

| | | | | |
|------------------|------------|------------|-----|-----|
| FFFFFFFFFFFFFFFF | 111 | 111 | XXX | XXX |
| FFFFFFFFFFFFFFFF | 111 | 111 | XXX | XXX |
| FFFFFFFFFFFFFFFF | 111 | 111 | XXX | XXX |
| FFF | 111111 | 111111 | XXX | XXX |
| FFF | 111111 | 111111 | XXX | XXX |
| FFF | 111111 | 111111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFFFFFFF.FFF | 111 | 111 | XXX | XXX |
| FFFFFFFFFFFFFF | 111 | 111 | XXX | XXX |
| FFFFFFFFFFFFFF | 111 | 111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFF | 111 | 111 | XXX | XXX |
| FFF | 1111111111 | 1111111111 | XXX | XXX |
| FFF | 1111111111 | 1111111111 | XXX | XXX |
| FFF | 1111111111 | 1111111111 | XXX | XXX |

Symb

IOCI
IO-C
IO-C
IO-C
IO-F
IO-S
KICL

KILL
KILL
LB_E
LB_C
LB_F
LB_H
LB_L
LOCAL
LOCAL
LOCK

LOCK
LOCK
LOCK

LOC
LOC
L_CC

L_CC
L_CC

L_DA
L_DA

MAIN
MAKE
MAKE
MAKE
MAKE

MAKE
MAKE

MAP
MAP

MAR
MAR
MAR
MAR
MAR

```

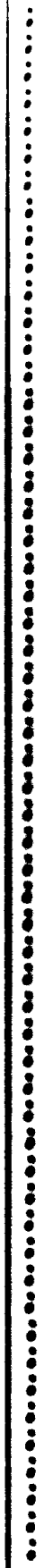
DDDDDDDD      EEEEEEEEEE      LL      FFFFFFFFFF      IIIIII      LL
DDDDDDDD      EEEEEEEEEE      LL      FFFFFFFFFF      IIIIII      LL
DD      DD      EE      LL      FF      II      LL
DD      DD      EE      LL      FF      II      LL
DD      DD      EE      LL      FF      II      LL
DD      DD      EE      LL      FFFFFFFF      II      LL
DD      DD      EEEEEEEE      LL      FFFFFFFF      II      LL
DD      DD      EEEEEEEE      LL      FF      II      LL
DD      DD      EE      LL      FF      II      LL
DD      DD      EE      LL      FF      II      LL
DD      DD      EE      LL      FF      II      LL
DD      DD      EE      LL      FF      II      LL
DDDDDDDD      EEEEEEEEEE      LLLLLLLLLL      FF      IIIIII      LLLLLLLLLL
DDDDDDDD      EEEEEEEEEE      LLLLLLLLLL      FF      IIIIII      LLLLLLLLLL

```

```

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

```



```

1 0001 0 MODULE DELFIL (
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000'
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
12 0012 1 * ALL RIGHTS RESERVED. *
13 0013 1 *
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 deletes a file, returning its blocks to the storage map
38 0038 1 and releasing the file header.
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: 4-Apr-1977 15:50
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 V03-012 CDS0008 Christian D. Saether 22-Aug-1984
53 0053 1 Don't complain about directories either (CDS0006).
54 0054 1
55 0055 1 V03-011 ACG0444 Andrew C. Goldstein, 21-Aug-1984 20:43
56 0056 1 Fix error recovery in file ID cache flush code
57 0057 1

```

```

58 0058 1 V03-010 CDS0007 Christian D. Saether 14-Aug-1984
59 0059 1 Don't complain (CDS0006) about extension headers.
60 0060 1
61 0061 1 V03-009 CDS0006 Christian D. Saether 10-Aug-1094
62 0062 1 Add bugchecks to guard against deleting the wrong file,
63 0063 1 and directories in particular.
64 0064 1
65 0065 1 V03-008 CDS0005 Christian D. Saether 7-Aug-1984
66 0066 1 Replace TOSS_CACHE_DATA call with KILL_BUFFERS call.
67 0067 1
68 0068 1 V03-007 ACG0438 Andrew C. Goldstein, 1-Aug-1984 17:14
69 0069 1 Add cache interlock logic
70 0070 1
71 0071 1 V03-006 ACG0409 Andrew C. Goldstein, 22-Mar-1984 0:08
72 0072 1 Don't invalidate deleted file headers, as they are
73 0073 1 likely to be reused soon, due to the file ID cache.
74 0074 1 Make APPLY_RVN and DEFAULT_RVN macros.
75 0075 1
76 0076 1 V03-005 CDS0004 Christian D. Saether 1-Mar-1984
77 0077 1 Replace call to FLUSH_FID with call to TOSS_CACHE_DATA.
78 0078 1
79 0079 1 V03-004 CDS0003 Christian D. Saether 29-Dec-1983
80 0080 1 Use L_NORM linkage and BIND_COMMON macro.
81 0081 1
82 0082 1 V03-003 CDS0002 Christian D. Saether 13-Sep-1983
83 0083 1 Change interface to allocation serialization.
84 0084 1
85 0085 1 V03-002 CDS0001 Christian D. Saether 13-May-1983
86 0086 1 Serialize file header deletion processing.
87 0087 1
88 0088 1 V03-001 LMP0077 L. Mark Pilant, 31-Jan-1983 10:26
89 0089 1 Eliminate the check made for extension headers as this is
90 0090 1 now done in the DELETE module. An access conflict error
91 0091 1 will result if an attempt is made to delete a file that
92 0092 1 has one of its extension headers accessed.
93 0093 1
94 0094 1 V02-007 ACG0229 Andrew C. Goldstein, 23-Dec-1981 21:59
95 0095 1 Count file ID cache hits and misses
96 0096 1
97 0097 1 V02-006 ACG0167 Andrew C. Goldstein, 16-Apr-1980 19:25
98 0098 1 Previous revision history moved to F11B.REV
99 0099 1 **
100 0100 1
101 0101 1
102 0102 1 LIBRARY 'SYS$LIBRARY:LIB.L32';
103 0103 1 REQUIRE 'SRC$:FCPDEF.B32';
104 1094 1
105 1095 1
106 1096 1 FORWARD ROUTINE
107 1097 1 DELETE_FILE : L_NORM NOVALUE, ! complete file deletion
108 1098 1 DELETE_FID : L_NORM NOVALUE, ! just release file header
109 1099 1 RETURN_FILE_NUM : L_NORM, ! return file number to cache
110 1100 1 REMOVE_FILE_NUM : L_NORM; ! remove file numbers from cache

```

```

112 1101 1 GLOBAL ROUTINE DELETE_FILE (FIB, FILEHEADER) : L_NORM NOVALUE =
113 1102 1
114 1103 1 **
115 1104 1
116 1105 1 FUNCTIONAL DESCRIPTION:
117 1106 1
118 1107 1     This routine deletes a file by releasing its blocks to the storage
119 1108 1     bitmap and then releasing the header.
120 1109 1
121 1110 1 CALLING SEQUENCE:
122 1111 1     DELETE_FILE (ARG1, ARG2)
123 1112 1
124 1113 1 INPUT PARAMETERS:
125 1114 1     ARG1: FIB of operation
126 1115 1     ARG2: address of file header buffer
127 1116 1
128 1117 1 IMPLICIT INPUTS:
129 1118 1     NONE
130 1119 1
131 1120 1 OUTPUT PARAMETERS:
132 1121 1     NONE
133 1122 1
134 1123 1 IMPLICIT OUTPUTS:
135 1124 1     NONE
136 1125 1
137 1126 1 ROUTINE VALUE:
138 1127 1     NONE
139 1128 1
140 1129 1 SIDE EFFECTS:
141 1130 1     File deleted, storage map and index file bitmap modified, VCB modified
142 1131 1
143 1132 1 --
144 1133 1
145 1134 2 BEGIN
146 1135 2
147 1136 2 MAP
148 1137 2     FIB          : REF BBLOCK,    ! address of user FIB
149 1138 2     FILEHEADER  : REF BBLOCK;    ! address of file header
150 1139 2
151 1140 2 LOCAL
152 1141 2     HEADER      : REF BBLOCK,    ! local address of file header
153 1142 2     FCB         : REF BBLOCK,    ! FCB of header in process
154 1143 2     FILE_NUMBR, :                ! file number of header being deleted
155 1144 2     EXT_FID     : BBLOCK [FIDSC_LENGTH], ! extension file ID
156 1145 2     EX_SEGNUM,  :                ! header extension segment number
157 1146 2     FIFESIZE;   :                ! size of file section
158 1147 2
159 1148 2 BIND_COMMON;
160 1149 2
161 1150 2 EXTERNAL ROUTINE
162 1151 2     FILE_SIZE   : L_NORM,        ! compute file section size
163 1152 2     CHARGE_QUOTA : L_NORM,        ! charge user's disk quota
164 1153 2     CHECKSUM    : L_NORM,        ! compute file header checksum
165 1154 2     SEND_BADSCAN : L_NORM ADDRESSING MODE (GENERAL),
166 1155 2     start bad block scan process
167 1156 2     WRITE_BLOCK : L_NORM,        ! write block to disk
168 1157 2     TRUNCATE_HEADER : L_NORM,    ! truncate file header

```

```
169      1158      NEXT_HEADER      : L_NORM;      . read next file extension header
170      1159
171      1160
172      1161      HEADER = .FILEHEADER;
173      1162
174      1163      IF (.HEADER [FH2$W_SEG_NUM] EQL 0
175      1164          AND (.FIB [FIB$W_FID]) NEQ (.HEADER [FH2$W_FID]) ! fid_num + fid_seq
176      1165          OR .FIB [FIB$B_FID_NMX] NEQ .HEADER [FH2$B_FID_NMX])
177      1166
178      1167      THEN
179      1168          BUG_CHECK (WRTINVBUFF, 'attempted to delete the wrong file');
180      1169
181      1170      ! If the file is marked bad and is not empty, we do not delete the file,
182      1171      ! but rather send it to the bad block scanner, who will analyze the file and
183      1172      ! delete it piecemeal.
184      1173
185      1174
186      1175      IF .HEADER[FH2$V_BADBLOCK]
187      1176      AND ( .HEADER[FH2$B_MAP_INUSE] NEQ 0
188      1177          OR .HEADER[FH2$W_EX_FIDNUM] NEQ 0
189      1178          OR .HEADER[FH2$W_EX_FIDRVN] NEQ 0)
190      1179      THEN
191      1180          BEGIN
192      1181              CHECKSUM (.HEADER);
193      1182              SEND_BADSCAN (HEADER[FH2$W_FID]);
194      1183              RETURN;
195      1184          END;
196      1185
197      1186      ! Loop for all headers, releasing the blocks mapped and the headers.
198      1187
199      1188
200      1189      WHILE 1 DO
201      1190          BEGIN
202      1191              FILE_NUMBER = .HEADER[FH2$W_FID_NUM];
203      1192              IF .CURRENT_VCB[VCB$V_EXTFID]
204      1193                  THEN FILE_NUMBER < 16,85 = .HEADER[FH2$B_FID_NMX];
205      1194              NEW_FID = FILE_NUMBER; ! record file number for cleanup
206      1195              NEW_FID_RVN = .CURRENT_RVN;
207      1196
208      1197              HEADER[FH2$W_FID_NUM] = 0; ! deleted header has zero file number
209      1198              HEADER[FH2$W_FID_RVN] = 0;
210      1199              HEADER[FH2$W_CHECKSUM] = 0; ! and zero checksum
211      1200              FILE_HEADER = 0;
212      1201              WRITE_BLOCK (.HEADER);
213      1202
214      1203              ! Credit the header and the blocks it maps to the owner's disk quota.
215      1204
216      1205
217      1206              FILESIZE = 0;
218      1207              IF NOT .CLEANUP_FLAGS[CLF_NOTCHARGED]
219      1208                  THEN FILESIZE = FILE_SIZE - (.HEADER);
220      1209              IF NOT .CLEANUP_FLAGS[CLF_HDRNOTCHG]
221      1210                  THEN FILESIZE = FILESIZE + 1;
222      1211              CHARGE_QUOTA (.HEADER[FH2$L_FILEOWNER], -.FILESIZE, BITLIST (QUOTA_CHARGE));
223      1212
224      1213      ! Now return the blocks mapped by the header to the storage map.
225      1214      ! Then extract the extension header data.
```

```

226 1215 3
227 1216
228 1217 TRUNCATE_HEADER (.FIB, .HEADER);
229 1218
230 1219 EX SEGNUM = .HEADER[FH2$W_SEG_NUM] + 1;
231 1220 CHSMOVE (FID$C_LENGTH, HEADER[FH2$W_EXT_FID], EXT_FID);
232 1221
233 1222 ! Now free the header in the index file bitmap. Then chain to the next header,
234 1223 ! if any, and repeat.
235 1224
236 1225
237 1226 NEW FID = 0;
238 1227 DELETE_FID (.FILE_NUMBER);
239 1228
240 1229 HEADER = NEXT_HEADER (0, 0, EXT_FID, .EX_SEGNUM);
241 1230 IF .HEADER EQ 0 THEN EXITLOOP;
242 1231 END;
243 1232
244 1233 1 END;
! end of routine DELETE_FILE

```

| | | | | | |
|--------|---|--|--|--|------|
| .TITLE | DELFIL | | | | |
| .IDENT | \V04-C00\ | | | | |
| .EXTRN | FILE SIZE, CHARGE QUOTA | | | | |
| .EXTRN | CHECKSUM, SEND_BADSCAN | | | | |
| .EXTRN | WRITE_BLOCK, TRUNCATE_HEADER | | | | |
| .EXTRN | NEXT_HEADER, BUGS_WRTINBUF | | | | |
| .PSECT | \$CODE\$,NOWRT,2 | | | | |
| .ENTRY | DELETE_FILE, Save R2,R3,R4,R5,R6,R7,R8,R9 | | | | 1101 |
| SUBL2 | #8, SP | | | | |
| MOVL | FILEHEADER, HEADER | | | | 1161 |
| TSTW | 4(HEADER) | | | | 1163 |
| BNEQ | 2\$ | | | | |
| MOVL | FIB, R0 | | | | 1164 |
| CMPL | 4(R0), 8(HEADER) | | | | |
| BNEQ | 1\$ | | | | |
| CMPB | 9(R0), 13(HEADER) | | | | 1165 |
| BEQL | 2\$ | | | | |
| BUGW | | | | | 1168 |
| .WORD | <BUGS_WRTINBUF'4> | | | | |
| BBC | #6, 53(HEADER), 4\$ | | | | 1175 |
| TSTB | 58(HEADER) | | | | 1176 |
| BNEQ | 3\$ | | | | |
| TSTW | 14(HEADER) | | | | 1177 |
| BNEQ | 3\$ | | | | |
| TSTW | 18(HEADER) | | | | 1178 |
| BEQL | 4\$ | | | | |
| PUSHL | HEADER | | | | 1181 |
| CALLS | #1, CHECKSUM | | | | |
| PUSHAB | 8(HEADER) | | | | 1182 |
| CALLS | #1, SEND_BADSCAN | | | | |
| RET | | | | | 1180 |
| MOVZWL | 8(HEADER), FILE_NUMBER | | | | 1191 |
| MOVL | -104(BASE), R0 | | | | 1192 |

| | | | | | |
|----|-----------|----|-------|-------|-------|
| | | | 03FC | 00000 | |
| | 5E | | 08 | C2 | 00002 |
| | 56 | 08 | AC | D0 | 00005 |
| | | 04 | A6 | B5 | 00009 |
| | | | 16 | 12 | 0000C |
| | 50 | 04 | AC | D0 | 0000E |
| 08 | A6 | 04 | A0 | D1 | 00012 |
| | | | 07 | 12 | 00017 |
| 0D | A6 | 09 | A0 | 91 | 00019 |
| | | | 04 | 13 | 0001E |
| | | | FEFF | 00020 | 1\$: |
| | | | 0000* | 00022 | |
| 21 | 35 | A6 | 06 | E1 | 00024 |
| | | 3A | A6 | 95 | 00029 |
| | | | 0A | 12 | 0002C |
| | | 0E | A6 | B5 | 0002E |
| | | | 05 | 12 | 00031 |
| | | 12 | A6 | B5 | 00033 |
| | | | 12 | 13 | 00036 |
| | | | 56 | DD | 00038 |
| | 0000G | CF | 01 | FB | 0003A |
| | | 08 | A6 | 9F | 0003F |
| | 00000000G | 00 | 01 | FB | 00042 |
| | | | 04 | 00049 | |
| | 57 | 08 | A6 | 3C | 0004A |
| | 50 | 98 | AA | D0 | 0004E |

| | | | | | | | | | |
|----|-------|----|------|----|----------|--------|----------------------------------|------------------------|------|
| 06 | 09 | A0 | 05 | E1 | 00052 | BBC | #5, 11(R0), 5\$ | : | |
| 08 | | 10 | 0D | A6 | F0 00057 | INSV | 13(HEADER), #16, #8, FILE_NUMBER | : | 1193 |
| | AB | AA | | 57 | D0 0005D | 5\$: | MOVL | FILE_NUMBER, -88(BASE) | 1194 |
| | AC | AA | A0 | AA | D0 00061 | MOVL | -96(BASE), -84(BASE) | : | 1195 |
| | | | 08 | A6 | B4 00066 | CLRW | 8(HEADER) | : | 1197 |
| | | | 0C | A6 | B4 00069 | CLRW | 12(HEADER) | : | 1198 |
| | | | 01FE | C6 | B4 0006C | CLRW | 510(HEADER) | : | 1199 |
| | | | 04 | AA | D4 00070 | CLRL | 4(BASE) | : | 1200 |
| | | | | 56 | DD 00073 | PUSHL | HEADER | : | 1201 |
| | 0000G | CF | | 01 | FB 00075 | CALLS | #1, WRITE_BLOCK | : | |
| | | | | 58 | D4 0007A | CLRL | FILESIZE | : | 1206 |
| 0A | | 6A | | 1D | E0 0007C | BBS | #29, (BASE), 6\$ | : | 1207 |
| | | | | 56 | DD 00080 | PUSHL | HEADER | : | 1208 |
| | 0000G | CF | | 01 | FB 00082 | CALLS | #1, FILE_SIZE | : | |
| | | 58 | | 50 | D0 00087 | MOVL | R0, FILESIZE | : | |
| 02 | | 6A | | 1B | E0 0008A | 6\$: | BBS | #27, (BASE), 7\$ | 1209 |
| | | | | 58 | D6 0008E | INCL | FILESIZE | : | 1210 |
| | | | | 02 | DD 00090 | 7\$: | PUSHL | #2 | 1211 |
| | | 7E | | 58 | CE 00092 | MNEGL | FILESIZE, -(SP) | : | |
| | | | 3C | A6 | DD 00095 | PUSHL | 60(HEADER) | : | |
| | 0000G | CF | | 03 | FB 00098 | CALLS | #3, CHARGE_QUOTA | : | |
| | | | | 56 | DD 0009D | PUSHL | HEADER | : | 1217 |
| | | | 04 | AC | DD 0009F | PUSHL | FIB | : | |
| | 0000G | CF | | 02 | FB 000A2 | CALLS | #2, TRUNCATE_HEADER | : | |
| | | 59 | 04 | A6 | 3C 000A7 | MOVZWL | 4(HEADER), EX_SEGNUM | : | 1219 |
| | | | | 59 | D6 000AB | INCL | EX_SEGNUM | : | |
| 6E | 0E | A6 | | 06 | 28 000AD | MOVCS | #6, 14(HEADER), EXT_FID | : | 1220 |
| | | | AB | AA | D4 000B2 | CLRL | -88(BASE) | : | 1226 |
| | | | | 57 | DD 000B5 | PUSHL | FILE_NUMBER | : | 1227 |
| | 0000V | CF | | 01 | FB 000B7 | CALLS | #1, DELETE_FID | : | |
| | | | | 59 | DD 000BC | PUSHL | EX_SEGNUM | : | 1229 |
| | | | 04 | AE | 9F 000BE | PUSHAB | EXT_FID | : | |
| | | | | 7E | 7C 000C1 | CLRW | -(SP) | : | |
| | 0000G | CF | | 04 | FB 000C3 | CALLS | #4, NEXT_HEADER | : | |
| | | 56 | | 50 | D0 000C8 | MOVL | R0, HEADER | : | |
| | | | | 03 | 13 000CB | BEQL | 8\$ | : | 1230 |
| | | | FF7A | 31 | 000CD | BRW | 4\$ | : | |
| | | | | 04 | 000D0 | 8\$: | RET | : | 1233 |

; Routine Size: 209 bytes. Routine Base: \$CODE\$ + 0000


```

246 1234 1 GLOBAL ROUTINE DELETE_FID (FILENUM) : _NORM NOVALUE =
247 1235 1
248 1236 1 '++
249 1237 1
250 1238 1 FUNCTIONAL DESCRIPTION:
251 1239 1
252 1240 1     This routine marks the indicated file header free in the index
253 1241 1     file bitmap.
254 1242 1
255 1243 1 CALLING SEQUENCE:
256 1244 1     DELETE_HEADER (ARG1)
257 1245 1
258 1246 1 INPUT PARAMETERS:
259 1247 1     ARG1: file number of header
260 1248 1
261 1249 1 IMPLICIT INPUTS:
262 1250 1     CURRENT_VCB: VCB of volume
263 1251 1
264 1252 1 OUTPUT PARAMETERS:
265 1253 1     NONE
266 1254 1
267 1255 1 IMPLICIT OUTPUTS:
268 1256 1     NONE
269 1257 1
270 1258 1 ROUTINE VALUE:
271 1259 1     NONE
272 1260 1
273 1261 1 SIDE EFFECTS:
274 1262 1     Header deleted - index file bitmap & VCB altered
275 1263 1
276 1264 1 --
277 1265 1
278 1266 2 BEGIN
279 1267 2
280 1268 2 BUILTIN
281 1269 2     FP;
282 1270 2
283 1271 2 LOCAL
284 1272 2     CACHE           : REF BBLOCK,      : address of cache block
285 1273 2     FID_CACHE       : REF BBLOCK,      : address of file number cache
286 1274 2     LOCK_STATUS     : VECTOR [2],      : lock status block
287 1275 2     VBN             :                   : relative block in bitmap
288 1276 2     BEST_VBN        :                   : best block to return bits to
289 1277 2     COUNT           :                   : number of FID's in current block
290 1278 2     BEST_COUNT      :                   : number of FID's in best block
291 1279 2     BLOCK           :                   : block number of current entry
292 1280 2     BUFFER          :                   : bitmap buffer
293 1281 2
294 1282 2 EXTERNAL
295 1283 2     PMS$GL_FIDHIT   : ADDRESSING_MODE (GENERAL)
296 1284 2                   : count of file ID cache hits
297 1285 2     PMS$GL_FIDMISS : ADDRESSING_MODE (GENERAL);
298 1286 2                   : count of file ID cache misses
299 1287 2
300 1288 2 BIND_COMMON;
301 1289 2
302 1290 2 EXTERNAL ROUTINE

```

```
303 1291 2 ALLOCATION_LOCK : L_NORM,
304 1292 2 INIT_FID_CACHE : L_NORM,      : initialize file ID cache lock
305 1293 2 READ_BLOCK   : L_NORM,      : read a block from the disk
306 1294 2 WRITE_BLOCK  : L_NORM,      : write it back
307 1295 2 ZERO_ON_ERROR; : return zero on error signal (handler)
308 1296 2
309 1297 2 : Serialize against other storage or file header allocation/deallocation
310 1298 2 operations.
311 1299 2
312 1300 2
313 1301 2 ALLOCATION_LOCK ();
314 1302 2
315 1303 2 : If this is not a flush call, we delete the file number by returning it
316 1304 2 to the file number cache. If the cache fills up, the kernel mode routine
317 1305 2 returns LBC. We then scan the cache, looking for the largest group of file
318 1306 2 numbers that are all in the same bitmap block (up to half of the cache),
319 1307 2 and then flush those from the cache. If this is a cache flush call or
320 1308 2 the volume is marked for dismount, however, we flush the entire cache.
321 1309 2
322 1310 2
323 1311 2 CACHE = .CURRENT_VCB[VCBSL_CACHE];
324 1312 2 FID_CACHE = .CACHE[VCASL_FIDCACHE];
325 1313 2
326 1314 2 IF .FILENUM NEQ 0
327 1315 2 THEN
328 1316 2 BEGIN
329 1317 2 IF NOT .CACHE[VCASV_FIDC_VALID]
330 1318 2 THEN INIT_FID_CACHE(.(CACHE);
331 1319 2 IF KERNEL_CALL (RETURN_FILE_NUM, .FILENUM)
332 1320 2 THEN
333 1321 2 BEGIN
334 1322 2 PMSSGL_FIDHIT = .PMSSGL_FIDHIT + 1;
335 1323 2 RETURN;
336 1324 2 END;
337 1325 2 END;
338 1326 2
339 1327 2 IF .FILENUM NEQ 0
340 1328 2 AND .CACHE[VCASV_FIDC_VALID]
341 1329 2 THEN
342 1330 2 BEGIN
343 1331 2 PMSSGL_FIDMISS = .PMSSGL_FIDMISS + 1;
344 1332 2 BEST_COUNT = 0;
345 1333 2 VBN = -1;
346 1334 2 INCR J FROM 1 TO .FID_CACHE[VCASW_FIDCOUNT]
347 1335 2 DO
348 1336 2 BEGIN
349 1337 2 BLOCK = (.VECTOR [FID_CACHE[VCASL_FIDLST], .J-1] - 1) / 4096;
350 1338 2 IF .BLOCK NEQ .VBN
351 1339 2 THEN
352 1340 2 BEGIN
353 1341 2 VBN = .BLOCK;
354 1342 2 COUNT = 0;
355 1343 2 END;
356 1344 2 COUNT = .COUNT + 1;
357 1345 2 IF .COUNT GTRU .BEST_COUNT
358 1346 2 THEN
359 1347 2 BEGIN
```

```

360 1348 S          BEST_COUNT = .COUNT;
361 1349 S          BEST_VBN = .VBN;
362 1350 S          END;
363 1351 S          IF .BEST_COUNT GEQU .FID_CACHE[VCASW_FIDCOUNT]/2
364 1352 S          THEN EXIT LOOP;
365 1353 S          END;
366 1354 S
367 1355 S          ! Read the appropriate block, return the desired number of file numbers to
368 1356 S          it, and write it back.
369 1357 S
370 1358 S
371 1359 S          IF .BEST_VBN GEQU .CURRENT_VCB[VCBSB_IBMAPSIZE]
372 1360 S          THEN BUG_CHECK (BADFID, FATAL, 'ACP file number out of range for this volume');
373 1361 S
374 1362 S          BUFFER = READ_BLOCK (.BEST_VBN + .CURRENT_VCB[VCBSL_IBMAPLBN], 1, INDEX_TYPE);
375 1363 S          KERNEL_CALL (REMOVE_FILE_NUM, .BEST_COUNT, .BEST_VBN, .BUFFER);
376 1364 S          WRITE_BLOCK (.BUFFER);
377 1365 S          END
378 1366 S
379 1367 S          ! If this is a cache flush, loop for all the blocks represented in the
380 1368 S          cache, read the block, return the file numbers, and write it. Then
381 1369 S          mark the cache invalid, and release the cache lock if there is one.
382 1370 S          This operation is done under a handler to ensure its completion in
383 1371 S          the face of I/O errors.
384 1372 S
385 1373 S
386 1374 S      ELSE
387 1375 S      BEGIN
388 1376 S      .FP = ZERO_ON_ERROR;
389 1377 S      UNTIL .FID_CACHE[VCASW_FIDCOUNT] EQL 0
390 1378 S      DO
391 1379 S      BEGIN
392 1380 S      VBN = (.FID_CACHE[VCASL_FIDLIST] - 1) / 4096;
393 1381 S      IF .VBN GEQU .CURRENT_VCB[VCBSB_IBMAPSIZE]
394 1382 S      THEN BUG_CHECK (BADFID, FATAL, 'ACP file number out of range for this volume');
395 1383 S
396 1384 S      BUFFER = READ_BLOCK (.VBN + .CURRENT_VCB[VCBSL_IBMAPLBN], 1, INDEX_TYPE);
397 1385 S      IF .BUFFER NEQ 0
398 1386 S      THEN
399 1387 S      BEGIN
400 1388 S      KERNEL_CALL (REMOVE_FILE_NUM, 0, .VBN, .BUFFER);
401 1389 S      WRITE_BLOCK (.BUFFER);
402 1390 S      END
403 1391 S      ELSE
404 1392 S      FID_CACHE[VCASW_FIDCOUNT] = 0;
405 1393 S      END;
406 1394 S      IF .FID_CACHE[VCASL_FIDCLKID] NEQ 0
407 1395 S      THEN
408 1396 S      BEGIN
409 1397 S      LOCK_STATUS[1] = .FID_CACHE[VCASL_FIDCLKID];
410 1398 S      IF NOT SENQW (EFN = EFN,
411 1399 S      LKMODE = LCK$K_NLMODE,
412 1400 S      FLAGS = LCK$M_NOQUEUE OR LCK$M_SYNCSTS OR LCK$M_CONVERT OR LCK$M_CVTSYS,
413 1401 S      LKSB = LOCK_STATUS
414 1402 S      )
415 1403 S      THEN BUG_CHECK (XOPERR, FATAL, 'Unexpected lock manager error');
416 1404 S      END;

```

: 417 1405 3 CACHE[VCASV_FIDC_VALID] = 0;
: 418 1406 2 END;
: 419 1407 2
: 420 1408 1 END;

: end of routine DELETE_HEADER

| | | | | | | |
|-------|---------|-----------|------|------------------|---|------|
| | | | | OBFC 00000 | .EXTRN PMSSGL_FIDHIT, PMSSGL_FIDMISS | |
| | | | | | .EXTRN ALLOCATION_LOCK | |
| | | | | | .EXTRN INIT_FID_CACHE, READ_BLOCK | |
| | | | | | .EXTRN ZERO_ON_ERROR, BUGS_BADFID | |
| | | | | | .EXTRN SYSENQ, BUGS_XQPERR | |
| | | | | | .ENTRY DELETE_FID, Save R2,R3,R4,R5,R6,R7,R8,R9,- | 1234 |
| | | | | | R11 | |
| | SE | | 08 | C2 00002 | SUBL2 #8, SP | |
| | | 98 | AA | 9F 00005 | PUSHAB -104(BASE) | 1285 |
| 0000G | CF | | 00 | FB 00008 | CALLS #0, ALLOCATION_LOCK | 1301 |
| | 50 | 00 | BE | D0 0000D | MOVL @0(SP), R0 | 1311 |
| | 54 | 58 | A0 | D0 00011 | MOVL 88(R0), CACHE | |
| | 52 | | 64 | D0 00015 | MOVL (CACHE), FID_CACHE | 1312 |
| | | 04 | AC | D5 00018 | TSTL FILENUM | 1314 |
| | | | 1D | 13 0001B | BEQL 2\$ | |
| | 07 | 08 | A4 | E8 0001D | BLBS 11(CACHE), 1\$ | 1317 |
| | | | 54 | DD 00021 | PUSHL CACHE | 1318 |
| 0000G | CF | | 01 | FB 00023 | CALLS #1, INIT_FID_CACHE | |
| | | 04 | AC | DD 00028 1\$: | PUSHL FILENUM | 1319 |
| 0000V | CF | | 01 | FB 0002B | CALLS #1, RETURN_FILE_NUM | |
| | 07 | | 50 | E9 00030 | BLBC R0, 2\$ | |
| | | 00000000G | 00 | D6 00033 | INCL PMSSGL_FIDHIT | 1322 |
| | | | 04 | 00039 | RET | 1321 |
| | | | 04 | AC D5 0003A 2\$: | TSTL FILENUM | 1327 |
| | | | 03 | 12 0003D | BNEQ 4\$ | |
| | | | 0087 | 31 0003F 3\$: | BRW 11\$ | |
| | F9 | 08 | A4 | E9 00042 4\$: | BLBC 11(CACHE), 3\$ | 1328 |
| | | 00000000G | 00 | D6 00046 | INCL PMSSGL_FIDMISS | 1331 |
| | | | 58 | D4 0004C | CLRL BEST_COUNT | 1332 |
| | 53 | | 01 | CE 0004E | MNEGL #1, VBN | 1333 |
| | 58 | 02 | A2 | 3C 00051 | MOVZWL 2(FID_CACHE), R11 | 1334 |
| | 56 | 24 | A2 | 9E 00055 | MOVAB 36(FID_CACHE), R6 | 1337 |
| | | | 50 | D4 00059 | CLRL J | |
| | | | 31 | 11 0005B | BRB 8\$ | |
| 51 | FC A640 | | 01 | C3 0005D 5\$: | SUBL3 #1, -4(R6)[J], R1 | |
| 59 | | 00001000 | 8F | C7 00063 | DIVL3 #4096, R1, BLOCK | |
| | | | 59 | D1 0006B | CMPL BLOCK, VBN | 1338 |
| | | | 05 | 13 0006E | BEQL 6\$ | |
| | 53 | | 59 | D0 00070 | MOVL BLOCK, VBN | 1341 |
| | | | 57 | D4 00073 | CLRL COUNT | 1342 |
| | | | 57 | D6 00075 6\$: | INCL COUNT | 1344 |
| | 58 | | 57 | D1 00077 | CMPL COUNT, BEST_COUNT | 1345 |
| | | | 06 | 1B 0007A | BLEQU 7\$ | |
| | 58 | | 57 | D0 0007C | MOVL COUNT, BEST_COUNT | 1348 |
| | 55 | | 53 | D0 0007F | MOVL VBN, BEST_VBN | 1349 |
| | 51 | 02 | A2 | 3C 00082 7\$: | MOVZWL 2(FID_CACHE), R1 | 1351 |
| | 51 | | 02 | C6 00086 | DIVL2 #2, RT | |
| | 51 | | 58 | D1 00089 | CMPL BEST_COUNT, R1 | |
| | | | 04 | 1E 0008C | BGEQU 9\$ | |

| | | | | | | | | | | | | |
|-----------|----|----|--|----|-------|----------|-------|-------------|-------|--------|-----------------------------|------|
| | | CB | | 50 | | 5B | F3 | 0008E | 8\$: | AOBLEQ | R11, J, 5\$ | 1334 |
| 55 | 38 | AD | | 50 | 00 | BE | DO | 00092 | 9\$: | MOVL | @0(SP), R0 | 1359 |
| | | | | 08 | | 00 | ED | 0C096 | | CMPTV | #0, #8, 56(R0), BEST_VBN | |
| | | | | | | 04 | 1A | 0009C | | BGTRU | 10\$ | |
| | | | | | | | | FEFF 0009E | | BUGW | | 1360 |
| | | | | | | | | 0000* 000A0 | | .WORD | <BUGS_BADFID!4> | |
| | | | | | | 03 | DD | 000A2 | 10\$: | PUSHL | #3 | 1362 |
| | | | | 50 | | 01 | DD | 000A4 | | PUSHL | #1 | |
| | | | | 08 | 08 | BE | DO | 000A6 | | MOVL | @8(SP), R0 | |
| | | | | | 30 | B045 | 9F | 000AA | | PUSHAB | @48(R0)[BEST_VBN] | |
| 0000G | | | | | | 03 | FB | 000AE | | CALLS | #3, READ_BLOCK | |
| | | | | 57 | | 50 | DO | 000B3 | | MOVL | R0, BUFFER | |
| | | | | | 00A0 | 8F | BB | 000B6 | | PUSHR | #^M<R5, R7> | 1363 |
| | | | | | | 58 | DD | 000BA | | PUSHL | BEST_COUNT | |
| 0000V | | | | | | 03 | FB | 000BC | | CALLS | #3, REMOVE_FILE_NUM | |
| | | | | 57 | | DD | 000C1 | | | PUSHL | BUFFER | 1364 |
| 0000G | | | | | | 01 | FB | 000C3 | | CALLS | #1, WRITE_BLOCK | |
| | | | | | | | | 04 000C8 | | RET | | 1327 |
| | | | | 6D | 0000G | CF | 9E | 000C9 | 11\$: | MOVAB | ZERO ON ERROR, (FP) | 1376 |
| | | | | | | 02 | A2 | B5 000CE | 12\$: | TSTW | 2(FID_CACHE) | 1377 |
| | | | | | | 4C | 13 | 000D1 | | BEQL | 15\$ | |
| | | 50 | | 24 | A2 | 01 | C3 | 000D3 | | SUBL3 | #1, 36(FID_CACHE), R0 | 1380 |
| | | 53 | | | 50 | 8F | C7 | 000D8 | | DIVL3 | #4096, R0, VBN | |
| | | | | | 50 | 00001000 | BE | DO 000E0 | | MOVL | @0(SP), R0 | 1381 |
| 53 | | 38 | | | 08 | 00 | ED | 000E4 | | CMPTV | #0, #8, 56(R0), VBN | |
| | | | | | | 04 | 1A | 000EA | | BGTRU | 13\$ | |
| | | | | | | | | FEFF 000EC | | BUGW | | 1382 |
| | | | | | | | | 0000* 000EE | | .WORD | <BUGS_BADFID!4> | |
| | | | | | | 03 | DD | 000F0 | 13\$: | PUSHL | #3 | 1384 |
| | | | | | | 01 | DD | 000F2 | | PUSHL | #1 | |
| | | | | 50 | 08 | BE | DO | 000F4 | | MOVL | @8(SP), R0 | |
| | | | | | 30 | B043 | 9F | 000F8 | | PUSHAB | @48(R0)[VBN] | |
| 0000G | | | | | | 03 | FB | 000FC | | CALLS | #3, READ_BLOCK | |
| | | | | 57 | | 50 | DO | 00101 | | MOVL | R0, BUFFER | |
| | | | | | | 14 | 13 | 00104 | | BEQL | 14\$ | 1385 |
| | | | | | 0088 | 8F | BB | 00106 | | PUSHR | #^M<R3, R7> | 1388 |
| | | | | | | 7E | D4 | 0010A | | CLRL | -(SP) | |
| 0000V | | | | | | 03 | FB | 0010C | | CALLS | #3, REMOVE_FILE_NUM | |
| | | | | 57 | | DD | 00111 | | | PUSHL | BUFFER | 1389 |
| 0000G | | | | | | 01 | FB | 00113 | | CALLS | #1, WRITE_BLOCK | |
| | | | | | | B4 | 11 | 00118 | | BRB | 12\$ | 1385 |
| | | | | | | 02 | A2 | B4 0011A | 14\$: | CLRW | 2(FID_CACHE) | 1392 |
| | | | | | | AF | 11 | 0011D | | BRB | 12\$ | 1377 |
| | | | | | | 04 | A2 | D5 0011F | 15\$: | TSTL | 4(FID_CACHE) | 1394 |
| | | | | | | 25 | 13 | 00122 | | BEQL | 16\$ | |
| | | | | 08 | AE | 04 | A2 | DO 00124 | | MOVL | 4(FID_CACHE), LOCK_STATUS+4 | 1397 |
| | | | | | | 7E | 7C | 00129 | | CLRQ | -(SP) | 1402 |
| | | | | | | 7E | 7C | 0012B | | CLRQ | -(SP) | |
| | | | | | | 7E | 7C | 0012D | | CLRQ | -(SP) | |
| | | | | | | 7E | D4 | 0012F | | CLRL | -(SP) | |
| | | | | 7E | 4E | 8F | 9A | 00131 | | MOVZBL | #78, -(SP) | |
| | | | | | 24 | AE | 9F | 00135 | | PUSHAB | LOCK_STATUS | |
| | | | | | | 1E | 7D | 00138 | | MOVQ | #30, -(SP) | |
| 00000000G | | | | | | 0B | FB | 0013B | | CALLS | #11, SYS\$ENQW | |
| | | | | | | 50 | E8 | 00142 | | BLBS | R0, 16\$ | |
| | | | | | | | | FEFF 00145 | | BUGW | | 1403 |
| | | | | | | | | 0000* 00147 | | .WORD | <BUGS_XQPERR!4> | |


```

422 1409 1 ROUTINE RETURN_FILE_NUM (FILE_NUMBER) : L_NORM =
423 1410 1
424 1411 1 |++
425 1412 1 |
426 1413 1 | FUNCTIONAL DESCRIPTION:
427 1414 1 |
428 1415 1 |     This routine returns a file number to the volume's file number
429 1416 1 |     cache. If the cache fills up as a result, it also sorts the
430 1417 1 |     entries and returns failure status to signal the caller that the
431 1418 1 |     cache should be emptied.
432 1419 1 |
433 1420 1 |
434 1421 1 | CALLING SEQUENCE:
435 1422 1 |     RETURN_FILE_NUM (ARG1)
436 1423 1 |
437 1424 1 | INPUT PARAMETERS:
438 1425 1 |     ARG1: file number to return
439 1426 1 |
440 1427 1 | IMPLICIT INPUTS:
441 1428 1 |     CURRENT_VCB: VCB of volume
442 1429 1 |     CURRENT_UCB: UCB of volume
443 1430 1 |
444 1431 1 | OUTPUT PARAMETERS:
445 1432 1 |     NONE
446 1433 1 |
447 1434 1 | IMPLICIT OUTPUTS:
448 1435 1 |     NONE
449 1436 1 |
450 1437 1 | ROUTINE VALUE:
451 1438 1 |     1 if success
452 1439 1 |     0 if cache is now full
453 1440 1 |
454 1441 1 | SIDE EFFECTS:
455 1442 1 |     file ID cache modified
456 1443 1 |
457 1444 1 | --
458 1445 1 |
459 1446 2 BEGIN
460 1447 2
461 1448 2 LOCAL
462 1449 2     CACHE          : REF BBLOCK,      ! address of cache block
463 1450 2     FID_CACHE      : REF BBLOCK,      ! address of file number cache
464 1451 2     J;              ! cache index
465 1452 2
466 1453 2 BIND_COMMON;
467 1454 2
468 1455 2
469 1456 2 ! Scan the cache for an entry higher than the file number being returned.
470 1457 2 ! Shuffle the cache upward and insert the file number in order. If the
471 1458 2 ! cache fills up, return failure to cause a cache flush.
472 1459 2 |
473 1460 2 |
474 1461 2 CACHE = .CURRENT_VCB[VCSL_CACHE];
475 1462 2 FID_CACHE = .CACHE[VCASL_FIDCACHE];
476 1463 2 J = 0;
477 1464 2 UNTIL .J GEQU .FID_CACHE[VCASW_FIDCOUNT]
478 1465 2 DO

```

```

: 479      1466 3 BEGIN
: 480      1467 3 IF .VECTOR [FID_CACHE[VCA$FIDLIST], .J] GTRU .FILE_NUMBER
: 481      1468 3 THEN EXITLOOP;
: 482      1469 3 IF .VECTOR [FID_CACHE[VCA$FIDLIST], .J] EQL .FILE_NUMBER
: 483      1470 3 THEN RETURN 1;
: 484      1471 3 J = .J + 1;
: 485      1472 2 END;
: 486      1473 2
: 487      1474 2 CH$MOVE ((.FID_CACHE[VCA$FIDCOUNT]-.J)*4,
: 488      1475 2 VECTOR [FID_CACHE[VCA$FIDLIST], .J],
: 489      1476 2 VECTOR [FID_CACHE[VCA$FIDLIST], .J+1]);
: 490      1477 2 VECTOR [FID_CACHE[VCA$FIDLIST], .J] = .FILE_NUMBER;
: 491      1478 2 FID_CACHE[VCA$FIDCOUNT] = .FID_CACHE[VCA$FIDCOUNT] + 1;
: 492      1479 2
: 493      1480 2 .FID_CACHE[VCA$FIDCOUNT] LSSU .FID_CACHE[VCA$FIDSIZE]
: 494      1481 2 AND .CACHE[VCA$FIDC_VALID]
: 495      1482 2
: 496      1483 1 END;

```

! end of routine RETURN_FILE_NUM

| 01FC 0000 RETURN_FILE_NUM: | | | | | | | | | | |
|----------------------------|----|----|--|--|--|--|--|--|--|------|
| | | | | | | | | | | 1409 |
| | | | | | | | | | | 1461 |
| | | | | | | | | | | 1462 |
| | | | | | | | | | | 1463 |
| 57 | 02 | A6 | | | | | | | | 1467 |
| | | | | | | | | | | 1464 |
| | | | | | | | | | | 1467 |
| | | | | | | | | | | 1469 |
| | | | | | | | | | | 1470 |
| | | | | | | | | | | 1471 |
| | | | | | | | | | | 1464 |
| | | | | | | | | | | 1474 |
| | | | | | | | | | | 1476 |
| | | | | | | | | | | 1477 |
| | | | | | | | | | | 1478 |
| | | | | | | | | | | 1480 |
| 51 | 0B | A8 | | | | | | | | 1481 |
| | | | | | | | | | | 1483 |

; Routine Size: 97 bytes. Routine Base: \$CODE\$ + 021F

DELFIL
V04-000

B-11
16-Sep-1984 00:17:11
14-Sep-1984 12:30:16

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]DELFIL.B32,1 Page 15
(4)

DJ
V

```

498 1484 1 ROUTINE REMOVE_FILE_NUM (COUNT, VBN, BUFFER) : L_NORM =
499 1485 1
500 1486 1 **
501 1487 1
502 1488 1 FUNCTIONAL DESCRIPTION:
503 1489 1
504 1490 1 This routine removes the specified entries from the file ID cache
505 1491 1 and marks them free in the index file bitmap block supplied.
506 1492 1
507 1493 1
508 1494 1 CALLING SEQUENCE:
509 1495 1 REMOVE_FILE_NUM (ARG1, ARG2, ARG3)
510 1496 1
511 1497 1 INPUT PARAMETERS:
512 1498 1 ARG1: number of entries to remove (0 to remove all)
513 1499 1 ARG2: VBN of bitmap buffer
514 1500 1 ARG3: address of bitmap buffer
515 1501 1
516 1502 1 IMPLICIT INPUTS:
517 1503 1 CURRENT_VCB: address of volume VCB
518 1504 1
519 1505 1 OUTPUT PARAMETERS:
520 1506 1 NONE
521 1507 1
522 1508 1 IMPLICIT OUTPUTS:
523 1509 1 NONE
524 1510 1
525 1511 1 ROUTINE VALUE:
526 1512 1 1
527 1513 1
528 1514 1 SIDE EFFECTS:
529 1515 1 file ID cache altered, bitmap buffer modified
530 1516 1
531 1517 1 --
532 1518 1
533 1519 2 BEGIN
534 1520 2
535 1521 2 MAP
536 1522 2 BUFFER : REF BITVECTOR; ! bitmap buffer
537 1523 2
538 1524 2 LOCAL
539 1525 2 FID_CACHE : REF BBLOCK, ! address of file number cache
540 1526 2 K, ! counter of entries removed
541 1527 2 J, ! index into cache
542 1528 2 FILE_NUMBER, ! file number-1 of entry
543 1529 2 BITPOS; ! bit position in buffer
544 1530 2
545 1531 2 BIND_COMMON;
546 1532 2
547 1533 2 ! Scan the file ID cache for entries whose bitmap VBN match those of the
548 1534 2 ! buffer. When one is found, clear the corresponding bit in the bitmap,
549 1535 2 ! decrement the count in the cache, and shuffle down the remaining entries
550 1536 2 ! to keep the cache compacted.
551 1537 2
552 1538 2
553 1539 2 FID_CACHE = .BBLOCK [.CURRENT_VCB[VCBSL_CACHE], VCASL_FIDCACHE];
554 1540 2 K = .COUNT;

```

```

555 1541 2 J = 1;
556 1542 2 DO
557 1543 2 BEGIN
558 1544 2 FILE_NUMBER = .VECTOR [FID_CACHE[VCASL_FIDLIST], .J-1] - 1;
559 1545 2 IF .FILE_NUMBER / 4096 EQL .VBN
560 1546 2 THEN
561 1547 2 BEGIN
562 1548 2 BITPOS = .FILE_NUMBER<0,12>;
563 1549 2 BUFFER[.BITPOS] = 0;
564 1550 2 CHSMOVE ((.FID_CACHE[VCASW_FIDCOUNT]-.J)*4,
565 1551 2 VECTOR [FID_CACHE[VCASL_FIDLIST], .J],
566 1552 2 VECTOR [FID_CACHE[VCASL_FIDLIST], .J-1]);
567 1553 2 FID_CACHE[VCASW_FIDCOUNT] = .FID_CACHE[VCASW_FIDCOUNT] - 1;
568 1554 2 J = .J - 1;
569 1555 2 K = .K - 1;
570 1556 2 END;
571 1557 2 J = .J + 1;
572 1558 2 END
573 1559 2 UNTIL .K EQL 0 OR .J GTRU .FID_CACHE[VCASW_FIDCOUNT];
574 1560 2
575 1561 2 ! If we have freed file numbers in a block that precedes the current bitmap
576 1562 2 ! scan point, reset the scan point.
577 1563 2
578 1564 2
579 1565 2 IF .VBN LSSU .CURRENT_VCB[VCBSB_IBMAPVBN]
580 1566 2 THEN CURRENT_VCB[VCBSB_IBMAPVBN] = .VBN;
581 1567 2
582 1568 2 1
583 1569 2 END;

```

! end of routine RETURN_FILE_NUM

| OBFC 0000 REMOVE_FILE_NUM: | | | | .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R11 | | |
|----------------------------|---------|----------|---------------------|--|-----------------------------------|--------|
| | 50 | 98 | AA D0 00002 | MOVL | -104(BASE), R0 | : 1484 |
| | 56 | 58 | B0 D0 00006 | MOVL | @88(R0), FID_CACHE | : 1539 |
| | 5B | 04 | AC D0 0000A | MOVL | COUNT, K | : 1540 |
| | 58 | | 01 D0 0000E | MOVL | #1, J | : 1541 |
| 57 | 20 A648 | | 01 C3 00011 1\$: | SUBL3 | #1, 32(FID_CACHE)[J], FILE_NUMBER | : 1544 |
| 50 | 08 AC | 00001000 | 8F C7 00017 | DIVL3 | #4096, FILE_NUMBER, R0 | : 1545 |
| | | | 50 D1 0001F | CML | R0, VBN | |
| | | | 27 12 00023 | BNEQ | 3\$ | |
| 59 | 57 | 0C | 00 EF 00025 | EXTZV | #0, #12, FILE_NUMBER, BITPOS | : 1548 |
| | 00 | 0C BC | 59 E5 0002A | BBCC | BITPOS, @BUFFER, 2\$ | : 1549 |
| | | 50 | 02 A6 3C 0002F 2\$: | MOVZWL | 2(FID_CACHE), R0 | : 1550 |
| | | 50 | 58 C2 00033 | SUBL2 | J, R0 | |
| | | 50 | 04 C4 00036 | MULL2 | #4, R0 | |
| | | 20 A648 | DF 00039 | PUSHAL | 32(FID_CACHE)[J] | : 1552 |
| | | 24 A648 | DF 0003D | PUSHAL | 36(FID_CACHE)[J] | |
| | 9E | 9E | 50 28 00041 | MOV C3 | R0, @(SP)+, @(SP)+ | |
| | | 02 | A6 B7 00045 | DECW | 2(FID_CACHE) | : 1553 |
| | | | 58 D7 00048 | DECL | J | : 1554 |
| | | | 58 D7 0004A | DECL | K | : 1555 |
| | | | 58 D6 0004C 3\$: | INCL | J | : 1557 |
| | | | 58 D5 0004E | TSTL | K | : 1559 |

| | | | | | | | | | | | | |
|----|----|----|----|----|----|-------|-------|--------------------------|-------|---------------------|--------|------|
| 58 | 02 | A6 | 10 | 08 | 13 | 00050 | BEQL | 4\$ | | | | |
| | | | | 00 | ED | 00052 | CMPZV | #0, #16, 2(FID_CACHE), J | | | | |
| | | | | B7 | 1E | 00058 | BGEQU | 1\$ | | | | |
| 08 | AC | 3A | A0 | 5C | 98 | AA D0 | 0005A | 4\$: | MOVL | -104(BASE), R0 | 1565 | |
| | | | | 08 | | 00 | ED | 0005E | CMPZV | #0, #8, 58(R0), VBN | | |
| | | | | | | 05 | 1B | 00065 | BLEQU | 5\$ | | |
| | | 3A | A0 | 08 | | AC | 90 | 00067 | MOVB | VBN, 58(R0) | 1566 | |
| | | | 50 | | | 01 | D0 | 0006C | 5\$: | MOVL | #1, R0 | 1569 |
| | | | | | | 04 | 0006F | | RET | | | |

: Routine Size: 112 bytes, Routine Base: \$CODE\$ + 0280

```

: 584      1570  1
: 585      1571  1 END
: 586      1572  0 ELUDOM

```

PSECT SUMMARY

| Name | Bytes | Attributes |
|----------|-------|--|
| \$CODE\$ | 752 | NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2) |

Library Statistics

| File | Total | Symbols Loaded | Percent | Pages Mapped | Processing Time |
|---------------------------------|-------|----------------|---------|--------------|-----------------|
| _\$255\$DUA28:[SYSLIB]LIB.L32;1 | 18619 | 50 | 0 | 1000 | 00:01.8 |

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DELFIL/OBJ=OBJ\$:DELFIL MSRC\$:DELFIL/UPDATE=(ENH\$:DELFIL)

```

: Size:          752 code + 0 data bytes
: Run Time:      00:49.2
: Elapsed Time: 01:43.4
: Lines/CPU Min: 1915
: Lexemes/CPU-Min: 58358
: Memory Used:  262 pages
: Compilation Complete

```

