF		2156	
	14	A3	14	AE	D0	003D4		2157	
	19	A3	08	AC	90	003D9		2158	
		50	D0	AA44	7E	003DE		2160	
	00	B0		63	0E	003E3			
		53		20	C2	003E7		2162	
		92		52	F4	003EA	67\$:	2133	
				00G0V	30	003ED		2166	
	02	00		54	CF	003F0		2171	
0016	000E	0006		003F4	68\$:	.WORD	69\$-68\$,- 70\$-68\$,- 71\$-68\$		
		00000000G	9F	D6	003FA	69\$:	INCL	@#PMSSGL_STORAGMAP_MISS	2173
			0E	11	00400		BRB	72\$	
		00000000G	9F	D6	00402	70\$:	INCL	@#PMSSGL_DIRDATA_MISS	2174
			06	11	00408		BRB	72\$	
		00000000G	9F	D6	0040A	71\$:	INCL	@#PMSSGL_FILHDR_MISS	2175


```

: 1121 2181 1 ROUTINE WAKE_WAITER (AOB) : NOVALUE =
: 1122 2182 1
: 1123 2183 1 |++
: 1124 2184 1 |
: 1125 2185 1 | FUNCTIONAL DESCRIPTION:
: 1126 2186 1 |
: 1127 2187 1 | Remove ourself from cache processing queue. Wake up
: 1128 2188 1 | the next process if queue is not empty.
: 1129 2189 1 |
: 1130 2190 1 | --
: 1131 2191 1 |
: 1132 2192 2 BEGIN
: 1133 2193 2
: 1134 2194 2 MAP
: 1135 2195 2 AOB : REF BBLOCK;
: 1136 2196 2
: 1137 2197 2 LINKAGE
: 1138 2198 2 L_SCH$QAST = JSB : GLOBAL (PINCL=2, ACB=5) NORESERVE (3,4)
: 1139 2199 2 NOTUSED (6,7,8,9,10,11);
: 1140 2200 2
: 1141 2201 2 GLOBAL REGISTER
: 1142 2202 2 PINCL = 2,
: 1143 2203 2 ACB = 5;
: 1144 2204 2
: 1145 2205 2 EXTERNAL ROUTINE
: 1146 2206 2 SCH$QAST : L_SCH$QAST ADDRESSING_MODE (ABSOLUTE);
: 1147 2207 2
: 1148 2208 2 ACB = .AOB [AOB$L ACPOFL] + IRP$C_CDRP; ! first waiter's ACB
: 1149 2209 2 PINCL = PRIS$RESAVL;
: 1150 2210 2 IF NOT SCH$QAST ()
: 1151 2211 2 THEN
: 1152 2212 2 BUG_CHECK (XQPERR, 'Failed to queue ast');
: 1153 2213 2
: 1154 2214 1 END;

```

```

                                .EXTRN SCH$QAST
                                003C 0000 WAKE_WAITER:
                                .WORD Save R2,R3,R4,R5
55      04 BC 00000060 8F C1 00002 ADDL3 #96, @AOB, ACB
                                52 02 D0 0000B MOVL #2, PINCL
                                04 000000C0G 9F 16 0000E JSB @#SCH$QAST
                                50 EB 00014 BLBS R0, 1$
                                FEF 00017 BUGW
                                00C0* 00019 .WORD <BUG$_XQPERR!4>
                                04 0001B 1$: RET
: 2181
: 2208
: 2209
: 2210
: 2212
: 2214

```

: Routine Size: 28 bytes, Routine Base: \$CODE\$ + 06FB


```
:001 :CDS0016 2244 1 ROUTINE RESOLVE_AMBIGUITY : L_NORM NOVALUE =
:002 :CDS0016 2245 1
:003 :CDS0016 2246 1 :++
:004 :CDS0016 2247 1
:005 :CDS0016 2248 1 : FUNCTIONAL DESCRIPTION:
:006 :CDS0016 2249 1
:007 :CDS0016 2250 1 : This routine stalls this thread on the ambiguity queue. It will
:008 :CDS0016 2251 1 : be awakened when the next thread completes. That does not mean
:009 :CDS0016 2252 1 : the ambiguity is resolved yet, but that it should be checked.
:1196-10 2253 1
:1197 2254 1 :--
:1198 2255 1
:1199 2256 2 BEGIN
:1200 2257 2
:1201 2258 2 BIND_COMMON;
:1202 2259 2
:1203 2260 2 LOCAL
:001 :CDS0016 2261 2 AQB : REF BBLOCK;
:002 :CDS0016 2262 2
:003 :CDS0016 2263 2 AQB = .CURRENT_VCB [VCBSL_AQB];
:004 :CDS0016 2264 2
:005 :CDS0016 2265 2 ! The cache serialization interlock is held coming into this routine.
:006 :CDS0016 2266 2
:007 :CDS0016 2267 2
:008 :CDS0016 2268 2 ! Insert the acb onto the head of cache serialization queue. This
:009 :CDS0016 2269 2 ! allows us to remove the irp from the head of the queue (where it is
:010 :CDS0016 2270 2 ! now) without releasing ownership of that queue yet.
:011 :CDS0016 2271 2
:012 :CDS0016 2272 2
:013 :CDS0016 2273 2 INSQUE (.ACB_ADDR, AQB [AQB$ACPOFL]);
:014 :CDS0016 2274 2
:015 :CDS0016 2275 2 REMQUE (.IO_PACKET, IO_PACKET);
:016 :CDS0016 2276 2
:017 :CDS0016 2277 2 ! The ambiguity queue is not initialized by MOUNT (setup_cache) as
:018 :CDS0016 2278 2 ! others are, so check for that now.
:019 :CDS0016 2279 2
:020 :CDS0016 2280 2
:021 :CDS0016 2281 2 IF .CACHE_HDR [F11BC$AMBIGQFL] EQL 0
:022 :CDS0016 2282 2 THEN
:023 :CDS0016 2283 3 BEGIN
:024 :CDS0016 2284 3 CACHE_HDR [F11BC$AMBIGQFL] = CACHE_HDR [F11BC$AMBIGQFL];
:025 :CDS0016 2285 3 CACHE_HDR [F11BC$AMBIGQBL] = CACHE_HDR [F11BC$AMBIGQFL];
:026 :CDS0016 2286 2 END;
:027 :CDS0016 2287 2
:028 :CDS0016 2288 2 ! Put us on the ambiguity queue.
:029 :CDS0016 2289 2
:030 :CDS0016 2290 2
:031 :CDS0016 2291 2 INSQUE (.IO_PACKET, .CACHE_HDR [F11BC$AMBIGQBL]);
:032 :CDS0016 2292 2
:033 :CDS0016 2293 2 ! Releasing the cache will remove our temp entry (the acb), leaving us
:034 :CDS0016 2294 2 ! on the ambiguity queue with our irp, while we await awakening.
:035 :CDS0016 2295 2
:036 :CDS0016 2296 2
:037 :CDS0016 2297 2 RELEASE_CACHE ();
:038 :CDS0016 2298 2
:039 :CDS0016 2299 2 WAIT_FOR_AST ();
:040 :CDS0016 2300 2
```

:041 !CDS0016 2301 1 END;

! of routine resolve_ambiguity

				000C 00000 RESOLVE_AMBIGUITY:				
	50	98	AA	D0	00002	.WORD	Save R2,R3	: 2244
	50	10	A0	D0	00006	MOVL	-104(BASE), R0	: 2263
	60	C8	BA	0E	0000A	MOVL	16(R0), AQB	
90	AA	90	BA	0F	0000E	INSQUE	@-56(BASE), (AQB)	: 2273
	50	FC	AA	D0	00013	REMQUE	@-112(BASE), -112(BASE)	: 2275
	50	0080	CO	9E	00017	MOVL	-4(BASE), R0	: 2281
			60	D5	0001C	MOVAB	128(R0), R0	
			0E	12	0001E	TSTL	(R0)	
	60		50	D0	00020	BNEQ	1\$: 2284
	50	FC	AA	D0	00023	MOVL	R0, (R0)	: 2285
0084	CO	0080	CO	9E	00027	MOVL	-4(BASE), R0	
	50	FC	AA	D0	0002E	MOVAB	128(R0), 132(R0)	: 2291
0084	D0	90	BA	0E	00032	MOVL	-4(BASE), R0	
			A6	10	00038	INSQUE	@-112(BASE), @132(R0)	: 2297
0000G	CF		00	FB	0003A	BSBB	RELEASE CACHE	: 2299
			04	0003F		CALLS	#0, WAIT_FOR_AST	: 2301
						RET		

; Routine Size. 64 bytes. Routine Base: \$CODE\$ + 0727


```
:043 :CDS0016 2302 1 GLOBAL ROUTINE GET_REQD_BFR_CREDITS : L_NORM NOVALUE =
:044 :CDS0016 2303 1
:045 :CDS0016 2304 1 :++
:046 :CDS0016 2305 1
:047 :CDS0016 2306 1 : FUNCTIONAL DESCRIPTION:
:048 :CDS0016 2307 1
:049 :CDS0016 2308 1 : Acquire minimum buffer credits so that this operation may proceed.
:050 :CDS0016 2309 1 : Wait for their availability if necessary.
:051 :CDS0016 2310 1 : Initialize the global cell CACHE_HDR, primarily used by the buffer
:052 :CDS0016 2311 1 : management routines.
:053 :CDS0016 2312 1
:054 :CDS0016 2313 1 :--
:055 :CDS0016 2314 1
:056 :CDS0016 2315 2 BEGIN
:057 :CDS0016 2316 2
:058 :CDS0016 2317 2 BIND_COMMON:
:059 :CDS0016 2318 2
:060 :CDS0016 2319 2 LOCAL
1204 : 2320 2 AQB : REF BBLOCK,
1205 : 2321 2 REQD;
1206 : 2322 2
1207 : 2323 2 AQB = .CURRENT_VCB [VCB$AQB];
1208 : 2324 2
1209 : 2325 2 CACHE_HDR = .AQB [AQB$BUFCACHE];
1210 : 2326 2
1211 : 2327 2 SERIAL_CACHE ();
1212 : 2328 2
1213 : 2329 3 BEGIN
1214 : 2330 3 BIND
1215 : 2331 3 POOLAVAIL = CACHE_HDR [F11BC$POOLAVAIL] : VECTOR,
1216 : 2332 3 POOL_WAITQ = CACHE_HDR [F11BC$POOL_WAITQ] : BLOCKVECTOR [8, BYTE];
1217 : 2333 3
1218 : 2334 3 DECR POOL FROM 3 TO 0
1219 : 2335 3 DO
1220 : 2336 4 BEGIN
1221 : 2337 4
1222 : 2338 4 ! The minimal buffer requirements are:
1223 : 2339 4 ! 1 for storage bitmap blocks (pool 0)
1224 : 2340 4 ! 2 for directory data blocks (pool 1)
1225 : 2341 4 ! 3 for file headers (pool 2)
1226 : 2342 4 ! 1 for directory index (pool 3)
1227 : 2343 4
1228 : 2344 4
1229 : 2345 4
1230 : 2346 4 REQD = .POOL + 1;
1231 : 2347 4
1232 : 2348 4 IF .POOL EQL 3
1233 : 2349 4 THEN
1234 : 2350 4 REQD = 1;
1235 : 2351 4
1236 : 2352 4 IF (POOLAVAIL [.POOL] = .POOLAVAIL [.POOL] - .REQD) LSS 0
1237 : 2353 4 THEN
1238 : 2354 5 BEGIN
1239 : 2355 5
1240 : 2356 5 ! Insert our ACB temporarily at the head of the queue. Then when
1241 : 2357 5 ! we pull our irp off of it to put on the wait queue, the cache interlock
: 1242 : 2358 5 ! queue will not be left empty until we do our release_cache and
```

```

: 1243      2359 5  ! at that point pull off our temp acb entry.
: 1244      2360 5  !
: 1245      2361 5
: 1246      2362 5      INSQUE (.ACB_ADDR, AQB [AQB$SL_ACPQFL]);
: 1247      2363 5      REMQUE (.IO_PACKET, IO_PACKET); ! take us out of cache queue
: 1248      2364 5      INSQUE (.IO_PACKET, .POOL_WAITQ [.POOL, QBLNK]); ! into pool queue
: 1249      2365 5      RELEASE_CACHE ();
: 1250      2366 5      WAIT_FOR_AST ();
: 1251      2367 5
: 1252      2368 5      IF .AQB [AQB$SL_ACPQFL] NEQ .IO_PACKET
: 1253      2369 5      THEN
: 1254      2370 5          BUG_CHECK (XQPERR, 'messed up cache interlock queue');
: 1255      2371 5
: 1256      2372 4      END:
: 1257      2373 4
: 1258      2374 4      BFR_CREDITS [.POOL] = .REQD;
: 1259      2375 4
: 1260      2376 3      END:
: 1261      2377 2      END:          ! of BIND
: 1262      2378 2
: 1263      2379 2      RELEASE_CACHE ();
: 1264      2380 2
: 1265      2381 1      END:

```

				007C 0000U	.ENTRY	GET REQD BFR CREDITS, Save R2,R3,R4,R5,R6	: 2302
	50	98	AA	D0 00002	MOVL	-10Z(BASE), R0	: 2323
	54	10	A0	D0 00006	MOVL	16(R0), AQB	
	FC	AA	A4	D0 0000A	MOVL	24(AQB), -4(BASE)	: 2325
			FB0D	30 0000F	BSBW	SERIAL_CACHE	: 2327
55	FC	AA	00000068	8F C1 00012	ADDL3	#104, -4(BASE), R5	: 2331
56	FC	AA	00000048	8F C1 0001B	ADDL3	#72, -4(BASE), R6	: 2332
	52		03	D0 00024	MOVL	#3, POOL	: 2334
	53	01	A2	9E 00027	MOVAB	1(R2), REQD	: 2346
	03		52	D1 0002B	CMPL	POOL, #3	: 2348
			03	12 0002E	BNEQ	2\$	
	53		01	D0 00030	MOVL	#1, REQD	: 2350
	6542		53	C2 00033	SUBL2	REQD, (R5)[POOL]	: 2352
			25	18 00037	BGEQ	3\$	
	64	C8	BA	0E 00039	INSQUE	@-56(BASE), (AQB)	: 2362
	90	AA	90	BA 0F 0003D	REMQUE	@-112(BASE), -112(BASE)	: 2363
	50	04	A642	7E 00042	MOVAQ	4(R6)[POOL], R0	: 2367
	00	B0	90	BA 0E 00047	INSQUE	@-112(BASE), @0(R0)	
			FF51	30 0004C	BSBW	RELEASE_CACHE	: 2365
0000G	CF		00	FB 0004F	CALLS	#0, WAIT_FOR_AST	: 2366
90	AA		64	D1 00054	CMPL	(AQB), -112(BASE)	: 2368
			04	13 00058	BEQL	3\$	
			FEFF	0005A	BUGW		: 2370
			0000*	0005C	.WORD	<BUG\$_XQPERR!4>	
EC	AA42		53	B0 0005E	MOVW	REQD, -20(BASE)[POOL]	: 2374
	C1		52	F4 00063	SOBGEQ	POOL, 1\$: 2334
			FF37	30 00066	BSBW	RELEASE_CACHE	: 2379
			04	00069	RET		: 2381


```
1267 2382 1 GLOBAL ROUTINE RETURN_CREDITS : L_NORM NOVALUE =
1268 2383 1
1269 2384 1 ++
1270 2385 1
1271 2386 1 FUNCTIONAL DESCRIPTION:
1272 2387 1
1273 2388 1 This routine returns the in-process buffer credits to the
1274 2389 1 buffer pool. It moves the next waiter in line to the head
1275 2390 1 of the cache processing queue, which will cause it to become
1276 2391 1 awakened when we release the cache inter.ock.
1277 2392 1
1278 2393 1 --
1279 2394 1
1280 2395 2 BEGIN
1281 2396 2
1282 2397 2 BIND_COMMON;
1283 2398 2
1284 2399 2 LOCAL
001 !CDS0016 2400 2 WAITER,
1285 2401 2 POOLAVAIL : REF VECTOR,
1286 2402 2 AQB : REF BBLOCK;
1287 2403 2
1288 2404 2 ! This may be called multiple times. Check if already returned credits.
1289 2405 2
1290 2406 2
1291 2407 2 IF .BFR_CREDITS [3] EQL 0
1292 2408 2 THEN RETURN;
1293 2409 2
1294 2410 2 AQB = .CURRENT_VCB [VCBSL_AQB];
1295 2411 2 POOLAVAIL = CACHE_HDR [F1TBCSL_POOLAVAIL];
1296 2412 2
1297 2413 2 SERIAL_CACHE ();
1298 2414 2
1299 2415 2 DECR POOL FROM 3 TO 0
1300 2416 2 DO
1301 2417 2 BEGIN
1302 2418 2
1303 2419 2 BIND
1304 2420 2 POOL_WAITQ = CACHE_HDR [F11BCSQ_POOL_WAITQ] + .POOL*8 : BBLOCK;
1305 2421 2
1306 2422 2 IF .BFRS_USED [.POOL] NEQ 0
1307 2423 2 THEN
1308 2424 2 BUG_CHECK (XQPERR, 'buffer use not to zero yet');
1309 2425 2
1310 2426 2 POOLAVAIL [.POOL] = .POOLAVAIL [.POOL] + .BFR_CREDITS [.POOL];
1311 2427 2 BFR_CREDITS [.POOL] = 0;
1312 2428 2
1313 2429 2 IF .POOL_WAITQ [QFLNK] NEQ POOL_WAITQ [QFLNK]
1314 2430 2 THEN
1315 2431 2 BEGIN
1318-2 2432 2
1319 2433 2 REMQUE (.POOL_WAITQ [QF_NK], WAITER);
1320 2434 2 INSQUE (.WAITER, .AQB [AQB$SL_ACQFL]); ! insert AFTER us.
1321 2435 2 END;
1322 2436 2
1323 2437 2
1324 2438 2 END;
```

```

:001 :CDS0016 2439 2 IF .CACHE_HDR [F11BC$$_AMBIGQFL] NEQ 0
:002 :CDS0016 2440 THEN
:003 :CDS0016 2441 BEGIN
:004 :CDS0016 2442
:005 :CDS0016 2443 REMQUE (.CACHE_HDR [F11BC$$_AMBIGQFL], WAITER);
:006 :CDS0016 2444 INSQUE (.WAITER, .AOB [AOB$$_A$$_POFL]);
:007 :CDS0016 2445
:008 :CDS0016 2446 IF .CACHE_HDR [F11BC$$_AMBIGQFL] EQL CACHE_HDR [F11BC$$_AMBIGQFL]
:009 :CDS0016 2447 THEN
:010 :CDS0016 2448 BEGIN
:011 :CDS0016 2449 CACHE_HDR [F11BC$$_AMBIGQFL] = 0;
:012 :CDS0016 2450 CACHE_HDR [F11BC$$_AMBIGQBL] = 0;
:013 :CDS0016 2451 END;
:014 :CDS0016 2452
:015 :CDS0016 2453
:016 :CDS0016 2454 END;
:017 :CDS0016 2455
:018 :CDS0016 2456
:1325 2457 RELEASE_CACHE ();
:1326 2458
:1327 2459 1 END;

```

			00FC 00000	.ENTRY	RETURN CREDITS, Save R2,R3,R4,R5,R6,R7	: 2382
	56	EC	AA 9E 00002	MOVAB	-20(BASE), R6	: 2395
	55	FC	AA 9E 00006	MOVAB	-4(BASE), R5	
		06	A6 B5 0000A	TSTW	6(R6)	: 2407
			6E 13 0000D	BEQL	5\$	
	50	98	AA D0 0000F	MOVL	-104(BASE), R0	: 2410
	54	10	A0 D0 00013	MOVL	16(R0), AOB	
52	65	00000068	8F C1 00017	ADDL3	#104, (R5), POOLAVAIL	: 2411
			FA93 30 0001F	BSBW	SERIAL CACHE	: 2413
	50		03 D0 00022	MOVL	#3, POOL	: 2415
	51	00	B540 7E 00025	MOVAJ	@0(R5)[POOL], R1	: 2420
	51	48	A1 9E 0002A	MOVAB	72(R1), R1	
		F4	AA40 B5 0002E	TSTW	-12(BASE)[POOL]	: 2422
			04 13 00032	BEQL	2\$	
			FEFF 00034	BUGW		: 2424
			0000* 00036	.WORD	<BUG\$ XQPERR!4>	
	57		6640 3C 00038	MOVZWL	(R6)[POOL], R7	: 2426
	6240		57 C0 0003C	ADDL2	R7, (POOLAVAIL)[POOL]	
			6640 B4 00040	CLRW	(R6)[POOL]	: 2427
	51		61 D1 00043	CMLP	(R1), R1	: 2429
			08 13 00046	BEQL	3\$	
	53	00	B1 0F 00048	REMQUE	@0(R1), WAITER	: 2433
00	B4		63 0E 0004C	INSQUE	(WAITER), @0(AOB)	: 2434
	D2		50 F4 00050	SOBGEQ	POOL, 1\$: 2435
	51		65 D0 00053	MOVL	(R5), R1	: 2439
	51	0080	C1 D0 00056	MOVL	128(R1), R1	
			1D 13 0005B	BEQL	4\$	
	53		61 0F 0005D	REMQUE	(R1), WAITER	: 2443
00	B4		63 0E 00060	INSQUE	(WAITER), @0(AOB)	: 2444
	50		65 D0 00064	MOVL	(R5), R0	: 2446
	50	0080	C0 9E 00067	MOVAB	128(R0), R0	

RDBLOK
V04-005

N 1
8-Jan-1985 18:20:35 VAX-11 Bliss-32 V4.0-742
2-Oct-1984 12:43:36 [F11X.BUGSRC]RDBLOK.B32;1

Page 44
(11)

50		60	D1	0006C		CMPL	(R0), R0
		09	12	0006F		BNEC	4\$
		60	D4	00071		CLRL	(R0)
50		65	D0	00073		MOVL	(R5), R0
	0084	C0	D4	00076		CLRL	132(R0)
		FEB9	30	0007A	4\$:	BSBW	RELEASE_CACHE
			04	0007D	5\$:	RET	

.....
: 2449
: 2450
: 2457
: 2459

; Routine Size: 126 bytes, Routine Base: \$CODE\$ + 07D1

RDB
V04

: R

```

: 1329      2460 1 GLOBAL ROUTINE WRITE_BLOCK (BUFFER) : L_NORM NGVALUE =
: 1330      2461 1
: 1331      2462 1 : **
: 1332      2463 1
: 1333      2464 1 FUNCTIONAL DESCRIPTION:
: 1334      2465 1
: 1335      2466 1     This routine writes the indicated block back to the disk.
: 1336      2467 1
: 1337      2468 1 CALLING SEQUENCE:
: 1338      2469 1     WRITE_BLOCK (ARG1)
: 1339      2470 1
: 1340      2471 1 INPUT PARAMETERS:
: 1341      2472 1     ARG1: address of block buffer
: 1342      2473 1
: 1343      2474 1 IMPLICIT INPUTS:
: 1344      2475 1     BUFFER DESCRIPTOR ARRAYS
: 1345      2476 1
: 1346      2477 1 OUTPUT PARAMETERS:
: 1347      2478 1     NONE
: 1348      2479 1
: 1349      2480 1 IMPLICIT OUTPUTS:
: 1350      2481 1     NONE
: 1351      2482 1
: 1352      2483 1 ROUTINE VALUE:
: 1353      2484 1     NONE
: 1354      2485 1
: 1355      2486 1 SIDE EFFECTS:
: 1356      2487 1     block written
: 1357      2488 1
: 1358      2489 1 --
: 1359      2490 1
: 1360      2491 2 BEGIN
: 1361      2492 2
: 1362      2493 2 LOCAL
: 1363      2494 2     BFRD           : REF BBLOCK,
: 1364      2495 2     STATUS,         : service status of QIO call
: 1365      2496 2     INDX;           : index of buffer
: 1366      2497 2
: 1367      2498 2 EXTERNAL
: 1368      2499 2     CLUSGL_CLUB   : ADDRESSING MODE (ABSOLUTE),
: 1369      2500 2     ACP$GB_DATACHK : BITVECTOR ADDRESSING MODE (ABSOLUTE);
: 1370      2501 2     : ACP datacheck enable flags
: 1371      2502 2
: 1372      2503 2 BIND_COMMON;
: 1373      2504 2
: 1374      2505 2 EXTERNAL LITERAL
: 1375      2506 2     ACPSV_WRITECHK : UNSIGNED (6); ! write check enable flag
: 1376      2507 2
: 1377      2508 2 INDX = (.BUFFER - .CACHE_HDR [F11BC$$_BUFBASE])/512 + 1;
: 1378      2509 2
: 1379      2510 2 IF .INDX<0,16> GTRU .CACHE_HDR [F11BC$$_BFRCNT]
: 1380      2511 2 THEN
: 1381      2512 2     BUG_CHECK (BADBUFADR, 'out of this cache, man');
: 1382      2513 2
: 1383      2514 2 BFRD = BFRD_ADDR (.INDX);
: 1384      2515 2
: 1385      2516 2 ! Stuff the UCB of IO_CHANNEL to go where we want.

```

```

: 1386      2517 2 !
: 1387      2518 2
: 1388      2519 2 IO_CCB [CCBSL_UCB] = .BFRD [BFRDSL_UCB];
: 1389      2520 2
: 1390      2521 2 BEGIN
: 1391      2522 2 LOCAL
: 1392      2523 2
: 1393      2524 2     PTR          : REF BBLOCK,
: 1394      2525 2     SAVE_PRIV   : VECTOR [4];
: 1395      2526 2
: 1396      2527 2 PTR = .CTL$GL PCB;
: 1397      2528 2 PTR [PCBSW_DIOCNT] = .PTR [PCBSW_DIOCNT] + 1;
: 1398      2529 2 PTR [PCBSW_ASTCNT] = .PTR [PCBSW_ASTCNT] + 1;
: 1399      2530 2 SAVE_PRIV [0] = .(PTR [PCBSQ_PRIV]);
: 1400      2531 2 SAVE_PRIV [1] = .(PTR [PCBSQ_PRIV]+4);
: 1401      2532 2 BBLOCK [PTR [PCBSQ_PRIV], PRVSV_LOG_IO] = 1;
: 1402      2533 2 BBLOCK [PTR [PCBSQ_PRIV], PRVSV_BYPASS] = 1;
: 1403      2534 2 PTR = .CTL$GL PHD;
: 1404      2535 2 SAVE_PRIV [2] = .(PTR [PHDSQ_PRIVMSK]);
: 1405      2536 2 SAVE_PRIV [3] = .(PTR [PHDSQ_PRIVMSK]+4);
: 1406      2537 2 BBLOCK [PTR [PHDSQ_PRIVMSK], PRVSV_LOG_IO] = 1;
: 1407      2538 2 BBLOCK [PTR [PHDSQ_PRIVMSK], PRVSV_BYPASS] = 1;
: 1408      2539 2
: 1409      2540 2 PMS TOT WRITE = .PMS_TOT_WRITE + 1;
: 1410      2541 2 STATUS = $QIO (
: 1411      2542 2     EFN = EFN,
: 1412      2543 2     ASTADR = CONTINUE_THREAD,
: 1413      2544 2     ASTPRM = .BASE,
: 1414      2545 2     CHAN = .IO_CHANNEL,
: 1415      2546 2     FUNC = (IOS_WRITEBLK
: 1416      2547 2             OR .ACPSGB_DATACHK[ACPSV_WRITECHK]
: 1417      2548 2             ^ $BITPOSITION (IOSV_DATACHECK)),
: 1418      2549 2     IOSB = IO_STATUS,
: 1419      2550 2     P1 = .BUFFER,
: 1420      2551 2     P2 = 512,
: 1421      2552 2     P3 = .BFRD [BFRDSL_LBN]
: 1422      2553 2 );
: 1423      2554 2 (PTR [PHDSQ_PRIVMSK]) = .SAVE_PRIV [2];
: 1424      2555 2 (PTR [PHDSQ_PRIVMSK]+4) = .SAVE_PRIV [3];
: 1425      2556 2 PTR = .CTL$GL PCB;
: 1426      2557 2 PTR [PCBSW_DIOCNT] = .PTR [PCBSW_DIOCNT] - 1;
: 1427      2558 2 PTR [PCBSW_ASTCNT] = .PTR [PCBSW_ASTCNT] - 1;
: 1428      2559 2 (PTR [PCBSQ_PRIV]) = .SAVE_PRIV [0];
: 1429      2560 2 (PTR [PCBSQ_PRIV]+4) = .SAVE_PRIV [1];
: 1430      2561 2
: 1431      2562 2 END:          ! of block defining PTR, SAVE_PRIV
: 1432      2563 2
: 1433      2564 2 IF NOT .STATUS
: 1434      2565 2 THEN IO_STATUS = .STATUS
: 1435      2566 2 ELSE WAIT_FOR_AST();
: 1436      2567 2
: 1437      2568 2 ! Restore CURRENT_UCB to IO_CHANNEL.
: 1438      2569 2 !
: 1439      2570 2
: 1440      2571 2 IO_CCB [CCBSL_UCB] = .CURRENT_UCB;
: 1441      2572 2
: 1442      2573 2 IF .IO_STATUS

```



```
: 1443      2574      2 THEN
: 1444      2575      BEGIN
: 1445      2576      LOCAL
: 1446      2577      CLUSTER,
: 1447      2578      LCKINDX;
: 1448      2579
: 1449      2580      BFRD [BFRDSV_DIRTY] = 0;
: 1450      2581
: 1451      2582      ! Now let's find the lock that backs this buffer and bump it's
: 1452      2583      ! sequence number so the rest of the world will know we've modified
: 1453      2584      ! this buffer.
: 1454      2585
: 1455      2586
: 1456      2587      LCKINDX = 0;
: 1457      2588
: 1458      2589      DO
: 1459      2590      BEGIN
: 1460      2591
: 1461      2592      IF .BFRD [BFRDSL_LOCKBASIS] EQL .LB_BASIS [.LCKINDX]
: 1462      2593      THEN
: 1463      2594      EXITLOOP;
: 1464      2595
: 1465      2596      LCKINDX = .LCKINDX + 1;
: 1466      2597      END
: 1467      2598      UNTIL .LCKINDX EQL LB_NUM;
: 1468      2599
: 1469      2600      IF .LCKINDX EQL LB_NUM
: 1470      2601      THEN
: 1471      2602      BUG_CHECK (XQPERR, 'no backing lock for dirty buffer');
: 1472      2603
: 1473      2604      CLUSTER = 0;
: 1474      2605
: 1475      2606      IF .BBLOCK [CURRENT_UCB [UCBSL_DEVCHAR2], DEV$V_CLU]
: 1476      2607      AND .CLU$GL_CLUB NEQ 0
: 1477      2608      THEN
: 1478      2609      CLUSTER = 1;
: 1479      2610
: 1480      2611      CASE .BFRD [BFRDSB_BTYPE] FROM 0 TO 5 OF
: 1481      2612      SET
: 1482      2613      [HEADER_TYPE]:
: 1483      2614      IF .CLUSTER
: 1484      2615      THEN
: 1485      2616      LB_HDRSEQ [.LCKINDX] = .LB_HDRSEQ [.LCKINDX] + 1;
: 1486      2617
: 1487      2618      [DIRECTORY_TYPE, DATA_TYPE]:
: 1488      2619      IF .CLUSTER
: 1489      2620      THEN
: 1490      2621      LB_DATASEQ [.LCKINDX] = .LB_DATASEQ [.LCKINDX] + 1;
: 1491      2622
: 1492      2623      [BITMAP_TYPE]:
: 1493      2624      (LB_DATASEQ [0])<0,16,0> = .(LB_DATASEQ [0])<0,16,0> + 1;
: 1494      2625
: 1495      2626      [INDEX_TYPE]:
: 1496      2627      (LB_DATASEQ [0])<16,16,0> = .(LB_DATASEQ [0])<16,16,0> + 1;
: 1497      2628
: 1498      2629      [QUOTA_TYPE]:
: 1499      2630      SAVE_VC_FLAGS<1,15,0> = .SAVE_VC_FLAGS<1,15,0> + 1;
```

```

: 1500      2631      3
: 1501      2632      TES;
: 1502      2633
: 1503      2634      END
: 1504      2635      ELSE
: 1505      2636      BEGIN
: 1506      2637
: 1507      2638      ! clean up will iterate on write failures if the volume is write locked until
: 1508      2639      ! the best effort clean up has been done. However, if the write error was on the
: 1509      2640      ! storage bitmap, we allocation lock the volume to prevent further damage.
: 1510      2641
: 1511      2642
: 1512      2643      IF .BFRD [BFRD$V POOL] EQL 0      ! this is storage bitmap pool
: 1513      2644      AND SURFACE_ERROR (.IO_STATUS [0])
: 1514      2645      THEN
: 1515      2646      CURRENT_VCB [VCB$V_NOALLOC] = 1;
: 1516      2647
: 1517      2648      INVALIDATE (.BUFFER);
: 1518      2649      ERR_EXIT (.IO_STATUS[0]);
: 1519      2650      END;
: 1520      2651
: 1521      2652      1 END;

```

! end of routine WRITE_BLOCK

```

                                .EXTRN CLUSGL CLUB, ACPSY_WRITECHK
                                .EXTRN BUGS_BADBUFADR
                                .ENTRY WRITE_BLOCK, Save R2,R3,R4,R5      : 2460
MOVAB CTL$GL PCB, R5
SUBL2 #16, SP
MOVAB 168(BASE), R4      : 2500
SUBL3 @-4(BASE), BUFFER, R2      : 2508
DIVL2 #512, R2
INCL INDX
MOVL -4(BASE), R0      : 2510
CMPW INDX, 22(R0)
BLEQU 1$
BUGW      : 2512
WORD <BUGS_BADBUFADR!4>
MOVL -4(BASE), R0      : 2514
MULL2 #32, R2
ADDL2 24(R0), R2
SUBL2 #32, BFRD
MOVL 12(BFRD), @-140(BASE)      : 2510
MOVL CTL$GL PCB, PTR      : 2526
INCL 62(PTR)      : 2527
INCL 56(PTR)      : 2528
MOVQ 132(PTR), SAVE_PRIV      : 2529
BISL2 #536871040, 132(PTR)      : 2532
MOVL CTL$GL PHD, PTR      : 2533
MOVQ (PTR), SAVE_PRIV+8      : 2534
BISL2 #536871040, -(PTR)      : 2537
INCL 2284(BASE)      : 2539
CLRQ -(SP)
CLRL -(SP)
PUSHL 8(BFRD)

```

		7E	0200	8F	3C	00076	MOVZWL	#512, -(SP)	
			04	AC	DD	0007B	PUSHL	BUFFÉR	
				5A	DD	0007E	PUSHL	BASE	
			0000G	CF	9F	00080	PUSHAB	CONTINUE THREAD	
			88	AA	9F	00084	PUSHAB	-120(BASE)	
50	00000000G	9F	01	00G	EF	00087	EXTZV	S^ACPSV_WRITECHK, #1, @#ACPSGB_DATACHK, RO	
		50	50	0E	78	00090	ASHL	#14, RO, RO	
		7E	50	20	C9	00094	BISL3	#32, RO, -(SP)	
				FF78	CA	00098	PUSHL	-136(BASE)	
					1E	0009C	PUSHL	#30	
	00000000G		00	0C	FB	0009E	CALLS	#12, SYSSQIO	
			63	08	AE	000A5	MOVQ	SAVE PRIV+8, (PTR)	2554
			53	65	D0	000A9	MOVL	CTL\$GL PCB, PTR	2556
				3E	A3	000AC	DECW	62(PTR)	2557
				38	A3	000AF	DECW	56(PTR)	2558
	0084	C3		6E	7D	000B2	MOVQ	SAVE PRIV, 132(PTR)	2559
		06		50	E8	000B7	BLBS	STATUS, 2\$	2564
	88	AA		50	D0	000BA	MOVL	STATUS, -120(BASE)	2565
				05	11	00^BE	BRB	3\$	
	0000G	CF		00	FB	000C0	CALLS	#0, WAIT FOR AST	2566
	FF74	DA	94	AA	D0	000C5	MOVL	-108(BASE), 3-140(BASE)	2571
		51	88	AA	D0	000CB	MOVL	-120(BASE), R1	2573
		6B		51	E9	000CF	BLBC	R1, 14\$	
	18	A2		04	8A	000D2	BICB2	#4, 24(BFRD)	2580
				50	D4	000D6	CLRL	LCKINDX	2587
	0080	CA40	10	A2	D1	000D8	CMPL	16(BFRD), 128(BASE)[LCKINDX]	2592
				07	13	000DF	BEQL	5\$	
				50	D6	000E1	INCL	LCKINDX	2596
		05		50	D1	000E3	CMPL	LCKINDX, #5	2598
				F0	12	000E6	BNEQ	4\$	
		05		50	D1	000E8	CMPL	LCKINDX, #5	2600
				04	12	000EB	BNEQ	6\$	
				FEFF	00CED		BUGW		2602
				0000*	000EF		.WORD	<BUG\$ XQPERR!4>	
				53	D4	000F1	CLRL	CLUSTER	2604
		51	94	AA	D0	000F3	MOVL	-108(BASE), R1	2606
		08	3C	A1	E9	000F7	BLBC	60(R1), 7\$	
			00000000G	9F	D5	000FB	TSTL	@#CLUSGL_CLUB	2607
				03	13	00101	BEQL	7\$	
		53		01	D0	00103	MOVL	#1, CLUSTER	2609
001F	05	00	19	A2	8F	00106	CASEB	25(BFRD), #0, #5	2611
	0015	001C		000C		0010B	.WORD	9\$-8\$, -	
		0023		0015		00113		11\$-8\$, -	
								10\$-8\$, -	
								12\$-8\$, -	
								10\$-8\$, -	
								13\$-8\$	
		60		53	E9	00117	BLBC	CLUSTER, 18\$	2614
			0094	CA40	D6	0011A	INCL	148(BASE)[LCKINDX]	2616
					04	0011F	RET		2614
		57		53	E9	00120	BLBC	(CLUSTER, 18\$	2619
				6440	D6	00123	INCL	(R4)[LCKINDX]	2621
					04	00126	RET		2619
				64	B6	00127	INCW	(R4)	2624
					04	00129	RET		
			02	A4	B6	0012A	INCW	2(R4)	2627
					04	0012D	RET		

RDBLOK
V04-005

G 2
8-Jan-1985 18:20:35
2-Oct-1984 12:43:36

VAX-11 Bliss-32 V4.0-742
[F11X.BUGSRC]RDBLOK.B32;1

Page 50
(12)

50	A4	AA	0F	01	EF	0012E	13\$:	EXTZV	#1, #15, -92(BASE), R0	: 2630	
				50	D6	00134		INCL	R0	: 2630	
A4	AA		0F	50	F0	00136		INSV	R0, #1 #15, -92(BASE)	: 2630	
					04	0013C		RET		: 2573	
			03	18	A2	93	0013D	14\$:	BITB	24(BFRD), #3	: 2643
		000001F4	8F		09	12	00141		BNEQ	15\$: 2644
					51	D1	00143		CMPL	R1, #500	: 2644
		0000005C	8F		iB	13	0014A		BEQL	16\$: 2644
					51	D1	0014C	15\$:	CMPL	R1, #92	: 2644
		000000BC	8F		12	13	00153		BEQL	16\$: 2644
					51	D1	00155		CMPL	R1, #188	: 2644
		00002144	8F		09	13	0015C		BEQL	16\$: 2644
					51	D1	0015E		CMPL	R1, #8516	: 2644
			50	98	08	12	00165		BNEQ	17\$: 2646
	OB	A0			AA	D0	00167	16\$:	MOVL	-104(BASE), R0	: 2646
					10	88	0016B		BISB2	#16, 11(R0)	: 2648
			04		AC	DD	0016F	17\$:	PUSHL	BUFFER	: 2648
	0000V	CF			01	FB	00172		CALLS	#1, INVALIDATE	: 2649
				88	AA	BF	00177		CHMU	-120(BASE)	: 2652
					04	0017A	18\$:	RET		: 2652	

; Routine Size: 379 bytes, Routine Base: \$CODE\$ + 084F

				00FC 00000	.ENTRY	CREATE_BLOCK, Save R2,R3,R4,R5,R6,R7	: 2653
		5E		04 C2 00002	SUBL2	#4, SP	: 2692
				5E DD 00005	PUSHL	SP	
				01 DD 00007	PUSHL	#1	
			0C	AC DD 00009	PUSHL	TYPE	
			04	AC DD 0000C	PUSHL	LBN	
	F8F8	CF		04 FB 0000F	CALLS	#4, FIND_BUFFER	
		52		50 D0 00014	MOVL	R0, INDX0	
		51	FC	AA D0 00017	MOVL	-4(BASE), R1	: 2694
	50	52		05 78 0001B	ASHL	#5, INDX0, R0	
	56	50	18	A1 C1 0001F	ADDL3	24(R1), R0, BFRD	
	50	52		09 78 00024	ASHL	#9, INDX0, R0	: 2696
	57	50	FC	BA C1 00028	ADDL3	@-4(BASE), R0, BUFFER	
G200	8F	00		00 2C 0002D	MOVC5	#0, (SP), #0, #512, (BUFFER)	: 2697
		6E		67 00034			
	50	04	AC	01 C1 00035	ADDL3	#1, LBN, R0	: 2699
				04 13 0003A	BEQL	1\$	
		18	A6	0C 88 0003C	BISB2	#12, 24(BFRD)	: 2703
		50		57 D0 00040	MOVL	BUFFER, R0	: 2706
				04 00043	RET		: 2708

: Routine Size. 68 bytes. Routine Base: \$CODE\$ + 09CA

```
1580 2709 1 GLOBAL ROUTINE UNHOOK_BFRD (INDXARG, BFRDARG) : L_NORM NOVALUE =
1581 2710 1
1582 2711 1 !++
1583 2712 1
1584 2713 1 FUNCTIONAL DESCRIPTION:
1585 2714 1
1586 2715 1 This routine extricates a BFRD and its associated BFRL from
1587 2716 1 their hash lookup lists. It also takes care of the BFRL if
1588 2717 1 this was the last reference to it.
1589 2718 1 The caller of this routine is responsible for clearing or
1590 2719 1 rewriting other relevant fields in the BFRD.
1591 2720 1
1592 2721 1 !--
1593 2722 1
1594 2723 2 BEGIN
1595 2724 2
1596 2725 2 BIND_COMMON;
1597 2726 2
1598 2727 2 EXTERNAL ROUTINE
1599 2728 2     DEL_EXTFCB      : L_NORM,
1600 2729 2     NUKE_HEAD_FCB  : L_NORM NOVALUE;
1601 2730 2
1602 2731 2 LOCAL
1603 2732 2     BFRD           : REF BBLOCK,
1604 2733 2     INDX           : WORD,
1605 2734 2     INDX_ADDR,
1606 2735 2     BFRL           : REF BBLOCK;
1607 2736 2
1608 2737 2 BFRD = .BFRDARG;
1609 2738 2 INDX = .INDXARG;
1610 2739 2
1611 2740 2 IF .BFRD [BFRD$V_DIRTY]
1612 2741 2 THEN
1613 2742 2     BUG_CHECK (XQPERR, 'Should not find dirty buffer');
1614 2743 2
1615 2744 2 BFRD [BFRD$V_VALID] = 0;
1616 2745 2
1617 2746 2 IF .BFRD [BFRD$B_BTYPE] EQL DIRINDX_TYPE
1618 2747 2 THEN
1619 2748 3     BEGIN
1620 2749 3     LOCAL
1621 2750 3         BUFR      : REF BBLOCK,
1622 2751 3         DIRFCB   : REF BBLOCK;
1623 2752 3
1624 2753 3 ! DIRINDX_TYPE blocks are not hashed by lbn, and the field is used
1625 2754 3 ! to point to the directory fcb.
1626 2755 3
1627 2756 3     IF (DIRFCB = .BFRD [BFRD$L_LBN]) NEQ 0
1628 2757 3     THEN
1629 2758 4         BEGIN
1630 2759 4
1631 2760 4             IF .DIRFCB [FCB$B_TYPE] NEQ DYN$C_FCB
1632 2761 4             OR .DIRFCB [FCB$L_DIRINDX] EQL 0
1633 2762 4             THEN
1634 2763 4                 BUG_CHECK (XQPERR, 'not legit dirfcb');
1635 2764 4
1636 2765 4 ! This will break the association of this directory index buffer
```

```
1637 2766 4 | from the directory fcb.
1638 2767 4 |
1639 2768 4 |
1640 2769 4 | DIRFCB [FCBSL_DIRINDX] = 0;
1641 2770 4 |
1642 2771 4 | IF .DIRFCB [FCBSW_REFCNT] EQL 0
1643 2772 4 | THEN
1644 2773 4 | IF TESTBITSC (DIRFCB [FCBSV_DIR])
1645 2774 4 |
1646 2775 4 | from this point on, because the refcnt was zero, and the DIR
1647 2776 4 | flag was set and is now clear, the search_fcb routine will no
1648 2777 4 | longer find this fcb, so we are free to deallocate it.
1649 2778 4 | Furthermore, because the DIR was set, we know that no search_fcb
1650 2779 4 | had found it at the time we cleared it because search_fcb will
1651 2780 4 | unconditionally clear it.
1652 2781 4 |
1653 2782 4 | THEN
1654 2783 5 | BEGIN
1655 2784 5 |
1656 2785 5 | Note that we are under the cache interlock here, and assume that
1657 2786 5 | the these routines will do nothing that could cause a stall.
1658 2787 5 |
1659 2788 5 |
1660 2789 5 | DEL_EXTFCB (.DIRFCB);
1661 2790 5 | NUKE_HEAD_FCB (.DIRFCB);
1662 2791 4 | END;
1663 2792 4 |
1664 2793 4 | BFRD [BFRDSL_LBN] = 0;
1665 2794 4 |
1666 2795 3 | END; ! of bfrd$l_lbn (dirfcb pointer) neq zero
1667 2796 3 |
1668 2797 2 | END;
1669 2798 2 |
1670 2799 2 | IF .BFRD [BFRDSL_LBN] NEQ 0
1671 2800 2 | THEN
1672 2801 2 |
1673 2802 2 | BFRDSL_LBN neq 0 means that this buffer is in the hash lookup list.
1674 2803 2 | We need to remove it from the list it is currently
1675 2804 2 | in as we are about to use the same buffer for some other LBN.
1676 2805 2 |
1677 2806 2 |
1678 2807 3 | BEGIN
1679 2808 3 |
1680 2809 3 | INDX_ADDR = LOOKUP_LBN (.BFRD [BFRDSL_LBN]);
1681 2810 3 |
1682 2811 3 | IF .(.INDX_ADDR)<0,16> EQL .INDX
1683 2812 3 | THEN
1684 2813 3 | (.INDX_ADDR)<0,16> = .BFRD [BFRDSW_NXTBFRD]
1685 2814 3 | ELSE
1686 2815 4 | BEGIN
1687 2816 4 | DO
1688 2817 4 | INDX_ADDR
1689 2818 4 | = BBLOCK [ BFRD_ADDR (.INDX_ADDR)<0,16>), BFRDSW_NXTBFRD]
1690 2819 4 | UNTIL .(.INDX_ADDR)<0,16> EQL .INDX;
1691 2820 4 |
1692 2821 4 | (.INDX_ADDR)<0,16> = .BFRD [BFRDSW_NXTBFRD];
1693 2822 3 | END;
```



```

: 1693      2823      2      END;
: 1694      2824      2
: 1695      2825      2      ! If it had a buffer lock hooked up with it, adjust the reference count
: 1696      2826      2      ! on the lock descriptor, and remove the lock descriptor from its hash
: 1697      2827      2      ! lookup list if this is the last reference to it.
: 1698      2828      2
: 1699      2829      2
: 1700      2830      2      IF .BFRD [BFRD$W_BFRL] NEQ 0
: 1701      2831      2      THEN
: 1702      2832      2          BEGIN
: 1703      2833      3
: 1704      2834      3          BFRL = BFRD_ADDR (.BFRD [BFRD$W_BFRL]);
: 1705      2835      3
: 1706      2836      3          BFRL [BFRL$W_REFCNT] = .BFRL [BFRL$W_REFCNT] - 1;
: 1707      2837      3          IF .BFRL [BFRL$W_REFCNT] EQL 0
: 1708      2838      3          THEN
: 1709      2839      4              BEGIN
: 1710      2840      4
: 1711      2841      4                  IF .BFRL [BFRL$L_LKID] NEQ 0
: 1712      2842      4                  THEN
: 1713      2843      5                      BEGIN
: 1714      2844      6                          IF NOT $DEQ (LKID = .BFPL [BFRL$L_LKID])
: 1715      2845      5                          THEN
: 1716      2846      5                              BUG_CHECK (XQPERR, 'unexpected lock manager reaction');
: 1717      2847      4                          END;
: 1718      2848      4
: 1719      2849      4      ! Hash on the lockbasis,parlkid pair and scan the list to find the
: 1720      2850      4      ! desired bfrl, if necessary. Pull it from the list when located.
: 1721      2851      4
: 1722      2852      4
: 1723      P 2853      4          INDX_ADDR = LOOKUP_LOCK (.BFRL [BFRL$L_LCKBASIS],
: 1724      2854      4          .BFRL [BFRL$L_PARLKID]);
: 1725      2855      4
: 1726      2856      4          IF .(.INDX_ADDR)<0,16> EQL .BFRD [BFRD$W_BFRL]
: 1727      2857      4          THEN
: 1728      2858      4              (.INDX_ADDR)<0,16> = .BFRL [BFRL$W_NXTBFRL]
: 1729      2859      4          ELSE
: 1730      2860      5              BEGIN
: 1731      2861      5
: 1732      2862      5                  DO
: 1733      2863      5                      INDX_ADDR = BBLOCK [ BFRD_ADDR (.(.INDX_ADDR)<0,16>), BFRL$W_NXTBFRL]
: 1734      2864      5                      UNTIL .(.INDX_ADDR)<0,16> EQL .BFRD [BFRD$W_BFRL];
: 1735      2865      5
: 1736      2866      5                      (.INDX_ADDR)<0,16> = .BFRL [BFRL$W_NXTBFRL];
: 1737      2867      4                  END;
: 1738      2868      4
: 1739      2869      4          BFRL [BFRL$L_LKID] = 0;
: 1740      2870      4          BFRL [BFRL$L_LCKBASIS] = 0;
: 1741      2871      4          BFRL [BFRL$L_PARLKID] = 0;
: 1742      2872      4
: 1743      2873      4          BFRL [BFRL$W_NXTBFRL] = .CACHE_HDR [F11BC$W_FREEBFRL];
: 1744      2874      4          CACHE_HDR [F11BC$W_FREEBFRL] = .BFRD [BFRD$W_BFRL];
: 1745      2875      3          END;
: 1746      2876      3
: 1747      2877      3      ! There is no longer a BFRL associated with this BFRD.
: 1748      2878      3
: 1749      2879      3

```


			64	1E	A3	B0	00094	8\$:	MOVW	30(BFRD), (INDX_ADDR)	2821
				1A	A3	B5	00098	9\$:	TSTW	26(BFRD)	2830
					01	12	0009B		BNEQ	10\$	
						04	0009D		RET		
			51		65	D0	0009E	10\$:	MOVL	(R5), R1	2834
			50	1A	A3	3C	000A1		MOVZWL	26(BFRD), R0	
			50		10	C4	000A5		MULL2	#16, R0	
			50	1C	A1	C0	000A8		ADDL2	28(R1), R0	
			52	F0	A0	9E	000AC		MOVAB	-16(R0), BFRL	
				02	A2	B7	000B0		DECW	2(BFRL)	2836
					72	12	000B3		BNEQ	15\$	2837
				04	A2	D5	000B5		TSTL	4(BFRL)	2841
					15	13	000B8		BEQL	11\$	
					7E	7C	000BA		CLRW	-(SP)	2844
					7E	D4	000BC		CLRL	-(SP)	
				04	A2	DD	000BE		PUSHL	4(BFRL)	
		00000000G	00		04	FB	000C1		CALLS	#4, SYSSDEQ	
			04		50	EB	000C8		BLBS	R0, 11\$	
						FEFF	000CB		BUGW		2846
						0000*	000CD		.WORD	<BUGS_XQPERR!4>	
			51		65	D0	000CF	11\$:	MOVL	(R5), R1	2854
	50	08		0C	A2	C1	000D2		ADDL3	12(BFRL), 8(BFRL), R0	
			56	24	A1	3C	000D8		MOVZWL	36(R1), R6	
7E			50		01	7A	000DC		EMUL	#1, R0, #0, -(SP)	
50	00		8E		56	7B	000E1		EDIV	R6, (SP)+, R0, R0	
					50	D5	000E6		ICTL	R0	
					03	18	000E8		BGEQ	12\$	
			50		50	CE	000EA		MNEGL	R0, R0	
			54	20	B140	3E	000ED	12\$:	MOVW	@32(R1)[R0], INDX_ADDR	
				1A	A3	B1	000F2		CMPW	(INDX_ADDR), 26(BFRD)	2856
					17	13	000F6		BEQL	14\$	
			50		65	D0	000F8	13\$:	MOVL	(R5), R0	2863
			51		64	3C	000FB		MOVZWL	(INDX_ADDR), R1	
			51		10	C4	00CFE		MULL2	#16, R1	
			51	1C	A0	C0	00101		ADDL2	28(R0), R1	
			54	F0	A1	9E	00105		MOVAB	-16(R1), INDX_ADDR	
				1A	A3	B1	00109		CMPW	(INDX_ADDR), 26(BFRD)	2864
					E9	12	0010D		BNEQ	13\$	
			64		62	B0	0010F	14\$:	MOVW	(BFRL), (INDX_ADDR)	2866
				04	A2	7C	00112		CLRW	4(BFRL)	2869
				0C	A2	D4	00115		CLRL	12(BFRL)	2871
			50		65	D0	00118		MOVL	(R5), R0	2873
			62	26	A0	B0	0011B		MOVW	38(R0), (BFRL)	
			50		65	D0	0011F		MOVL	(R5), R0	2874
	26		A0	1A	A3	B0	00122		MOVW	26(BFRD), 38(R0)	
				1A	A3	B4	00127	15\$:	CLRW	26(BFRD)	2880
					04	0012A			RET		2884

; Routine Size: 299 bytes, Routine Base: \$CODE\$ + 0A0E

```

:001 CDS0020 2885 1 ROUTINE FREE_ONE (POOL) : L_NORM =
:1757-1 2886 1
:1758 2887 1 **
:1759 2888 1
:1760 2889 1 FUNCTIONAL DESCRIPTION:
:1761 2890 1
:1762 2891 1 This routine tosses a buffer off the in-process list to make room
:1763 2892 1 for a new one. This only happens when we already have used up
:1764 2893 1 the minimum number of buffers for this pool, and there are not
:1765 2894 1 enough available in the cache to steal more from it.
:1766 2895 1
:001 CDS0020 2896 1 ROUTINE VALUE:
:002 CDS0020 2897 1 1 - if cache interlock released and reacquired
:003 CDS0020 2898 1 0 - if cache interlock never released.
:004 CDS0020 2899 1
:1767 2900 1 --
:1768 2901 1
:1769 2902 2 BEGIN
:1770 2903 2
:1771 2904 2 BIND_COMMON:
:1772 2905 2
:1773 2906 2 LOCAL
:001 CDS0020 2907 2 STATUS,
:002 CDS0020 2908 2 INDX0,
:003 CDS0020 2909 2 POOL_LRU : REF BBLOCK,
:004 CDS0020 2910 2 FREED : REF BBLOCK;
:005 CDS0020 2911 2
:006 CDS0020 2912 2 STATUS = 0;
:007 CDS0020 2913 2
:008 CDS0016 2914 2 FREED = .BFR_LIST [.POOL, QFLNK];
:009 CDS0020 2915 2
:010 CDS0020 2916 2 INDX0 = (.FREED - .CACHE_HDR [F11BC$L_BFRDBAS])/BFRD$$_BFRDDEF;
:011 CDS0020 2917 2
:012 CDS0020 2918 2 IF .FREED [BFRD$$_DIRTY]
:013 CDS0020 2919 2 THEN
:014 CDS0020 2920 3 BEGIN
:015 CDS0020 2921 3 RELEASE_CACHE ();
:016 CDS0020 2922 3 WRITE_BLOCK (.INDX0*512 + .CACHE_HDR [F11BC$L_BUFBASE]);
:017 CDS0020 2923 3 SERIAL_CACHE ();
:018 CDS0020 2924 3
:019 CDS0020 2925 3 STATUS = 1;
:020 CDS0020 2926 3
:021 CDS0020 2927 2 END;
:022 CDS0020 2928 2
:023 CDS0020 2929 2 Remove the bfrd out of the list now that it is written. It cannot
:024 CDS0020 2930 2 be pulled out prior to the possible write because the error path
:025 CDS0020 2931 2 from the write will call invalidate, which will also do a remque on it.
:026 CDS0020 2932 2
:027 CDS0020 2933 2 If a BFRL (lock) has not been associated with this bfrd yet, unhook it
:028 CDS0020 2934 2 from the hash lists and completely return it (zero fields, etc). We
:029 CDS0020 2935 2 do not attempt to associate a lock with it now because that logic is
:030 CDS0020 2936 2 part of the RELEASE_LOCKBASIS routine and is done in combination with
:031 CDS0020 2937 2 dequeuing the synchronization lock. To fabricate a NL lock here without
:032 CDS0020 2938 2 releasing the synch lock would be a pain and not worth it.
:033 CDS0020 2939 2
:034 CDS0020 2940 2 Also free the buffer entirely if not valid.
:035 CDS0020 2941 2

```

```

:036 :CDS0020 2942 2 : If there already is a BFRL with the BFRD, and it is valid, simply pull
:037 :CDS0020 2943 2 : the bfrd from the in-process list and return it to the cache lru, clearing
:038 :CDS0020 2944 2 : the CURPID field.
:039 :CDS0020 2945 2 :
:040 :CDS0020 2946 2 : RETURN_BFRD will fix BFRS_USED, otherwise do it here.
:041 :CDS0020 2947 2 :
:042 :CDS0020 2948 2 :
:043 :CDS0020 2949 2 IF .FREED [BFRD$W_BFRL] EQL 0
:044 :CDS0020 2950 2 OR NOT .FREED [BFRD$V_VALID]
:045 :CDS0020 2951 2 THEN
:046 :CDS0020 2952 2 BEGIN
:047 :CDS0020 2953 2 UNHOOK_BFRD (.INDX0+1, .FREED);
:048 :CDS0020 2954 2 RETURN_BFRD (.FREED);
:049 :CDS0020 2955 2 END
:050 :CDS0020 2956 2 ELSE
:051 :CDS0020 2957 2 BEGIN
:052 :CDS0020 2958 2
:053 :CDS0020 2959 2 REMQUE (.FREED, FREED);
:054 :CDS0020 2960 2
:055 :CDS0020 2961 2 POOL_LRU = CACHE HDR [F11BC$Q POOL_LRU] + .POOL*8;
:056 :CDS0020 2962 2 INSQUE (.FREED, POOL_LRU [QFLNK]);
:057 :CDS0020 2963 2 FREED [BFRD$W_CURPID] = 0;
:058 :CDS0020 2964 2 BFRS_USED [.POOL] = .BFRS_USED [.POOL] - 1;
:059 :CDS0020 2965 2 END;
:060 :CDS0020 2966 2
:061 :CDS0020 2967 2 RETURN .STATUS
: 1798-24 2968 2
: 1799 2969 1 END;

```

001C 00000 FREE_ONE:

					.WORD	Save R2,R3,R4		2885
			54	D4	00002	CLRL	STATUS	2912
	50	04	AC	D0	00004	MOVL	POOL, R0	2914
		CC	AA40	7F	00008	PUSHAQ	-52(BASE)[R0]	
	53		9E	D0	0000C	MOVL	@(SP)+, FREED	
	50	FC	AA	D0	0000F	MOVL	-4(BASE), R0	2916
50	53	18	A0	C3	00013	SUBL3	24(R0), FREED, R0	
52	50		20	C7	00018	DIVL3	#32, R0, INDX0	
19	18	A3	02	E1	0001C	BBC	#2, 24(FREED), 1\$	2918
			FBAA	30	00021	BSBW	RELEASE CACHE	2921
50	52		09	78	00024	ASHL	#9, INDX0, R0	2927
	51	FC	BA	D0	00028	MOVL	@-4(BASE), R1	
			6140	9F	0002C	PUSHAB	(R1)[R0]	
	FCE2	CF	01	FB	0002F	CALLS	#1, WRITE BLOCK	
			F716	30	00034	BSBW	SERIAL CACHE	2923
	54		01	D0	00037	MOVL	#1, STATUS	2925
		1A	A3	B5	0003A	TSTW	26(FREED)	2949
			05	13	0003D	BEQL	2\$	
12	18	A3	03	E0	0003F	BBS	#3, 24(FREED), 3\$	2950
			53	DD	00044	PUSHL	FREED	2953
		01	A2	9F	00046	PUSHAB	1(INDX0)	
	FE87	CF	02	FB	00049	CALLS	#2, UNHOOK_BFRD	
	50		53	D0	0004E	MOVL	FREED, R0	2954

RDBLOK
V04-005

D 3
8-Jan-1985 18:20:35 VAX-11 Bliss-32 V4.0-742
2-Oct-1984 12:43:36 [F11X.BUGSRC]RDBLOK.B32;1

Page 60
(15)

	0000V	30	00051	BSBW	RETURN_BFRD		
	1D	11	00054	BRB	4\$: 2949	
53	63	0F	00056	3\$: REMQUE	(FREED), FREED	: 2959	
50	04	AC	D0	00059	MOVL	POOL, R0	: 2961
50	FC	BA40	7E	0005D	MOVAQ	@-4(BASE)[R0], POOL_LRU	
50		28	C0	00062	ADDL2	#40, POOL_LRU	
60		63	0E	00065	INSQUE	(FREED), TPOOL_LRU	: 2962
	1C	A3	B4	00068	CLRW	28(FREED)	: 2963
50	04	AC	D0	0006B	MOVL	POOL, R0	: 2964
	F4	AA40	B7	0006F	DECW	-12(BASE)[R0]	
50		54	D0	00073	4\$: MOVL	STATUS, R0	: 2967
		04	00076	RET		: 2969	

: Routine Size: 119 bytes, Routine Base: \$CODE\$ + 0B39

```

: 1801      2970  1 GLOBAL ROUTINE RELEASE_LOCKBASIS (LCKINDX) : L_NORM =
: 1802      2971  1
: 1803      2972  1 :++
: 1804      2973  1
: 1805      2974  1 : FUNCTIONAL DESCRIPTION:
: 1806      2975  1
: 1807      2976  1 : This routine scans in-process buffers and prepares them for lowering
: 1808      2977  1 : the synchronization lock associated with LOCKBASIS.
: 1809      2978  1 : All dirty buffers associated with the given LOCKBASIS must have been
: 1810      2979  1 : written to disk previously. This routine runs under the cache serialization
: 1811      2980  1 : interlock, and if a conversion is required, will exit without releasing
: 1812      2981  1 : the cache serialization interlock.
: 1813      2982  1
: 1814      2983  1 :--
: 1815      2984  1
: 1816      2985  2 BEGIN
: 1817      2986  2
: 1818      2987  2 BIND_COMMON;
: 1819      2988  2
: 1820      2989  2 EXTERNAL
: 1821      2990  2     CLUSGL_CLUB      : ADDRESSING_MODE (GENERAL);
: 1822      2991  2
: 1823      2992  2 LOCAL
: 1824      2993  2     LOCKBASIS,
: 1825      2994  2     LKID_ADDR,
: 1826      2995  2     INDX;
: 1827      2996  2
: 1828      2997  2 LABEL
: 1829      2998  2     FIXBFRL,
: 1830      2999  2     LBMATCH;
: 1831      3000  2
: 1832      3001  2 LKID_ADDR = 0;
: 1833      3002  2 INDX = 0;
: 1834      3003  2 LOCKBASIS = .LB_BASIS [.LCKINDX];
: 1835      3004  2
: 1836      3005  2 SERIAL_CACHE ();
: 1837      3006  2
: 1838      3007  2 INCR POOL FROM 0 TO 3
: 1839      3008  2 DO
: 1840      3009  3     BEGIN
: 1841      3010  3     LOCAL
: 1842      3011  3     I,
: 1843      3012  3     QHEAD  : REF BBLOCK,
: 1844      3013  3     BFRL   : REF BBLOCK,
: 1845      3014  3     BFRD   : REF BBLOCK;
: 1846      3015  3
: 1847      3016  3     QHEAD = BFR_LIST [.POOL, QFLNK];
: 1848      3017  3     BFRD = .QHEAD [QFLNK];
: 1849      3018  3
: 1850      3019  3     I = .PFRS_USED [.POOL];
: 1851      3020  3
: 1852      3021  3 UNTIL .I EQL 0
: 1853      3022  3 DO
: 1854      3023  4     BEGIN
: 1855      3024  4     LOCAL
: 1856      3025  4     NXTBFRD;
: 1857      3026  4

```

```

: 1858      3027  4      I = .I - 1;
: 1859      3028  4      NXTBFRD = .BFRD [BFRD$$_QFL];
: 1860      3029  4
: 1861      3030  4      ! Check this BFRD for LOCKBASIS match. Note that we are scanning an
: 1862      3031  4      ! in-process list which can only contain buffers from a given volume
: 1863      3032  4      ! or volume set. Because the LOCKBASIS contains the RVN, it is not
: 1864      3033  4      ! necessary to make a separate check for a volume set match.
: 1865      3034  4      ! When the buffer was initially brought into the in-process list, it
: 1866      3035  4      ! matched correctly on the desired UCB.
: 1867      3036  4
: 1868      3037  4
: 1869      3038  4      IF .BFRD [BFRD$$_LOCKBASIS] EQL .LOCKBASIS
: 1870      3039  4      THEN
: 1871      3040  5      LBMATCH: BEGIN
: 1872      3041  5
: 1873      3042  5      LOCAL
: 1874      3043  5      POOL_LRU : REF BBLOCK;
: 1875      3044  5
: 1876      3045  5      IF NOT .BFRD [BFRD$$_VALID]
: 1877      3046  5      THEN
: 1878      3047  6      BEGIN
: 1879      3048  6      LOCAL
: 1880      3049  6      INDX0;
: 1881      3050  6
: 1882      3051  6      INDX0 = (.BFRD - .CACHE_HDR [F11BC$$_BFRDBAS])/BFRD$$_BFRDDEF;
: 1883      3052  6
: 1884      3053  6      UNHOOK_BFRD (.INDX0 + 1, .BFRD);
: 1885      3054  6
: 1886      3055  6      RETURN_BFRD (.BFRD);
: 1887      3056  6
: 1888      3057  6      LEAVE LBMATCH;
: 1889      3058  5      END;
: 1890      3059  5
: 1891      3060  5      ! This is a valid buffer to be returned to the cache.
: 1892      3061  5
: 1893      3062  5
: 1894      3063  6      FIXBFRL: BEGIN
: 1895      3064  6
: 1896      3065  6      LOCAL
: 1897      3066  6      LBINDX_ADDR,
: 1898      3067  6      PARLKID;
: 1899      3068  6
: 1900      3069  6      IF .BFRD [BFRD$$_DIRTY]
: 1901      3070  6      THEN
: 1902      3071  6      BUG_CHECK (XQPERR, 'should not have dirty buffers now');
: 1903      3072  6
: 1904      3073  7      PARLKID = .(IF .CURRENT_VCB [VCBSW_RVN] EQL 0
: 1905      3074  7      THEN CURRENT_VCB [VCBSL_VOLLKID]
: 1906      3075  6      ELSE CURRENT_RVT [RVTSL_STRUCLKID]);
: 1907      3076  6
: 1908      3077  6      IF .BFRD [BFRD$$_BFRL] NEQ 0
: 1909      3078  6      THEN
: 1910      3079  6
: 1911      3080  6      ! We're already hooked up to a BFRL. This means we found it already
: 1912      3081  6      ! in the cache in the first place. Nothing needs to be done with
: 1913      3082  6      ! the BFRL. However, let's be paranoid and make sure that it is
: 1914      3083  6      ! hooked up to the correct lock.
```



```

: 1915 3084 6
: 1916 3085 6
: 1917 3086 7
: 1918 3087 7
: 1919 3088 7
: 1920 3089 7
: 1921 3090 7
: 1922 3091 7
: 1923 3092 7
: 1924 3093 7
: 1925 3094 7
: 1926 3095 7
: 1927 3096 6
: 1928 3097 6
: 1929 3098 6
: 1930 3099 6 ! This BFRD is not hooked up with a BFRL yet, but we've already
: 1931 3100 6 ! got a BFRL in hand (from an earlier hit) so just hook it up
: 1932 3101 6 ! and bump the REFCNT in the BFRL.
: 1933 3102 6
: 1934 3103 7
: 1935 3104 7
: 1936 3105 7
: 1937 3106 7
: 1938 3107 7
: 1939 3108 7
: 1940 3109 6
: 1941 3110 6
: 1942 3111 6 ! There is no BFRL for this BFRD yet, so let's first see if a BFRL
: 1943 3112 6 ! for this LOCKBASIS is already in the cache.
: 1944 3113 6
: 1945 3114 6
: 1946 3115 6
: 1947 3116 6
: 1948 3117 6
: 1949 3118 6
: 1950 3119 6
: 1951 3120 7
: 1952 3121 7
: 1953 3122 7
: 1954 3123 7
: 1955 3124 7
: 1956 3125 7
: 1957 3126 7
: 1958 3127 7
: 1959 3128 7
: 1960 3129 7
: 1961 3130 6
: 1962 3131 6
: 1963 3132 6
: 1964 3133 6
: 1965 3134 6
: 1966 3135 6 ! We did not find a BFRL in the cache already, so take one off
: 1967 3136 6 ! the free list and use it.
: 1968 3137 6
: 1969 3138 7
: 1970 3139 7
: 1971 3140 7

      BEGIN
      BFRL = BFRD_ADDR (.BFRD [BFRD$W_BFRL]);
      IF .BFRL [BFRL$S_LCKBASIS] NEQ .LOCKBASIS
      OR .BFRL [BFRL$S_PARLKID] NEQ .PARLKID
      THEN
        BUG_CHECK (XQPERR, 'wrong lockbasis');
      LEAVE FIXBFRL;
      END
      ELSE IF .INDX NEQ 0
      THEN
        BEGIN
        BFRD [BFRD$W_BFRL] = .INDX;
        BFRL [BFRL$W_REFCNT] = .BFRL [BFRL$W_REFCNT] + 1;
        LEAVE FIXBFRL;
        END;
        ! There is no BFRL for this BFRD yet, so let's first see if a BFRL
        ! for this LOCKBASIS is already in the cache.
        LBINDX_ADDR = LOOKUP_LOCK (.LOCKBASIS, .PARLKID);
        INDX = (.LBINDX_ADDR)<0,1c>;
        WHILE .INDX NEQ 0
        DO
          BEGIN
          BFRL = BFRD_ADDR (.INDX);
          IF .BFRL [BFRL$S_LCKBASIS] EQL .LOCKBASIS
          AND .BFRL [BFRL$S_PARLKID] EQL .PARLKID
          THEN
            EXITLOOP;
          INDX = .BFRL [BFRL$W_MATBFRL];
          END;
          IF .INDX EQL 0
          THEN
            ! We did not find a BFRL in the cache already, so take one off
            ! the free list and use it.
            BEGIN
            IF (INDX = .CACHE_HDR [F11BC$W_FREEBFRL]) EQL 0
```

```

1972 3141 7      THEN
1973 3142 7      BUG_CHECK (XQPERR, 'no free lock block');
1974 3143 7
1975 3144 7 ! Take our new BFRL out of the free BFRL list.
1976 3145 7 ! Link it into the front of the hash chain.
1977 3146 7 ! Fill in the lockbasis and parent lock ID fields.
1978 3147 7
1979 3148 7
1980 3149 7      BFRL = BFRD_ADDR (.INDX);
1981 3150 7      CACHE_HDR [FT1BCSW FREEBFRL] = .BFRL [BFRLSW NXTBFRL];
1982 3151 7      BFRL [BFRLSW NXTBFRL] = (.LBINDEX_ADDR)<0,16>;
1983 3152 7      (.LBINDEX_ADDR)<0,16> = .INDX;
1984 3153 7
1985 3154 7      BFRL [BFRLSL_LCKBASIS] = .LOCKBASIS;
1986 3155 7      BFRL [BFRLSL_PARLKID] = .PARLKID;
1987 3156 7
1988 3157 7 ! Test for LOCKBASIS of -1. -1 is used for all buffers backed by the
1989 3158 7 ! volume allocation lock. Because there is a volume allocation lock
1990 3159 7 ! associated with the VCB at all times, there is no need to separately
1991 3160 7 ! keep a lock on those buffers in the cache. However, we will keep
1992 3161 7 ! a BFRL around to avoid a lot of special casing, but it's LKID field
1993 3162 7 ! will be zero.
1994 3163 7
1995 3164 7
1996 3165 7      IF (.LOCKBASIS + 1) NEQ 0
1997 3166 7          AND .BBLOCK [CURRENT_UCB [UCBSL_DEVCHAR2], DEVSV_CLU]
1998 3167 7          AND .CLUSGL_CLUB NEQ 0
1999 3168 7      THEN
2000 3169 7          LKID_ADDR = BFRL [BFRLSL_LKID];
2001 3170 7
2002 3171 6      END;
2003 3172 6
2004 3173 6 ! Fill in the BFRD field that points to the BFRL. Bump the REFCNT
2005 3174 6 ! in the BFRL.
2006 3175 6
2007 3176 6
2008 3177 6      BFRD [BFRD$W_BFRL] = .INDX;
2009 3178 6      BFRL [BFRLSW_REFCNT] = .BFRL [BFRLSW_REFCNT] + 1;
2010 3179 6
2011 3180 5      END;          ! of FIXBFRL block
2012 3181 5
2013 3182 5 ! Store the current sequence number from the backing lock for this buffer.
2014 3183 5 ! This must be done here because multiple buffers may be backed by a
2015 3184 5 ! single lock. If one of them has been modified, the sequence number
2016 3185 5 ! for that lock will have changed, yet unmodified buffers backed by
2017 3186 5 ! the same do not have the same sequence number, even though they
2018 3187 5 ! are still valid. Hence, stamp all buffers with the current sequence
2019 3188 5 ! number.
2020 3189 5
2021 3190 5      CASE .BFRD [BFRD$B_BTYPE] FROM 0 TO 6 OF
2022 3191 5      SET
2023 3192 5      [HEADER TYPE]:
2024 3193 5          BFRD [BFRD$SL_SEQNUM] = .LB_HDRSEQ [.LCKINDEX];
2025 3194 5
2026 3195 5      [DIRECTORY TYPE, DATA TYPE]:
2027 3196 5          BFRD [BFRD$SL_SEQNUM] = .LB_DATASEQ [.LCKINDEX];
2028 3197 5

```

```

2029      3198 5      [DIRINDX TYPE]:
2030      3199 6      BEGIN
2031      3200 6      LOCAL
2032      3201 6      DIRINDX      : REF BBLOCK FIELD (DIRC);
2033      3202 6
2034      3203 6      BFRD [BFRD$$_SEQNUM] = .LB_HDRSEQ [.LCKINDX];
2035      3204 6      DIRINDX = .CACHE_HDR [F11BC$_BUFBASE]
2036      3205 6      + $12*(.BFRD - .CACHE_HDR [F11BC$_BFRDBAS])/BFRD$$_BFRDDEF;
2037      3206 6      DIRINDX [DIRC$_DATASEQ] = .LB_DATASEQ [.LCKINDX];
2038      3207 5      END;
2039      3208 5
2040      3209 5      [BITMAP TYPE]:
2041      3210 5      BFRD [BFRD$_SEQNUM] = .(LB_DATASEQ [0])<0,16,0>;
2042      3211 5
2043      3212 5      [INDEX TYPE]:
2044      3213 5      BFRD [BFRD$_SEQNUM] = .(LB_DATASEQ [0])<16,16,0>;
2045      3214 5
2046      3215 5      [QUOTA TYPE]:
2047      3216 5      BFRD [BFRD$_SEQNUM] = .SAVE_VC_FLAGS<1,15,0>;
2048      3217 5
2049      3218 5      TES;
2050      3219 5
2051      3220 5
2052      3221 5      ! Take the buffer off of the in-process queue and put it back
2053      3222 5      ! on the appropriate pool LRU list in the cache.
2054      3223 5      ! Clear out CURPID in the BFRD to note we aren't using it anymore.
2055      3224 5
2056      3225 5
2057      3226 5      REMQUE (.BFRD, BFRD);
2058      3227 5      BFRS_USED [.POOL] = .BFRS_USED [.POOL] - 1;
2059      3228 5      BFRD [BFRD$_CURPID] = 0;
2060      3229 5      POOL_LRU = CACHE_HDR [F11BC$_POOL_LRU] + .POOL*8;
2061      3230 5      INSQUE (.BFRD, .POOL_LRU [QBLK]);
2062      3231 4      END;      ! of LOCKBASIS match.
2063      3232 4
2064      3233 4      BFRD = .NXTBFRD;
2065      3234 4
2066      3235 3      END;      ! of loop through pool bfrds.
2067      3236 3
2068      3237 3      IF .QHEAD NEQA .BFRD
2069      3238 3      THEN
2070      3239 3      BUG_CHECK (XQPERR, 'screwed up in-process list');
2071      3240 3
2072      3241 2      END;      ! of loop through all pcals
2073      3242 2
2074      3243 2      IF .LKID_ADDR EQL 0
2075      3244 2      THEN
2076      3245 2      RELEASE_CACHE ();
2077      3246 2
2078      3247 2      RETURN .LKID_ADDR
2079      3248 2
2080      3249 1      END;

```


		51	67	D0	000B6	MOVL	(R7), R1	3115	
	50	59	55	C1	000B9	ADDL3	PARLKID, LOCKBASIS, R0		
		58	24	A1	3C 000BD	MOVZWL	36(R1), R8		
7E	00	50	01	7A	000C1	EMIJL	#1, R0, #0, -(SP)		
50	50	8E	58	7B	000C6	EDIV	R8, (SP)+, R0, R0		
			50	D5	000CB	TSTL	R0		
			03	18	000CD	BGEQ	12\$		
		50	50	CE	000CF	MNEGL	R0, R0		
		58	20	B140	3E 000D2	MOVAV	32(R1)[R0], LBINDX_ADDR		
		56	68	3C	000D7	MOVZWL	(LBINDX_ADDR), INDX	3116	
			20	13	000DA	BEQL	15\$	3118	
	51	50	67	D0	000DC	MOVL	(R7), R0	3122	
		56	04	78	000DF	ASHL	#4, INDX, R1		
		51	1C	A0	C0 000E3	ADDL2	28(R0), R1		
		52	FO	A1	9E 000E7	MOVAB	-16(R1), BFRL		
		59	08	A2	D1 000EB	CMPL	8(BFRL), LOCKBASIS	3124	
			06	12	000EF	BNEQ	14\$		
		55	OC	A2	D1 000F1	CMPL	12(BFRL), PARLKID	3125	
			05	13	000F5	BEQL	15\$		
		56	62	3C	000F7	MOVZWL	(BFRL), INDX	3129	
			DE	11	000FA	BRB	13\$	3118	
			56	D5	000FC	TSTL	INDX	3132	
			4C	12	000FE	BNEQ	18\$		
		50	67	D0	00100	MOVL	(R7), R0	3140	
		56	26	A0	3C 00103	MOVZWL	38(R0), INDX		
			04	12	00107	BNEC	17\$		
				FEFF	00109	BUGW		3142	
				0000*	0010B	.WORD	<BUGS_XQPERR!4>		
		50	67	D0	0010D	MOVL	(R7), R0	3149	
	51	56	04	78	00110	ASHL	#4, INDX, R1		
		51	1C	A0	C0 00114	ADDL2	28(R0), R1		
		52	FO	A1	9E 00118	MOVAB	-16(R1), BFRL		
		51	67	D0	0011C	MOVL	(R7), R1	3150	
	26	A1	62	B0	0011F	MOVW	(BFRL), 38(R1)		
		62	68	B0	00123	MOVW	(LBINDX_ADDR), (BFRL)	3151	
		68	56	B0	00126	MOVW	INDX, (LBINDX_ADDR)	3152	
	08	A2	59	D0	00129	MOVL	LOCKBASIS, 8(BFRL)	3154	
	0C	A2	55	D0	0012D	MOVL	PARLKID, 12(BFRL)	3155	
		50	01	A9	9E 00131	MOVAB	1(R9), R0	3165	
			15	13	00135	BEQL	18\$		
		50	94	AA	D0 00137	MOVL	-108(BASE), R0	3166	
		0D	3C	A0	E9 0013B	BLBC	60(R0), 18\$		
			00000000G	00	D5 0013F	TSTL	CLUSGL_CLUG	3167	
			05	13	00145	BEQL	18\$		
	0C	AE	04	A2	9E 00147	MOVAB	4(R2), LKID_ADDR	3160	
	1A	A4	56	B0	0014C	MOVW	INDX, 26(BFRD)	3177	
			02	A2	B6 00150	INCW	2(BFRL)	3178	
		51	14	A4	9E 00153	MOVAB	20(BFRD), R1	3193	
	06	00	19	A4	8F 00157	CASEB	25(BFRD), #0, #6	3190	
0050	001A	004B	000E	0015C	20\$:	.WORD	21\$-20\$,-		
	0024	0056	001A	00164			24\$-20\$,-		
							22\$-20\$,-		
							25\$-20\$,-		
							22\$-20\$,-		
							26\$-20\$,-		
							23\$-20\$		
		50	04	AC	D0 0016A	21\$:	MOVL	LCKINDX, R0	3193

	61	0094	CA40	DO	0016E	MOVL	148(BASE)[R0], (R1)	
			42	11	00174	BRB	27\$	
	50	04	AC	DO	00176	22\$:	MOVL	LCKINDX, R0
	61		6B40	DO	0017A		MOVL	(R11)[R0], (R1)
			38	11	0017E		BRB	27\$
	50	04	AC	DO	00180	23\$:	MOVL	LCKINDX, R0
	61	0094	CA40	DO	00184		MOVL	148(BASE)[R0], (R1)
	50		67	DO	0018A		MOVL	(R7), R0
	54	18	A0	C3	0018D		SUBL3	24(R0), BFRD, R1
51			09	78	00192		ASHL	#9, R1, R1
51			20	C6	00196		DIVL2	#32, R1
	51		60	C0	00199		ADDL2	(R0), DIRINDX
	50	04	AC	DO	0019C		MOVL	LCKINDX, R0
08	A1		6B40	DO	001A0		MOVL	(R11)[R0], 8(DIRINDX)
			11	11	001A5		BRB	27\$
	61		6B	3C	001A7	24\$:	MOVZWL	(R11), (R1)
			0C	11	001AA		BRB	27\$
	61	02	AB	3C	001AC	25\$:	MOVZWL	2(R11), (R1)
			06	11	001B0		BRB	27\$
61	A4	AA	0F	EF	001B2	26\$:	EXTZV	#1, #15, -92(BASE), (R1)
			54	OF	001B8	27\$:	REMQUE	(BFRD), BFRD
			F4	AA43	B7	001BB	DECW	-12(BASE)[POOL]
			1C	A4	B4	001BF	CLRW	28(BFRD)
	50	00	B743	7E	001C2		MOVAQ	@0(R7)[POOL], POOL_LRU
	50		28	C0	001C7		ADDL2	#40, POOL_LRU
04	B0		64	0E	001CA		INSQUE	(BFRD), @4(POOL_LRU)
	54		6E	DO	001CE	28\$:	MOVL	NXTBFRD, BFRD
			FE5E	31	001D1		BRW	2\$
	54	08	AE	D1	001D4	29\$:	CMP	QHEAD, BFRD
			04	13	001D8		BEQL	30\$
			FEFF	001DA		BUGW		3239
			0000*	001DC		.WORD	<BUGS_XOPERR!4>	
FE3E			03	F1	001DE	30\$:	ACBL	#3, #T, POOL, 1\$
			0C	AE	D5	001E4	TSTL	LKID_ADDR
			03	12	001E7		BNEQ	31\$
			F96B	30	001E9		BSBW	RELEASE CACHE
	50	0C	AE	DO	001EC	31\$:	MOVL	LKID_ADDR, R0
			04	001FO		RET		3247
								3249

; Routine Size: 497 bytes, Routine Base: \$CODE\$ + 0B80

```
2082 3250 1 GLOBAL ROUTINE RESET_LBN (BUFFER, LBN) : L_NORM NOVALUE =
2083 3251 1
2084 3252 1 !++
2085 3253 1
2086 3254 1 FUNCTIONAL DESCRIPTION:
2087 3255 1
2088 3256 1 This routine changes the resident LBN of the indicated block
2089 3257 1 and marks it as dirty and valid.
2090 3258 1 It is assumed this will be followed soon by a buffer write
2091 3259 1 operation for the new lbn.
2092 3260 1
2093 3261 1 CALLING SEQUENCE:
2094 3262 1 RESET_LBN (ARG1, ARG2)
2095 3263 1
2096 3264 1 INPUT PARAMETERS:
2097 3265 1 ARG1: address of block buffer
2098 3266 1 ARG2: new LBN
2099 3267 1
2100 3268 1 IMPLICIT INPUTS:
2101 3269 1 buffer descriptor arrays
2102 3270 1
2103 3271 1 OUTPUT PARAMETERS:
2104 3272 1 NONE
2105 3273 1
2106 3274 1 IMPLICIT OUTPUTS:
2107 3275 1 NONE
2108 3276 1
2109 3277 1 ROUTINE VALUE:
2110 3278 1 NONE
2111 3279 1
2112 3280 1 SIDE EFFECTS:
2113 3281 1 backing LBN for buffer altered
2114 3282 1 buffer marked as dirty and valid.
2115 3283 1
2116 3284 1 --
2117 3285 1
2118 3286 2 BEGIN
2119 3287 2
2120 3288 2 BIND_COMMON;
2121 3289 2
2122 3290 2 LOCAL
2123 3291 2 INDX,
2124 3292 2 BFRD : REF BBLOCK,
2125 3293 2 INDX_ADDR;
2126 3294 2
2127 3295 2 INDX = (.BUFFER - .CACHE_HDR [F11BC$L_BUFBASE])/512 + 1;
2128 3296 2
2129 3297 2 BFRD = BFRD_ADDR (.INDX);
2130 3298 2
2131 3299 2 SERIAL_CACHE ();
2132 3300 2
2133 3301 2 IF .BFRD [BFRD$L_LBN] NEQ 0
2134 3302 2 THEN
2135 3303 2
2136 3304 2 ! BFRD$L_LBN neq 0 means that this buffer is in the hash lookup list.
2137 3305 2 ! We need to remove it from the list it is currently
2138 3306 2 ! in prior to resetting the LBN field.
```

```

: 2139      3307 2 !
: 2140      3308 2 !
: 2141      3309 2 !
: 2142      3310 2 !
: 2143      3311 2 !
: 2144      3312 2 !
: 2145      3313 2 !
: 2146      3314 2 !
: 2147      3315 2 !
: 2148      3316 2 !
: 2149      3317 2 !
: 2150      3318 2 !
: 2151      3319 2 !
: 2152      3320 2 !
: 2153      3321 2 !
: 2154      3322 2 !
: 2155      3323 2 !
: 2156      3324 2 !
: 2157      3325 2 !
: 2158      3326 2 !
: 2159      3327 2 !
: 2160      3328 2 !
: 2161      3329 2 !
: 2162      3330 2 !
: 2163      3331 2 !
: 2164      3332 2 !
: 2165      3333 2 !
: 2166      3334 2 !
: 2167      3335 2 !
: 2168      3336 2 !
: 2169      3337 2 !
: 2170      3338 2 !
: 2171      3339 2 !
: 2172      3340 1 !

BEGIN
INDX_ADDR = LOOKUP_LBN (.BFRD [BFRD$L_LBN]);
IF (.INDX_ADDR)<0,16> EQL .INDX
THEN
  (.INDX_ADDR)<0,16> = .BFRD [BFRD$W_NXTBFRD]
ELSE
  BEGIN
  DO
    INDX_ADDR = BBLOCK [ BFRD_ADDR (.INDX_ADDR)<0,16>, BFRD$W_NXTBFRD]
  UNTIL (.INDX_ADDR)<0,16> EQL .INDX;
  (.INDX_ADDR)<0,16> = .BFRD [BFRD$W_NXTBFRD];
  END;
END;

! Store the desired LBN and hook it up in the LBN hash list.
BFRD [BFRD$L_LBN] = .LBN;
INDX_ADDR = LOOKUP_LBN (.LBN);
BFRD [BFRD$W_NXTBFRD] = (.INDX_ADDR)<0,16>;
(.INDX_ADDR)<0,16> = .INDX;
BFRD [BFRD$V_VALID] = 1;
BFRD [BFRD$V_DIRTY] = 1;
RELEASE_CACHE ();
END;

```

! end of routine RESET_LBN

					003C 0000	.ENTRY	RESET_LBN, Save R2,R3,R4,R5	3250
		55	FC	AA	9E 00002	MOVAB	-4(BASE), R5	3286
52	04	AC	00	B5	C3 00006	SUBL3	20(R5), BUFFER, R2	3295
		52	00000200	8F	C6 0000C	DIVL2	#512, R2	
		53	01	A2	9E 00013	MOVAB	1(R2), INDX	
		50		65	D0 00017	MOVL	(R5), R0	3297
52		53		05	78 0001A	ASHL	#5, INDX, R2	
		52	18	A0	C0 0001E	ADDL2	24(R0), R2	
		52		20	C2 00022	SUBL2	#32, BFRD	
				F48D	30 00025	BSBW	SERIAL CACHE	3299
			08	A2	D5 00028	TSTL	8(BFRD)	3301
				41	13 0002B	BEQL	48	
		51		65	D0 0002D	MOVL	(R5), R1	3311
		50	14	A1	3C 00030	MOVZWL	20(R1), R0	
7E	00	08	A2	01	7A 00034	EMUL	#1, 8(BFRD), #0, -(SP)	
50	50	8E		50	7B 0003A	EDIV	R0, (SP)+, R0, R0	
				50	D5 0003F	TSTL	R0	
				03	18 00041	BGEQ	18	

		50		50	CE 00043		MNEGL	R0, R0		
		50		10 B140	3E 00046	18:	MOVAW	@16(R1)[R0], INDX_ADDR		
53	60	10			0C ED 0004B		CMPZV	#0, #16, (INDX_ADDR), INDX		3313
					18 13 00050		BEQL	3\$		
		51			65 D0 00052	28:	MOVL	(R5), R1		3319
		54			60 3C 00055		MOVZWL	(INDX_ADDR), R4		
		54			20 C4 00058		MULL2	#32, R4		
		54		18	A1 C0 0005B		ADDL2	24(R1), R4		
53	60	50		FE	A4 9E 0005F		MOVAB	-2(R4), INDX_ADDR		
		10			00 ED 00063		CMPZV	#0, #16, (INDX_ADDR), INDX		3320
					E8 12 00068		BNEQ	2\$		
		60		1E	A2 B0 0006A	3\$:	MOVW	30(BFRD), (INDX_ADDR)		3322
		08		08	AC D0 0006E	4\$:	MOVL	LBN, 8(BFRD)		3329
		54			65 D0 00073		MOVL	(R5), R4		3331
		51		14	A4 3C 00076		MOVZWL	20(R4), R1		
7E	00	08			01 7A 0007A		EMUL	#1, LBN, #0, -(SP)		
51	51				51 7B 00080		EDIV	R1, (SP)+, R1, R1		
					51 D5 00085		TSTL	R1		
					03 18 00087		BGEQ	5\$		
		51			51 CE 00089		MNEGL	R1, R1		
		50		10 B441	3E 0008C	5\$:	MOVAW	@16(R4)[R1], INDX_ADDR		3332
		1E			60 B0 00091		MOVW	(INDX_ADDR), 30(BFRD)		3333
		60			53 B0 0C095		MOVW	INDX, (INDX_ADDR)		3336
		18			0C 88 00098		BISB2	#12, 24(BFRD)		3338
					F8C7 3C 0009C		BSBW	RELEASE_CACHE		3340
					04 0009F		RET			

; Routine Size: 160 bytes, Routine Base: \$CODE\$ + 0DA1

```

: 2174 3341 1 GLOBAL ROUTINE MARK_DIRTY (BUFFER) : L_NORM NOVALUE =
2175 3342 1
: 2176 3343 1 !++
2177 3344 1
: 2178 3345 1 FUNCTIONAL DESCRIPTION:
2179 3346 1
2180 3347 1 This routine marks the indicated buffer for write-back.
2181 3348 1
: 2182 3349 1 CALLING SEQUENCE:
2183 3350 1 MARK_DIRTY (ARG1)
2184 3351 1
: 2185 3352 1 INPUT PARAMETERS:
2186 3353 1 ARG1: address of block buffer
2187 3354 1
: 2188 3355 1 IMPLICIT INPUTS:
2189 3356 1 CURRENT_VCB: VCB of volume
2190 3357 1
: 2191 3358 1 OUTPUT PARAMETERS:
2192 3359 1 NONE
2193 3360 1
: 2194 3361 1 IMPLICIT OUTPUTS:
2195 3362 1 NONE
2196 3363 1
: 2197 3364 1 ROUTINE VALUE:
2198 3365 1 NONE
2199 3366 1
: 2200 3367 1 SIDE EFFECTS:
2201 3368 1 buffer marked for write-back
2202 3369 1
: 2203 3370 1 --
2204 3371 1
: 2205 3372 2 BEGIN
2206 3373 2
: 2207 3374 2 BIND_COMMON;
2208 3375 2
: 2209 3376 2 LOCAL
2210 3377 2 BFRD : REF BBLOCK,
2211 3378 2 INDX0;
2212 3379 2
: 2213 3380 2 INDX0 = (.BUFFER - .CACHE_HDR [F11BC$$_BUFBASE])/512;
2214 3381 2
: 2215 3382 2 IF .INDX0<0,16> GEQU .CACHE_HDR [F11BC$$_BFRCNT]
2216 3383 2 THEN
2217 3384 2 BUG_CHECK (BADBUFADR, 'buffer addr out of range');
2218 3385 2
: 2219 3386 2 BFRD = .CACHE_HDR [F11BC$$_BFRD$$_BAS] + .INDX0*BFRD$$_BFRDDEF;
2220 3387 2
: 2221 3388 2 IF NOT .BFRD [BFRD$$_VALID] OR .BFRD [BFRD$$_CURPID] EQL 0
2222 3389 2 THEN
2223 3390 2 BUG_CHECK (XOPERR, 'expect it to be valid and in-process');
2224 3391 2
: 2225 3392 2 BFRD [BFRD$$_DIRTY] = 1;
2226 3393 2
: 2227 3394 1 END; ! end of routine MARK_DIRTY
```

50	04	AC	FC	0000	00000	.ENTRY	MARK DIRTY. Save nothing	3341
		50	00000200	BA	C3 00002	SUBL3	@-4(BASE), BUFFER, R0	3380
		51	FC	8F	C6 00008	DIVL2	#512, INDX0	
	16	A1		AA	D0 0000F	MOVL	-4(BASE), R1	3382
				50	B1 00013	CMPW	INDX0, 22(R1)	
				04	1F 00017	BLSSU	1\$	
					FEFF 00019	BUGW		3384
					0000* 0001B	.WORD	<BUG\$ BADBUFADR!4>	
		51	FC	AA	D0 0001D	MOVL	-4(BASE), R1	3386
		50		20	C4 00021	MULL2	#32, R0	
		50	18	A1	C0 00024	ADDL2	24(R1), BFRD	
05	18	A0		03	E1 00028	RBC	#3, 24(BFRD), 2\$	3388
			1C	A0	B5 0002D	TSTW	28(BFRD)	
				04	12 00030	BNEQ	3\$	
					FEFF 00032	BUGW		3390
					0000* 00034	.WORD	<BUG\$ XQPERR!4>	
	18	A0		04	88 00036	BISB2	#4, 24(BFRD)	3392
				04	0003A	RET		3394

: Routine Size: 59 bytes, Routine Base: \$CODE\$ + 0E41

```

: 2229      3395 1 ROUTINE RETURN_BFRD (BFRD) : L_JSB_1ARG NOVALUE =
: 2230      3396 1
: 2231      3397 1 |++
: 2232      3398 1 |
: 2233      3399 1 | This routine completes the return of the given BFRD to the appropriate
: 2234      3400 1 | pool in the cache. It expects that it has already been unhooked from
: 2235      3401 1 | the hash lists and the BFRD taken care of.
: 2236      3402 1 |
: 2237      3403 1 |--
: 2238      3404 1
: 2239      3405 2 BEGIN
: 2240      3406 2
: 2241      3407 2 MAP
: 2242      3408 2     BFRD : REF BBLOCK;
: 2243      3409 2
: 2244      3410 2 BIND_COMMON;
: 2245      3411 2
: 2246      3412 2 LOCAL
: 2247      3413 2     POOL,
: 2248      3414 2     POOL_LRU : REF BBLOCK;
: 2249      3415 2
: 2250      3416 2
: 2251      3417 2 POOL = .BFRD [BFRD$V_POOL];
: 2252      3418 2 POOL_LRU = CACHE_HDR[F11B($Q_POOL_LRU) + .POOL*8;
: 2253      3419 2
: 2254      3420 2 ! Credit our buffers in use count if this was on our in-process list.
: 2255      3421 2 !
: 2256      3422 2
: 2257      3423 2 IF .BFRD [BFRD$W_CURPID] NEQ 0
: 2258      3424 2 THEN
: 2259      3425 2     BFRS_USED [.POOL] = .BFRS_USED [.POOL] - 1;
: 2260      3426 2
: 2261      3427 2 BFRD [BFRD$L_LBN] = 0;
: 2262      3428 2 BFRD [BFRD$L_UCB] = 0;
: 2263      3429 2 BFRD [BFRD$L_LOCKBASIS] = 0;
: 2264      3430 2 BFRD [BFRD$L_SEQNUM] = 0;
: 2265      3431 2 BFRD [BFRD$W_CURPID] = 0;
: 2266      3432 2 BFRD [BFRD$W_NXTBFRD] = 0;
: 2267      3433 2
: 2268      3434 2 ! Remove from either the in-process list or it's current position in
: 2269      3435 2 ! the cache LRU (whether curpid was zero or not) and move to the head
: 2270      3436 2 ! of the cache LRU list.
: 2271      3437 2 !
: 2272      3438 2
: 2273      3439 2 REMQUE (.BFRD, BFRD);
: 2274      3440 2 INSQUE (.BFRD, POOL_LRU [QFLNR]);
: 2275      3441 2
: 2276      3442 1 END;

```

```

51      18  A0      02      00  EF 00002      PUSHL  R2      : 3395
          52      FC BA41 7E 00008      EXTZV  #0, #2, 24(BFRD), POOL : 3417
          : 3418

```

RDBLOK
V04-005

F 4
8-Jan-1985 18:20:35
2-Oct-1984 12:43:36

VAX-11 Bliss-32 V4.0-742
[F11X.BUGSRC]RDBLOK.B32;1

Page 75
(19)

52	1C	28	C0	0000D	ADDL2	#40, POOL_LRU	:	
		A0	B5	00010	TSTW	28(BFRD)	:	3423
		04	13	00013	BEQ	1\$:	
	F4	AA	41	B7	00015	DECW	-12(BASE)[PCOL]	3425
	08	A0	7C	00019	1\$:	CLRQ	8(BFRD)	3427
	10	A0	7C	0001C		CLRQ	16(BFRD)	3429
	1C	A0	D4	0001F		CLRL	28(BFRD)	3431
50		60	0F	00022		REMQUE	(BFRD), BFRD	3439
62		60	0E	00025		INSQUE	(BFRD), (POOL_LRU)	3440
		04	BA	00028		POPR	#*M<R2>	3442
		05	00	0002A		RSB		

; Routine Size: 43 bytes, Routine Base: \$CODE\$ + 0E7C

```
: 2278 3443 1 GLOBAL ROUTINE INVALIDATE (BUFFER) : L_NORM NOVALUE =
: 2279 3444 1
: 2280 3445 1 ++
: 2281 3446 1
: 2282 3447 1 FUNCTIONAL DESCRIPTION:
: 2283 3448 1
: 2284 3449 1 This routine invalidates the indicated buffer.
: 2285 3450 1
: 2286 3451 1 CALLING SEQUENCE:
: 2287 3452 1 INVALIDATE (ARG1)
: 2288 3453 1
: 2289 3454 1 INPUT PARAMETERS:
: 2290 3455 1 ARG1: address of block buffer
: 2291 3456 1
: 2292 3457 1 IMPLICIT INPUTS:
: 2293 3458 1 NONE
: 2294 3459 1
: 2295 3460 1 OUTPUT PARAMETERS:
: 2296 3461 1 NONE
: 2297 3462 1
: 2298 3463 1 IMPLICIT OUTPUTS:
: 2299 3464 1 NONE
: 2300 3465 1
: 2301 3466 1 ROUTINE VALUE:
: 2302 3467 1 NONE
: 2303 3468 1
: 2304 3469 1 SIDE EFFECTS:
: 2305 3470 1 buffer contents forgotten
: 2306 3471 1
: 2307 3472 1 --
: 2308 3473 1
: 2309 3474 2 BEGIN
: 2310 3475 2
: 2311 3476 2 BIND_COMMON;
: 2312 3477 2
: 2313 3478 2 LOCAL
: 2314 3479 2 POOL,
: 2315 3480 2 BFRD : REF BBLOCK;
: 2316 3481 2
: 2317 3482 2 BFRD = ((.BUFFER - .CACHE_HDR [F11BC$$_BUFBASE])/512)*BFRD$$_BFRDDEF
: 2318 3483 2 + .CACHE_HDR [F11BC$$_BFRDBAS];
: 2319 3484 2
: 2320 3485 2 BFRD [BFRD$$_DIRTY] = 0;
: 2321 3486 2 BFRD [BFRD$$_VALID] = 0;
: 2322 3487 2
: 2323 3488 2 IF .BFRD [BFRD$$_CURPID] NEQ .(CTL$GL_PCB [PCBS$$_PID])<0,16>
: 2324 3489 2 THEN
: 2325 3490 2 BUG_CHECK (XQPFRR, 'expected this to be in-process');
: 2326 3491 2
: 2327 3492 2 ! Pull this buffer out of it's current position and place it at the
: 2328 3493 2 ! front of the list so it will be the next buffer tossed off of our
: 2329 3494 2 ! in-process list.
: 2330 3495 2
: 2331 3496 2
: 2332 3497 2 POOL = .BFRD [BFRD$$_POOL];
: 2333 3498 2 REMQUE (.BFRD, BFRD);
: 2334 3499 2 INSQUE (.BFRD, BFR_LIST [.POOL, QFLNK]);
```

```

: 2335      3500  2
: 2336      3501  2 IF .BUFFER EQL .BITMAP_BUFFER
: 2337      3502  2 THEN
: 2338      3503  2     BITMAP_VBN = 0;
: 2339      3504  2
: 2340      3505  1 END;

```

! end of routine INVALIDATE

					0000 00000		.ENTRY	INVALIDATE, Save nothing		3443
		50	FC	AA	D0	00002	MOVL	-4(BASE), R0		3482
51	04	AC		60	C3	00006	SUBL3	(R0), BUFFER, R1		
		51	00000200	8F	C6	0000B	DIVL2	#512, R1		
		51		20	C4	00012	MULL2	#32, R1		
		51	18	A0	C0	00015	ADDL2	24(R0), BFRD		3483
	18	A1		0C	8A	00019	BICB2	#12, 24(BFRD)		3486
		50	00000000G	00	D0	0001D	MOVL	CTL\$GL PCB, R0		3488
	60	A0	1C	A1	B1	00024	CMPW	28(BFRD), 96(R0)		
				04	13	00029	BEQL	1\$		
					FEFF	0002B	BUGW			3490
					0000*	0002D	.WORD	<BUG\$ XQPERR!4>		
50	18	A1		02	00	EF	EXTZV	#0, #2, 24(BFRD), POOL		3497
		51		61	0F	00035	REMQUE	(BFRD), BFRD		3498
		50	CC	AA40	7E	00038	MJVAQ	-52(BASE)[POOL], R0		3499
		60		61	0E	0003D	INSQUE	(BFRD), (R0)		
	BC	AA	04	AC	D1	00040	CMP	BUFFER, -68(BASE)		3501
				03	12	00045	BNEQ	2\$		
			B4	AA	D4	00047	CLRL	-76(BASE)		3503
				04	0004A	2\$:	RET			3505

; Routine Size: 75 bytes, Routine Base: \$CODE\$ + 0EA7

```

2342 3506 1 GLOBAL ROUTINE WRITE_HEADER : L_NORM NOVALUE =
2343 3507 1
2344 3508 1 !++
2345 3509 1
2346 3510 1 FUNCTIONAL DESCRIPTION:
2347 3511 1
2348 3512 1 This routine writes out the currently resident file header.
2349 3513 1
2350 3514 1 CALLING SEQUENCE:
2351 3515 1 WRITE_HEADER ()
2352 3516 1
2353 3517 1 INPUT PARAMETERS:
2354 3518 1 NONE
2355 3519 1
2356 3520 1 IMPLICIT INPUTS:
2357 3521 1 FILE_HEADER: address of current file header
2358 3522 1
2359 3523 1 OUTPUT PARAMETERS:
2360 3524 1 NONE
2361 3525 1
2362 3526 1 IMPLICIT OUTPUTS:
2363 3527 1 IO_STATUS: status of I/O transfer
2364 3528 1
2365 3529 1 ROUTINE VALUE:
2366 3530 1 NONE
2367 3531 1
2368 3532 1 SIDE EFFECTS:
2369 3533 1 checksum checked, header written
2370 3534 1
2371 3535 1 --
2372 3536 1
2373 3537 2 BEGIN
2374 3538 2
2375 3539 2 BIND_COMMON;
2376 3540 2
2377 3541 2 EXTERNAL ROUTINE
2378 3542 2 CHECKSUM : L_NORM; ! compute file header checksum
2379 3543 2
2380 3544 2
2381 3545 2 ! The checksum of the header should be good, since all routines that modify
2382 3546 2 ! the header bless it with a new checksum when they are finished. Check the
2383 3547 2 ! checksum and write the header.
2384 3548 2
2385 3549 2
2386 3550 2 IF NOT CHECKSUM (.FILE_HEADER)
2387 3551 2 THEN BUG_CHECK (WRTINVHDR, FATAL, 'ACP attempted to write an invalid file header');
2388 3552 2
2389 3553 2 WRITE_BLOCK (.FILE_HEADER);
2390 3554 2
2391 3555 1 END; ! end of routine WRITE_HEADER

```

```

04 AA DD 0002

```

```

.EXTRN CHECKSUM, BUG$WRTINVHDR
.ENTRY WRITE_HEADER, Save nothing
PUSHL 4(BASE)

```

```

: 3506
: 3550

```



```
2393 3556 1 GLOBAL ROUTINE WRITE_DIRTY (LOCKBASIS) : L_NORM NOVALUE =
2394 3557 1
2395 3558 1 !++
2396 3559 1
2397 3560 1 FUNCTIONAL DESCRIPTION:
2398 3561 1
2399 3562 1 This routine writes all buffers which were modified back to the
2400 3563 1 disk from whence they came.
2401 3564 1
2402 3565 1 INPUT PARAMETERS:
2403 3566 1
2404 3567 1 SIDE EFFECTS:
2405 3568 1 dirty buffers written.
2406 3569 1
2407 3570 1 !--
2408 3571 1
2409 3572 2 BEGIN
2410 3573 2
2411 3574 2 BIND_COMMON;
2412 3575 2
2413 3576 2 INCR POOL FROM 0 TO 2
2414 3577 2 DO
2415 3578 3 BEGIN
2416 3579 3 LOCAL
2417 3580 3 I,
2418 3581 3 QHEAD : REF BBLOCK,
2419 3582 3 BFRD : REF BBLOCK;
2420 3583 3
2421 3584 3 QHEAD = BFR_LIST [.POOL, QFLNK];
2422 3585 3 BFRD = .QHEAD [QFLNK];
2423 3586 3
2424 3587 3 I = .BFRS_USED [.POOL];
2425 3588 3
2426 3589 3 UNTIL .I EQL 0
2427 3590 3 DO
2428 3591 4 BEGIN
2429 3592 4 I = .I - 1;
2430 3593 4
2431 3594 5 IF (.LOCKBASIS EQL 0
2432 3595 5 OR .BFRD [BFRD$L_LOCKBASIS] EQL .LOCKBASIS)
2433 3596 4 AND .BFRD [BFRD$V_DIRTY]
2434 3597 4 THEN
2435 3598 4 WRITE_BLOCK (((.BFRD-.CACHE_HDR [F11BC$L_BFRDBAS])/BFRD$$_BFRDDEF)*512
2436 3599 4 + .CACHE_HDR [F11BC$L_BUFBASE]);
2437 3600 4
2438 3601 4 BFRD = .BFRD [BFRD$L_QFL];
2439 3602 4 END;
2440 3603 3
2441 3604 3 IF .QHEAD NEQA .BFRD
2442 3605 3 THEN
2443 3606 3 BUG_CHECK (XQPERR, 'in-process queue screwed up');
2444 3607 3
2445 3608 2 END;
2446 3609 2
2447 3610 1 END; ! of routine WRITE_DIRTY
```

			003C 00000	.ENTRY	WRITE_DIRTY, Save R2,R3,R4,R5	3556
			53 D4 00002	CLRL	POOL	3576
	54	CC	AA43 7E 00004 1\$:	MOVAQ	-52(BASE)[POOL], QHEAD	3584
	52		64 D0 00009	MOVL	(QHEAD), BFRD	3585
	55	F4	AA43 3C 0000C	MOVZWL	-12(BASE)[POOL], I	3587
			55 D5 00011 2\$:	TSTL	I	3589
			33 13 00013	BEQL	5\$	
			55 D7 00015	DECL	I	3592
		04	AC D5 00017	TSTL	LOCKBASIS	3594
			07 13 0001A	BEQL	3\$	
	04	AC	10 A2 D1 0001C	CML	16(BFRD), LOCKBASIS	3595
			20 12 00021	BNEQ	4\$	
18	18	A2	02 E1 00023 3\$:	BBC	#2, 24(BFRD), 4\$	3596
		S1	FC AA D0 00028	MOVL	-4(BASE), R1	3598
50		S2	18 A1 C3 0002C	SUBL3	24(R1), BFRD, R0	
		S0	20 C6 00031	DIVL2	#32, R0	
50		S0	09 78 00034	ASHL	#9, R0, R0	
		S1	61 D0 00038	MOVL	(R1), R1	3599
			6140 9F 0003B	PUSHAB	(R1)[R0]	
	F900	CF	01 FB 0003E	CALLS	#1, WRITE_BLOCK	
		S2	62 D0 00043 4\$:	MOVL	(BFRD), BFRD	3601
			C9 11 00046	BRB	2\$	3589
		S2	54 D1 00048 5\$:	CML	QHEAD, BFRD	3604
			04 13 0004B	BEQL	6\$	
			FEFF 0004D	BUGW		3606
			0000+ 0004F	.WORD	<BUG\$ XOPERR!4>	
AF		S3	02 F3 00051 6\$:	AOBLEQ	#2, POOL, 1\$	3576
			04 00055	RET		3610

: Routine Size: 86 bytes, Routine Base: \$CODE\$ + 0F0C

```

: 2449      3611 1 GLOBAL ROUTINE TOSS_CACHE_DATA (LCKINDX) : L_NORM NOVALUE =
: 2450      3612 1
: 2451      3613 1 +-
: 2452      3614 1
: 2453      3615 1 FUNCTIONAL DESCRIPTION:
: 2454      3616 1
: 2455      3617 1     This routine invalidates all data blocks associated with
: 2456      3618 1     the given lock basis, after first writing dirty blocks.
: 2457      3619 1     The data sequence number is incremented, which invalidates
: 2458      3620 1     copies of these blocks on other nodes, and blocks found in
: 2459      3621 1     our cache will be marked invalid, causing them to be tossed
: 2460      3622 1     from the cache when the serialization lock is lowered.
: 2461      3623 1     Only the data block pool is scanned.
: 2462      3624 1
: 2463      3625 1 SIDE EFFECTS:
: 2464      3626 1     dirty buffers written, appropriate buffers invalidated
: 2465      3627 1
: 2466      3628 1 --
: 2467      3629 1
: 2468      3630 2 BEGIN
: 2469      3631 2
: 2470      3632 2 BIND_COMMON;
: 2471      3633 2
: 2472      3634 2 LOCAL
: 2473      3635 2     I,
: 2474      3636 2     LOCKBASIS,
: 2475      3637 2     QHEAD      : REF BBLOCK,
: 2476      3638 2     BFRD       : REF BBLOCK;
: 2477      3639 2
: 2478      3640 2 LOCKBASIS = .LB_BASIS [.LCKINDX];
: 2479      3641 2
: 2480      3642 2 QHEAD = BFR_LIST [1, QFLNK];
: 2481      3643 2 BFRD = .QHEAD [QFLNK];
: 2482      3644 2
: 2483      3645 2 I = .BFRS_USED [1];
: 2484      3646 2
: 2485      3647 2 UNTIL .I EQL 0
: 2486      3648 2 DO
: 2487      3649 3     BEGIN
: 2488      3650 3     I = .I - 1;
: 2489      3651 3
: 2490      3652 3     IF .BFRD [BFRDSL_LOCKBASIS] EQL .LOCKBASIS
: 2491      3653 3     THEN
: 2492      3654 4         BEGIN
: 2493      3655 4
: 2494      3656 4             IF .BFRD [BFRDSV_DIRTY]
: 2495      3657 4             THEN
: 2496      3658 4                 WRITE_BLOCK (((.BFRD-.CACHE_HDR [F11BCSL_BFRDBAS])/BFRDSS_BFRDDEF)*512
: 2497      3659 4                 + .CACHE_HDR [F11BCSL_BUFBASE]);
: 2498      3660 4
: 2499      3661 4                 BFRD [BFRDSV_VALID] = 0;
: 2500      3662 3             END;
: 2501      3663 3
: 2502      3664 3     BFRD = .BFRD [BFRDSL_QFL];
: 2503      3665 2     END;
: 2504      3666 2     ? of loop through in-process data block pool
: 2505      3667 2 IF .QHEAD NEQA .BFRD
```

```

: 2506      3668 2 THEN
: 2507      3669 2     BUG_CHECK (XQPERR, 'in-process data block queue screwed up');
: 2508      3670 2
: 2509      3671 2 LB_DATASEQ [.LCKINDX] = .LB_DATASEQ [.LCKINDX] + 1;
: 2510      3672 2
: 2511      3673 1 END;

```

! of routine TOSS_CACHE_DATA

				003C 00000	.ENTRY	TOSS_CACHE_DATA, Save R2,R3,R4,R5		3611
	51	F4	AA	9E 00002	MOVAB	-12(BASE), R1		3630
	50	04	AC	D0 00036	MOVL	LCKINDX, R0		3640
	55	0080	CA40	D0 0000A	MOVL	128(BASE)[R0], LOCKBASIS		
	53	D4	AA	9E 00010	MOVAB	-44(BASE), QHEAD		3642
	52		63	D0 00014	MOVL	(QHEAD), BFRD		3643
	54	02	A1	3C 00017	MOVZWL	2(R1), i		3645
			54	D5 0001B 1\$:	TSTL	i		3647
			31	13 0001D	BEQL	4\$		
			54	D7 0001F	DECL	i		3650
	55	10	A2	D1 00021	CMPL	16(BFRD), LOCKBASIS		3652
			24	12 00025	BNEQ	3\$		
1B	18	A2	02	E1 00027	BBC	# 24(BFRD), 2\$		3656
	51	FC	AA	D0 0002C	MOVL	-4(BASE), R1		3658
50	52	18	A1	C3 00030	SUBL3	24(R1), BFRD, R0		
	50		20	C6 00035	DIVL2	#32, R0		
50	50		09	78 00038	ASHL	#0, R0, R0		
	51		61	D0 0003C	MOVL	(R1), R1		3659
			6140	9F 0003F	PUSHAB	(R1)[R0]		
	F8A6	CF	01	FB 00042	CALLS	#1, WRITE_BLOCK		
	18	A2	08	8A 00047 2\$:	BICB2	#8, 24(BFRD)		3661
	52		62	D0 0004B 3\$:	MOVL	(BFRD), BFRD		3664
			CB	11 0004E	BRB	1\$		3647
	52		53	D1 00050 4\$:	CMPL	QHEAD, BFRD		3667
			04	13 00053	BEQL	5\$		
				FEFF 00055	BUGW			3669
				0000* 00057	.WORD	<BUG\$ XQPERR!4>		
	50	04	AC	D0 00059 5\$:	MOVL	LCKINDX, R0		3671
		00A8	CA40	D6 0005D	INCL	168(BASE)[R0]		
			04	00062	RET			3673

: Routine Size: 99 bytes. Routine Base: \$CODE\$ + 0F62

: 2512 3674 1

```

: 2514      3675 1 GLOBAL ROUTINE KILL_CACHE (UCB) : L_NORM NOVALUE =
: 2515      3676 1
: 2516      3677 1 |++
: 2517      3678 1 |
: 2518      3679 1 | This routine scans through all BFRD's in the cache tossing out
: 2519      3680 1 | all those that match the specified UCB.
: 2520      3681 1 |
: 2521      3682 1 |--
: 2522      3683 1
: 2523      3684 2 BEGIN
: 2524      3685 2
: 2525      3686 2 BIND_COMMON:
: 2526      3687 2
: 2527      3688 2 LOCAL
: 2528      3689 2     BFRD : REF BBLOCK,
: 2529      3690 2     PIDINDX : WORD;
: 2530      3691 2
: 2531      3692 2 BFRD = .CACHE_HDR [F11BC$] BFRDBAS];
: 2532      3693 2 PIDINDX = .CT[$GL_PCB [PCB$] PID];
: 2533      3694 2
: 2534      3695 2 SERIAL_CACHE ();
: 2535      3696 2
: 2536      3697 3 INCR INDX0 FROM 0 TO (.CACHE_HDR [F11BC$] BFRCNT) - 1)
: 2537      3698 2 DO
: 2538      3699 2     BEGIN
: 2539      3700 3
: 2540      3701 3     IF .BFRD [BFRD$] UCB] EQLA .UCB
: 2541      3702 3     THEN
: 2542      3703 4         BEGIN
: 2543      3704 4
: 2544      3705 4             IF .BFRD [BFRD$] LURPID] NEQ 0
: 2545      3706 4             THEN
: 2546      3707 5                 BEGIN
: 2547      3708 5                     IF .BFRD [BFRD$] CURPID] EQL .PIDINDX
: 2548      3709 5                     THEN
: 2549      3710 5                         BFRD [BFRD$] VALID] = 0;           ! it will get tossed on unlock
: 2550      3711 5                     END
: 2551      3712 4                 ELSE
: 2552      3713 5                     BEGIN
: 2553      3714 5                         UNHOOK_BFRD (.INDX0+1, .BFRD);
: 2554      3715 5                         RETURN_BFRD (.BFRD);
: 2555      3716 5                     END;
: 2556      3717 4                 END;
: 2557      3718 3
: 2558      3719 3     BFRD = .BFRD + BFRD$] BFRDDEF];
: 2559      3720 3
: 2560      3721 2     END;
: 2561      3722 2
: 2562      3723 2 RELEASE_CACHE ();
: 2563      3724 2
: 2564      3725 1 END;
```

003C 0000

.ENTRY KILL_CACHE, Save R2,R3,R4,R5

: 3675

50	FC	AA	D0	00002		MOVL	-4(BASE), R0		3692
52	18	A0	D0	00006		MOVL	24(R0), BFRD		
50	00000000G	0C	D0	0000A		MOVL	(TL\$GL_PCB, R0		3693
55	60	A0	B0	00011		MOVW	96(R0), PIDINDX		
		F2A9	0	00015		BSBW	SERIAL_CACHE		3695
50	FC	AA	J0	00018		MOVL	-4(BASE), R0		3697
54	16	A0	3C	0001C		MOVZWL	22(R0), R4		
53			01	CE 00020		MNEGL	#1, INDX0		
			2B	11 00023		BRB	4\$		
04	AC	0C	A2	D1 00025	1\$:	CMPL	12(BFRD), UCB		3701
			21	12 0002A		BNEQ	3\$		
			1C	A2 B5 0002C		TSTW	28(BFRD)		3705
			0C	13 0002F		BEQL	2\$		
			A2	B1 C0 00031		CMPL	28(BFRD), PIDINDX		3708
			16	12 00035		BNEQ	3\$		
18	A2		08	8A 00037		BICB2	#8, 24(BFRD)		3710
			10	11 0003E		BRB	3\$		3705
			52	DD 0003D	2\$:	PUSHL	BFRD		3714
		01	A3	9F 0003F		PUSHAB	1(INDX0)		
FA02	CF		02	FB 00042		CALLS	#2, UNHOOK_BFRD		
	50		52	D0 00047		MOVL	BFRD, R0		3715
			FE6A	30 0004A		BSBW	RETURN_BFRD		
			20	C0 0004D	3\$:	ADDL2	#32, BFRD		3719
D1			54	F2 00050	4\$:	AOBLSS	R4, INDX0, 1\$		3697
			F6EB	30 00054		BSBW	RELEASE_CACHE		3723
			04	00057		RET			3725

: Routine Size: 88 bytes, Routine Base: \$CODE\$ + 0FC5

```

: 2566 3726 1 GLOBAL ROUTINE WRONG_LOCKBASIS (HEADER) : L_NORM NOVALUE =
: 2567 3727 1
: 2568 3728 1 |++
: 2569 3729 1 |
: 2570 3730 1 | This routine returns the given buffer as a result of discovering
: 2571 3731 1 | we had the wrong lockbasis when finding it.
: 2572 3732 1 | If it was read from disk, we must invalidate it, as we stored the
: 2573 3733 1 | wrong lockbasis in its bfrd. If it was found in the cache,
: 2574 3734 1 | simply return it and credit us for returning it.
: 2575 3735 1 |
: 2576 3736 1 |--
: 2577 3737 1
: 2578 3738 2 BEGIN
: 2579 3739 2
: 2580 3740 2 BIND_COMMON;
: 2581 3741 2
: 2582 3742 2 LOCAL
: 2583 3743 2     POOL,
: 2584 3744 2     BFRD      : REF BBLOCK,
: 2585 3745 2     POOL_LRU  : REF BBLOCK;
: 2586 3746 2
: 2587 3747 2 SERIAL_CACHE ();
: 2588 3748 2
: 2589 3749 2 BFRD = ((.HEADER - .CACHE_HDR [F11BC$L_BUFBASE])/512)*BFRD$$_BFRDDEF
: 2590 3750 2     + .CACHE_HDR [F11BC$L_BFRDBAS];
: 2591 3751 2
: 2592 3752 2 IF .BFRD [BFRD$W_BFRL] EQL 0
: 2593 3753 2 THEN
: 2594 3754 2     BFRD [BFRD$V_VALID] = 0
: 2595 3755 2 ELSE
: 2596 3756 2     BEGIN
: 2597 3757 2     POOL = .BFRD [BFRD$V_POOL];
: 2598 3758 2     REMQUE (.BFRD, BFRD);
: 2599 3759 2     BFRS_USED [.POOL] = .BFRS_USED [.POOL] - 1;
: 2600 3760 2     BFRD [BFRD$W_CURPID] = 0;
: 2601 3761 2     POOL_LRU = .CACHE_HDR [F11BC$L_POOL_LRU] + .POOL*8;
: 2602 3762 2     INSQUE (.BFRD, .POOL_LRU [QBLNK]);
: 2603 3763 2     END;
: 2604 3764 2
: 001 3765 2 CDS0016
: 002 3766 2 | Check the ambiguity queue. The fact that we have just realized we
: 003 3767 2 | have the wrong lockbasis means that another process with the correct
: 004 3768 2 | serialization lockbasis may have found the buffer in-process by us
: 005 3769 2 | in the cache and placed himself on the ambiguity queue.
: 006 3770 2 |
: 007 3771 2
: 008 3772 2 IF .CACHE_HDR [F11BC$L_AMBIGQFL] NEQ 0
: 009 3773 2 THEN
: 010 3774 2     BEGIN
: 011 3775 2     LOCAL
: 012 3776 2     AQB      : REF BBLOCK,
: 013 3777 2     WAITER;
: 014 3778 2
: 015 3779 2     AQB = .CURRENT_VCB [VCB$L_AQB];
: 016 3780 2
: 017 3781 2 | Remove the stalled thread from the ambiguity queue.
: 018 3782 2 | Insert it immediately after us, so that it will be awakened by
```



```

:019 CDS0016 3783 3 : release_cache.
:020 CDS0016 3784 3 :
:021 CDS0016 3785 3 :
:022 CDS0016 3786 3 REMQUE (.CACHE_HDR [F11BC$$_AMBIGQFL], WAITER);
:023 CDS0016 3787 3 INSQUE (.WAITER, .AQB [AQB$$_ACPOFL]);
:024 CDS0016 3788 3 :
:025 CDS0016 3789 3 IF .CACHE_HDR [F11BC$$_AMBIGQFL] EQL CACHE_HDR [F11BC$$_AMBIGQFL]
:026 CDS0016 3790 3 THEN
:027 CDS0016 3791 4 BEGIN
:028 CDS0016 3792 4 CACHE_HDR [F11BC$$_AMBIGQFL] = 0;
:029 CDS0016 3793 4 CACHE_HDR [F11BC$$_AMBIGQBL] = 0;
:030 CDS0016 3794 3 END;
:031 CDS0016 3795 3 :
:032 CDS0016 3796 3 :
:033 CDS0016 3797 2 END;
:034 CDS0016 3798 2 :
: 2605 3799 2 RELEASE_CACHE ();
: 2606 3800 1 END;

```

				000C 00000	.ENTRY	WRONG LOCKBASIS, save R2,R3		3726
	53	FC	AA	9E 00002	MOVAB	-4(BASE), R3		3738
			F260	30 00006	BSBW	SERIAL CACHE		3747
	50		63	D0 00009	MOVL	(R3), R0		3749
51	04	AC	60	C3 0000C	SUBL3	(R0), HEADER, R1		
	51	00000200	8F	C6 00011	DIVL2	#512, R1		
	51		20	C4 00018	MULL2	#32, R1		
	51	18	A0	C0 0001B	ADDL2	24(R0), BFRD		3750
		1A	A1	B5 0001F	TSTW	26(BFRD)		3752
			06	12 00022	BNEQ	1\$		
	18	A1	08	8A 00024	BICB2	#8, 24(BFRD)		3754
			1C	11 00028	BRB	2\$		
50	18	A1	00	EF 0002A	EXTZV	#0, #2, 24(BFRD), POOL		3757
	51		61	0F 00030	REMQUE	(BFRD), BFRD		3758
		F4	AA40	B7 00033	DECW	-12(BASE)[POOL]		3759
		1C	A1	B4 00037	CLRW	28(BFRD)		3760
	50	00	B340	7E 0003A	MOVAQ	@0(R3)[POOL], POOL_LRU		3761
	50		28	C0 0003F	ADDL2	#40, POOL_LRU		
	04	B0	61	0E 00042	INSQUE	(BFRD), @4(POOL_LRU)		3762
	50		63	D0 00046	MOVL	(R3), R0		3772
		0080	C0	D5 00049	TSTL	128(R0)		
			2A	13 0004D	BEQL	3\$		
	50	98	AA	D0 0004F	MOVL	-104(BASE), R0		3779
	52	10	A0	D0 00053	MOVL	16(R0), AQB		
	50		63	D0 00057	MOVL	(R3), R0		3786
	51	0080	D0	0F 0005A	REMQUE	@128(R0), WAITER		
	00	B2	61	0E 0005F	INSQUE	(WAITER), @0(AQB)		3787
	50		63	D0 00053	MOVL	(R3), R0		3789
	50	0080	C0	9E 00066	MOVAB	128(R0), R0		
	50		60	D1 0006B	CMPL	(R0), R0		
			09	12 0006E	BNEQ	3\$		
			60	D4 00070	CLRL	(R0)		3792
	50		63	D0 00072	MOVL	(R3), R0		3793
		0084	C0	D4 00075	CLRL	132(R0)		

RDBLOK
V04-005

F 5
8-Jan-1985 18:20:35 VAX-11 Bliss-32 V4.0-742
2-Oct-1984 12:43:36 [F11X.BUGSRC]RDBLOK.B32;1

Page 88
(25)

66E 30 00079 3\$: BSBW RELEASE_CACHE
04 0007C RET

: 3799
: 3800

: Routine Size: 125 bytes, Routine Base: \$CODE\$ + 101D

RW
VO
:
:
:

```

: 2608      3801 1 GLOBAL ROUTINE KILL_BUFFERS (POOL, LOCKBASIS) : L_NORM NOVALUE =
: 2609      3802 1
: 2610      3803 1 !++
: 2611      3804 1
: 2612      3805 1 This routine scans through all BFRD's in the cache tossing out
: 2613      3806 1 all those in the specified POOL for the CURRENT_UCB.
:001 :CDS0017 3807 1 If the directory data (1) or directory index (3) pools are specified,
:002 :CDS0017 3808 1 LOCKBASIS must match also.
:003 :CDS0017 3809 1
:004 :CDS0017 3810 1 Legitimate pool values are 0 through 3.
: 2615-1    3811 1
: 2616      3812 1 --
: 2617      3813 1
: 2618      3814 2 BEGIN
: 2619      3815 2
: 2620      3816 2 BIND_COMMON;
: 2621      3817 2
: 2622      3818 2 LOCAL
:001 :CDS0017 3819 2 UCB,
:002 :CDS0017 3820 2 TSTLOCKBASIS,
: 2623      3821 2 BFRD : REF BBLOCK,
: 2624      3822 2 BFRD CNT : WORD,
: 2625      3823 2 START_BFRD : WORD,
: 2626      3824 2 PIDINDEX : WORD;
: 2627      3825 2
: 2628      3826 3 BEGIN
: 2629      3827 3
: 2630      3828 3 BIND
: 2631      3829 3 POOLCNT = CACHE_HDR [F11BC$W_POOLCNT] : VECTOR [,WORD];
: 2632      3830 3
:001 :CDS0017 3831 3 TSTLOCKBASIS = 0;
:002 :CDS0017 3832 3 UCB = .CURRENT_UCB;
:003 :CDS0017 3833 3
:004 :CDS0017 3834 3 START_BFRD = 0;
:005 :CDS0017 3835 3
:006 :CDS0017 3836 3 IF .POOL GTRU 0
:007 :CDS0017 3837 3 THEN
:008 :CDS0017 3838 4 BEGIN
:009 :CDS0017 3839 4 START_BFRD = .POOLCNT [0];
:010 :CDS0017 3840 4 IF .POOL GTRU 1
:011 :CDS0017 3841 4 THEN
:012 :CDS0017 3842 5 BEGIN
:013 :CDS0017 3843 5 START_BFRD = .START_BFRD + .POOLCNT [1];
:014 :CDS0017 3844 5 IF .POOL EQL 3
:015 :CDS0017 3845 5 THEN
:016 :CDS0017 3846 6 BEGIN
:017 :CDS0017 3847 6 START_BFRD = .START_BFRD + .POOLCNT [2];
:018 :CDS0017 3848 6 TSTLOCKBASIS = 1;
:019 :CDS0017 3849 6 END;
:020 :CDS0017 3850 5 END
:021 :CDS0017 3851 4 ELSE ! This is pool 1.
:022 :CDS0017 3852 4 TSTLOCKBASIS = 1;
: 2641-8    3853 3 END;
: 2642      3854 3
: 2643      3855 3 BFRD_CNT = .POOLCNT [.POOL];
: 2644      3856 2 END;
: 2645      3857 2
```

```

: 2646      3858 2 BFRD = .CACHE HDR [F11BC$L BFRDBAS] + .START_BFRD*BFRD$$_BFRDDEF;
: 2647      3859 2 PIDINDX = .CTC$GL_PCB [PCB$L_PID];
: 2648      3860 2
: 2649      3861 2 SERIAL_CACHE ();
: 2650      3862 2
: 2651      3863 3 INCR INDX0 FROM .START_BFRD TO (.START_BFRD + .BFRD_CNT - 1)
: 2652      3864 2 DO
: 2653      3865 3 BEGIN
: 2654      3866 3
: 001 !CDS0017 3867 3 IF .UCB EQL .BFRD [BFRD$L_UCB]
: 002 .CDS0017 3868 4 AND (NOT .TSTLOCKBASIS OR
: 2657-2      3869 4 .BFRD [BFRD$L_LOCKBASIS] EQL .LOCKBASIS)
: 2658      3870 3 THEN
: 2659      3871 4 BEGIN
: 2660      3872 4
: 2661      3873 4 IF .BFRD [BFRD$W_CURPID] NEQ 0
: 2662      3874 4 THEN
: 2663      3875 5 BEGIN
: 2664      3876 5 IF .BFRD [BFRD$W_CURPID] EQL .PIDINDX
: 2665      3877 5 THEN
: 2666      3878 5 BFRD [BFRD$V_VALID] = 0; ! it will get tossed on unlock
: 2667      3879 5 END
: 2668      3880 4 ELSE
: 2669      3881 5 BEGIN
: 2670      3882 5 UNHOOK_BFRD (.INDX0+1, .BFRD);
: 2671      3883 5 RETURN_BFRD (.BFRD);
: 2672      3884 4 END;
: 2673      3885 3 END;
: 2674      3886 3
: 2675      3887 3 BFRD = .BFRD + BFRD$$_BFRDDEF;
: 2676      3888 3
: 2677      3889 2 END;
: 2678      3890 2
: 2679      3891 2 RELEASE_CACHE ();
: 2680      3892 2
: 2681      3893 1 END;

```

50	FC	AA	00000078	8F	C1	00002	.ENTRY	KILL_BUFFERS, Save R2,R3,R4,R5,R6,R7	3801
				56	D4	0000B	ADDL3	#120, -4(BASE), R0	3829
		57	94	AA	D0	0000D	CLRL	TSTLOCKBASIS	3831
				53	B4	00011	MOVL	-108(BASE), UCB	3833
		51	04	AC	D0	00013	CLRW	START_BFRD	3834
				18	13	00017	MOVL	POOL, R1	3836
		53		60	B0	00019	BEQL	2\$	
		01		51	D1	0001C	MOVW	(R0), START_BFRD	3839
				0D	1B	0001F	CMPL	R1, #1	3840
		53	02	A0	A0	00021	BLEQU	1\$	
		03		51	D1	00025	ADDW2	2(R0), START_BFRD	3843
				07	12	00028	CMPL	R1, #3	3844
		53	04	A0	A0	0002A	BNEQ	2\$	
		56		01	D0	0002E	ADDW2	4(R0), START_BFRD	3847
		54		6041	B0	00031	MOVL	#1, TSTLOCKBASIS	3852
							MOVW	(R0)[R1], BFRD_CNT	3855

	50	FC	AA	D0	00035	MOVL	-4(BASE), R0	3858		
	51		53	3C	00039	MOVZWL	START_BFRD, R1			
	51		20	C4	0003C	MULL2	#32, R1			
52	51	18	A0	C1	0003F	ADDL3	24(R0), R1, BFRD			
	50	00000000G	00	D0	00044	MOVL	CTL\$GL_PCB, R0	3859		
	55	60	A0	B0	0004B	MOVW	96(R0), PIDINDX			
			F19A	30	0004F	BSBW	SERIAL_CACHE	3861		
	50		53	3C	00052	MOVZWL	START_BFRD, R0	3863		
	54		54	3C	00055	MOVZWL	BFRD_CNT, R4			
	54		50	C0	00058	ADDL2	R0, R4			
	53		53	3C	0005B	MOVZWL	START_BFRD, INDX0			
			53	D7	0005E	DECL	INDX0			
			34	11	00060	BRB	7\$			
	0C	A2	57	D1	00062	3\$:	CPL	UCB, 12(BFRD)	3867	
			2B	12	00066	BNEQ	6\$			
	07		56	E9	00068	BLBC	TSTLOCKBASIS, 4\$	3868		
	08	AC	10	A2	D1	0006B	CPL	16(BFRD), LOCKBASIS	3869	
			21	12	00070	BNEQ	6\$			
			1C	A2	B5	00072	4\$:	TSTW	28(BFRD)	3873
			0C	13	00075	BEQL	5\$			
	55	1C	A2	B1	00077	CMPW	28(BFRD), PIDINDX	3876		
			16	12	C007B	BNEQ	6\$			
	18	A2	08	8A	0007D	BICB2	#8, 24(BFRD)	3878		
			10	11	00081	BRB	6\$	3873		
			52	DD	00083	5\$:	PUSHL	BFRD	3882	
			01	A3	9F	00085	PUSHAB	1(INDX0)		
	F8E7	CF	02	FB	00088	CALLS	#2, UNHOOK_BFRD			
		50	52	D0	0008D	MOVL	BFRD, R0	3883		
			FD4F	30	00090	BSBW	RETURN_BFRD			
			20	C0	00093	6\$:	ADDL2	#32, BFRD	3887	
	C8	52	54	F2	00096	7\$:	AOBLSS	R4, INDX0	3863	
			F5D0	30	0009A	BSBW	RELEASE_CACHE	3891		
			04	0009D	RET			3893		

: Routine Size: 158 bytes, Routine Base: \$CODE\$ + 109A

```

: 2682          3894 1
: 2683          3895 1 END
: 2684          3896 0 ELUDOM

```

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	4408	NOVEC,NOWRT, RD, EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

RDBLOK
V04-005

J 5
8-Jan-1985 18:20:35
2-Oct-1984 12:43:36

VAX-11 Bliss-32 V4.0-742
[F11X.BUGSRC]RDBLOK.B32;1

Page 92
(26)

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
:_\$255\$DUA18:[SYSLIB]LIB.L32;1	18619	92	0	1000	00:01.9

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:RDBLOK/OBJ=OBJ\$:RDBLOK MSRC\$:RDBLOK/UPDATE=(BUG\$:RDBLOK)

: Size: 4401 code + 7 data bytes
: Run Time: 04:22.7
: Elapsed Time: 06:02.3
: Lines/CPU Min: 889
: Lexemes/CPU-Min: 62096
: Memory Used: 490 pages
: Compilation Complete

0444 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

Grid of source code listings (LST files) for MCRF UPD.

Key titles visible in the grid:

- LTDRIVER MAP
- LTDRIVER LIS
- TRUNC LIS
- RWUB LIS
- SMALOC LIS
- SNDR LIS
- LAT

The grid contains numerous smaller listings, many of which are partially legible but mostly obscured by the dark background and low resolution.