

FFFFFFFFFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFF	111111	111111	111111	XXX
FFF	111111	111111	111111	XXX
FFF	111111	111111	111111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111111111	111111111	111111111	XXX
FFF	111111111	111111111	111111111	XXX
FFF	111111111	111111111	111111111	XXX

\*\*FILE\*\*ID\*\*WITURN

WW	WW		TTTTTTTTTT	UU	UU	RRRRRRRR	NN	NN
WW	WW		TT	UU	UU	RR	RR	NN
WW	WW		TT	UU	UU	RR	RR	NN
WW	WW		TT	UU	UU	RR	RR	NNNN
WW	WW		TT	UU	UU	RR	RR	NNNN
WW	WW		TT	UU	UU	RRRRRRRR	NN	NN
WW	WW		TT	UU	UU	RRRRRRRR	NN	NN
WW	WW		TT	UU	UU	RR	RR	NNNN
WWWW	WWWW		TT	UU	UU	RR	RR	NN
WWWW	WWWW		TT	UU	UU	RR	RR	NNNN
WW	WW		TT	UUUUUUUUUU	RR	RR	NN	NN
WW	WW		TT	UUUUUUUUUU	RR	RR	NN	NN
							....	
							....	
							....	

LL		SSSSSSSS
LL		SSSSSSSS
LL		SS
LL		SS
LL		SS
LL		SSSSSS
LL		SSSSSS
LL		SS
LL		SS
LL		SS
LLLLLLLL		SSSSSSSS
LLLLLLLL		SSSSSSSS

```
1 0001 0 MODULE WITURN (
2 0002 0   LANGUAGE (BLISS32),
3 0003 0   IDENT = 'V04-000'
4 0004 0   ) =
5 0005 1 BEGIN
6
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 generates a window mapping the desired VBN from
38 0038 1 the supplied 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: 7-Dec-1976 14:38
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 V03-012 CDS0002 Christian D. Saether 31-July-1984
53 0053 1 Remove local declaration of get_map_pointer linkage.
54 0054 1
55 0055 1 V03-011 LMP0254 L. Mark Pilant, 25-Jun-1984 10:12
56 0056 1 Don't clear the forward link for cathedral windows. This
57 0057 1 prevents the ACP from ACCVIOing when remapping already
```

58 0058 1 | existing cathedral windows.  
59 0059 1 |  
60 0060 1 | V03-010 CDS0001 Christian D. Saether 30-Dec-1983  
61 0061 1 | Use L\_NORM linkage and BIND\_COMMON macro.  
62 0062 1 |  
63 0063 1 | V03-009 LMP0037 L. Mark Pilant, 28-Jun-1982 15:10  
64 0064 1 | Remove the addressing mode module switch.  
65 0065 1 |  
66 0066 1 | V03-008 LMP0027 L. Mark Pilant, 18-May-1982 11:25  
67 0067 1 | Rearrange several instruction sequences to avoid possible  
68 0068 1 | page faults at an elevated IPL.  
69 0069 1 |  
70 0070 1 | V03-007 ACG0285 Andrew C. Goldstein, 12-Apr-1982 16:58  
71 0071 1 | Correct cathedral window logic for empty headers  
72 0072 1 |  
73 0073 1 | V03-006 LMP0016 L. Mark Pilant, 25-Mar-1982 13:23  
74 0074 1 | Remove diddling of the COMPLETE bit in the window segments.  
75 0075 1 |  
76 0076 1 | V02-005 LMP0013 L. Mark Pilant, 14-Mar-1982 13:25  
77 0077 1 | Fix window building logic to correctly handle Cathedral  
78 0078 1 | windows as well as the garden variety windows.  
79 0079 1 |  
80 0080 1 | V02-004 ACG0229 Andrew C. Goldstein, 23-Dec-1981 21:13  
81 0081 1 | Move counting of window turn to MAPVBN  
82 0082 1 |  
83 0083 1 | V02-003 LMP0003 L. Mark Pilant, 17-Nov-1981 14:35  
84 0084 1 | Added cathedral (segmented) window support.  
85 0085 1 |  
86 0086 1 | V02-002 ACG0167 Andrew C. Goldstein, 16-Apr-1980 19:28  
87 0087 1 | Previous revision history moved to F11B.REV  
88 0088 1 | \*\*  
89 0089 1 |  
90 0090 1 |  
91 0091 1 LIBRARY 'SYSSLIBRARY:LIB:L32';  
92 0092 1 REQUIRE 'SRCS:FCPDEF.B32';  
93 1083 1 |  
94 1084 1 |  
95 1085 1 |  
96 1086 1 | Lock the code into the working set since this routine runs at raised IPL.  
97 1087 1 |  
98 1088 1 |  
99 1089 1 | LOCK\_CODE;  
100 1090 1 GLOBAL ROUTINE TURN\_WINDOW (WINDOW, HEADER, DESIRED\_VBN, START\_VBN) : L\_NORM =  
101 1091 1 |  
102 1092 1 | ++  
103 1093 1 |  
104 1094 1 | FUNCTIONAL DESCRIPTION:  
105 1095 1 |  
106 1096 1 | This routine scans the map area of the supplied file header  
107 1097 1 | and builds retrieval pointers in the window until  
108 1098 1 | (1) the entire header has been scanned, or  
109 1099 1 | (2) the first retrieval pointer in the window maps the desired VBN  
110 1100 1 |  
111 1101 1 | CALLING SEQUENCE:  
112 1102 1 | TURN\_WINDOW (ARG1, ARG2, ARG3, ARG4)  
113 1103 1 |  
114 1104 1 | INPUT PARAMETERS:

: 115 1105 1 | ARG1: address of window block or 0 if to be created  
: 116 1106 1 | ARG2: address of file header  
: 117 1107 1 | ARG3: desired VBN  
: 118 1108 1 | ARG4: starting VBN of file header  
  
: 119 1109 1 |  
: 120 1110 1 | IMPLICIT INPUTS:  
: 121 1111 1 | CURRENT\_RVN: RVN of current volume  
: 122 1112 1 |  
: 123 1113 1 | OUTPUT PARAMETERS:  
: 124 1114 1 | updated window  
: 125 1115 1 |  
: 126 1116 1 | IMPLICIT OUTPUTS:  
: 127 1117 1 | NONE  
: 128 1118 1 |  
: 129 1119 1 | ROUTINE VALUE:  
: 130 1120 1 | address of created window  
: 131 1121 1 | or 1 if none created  
: 132 1122 1 |  
: 133 1123 1 | SIDE EFFECTS:  
: 134 1124 1 | NONE  
: 135 1125 1 |  
: 136 1126 1 |--  
: 137 1127 1 |  
: 138 1128 2 BEGIN  
: 139 1129 2 |  
: 140 1130 2 MAP  
: 141 1131 2 | WINDOW : REF BBLOCK; ! pointer to window  
: 142 1132 2 | HEADER : REF BBLOCK; ! pointer to file header  
: 143 1133 2 |  
: 144 1134 2 GLOBAL REGISTER  
: 145 1135 2 | COUNT = 6, ! retrieval pointer count  
: 146 1136 2 | LBN = 7, ! retrieval pointer start LBN  
: 147 1137 2 | MAP\_POINTER = 8 : REF BBLOCK; ! pointer to scan header map area  
: 148 1138 2 |  
: 149 1139 2 LABEL  
: 150 1140 2 | WINDOW\_INIT, ! window initialization logic  
: 151 1141 2 | MAP\_BUILD; ! loop to build window map  
: 152 1142 2 |  
: 153 1143 2 LOCAL  
: 154 1144 2 | WINDOW\_BUFFER : BBLOCK [MAX\_WINDOW\*6], ! buffer in which to build window  
: 155 1145 2 | POINTER\_COUNT, ! count of pointers in window  
: 156 1146 2 | WINDOW\_SIZE, ! size of window in pointers  
: 157 1147 2 | BASE\_VBN, ! starting VBN in window  
: 158 1148 2 | VBN, ! VBN in scanning window  
: 159 1149 2 | W\_POINTER : REF BBLOCK, ! pointer to scan window  
: 160 1150 2 | M\_POINTER : REF BBLOCK, ! previous header pointer  
: 161 1151 2 | NEW\_WINDOW : REF BBLOCK, ! address of newly allocated window  
: 162 1152 2 | WINDOW\_COUNT, ! number of block mapped by the window  
: 163 1153 2 | PRIMARY\_WINDOW : REF BBLOCK; ! address of the primary window segment  
: 164 1154 2 |  
: 165 1155 2 MACRO  
: 166 1156 2 | WINDOW\_MAP = (.WINDOW+WCBSC\_MAP)%; !start of window map area  
: 167 1157 2 |  
: 168 1158 2 BIND\_COMMON;  
: 169 1159 2 |  
: 170 1160 2 EXTERNAL ROUTINE  
: 171 1161 2 | ALLOCATE : L\_NORM; ! allocate system dynamic memory

```
172  
173 1162 2 EXTERNAL ROUTINE  
174 1163 2 GET_MAP_POINTER : L_MAP_POINTER; ! get value of next header map pointer  
175 1164 2  
176 1165 2 ! There are two general cases that can occur upon entering the window turner:  
177 1166 2 1) a window previously exists, in which case things get a little messy; or  
178 1167 2 2) no window previously exists, in which case things are very simply.  
179 1168 2  
180 1169 2  
181 1170 2  
182 1171 2 W_POINTER = WINDOW_BUFFER;  
183 1172 2 WINDOW_COUNT = 0;  
184 1173 2  
185 1174 2 M_POINTER = 0;  
186 1175 2 MAP_POINTER = .HEADER + .HEADER[FH2$B_MPOFFSET] * 2; ! point to map area  
187 1176 2  
188 1177 2 IF .WINDOW EQ 0  
189 1178 2 THEN  
190 1179 2  
191 1180 2 ! Create a new window. All that is necessary is to initialize a few variables  
192 1181 2 and pointers and then start reading the file header.  
193 1182 2  
194 1183 2  
195 1184 3 BEGIN  
196 1185 3 BASE_VBN = .START_VBN;  
197 1186 3 WINDOW_SIZE = MAX_WINDOW;  
198 1187 3 PRIMARY_WINDOW = NEW_WINDOW = 0;  
199 1188 3 POINTER_COUNT = 0;  
200 1189 3 END  
201 1190 2 ELSE  
202 1191 2  
203 1192 2 ! Use an existing window. Several situation may occur in this case: 1) the  
204 1193 2 window must be turned to point to a different portion of the file; 2) the  
205 1194 2 header contains pointers which may be added after truncating the existing  
206 1195 2 window, this usually occurs when a file is extended without causing a new  
207 1196 2 file header to be created; 3) the desired VBN is less than the specified  
208 1197 2 starting VBN and the starting VBN is greater than 1, this occurs when a  
209 1198 2 file is extended and a new file header had to be created; or 4) the window  
210 1199 2 already maps a portion of the header and only the new pointers (which may  
211 1200 2 include a partial map pointer if two contiguous extents were collapsed into  
212 1201 2 one map pointer in the header).  
213 1202 2  
214 1203 2  
215 1204 3 WINDOW_INIT: BEGIN  
216 1205 3 BASE_VBN = .WINDOW[WCB$L STVBN];  
217 1206 3 WINDOW_SIZE = MINU ((.WINDOW[WCB$W_SIZE] - WCB$C_LENGTH) / 6, MAX_WINDOW);  
218 1207 3 PRIMARY_WINDOW = NEW_WINDOW = .WINDOW;  
219 1208 3 POINTER_COUNT = .WINDOW[WCB$W_NMAP];  
220 1209 3 CH$MOVE (.POINTER_COUNT * 6, WINDOW_MAP, WINDOW_BUFFER); ! copy current  
221 1210 3 VBN = .BASE_VBN;  
222 1211 3  
223 1212 3 IF .START_VBN LEQU .VBN  
224 1213 3 OR  
225 1214 3  
226 1215 3 ! Determine if the existing window may be truncated.  
227 1216 3  
228 1217 3  
1218 4 BEGIN
```

```
229      1219 4      INCR J FROM 1 TO .POINTER_COUNT
230      1220 4
231      1221 5      DO
232      1222 5      BEGIN
233      1223 5      VBN = .VBN + .W_POINTER[WCB$W_COUNT];      ! VBN at the end of the pointer
234      1224 5      WINDOW_COUNT = -WINDOW_COUNT + .W_POINTER[WCB$W_COUNT];
235      1225 5      W_POINTER = .W_POINTER + 6;
236      1226 5      IF .START_VBN EQL .VBN
237      1227 6      THEN
238      1228 6      BEGIN
239      1229 6      POINTER_COUNT = .J;                      ! note where window truncated
240      1230 5      LEAVE WINDOW_INIT;
241      1231 4      END;
242      1232 4      NOT .WINDOW[WCB$V_CATHEDRAL]
243      1233 4      END
244      1234 3      THEN
245      1235 3      | Either the window cannot be truncated or the header maps before the beginning
246      1236 3      | of the existing window. In the latter case if cathedral windows are not in
247      1237 3      | use the window may be discarded; if cathedral windows are in use then it is
248      1238 3      | possible that a beginning portion of the header may be discarded.
249      1239 3
250      1240 3
251      1241 4      BEGIN
252      1242 4      IF .WINDOW[WCB$V_CATHEDRAL]
253      1243 4      THEN
254      1244 5      BEGIN
255      1245 5      IF .BASE_VBN EQL .START_VBN
256      1246 5      THEN
257      1247 6      BEGIN
258      1248 6      POINTER_COUNT = 0;
259      1249 6      LEAVE WINDOW_INIT;
260      1250 5      END;
261      1251 5      IF
262      1252 6      BEGIN
263      1253 6      VBN = .START_VBN;
264      1254 6      UNTIL .MAP_POINTER GEQA .HEADER + (.HEADER[FH2$B_MPOFFSET] + .HEADER[FH2$B_MAP_INUSE]) * 2
265      1255 6      DO
266      1256 7      BEGIN
267      1257 7      M_POINTER = MAP_POINTER;
268      1258 7      GET_MAP_POINTER (?);
269      1259 7      IF .BASE_VBN GEQ .VBN
270      1260 7      AND .BASE_VBN LSS .VBN + .COUNT
271      1261 7      THEN
272      1262 8      BEGIN
273      1263 8      MAP_POINTER = .M_POINTER;          ! back up the header pointer
274      1264 8      POINTER_COUNT = 0;
275      1265 8      LEAVE WINDOW_INIT;
276      1266 7      END;
277      1267 7      VBN = .VBN + .COUNT;
278      1268 7      END
279      1269 6      END
280      1270 5      THEN BUG_CHECK (WCBFCBMNG, FATAL, 'WCB/FCB correspondence broken');
281      1271 5
282      1272 4      ELSE
283      1273 5      BEGIN
284      1274 5      IF .VBN EQLU .BASE_VBN
285      1275 5      AND .DESIRED_VBN LSSU .START_VBN
```

```
: 286      1276 5     AND .START VBN GTRU 1
: 287      1277 5     THEN RETURN 1;                      ! note window already best fit
: 288
: 289      1279 5     | The header maps before the existing window and cathedral windows are not in
: 290      1280 5     | use, or a window truncation was attempted and starting VBN of the FCB was not
: 291      1281 5     | found in the window. In either case a new window must be formed.
: 292
: 293      1283 5
: 294      1284 5     POINTER COUNT = 0;
: 295      1285 5     WINDOW COUNT = 0;
: 296      1286 5     BASE VBN = .START VBN;
: 297      1287 5     W_POINTER = WINDOW_BUFFER;
: 298      1288 4     END;
: 299      1289 3     END;
: 300      1290 2     END;                                ! end of block WINDOW_INIT
: 301
: 302      1292 2     | The window is now suitably initialized. Set up necessary pointers.
: 303      1293 2     | Now scan the map area, extracting retrieval pointers.
: 304
: 305      1295 2
: 306      1296 3     MAP BUILD: BEGIN
: 307      1297 3     UNTIL .MAP_POINTER GEQA .HEADER + (.HEADER[FH2$B_MPOFFSET] + .HEADER[FH2$B_MAP_INUSE]) * 2
: 308      1298 3     DO
: 309      1299 4     BEGIN
: 310      1300 4
: 311      1301 4     GET_MAP_POINTER ();
: 312      1302 4
: 313      1303 4     | If the existing window's first map pointer is part of a set needed to map
: 314      1304 4     | a map pointer from the header (this only occurs if the map pointer from the
: 315      1305 4     | file header maps more than 65535 blocks), it is necessary to adjust the
: 316      1306 4     | block count and LBN returned from the header.
: 317      1307 4
: 318      1308 4
: 319      1309 4     IF .M_POINTER NEQ 0
: 320      1310 4     THEN
: 321      1311 5     BEGIN
: 322      1312 5     COUNT = .COUNT - (.BASE_VBN - .VBN);
: 323      1313 5     LBN = .LBN + (.BASE_VBN - .VBN);
: 324      1314 5     M_POINTER = 0;                      ! only needed once
: 325      1315 4     END;
: 326      1316 4
: 327      1317 4     | Build new retrieval pointers, using as many as needed to run out the
: 328      1318 4     | count. If the window is full, and segmented windows are not allowed,
: 329      1319 4     | shuffle the entries up by one. If this would cause the pointer mapping
: 330      1320 4     | the desired VBN to fall off the top, we are done. If segmented windows
: 331      1321 4     | are allowed, write out the current segment, allocate room for the next
: 332      1322 4     | segment, reset the critical counts and pointers, and continue on my
: 333      1323 4     | merry way.
: 334
: 335      1325 4
: 336      1326 4     IF .COUNT NEQ 0
: 337      1327 4     THEN
: 338      1328 4     WHILE 1 DO
: 339      1329 5     BEGIN
: 340      1330 5     IF .POINTER_COUNT GEQU .WINDOW_SIZE
: 341      1331 5     THEN
: 342      1332 6     BEGIN
```

```
: 343      1333 6      IF .WINDOW NEQ 0 AND NOT .WINDOW[WCB$V_CATHEDRAL]
: 344      1334 6      THEN
: 345      1335 7      BEGIN
: 346      1336 7      CLEANUP_FLAGS[CLF_INCOMPLETE] = 1;
: 347      1337 7      IF .BASE_VBN + .WINDOW_BUFFER[WCB$W_COUNT] GTRU .DESIRED_VBN
: 348      1338 7      THEN LEAVE MAP_BUILD;
: 349      1339 7
: 350      1340 7      POINTER_COUNT = .POINTER_COUNT - 1;
: 351      1341 7      BASE_VBN = .BASE_VBN + .WINDOW_BUFFER[WCB$W_COUNT];
: 352      1342 7      CHSMOVE (.POINTER_COUNT*6, WINDOW_BUFFER+6, WINDOW_BUFFER);
: 353      1343 7      W_POINTER = .W_POINTER - 6;
: 354      1344 7      END
: 355      1345 6      ELSE
: 356      1346 7      BEGIN
: 357      1347 7      LOCAL TEMP_LINK : REF BBLOCK; ! addr of temp window link
: 358      1348 7      IF .PRIMARY_WINDOW EQ 0
: 359      1349 7      THEN
: 360      1350 8      BEGIN
: 361      1351 9      PRIMARY_WINDOW = NEW_WINDOW = ALLOCATE ((MAXU (.POINTER_COUNT, MIN_WINDOW))
: 362      1352 8                      * 6 + WCB$C_LENGTH, WCB_TYPE);
: 363      1353 8
: 364      1354 8      IF .NEW_WINDOW EQ 0
: 365      1355 9      THEN
: 366      1356 9      BEGIN
: 367      1357 9      CLEANUP_FLAGS[CLF_INCOMPLETE] = 1;
: 368      1358 8      RETURN 0;
: 369      1359 7      END;
: 370      1360 7      IF .NEW_WINDOW[WCB$L_LINK] NEQ 0
: 371      1361 7      THEN TEMP_LINK = .NEW_WINDOW[WCB$L_LINK]
: 372      1362 7      ELSE
: 373      1363 8      BEGIN
: 374      1364 8      WINDOW_SIZE = MAX_WINDOW;
: 375      1365 8      TEMP_LINK = ALLOCATE (.WINDOW_SIZE * 6 + WCB$C_LENGTH, WCB_TYPE);
: 376      1366 8      IF .TEMP_LINK EQ 0
: 377      1367 8      THEN
: 378      1368 9      BEGIN
: 379      1369 9      CLEANUP_FLAGS[CLF_INCOMPLETE] = 1;
: 380      1370 9      LEAVE MAP_BUILD;
: 381      1371 8      END;
: 382      1372 7      END;
: 383      1373 7      SET_IPL (IPL$_SYNCH);
: 384      1374 7
: 385      1375 7 ! Copy the needed information from the previous window segment.
: 386      1376 7
: 387      1377 7
: 388      1378 7      TEMP_LINK[WCB$L_PID] = .NEW_WINDOW[WCB$L_PID];
: 389      1379 7      TEMP_LINK[WCB$L_ORGUCB] = .NEW_WINDOW[WCB$L_ORGUCB];
: 390      1380 7      TEMP_LINK[WCB$W_ACON] = .NEW_WINDOW[WCB$W_ACON];
: 391      1381 7      TEMP_LINK[WCB$L_FCB] = .NEW_WINDOW[WCB$L_FCB];
: 392      1382 7      TEMP_LINK[WCB$B_ACCESS] = .NEW_WINDOW[WCB$B_ACCESS];
: 393      1383 7      TEMP_LINK[WCB$L_RVT] = .NEW_WINDOW[WCB$L_RVT];
: 394      1384 7
: 395      1385 7 ! Finish up the current segment and create another.
: 396      1386 7
: 397      1387 7
: 398      1388 7      NEW_WINDOW[WCB$L_LINK] = .TEMP_LINK;
: 399      1389 7      NEW_WINDOW[WCB$V_CATHEDRAL] = T;
```

```
: 400      1390 7      NEW_WINDOW[WCB$W_NMAP] = .POINTER_COUNT;
: 401      1391 7      NEW_WINDOW[WCB$L-STVBN] = .BASE_VBN;
: 402      1392 7      CHSMOVE (.POINTER_COUNT*6, WINDOW_BUFFER, .NEW_WINDOW+WCB$C_MAP);
: 403      1393 7      NEW_WINDOW = .NEW_WINDOW[WCB$L_LINK];
: 404      1394 7      BASE_VBN = .BASE_VBN + .WINDOW_COUNT;
: 405      1395 7      NEW_WINDOW[WCB$W_NMAP] = 0;
: 406      1396 7      NEW_WINDOW[WCB$L-STVBN] = BASE_VBN;
: 407      1397 7      NEW_WINDOW[WCB$V_CATHEDRAL] = 1;
: 408      1398 7      SET_IPL (0);
: 409      1399 7      W_POINTER = WINDOW_BUFFER;
: 410      1400 7      POINTER_COUNT = 0;
: 411      1401 7      WINDOW_COUNT = 0;
: 412      1402 6      END;
: 413      1403 5      END;
: 414      1404 5
: 415      1405 5 ! Finally build the pointer and count it.
: 416      1406 5 !
: 417      1407 5
: 418      1408 5      W_POINTER[WCB$W_COUNT] = MINU (.COUNT, 65535);
: 419      1409 5      WINDOW_COUNT = .WINDOW_COUNT + .W_POINTER[WCB$W_COUNT];
: 420      1410 5      W_POINTER[WCB$L_LBN] = .LBN;
: 421      1411 5      (@_POINTER[WCB$C_LBN])<24,8> = .CURRENT_RVN;
: 422      1412 5      W_POINTER = .W_POINTER + 6;
: 423      1413 5      POINTER_COUNT = .POINTER_COUNT + 1;
: 424      1414 5      LBN = .[BN + 65535];
: 425      1415 5      COUNT = .COUNT - MINU (.COUNT, 65535);
: 426      1416 5      IF .COUNT EQL 0 THEN EXITLOOP;
: 427      1417 4      END;
: 428      1418 4
: 429      1419 3      END;                                ! end of header scan loop
: 430      1420 2      END;                                ! end of block MAP_BUILD
: 431      1421 2
: 432      1422 2 ! Having built a new window in the buffer, update the real one. Then interlock
: 433      1423 2 the system data base and copy the buffer into the window.
: 434      1424 2
: 435      1425 2
: 436      1426 2 IF .NEW_WINDOW EQL 0
: 437      1427 2 THEN
: 438      1428 3      BEGIN
: 439      1429 3      IF .WINDOW EQL 0
: 440      1430 3      THEN
: 441      1431 4      BEGIN
: 442      1432 5      PRIMARY_WINDOW = NEW_WINDOW = ALLOCATE ((MAXU (.POINTER_COUNT, MIN_WINDOW))
: 443      1433 4                                  * 6 + WCB$C_LENGTH, WCB_TYPE);
: 444      1434 4      IF .NEW_WINDOW EQL 0
: 445      1435 4      THEN
: 446      1436 5      BEGIN
: 447      1437 5      CLEANUP_FLAGS[CLF_INCOMPLETE] = 1;
: 448      1438 5      RETURN 0;
: 449      1439 4      END;
: 450      1440 4      NEW_WINDOW[WCB$V_CATHEDRAL] = 1;
: 451      1441 4      END
: 452      1442 3      ELSE NEW_WINDOW = .WINDOW;
: 453      1443 2      END;
: 454      1444 2
: 455      1445 2      SET_IPL (IPL$ SYNCH);
: 456      1446 2      NEW_WINDOW[WCB$W_NMAP] = .POINTER_COUNT;
```

! end of routine TURN\_WINDOW

						.TITLE	WITURN	
						.IDENT	\V04-000\	
						.EXTRN	ALLOCATE, GET_MAP_POINTER	
						.EXTRN	BUGS_WCBFCBMNG	
						.PSECT	\$LOCKEDC1\$,NOWRT,2	
			OBFC 00000			.ENTRY	TURN_WINDOW, Save R2,R3,R4,R5,R6,R7,R8,R9,-	1090
						R11		
						MOVAB -508(SP), SP		1171
						MOVAB WINDOW_BUFFER, W_POINTER		1174
						CLRQ M_POINTER		1175
						MOVL HEADER, R1		
						MOVZBL 1(R1), R0		
						MOVAW (R1)[R0], MAP_POINTER		
						MOVL WINDOW, R6		1177
						BNEQ 1\$		
						MOVL START_VBN, BASE_VBN		1185
						MOVZBL #80, WINDOW_SIZE		1186
						CLRQ PRIMARY_WINDOW		1187
						BRW 7\$		1188
						MOVL 44(R6), BASE_VBN		1205
						MOVZWL 8(R6), R0		1206
						SUBL2 #48, R0		
						DIVL2 #6, R0		
						CMPL R0, #80		
						BLEQU 2\$		
						MOVZBL #80, R0		
						MOVL R0, WINDOW_SIZE		
						MOVL R6, NEW_WINDOW		1207
						MOVL R6, PRIMARY_WINDOW		1208
						MOVZWL 22(R6), POINTER_COUNT		1209
						MOVL POINTER_COUNT, R7		
						MULL3 #6, R7-R0		
						MOVC3 R0, 48(R6), WINDOW_BUFFER		
						MOVL BASE_VBN, VBN		1210
						CMPL START_VBN, VBN		1211
						BLEQU 5\$		1212
						CLRL J		1219
						BRB 4\$		
						MOVZWL RW_POINTER, R1		1222
						ADDL2 R1, VBN		
						MOVZWL RW_POINTER, R1		1223
						ADDL2 R1, WINDOW_COUNT		
						ADDL2 #6, W_POINTER		1224

	18 AE	10 AC	D1 00089	CMPL	START_VBN, VBN	1225
	59	05 12	0008E	BNEQ	4\$	1228
		50 D0	00090	MOVL	J POINTER_COUNT	1229
DC	50	5A 11	00093	BRB	10\$	1219
77	0B A6	57 F3	00095	AOBLEQ	R7, J 3\$	1232
4E	0B A6	06 E0	00099	BBS	#6, 11(R6), 13\$	1242
	10 AC	06 E1	0009E	BBC	#6, 11(R6), 11\$	1245
		5B D1	000A3	CMPL	BASE_VBN, START_VBN	DE
		38 13	000A7	BEQL	7\$	L1
	18 AE	10 AC	D0 000A9	MOVL	START_VBN, VBN	1253
	51 08	AC 00	000AE	MOVL	HEADER, R1	1254
	50 01	A1 9A	000B2	MOVZBL	1(R1), R0	
	52 3A	A1 9A	000B6	MOVZBL	58(R1), R2	
	50	52 C0	000BA	ADDL2	R2, R0	
	50	6140 3E	000BD	MOVAW	(R1)[R0], R0	
	50	58 D1	000C1	CMPL	MAP_POINTER, R0	
		25 1E	000C4	BGEQU	9\$	
	OC AE	58 D0	000C6	MOVL	MAP_POINTER, M_POINTER	1257
		0000G 30	000CA	BSBW	GET_MAP_POINTER	1258
	18 AE	5B D1	000CD	CMPL	BASE_VBN, VBN	1259
50	56	12 19	000D1	BLSS	8\$	
	50	18 AE	C1 000D3	ADDL3	VBN, COUNT, R0	1260
	58 08	18 000DB	5B D1	CMPL	BASE_VBN, R0	
	58 0C	AE D0	000DD	BGEQ	8\$	
		59 D4	000E1	MOVL	M_POINTER, MAP_POINTER	1263
		30 11	000E3	CLRL	POINTER_COUNT	1264
	18 AE	56 C0	000E5	7\$: BRB	13\$	1265
		C3 11	000E9	ADDL2	COUNT, VBN	1267
		FEFF 000EB	9\$: BRB	6\$	1254	
		0000* 000ED	BUGW	<BUGS_WCBFCBMNG!4>	1270	
	5B 18	AE D1	000EF	WORD		
		10 12	000F1	10\$: BRB	13\$	1242
	10 AC	0C AC	D1 000F7	CMPL	VBN, BASE_VBN	1274
	01 10	AC D1	000FC	BNEQ	12\$	
		09 1E	000FE	CMPL	DESIRED_VBN, START_VBN	1275
		03 1B	00102	BGEQU	12\$	
		0222 31	00104	CMPL	START_VBN, #1	1276
	14 AE	10 AE	D4 00107	BLEQU	12\$	
		10 AC	D4 00109	BRW	33\$	
		10 AC	D0 0010C	CLRL	POINTER_COUNT	1284
		1C AE	9E 00110	CLRL	WINDOW_COUNT	1285
		51 08	AC D0	MOVAB	START_VBN, BASE_VBN	1286
		00115	00115	MOVAB	WINDOW_BUFFER, Q_POINTER	1287
		50 01	A1 9A	MOVZBL	HEADER, R1	1297
		52 3A	A1 9A	MOVZBL	1(R1), R0	
		52 C0	0011D	MOVZBL	58(R1), R2	
		50	6140 3E	ADDL2	R2, R0	
		50	58 D1	MOVAW	(R1)[R0], R0	
		03 1F	00128	CMPL	MAP_POINTER, R0	
		018E 31	0012D	BLSSU	14\$	
		0000G 30	00130	BRW	27\$	
	50 18	AE D5	00133	BSBW	GET_MAP_POINTER	1301
		OE 13	00136	TSTL	M_POINTER	1309
		5B C3	00138	BEQL	15\$	
		50 C0	0013D	SUBL3	BASE_VBN, VBN, R0	1312
		57 50	C2 00140	ADDL2	RO, COUNT	
				SUBL2	RO, LBN	1313

			OC	AE	D4	00143		CLRL	M_POINTER		1314		
				56	D5	00146	15\$:	TSTL	COUNT		1326		
				CB	13	00148		BEQL	13\$				
			08	AE	59	D1	0014A	16\$:	CMPL	POINTER_COUNT, WINDOW_SIZE		1330	
				50	50	33	1F	0014E	BLSSU	17\$			
				04	AC	30	13	00150	MOVL	WINDOW, R0		1333	
			2B	0B	A0	06	E0	00156	BEQL	18\$			
				01	AA	04	88	0015B	BBS	#6, 11(R0), 18\$			
				50	50	AE	3C	0015F	BISB2	#4, 1(BASE)		1336	
				50	50	5B	C0	00163	MOVZWL	WINDOW_BUFFER, R0		1337	
				OC	AC	50	D1	00166	ADDL2	BASE_VBN, R0			
						6F	1A	0016A	CMPL	R0 DESIRED_VBN			
						59	D7	0016C	BGTRU	22\$			
						AE	3C	0016E	DECL	POINTER_COUNT		1340	
						50	50	00172	MOVZWL	WINDOW_BUFFER, R0		1341	
			1C			06	C5	00175	ADDL2	R0, BASE_VBN			
						50	28	00179	MULL3	#6, POINTER_COUNT, R0		1342	
						06	C2	0017F	MOVC3	R0, WINDOW_BUFFER+6, WINDOW_BUFFER			
						00F2	31	00183	SUBL2	#6, W_POINTER			
						6E	D5	00186	17\$:	BRW	24\$		
						22	12	00188	TSTL	PRIMARY_WINDOW			
						01	DD	0018A	BNEQ	20\$			
						50	59	0018C	PUSHL	#1		1343	
						03	12	0018F	MOVL	POINTER_COUNT, R0			
						50	01	00191	BNEQ	19\$			
						50	06	C4	00194	19\$:	MOVL	#1, R0	
						30	A0	9F	00197	MULL2	#6, R0		
			0000G	CF	02	FB	0019A	PUSHAB	48(R0)	CALLS	#2, ALLOCATE		
			04	AE	50	D0	0019F			MOVL	R0, NEW_WINDOW		
				6E	04	AE	D0	001A3		MOVL	NEW_WINDOW, PRIMARY_WINDOW		
						03	12	001A7	BNEQ	20\$			
						013B	31	001A9	BRW	29\$			
						20	C1	001AC	ADDL3	#32, NEW_WINDOW, R0		1352	
						60	D5	001B1	TSTL	(R0)			
						0A	13	001B3	BEQL	21\$			
						20	C1	001B5	ADDL3	#32, NEW_WINDOW, R1			
			51	04	AE	61	D0	001BA	MOVL	(R1), TEMP_LINK		1361	
						1F	11	001BD	BRB	23\$			
						08	AE	50	MOVZBL	#80, WINDOW_SIZE		1364	
						01	DD	001C4	PUSHL	#1		1365	
			51	0C	AE	06	C5	001C6	MULL3	#6, WINDOW_SIZE, R1			
				0000G	CF	30	A1	9F	PUSHAB	48(R1)	CALLS	#2, ALLOCATE	
						02	FB	001CE		TSTL	TEMP_LINK		
						50	D5	001D3		BNEQ	23\$		
						07	12	001D5	BRW	24\$			
						01	AA	04	BISB2	#4, 1(BASE)		1369	
						08	DA	001DE	22\$:	BRW	27\$		
						08	DA	001DE	23\$:	MTPR	#8, #18		
			51	04	AE	0C	C1	001E1	ADDL3	#12, NEW_WINDOW, R1			
				0C	A0	61	D0	001E6	MOVL	(R1), 12TEMP_LINK)		1370	
				51	04	AE	10	C1	001EA	ADDL3	#16, NEW_WINDOW, R1		
						61	D0	001EF	MOVL	(R1), 16TEMP_LINK)		1373	
				51	04	AE	14	C1	001F3	ADDL3	#20, NEW_WINDOW, R1		
						61	B0	001F8	MOVW	(R1), 20TEMP_LINK)		1378	
				51	04	AE	18	C1	001FC	ADDL3	#24, NEW_WINDOW, R1		
						61	D0	00201	MOVL	(R1), 24TEMP_LINK)		1379	
												1380	
												1381	

51	04	AE		OB	C1	00205	ADDL3	#11, NEW WINDOW, R1 (R15, 11TEMP LINK)	1382	
51	04	AE		61	90	0020A	MOVB		1383	
51	04	AE		1C	C1	0020E	ADDL3	#28, NEW WINDOW, R1 (R15, 28TEMP LINK)	Sy --	
51	04	AE		61	D0	00213	MOVL		1388	
51	04	AE		20	C1	00217	ADDL3	#32, NEW WINDOW, R1 TEMP LINR, (R1)	FA	
50	04	61		50	D0	0021C	ADDL3	#11, NEW WINDOW, R0 BISB2 #64, (R0)	FA	
50	04	AE	40	OB	C1	0021F	ADDL3	#22, NEW WINDOW, R0 MOVW POINTER COUNT, (R0)	FA	
50	04	60		8F	88	00224	ADDL3	#44, NEW WINDOW, R0 ADDL3 #44, NEW_WINDOW, R0	FA	
50	04	AE		16	C1	00228	MOVL	BASE_VBN-, (R0)	FA	
50	04	60		59	B0	0022D	MULL3	#6, POINTER COUNT, R0 #48, NEW WINDOW, -(SP)	FA	
50	04	AE		2C	C1	00230	ADDL3	RO, WINDOW BUFFER, @SP+ #32, NEW WINDOW, R0	FA	
50	04	60		5B	D0	00235	MOVL	(R0), NEW WINDOW ADDL2 WINDOW COUNT, BASE_VBN	FA	
50	04	59		06	C5	00238	MULL3	#6, POINTER COUNT, R0 #48, NEW WINDOW, -(SP)	FA	
7E	04	AE		30	C1	0023C	ADDL3	RO, WINDOW BUFFER, @SP+ #32, NEW WINDOW, R0	FA	
9E	20	AE		50	28	00241	MOVC3	(R0), NEW WINDOW ADDL3 #32, NEW WINDOW, R0	FA	
50	04	AE		20	C1	00246	MOVL	ADDL2 WINDOW COUNT, BASE_VBN #22, NEW_WINDOW, R0	FA	
50	04	AE		60	D0	0024B	CLRW	(R0)	FA	
50	04	5B	10	AE	C0	0024F	ADDL3	#44, NEW_WINDOW, R0 ADDL3 #11, NEW_WINDOW, R0	FA	
50	04	AE		16	C1	00253	MOVL	BASE_VBN-, (R0)	FA	
50	04	60		60	B4	00258	ADDL3	#64, (R0)	FA	
50	04	AE		2C	C1	0025A	MOVL	MTPR #0, #18	FA	
50	04	60		5B	D0	0025F	ADDL3	WINDOW BUFFER, W_POINTER CLRL POINTER COUNT	FA	
50	04	AE	40	OB	C1	00262	MOVA	CLRL WINDOW_COUNT COUNT, R0	FA	
50	04	60		8F	88	00267	CLRL	CMPL RO, #65535	FA	
14	AE		12	00	DA	0026B	BLEQU	25\$	FA	
14	AE		1C	AE	9E	0026E	MOVZWL	#65535, R0	FA	
			10	59	D4	00273	MOVW	RO, @W_POINTER	FA	
			10	AE	D4	00275	MOVZWL	@W_POINTER, R1	FA	
0000FFFF	50		50	56	D0	00278	24\$: ADDL2	R1, WINDOW COUNT	FA	
	8F		8F	50	D1	0027B	MOVW	#2, W_POINTER, R1	FA	
		FFFF	05	1B	00282	MOVZWL	MOVAQ (LBN)‡, (R1)	1409		
			14	50	B0	00284	25\$: ADDL3	#5, W_POINTER, R1	FA	
			14	51	BE	00289	MOVW	-96(BASE), (R1)	1410	
			10	AE	51	C0	00291	ADDL2	ADDL2 #6, W_POINTER POINTER COUNT	FA
51	14	AE		02	C1	00295	ADDL3	INCL #2, W_POINTER, R1	1411	
51	14	61		87	7E	0029A	MOVAQ	65527(R7), LBN	FA	
51	14	AE		05	C1	0029D	ADDL3	SUBL2 RO, COUNT	1412	
			14	61	AA	90	002A2	MOVB	26\$	FA
			14	AE	06	C0	002A6	ADDL2	BRW 13\$	FA
			57	0000FFF7	59	D6	002AA	INCL	BRW 16\$	FA
			56		50	C2	002B3	MOVAB	NEW_WINDOW	FA
				03	12	002B6	SUBL2	BNEQ 32\$	FA	
				FE5A	31	002B8	BNEQ	WINDOW	FA	
				FE8C	31	002BB	26\$: TSTL	BNEQ 31\$	FA	
				04	AE	D5	002BE	TSTL	PUSHL #1	FA
				3A	12	002C1	BNEQ	POINTERS COUNT, R0	FA	
				04	AC	D5	002C3	TSTL	28\$	FA
				30	12	002C6	BNEQ	MOVBL #1, R0	FA	
				01	DD	002C8	PUSHL	MULL2 #6, R0	FA	
			50	59	D0	002CA	MOVBL	PUSHAB 48(R0)	FA	
			50	03	12	002CD	BNEQ	CALLS #2, ALLOCATE	FA	
			50	01	DD	002CF	MOVL		FA	
			50	06	C4	002D2	28\$: MULL2		FA	
			30	A0	9F	002D5	PUSHAB		FA	
0000G	CF		02	FB	002D8		CALLS		FA	

	04	AE		04	50	D0	002DD	MOVL	R0, NEW_WINDOW		
		6E			AE	D0	002E1	MOVL	NEW_WINDOW, PRIMARY_WINDOW		1432
					06	12	002E5	BNEQ	30\$:		1434
	01	AA			04	88	002E7	BISB2	#4, 1(BASE)		1437
					40	11	002EB	BRB	34\$:		1438
50	04	AE		40	0B	C1	002ED	ADDL3	#11, NEW_WINDOW, R0		1440
		60			8F	88	002F2	BISB2	#64, (R0)		
					05	11	002F6	BRB	32\$:		1429
	04	AE		04	AC	D0	002F8	MOVL	WINDOW, NEW_WINDOW		1442
		12			08	DA	002FD	MTPR	#8, #18		1445
50	04	AE			16	C1	00300	ADDL3	#22, NEW_WINDOW, R0		1446
		60			59	B0	00305	MOVW	POINTER COUNT, (R0)		
50	04	AE			2C	C1	00308	ADDL3	#44, NEW_WINDOW, R0		1447
		60			5B	D0	0030D	MOVL	BASE_VBN, (R0)		
		59			06	C4	00310	MULL2	#6, R9		1448
56	04	AE			30	C1	00313	ADDL3	#48, NEW_WINDOW, R6		
66	04	AE			59	28	00318	MOVC3	R9, WINDOW_BUFFER, (R6)		
		1C			00	DA	0031D	MTPR	#0, #18		1449
				04	AC	D5	00320	TSTL	WINDOW		1451
		12			04	12	00323	BNEQ	33\$:		
					50	6E	00325	MOVL	PRIMARY_WINDOW, R0		1453
						04	00328	RET			
					50	01	00329	MOVL	#1, R0		
						04	0032C	RET			
					50	D4	0032D	CLRL	RO		1455
						04	0032F	RET			

; Routine Size: 816 bytes. Routine Base: \$LOCKEDC1\$ + 0000

```
: 467      1456 1 GLOBAL ROUTINE MARK_INCOMPLETE (FIRST_BLOCK) : NOVALUE =
: 468      1457 1
: 469      1458 1 ++
: 470      1459 1
: 471      1460 1 FUNCTIONAL DESCRIPTION:
: 472      1461 1
: 473      1462 1 This routine starts from the specified window and marks all
: 474      1463 1 successive windows in the list as being incomplete. The list
: 475      1464 1 may start with the CURRENT_WINDOW; in which case the current
: 476      1465 1 and all successive window segments are marked as incomplete.
: 477      1466 1 It may also start with the PRIMARY_FCB; in which case all the
: 478      1467 1 window segments associated with the file are marked incomplete.
: 479      1468 1
: 480      1469 1 CALLING SEQUENCE:
: 481      1470 1     MARK_INCOMPLETE (ARG1)
: 482      1471 1
: 483      1472 1 INPUT PARAMETERS:
: 484      1473 1     ARG1: address of the first block in the list (either a WCB or FCB)
: 485      1474 1
: 486      1475 1 IMPLICIT INPUTS:
: 487      1476 1     none
: 488      1477 1
: 489      1478 1 OUTPUT PARAMETERS:
: 490      1479 1     none
: 491      1480 1
: 492      1481 1 IMPLICIT OUTPUTS:
: 493      1482 1     none
: 494      1483 1
: 495      1484 1 ROUTINE VALUE:
: 496      1485 1     none
: 497      1486 1
: 498      1487 1 SIDE EFFECTS:
: 499      1488 1     none
: 500      1489 1
: 501      1490 1 --
: 502      1491 1
: 503      1492 2 BEGIN
: 504      1493 2
: 505      1494 2 MAP
: 506      1495 2     FIRST_BLOCK : REF BBLOCK;           ! address of the first block
: 507      1496 2
: 508      1497 2 LOCAL
: 509      1498 2     STARTP   : REF BBLOCK;           ! address of the queue head
: 510      1499 2     POINTER   : REF BBLOCK;           ! address of the current block
: 511      1500 2
: 512      1501 2 ! Determine what type of block the first one is. If it is an FCB then it is
: 513      1502 2 necessary to traverse the WCB queue and mark all the WCB's associated with
: 514      1503 2 the FCB as incomplete. If it is a WCB then simply follow the links marking
: 515      1504 2 the WCB's as incomplete until a zero link is found.
: 516      1505 2 !
: 517      1506 2
: 518      1507 2 SELECTONE .FIRST_BLOCK[WCB$B_TYPE] OF
: 519      1508 2     SET
: 520      1509 2
: 521      1510 3     [DYN$C_WCB]:    BEGIN
: 522      1511 3             P$INTER = .FIRST_BLOCK;
: 523      1512 3             SET_IPL (IPL$_SYNCH);
```

```

: 524      1513 3
: 525      1514 4
: 526      1515 4
: 527      1516 4
: 528      1517 4
: 529      1518 4
: 530      1519 4
: 531      1520 3
: 532      1521 3
: 533      1522 3
: 534      1523 2
: 535      1524 2
: 536      1525 3
: 537      1526 3
: 538      1527 3
: 539      1528 3
: 540      1529 3
: 541      1530 3
: 542      1531 3
: 543      1532 4
: 544      1533 4
: 545      1534 4
: 546      1535 4
: 547      1536 4
: 548      1537 3
: 549      1538 3
: 550      1539 3
: 551      1540 2
: 552      1541 2
: 553      1542 2
: 554      1543 2
: 555      1544 2
: 556      1545 2
: 557      1546 1 END;

      DO BEGIN
        IF .POINTER[WCB$B_TYPE] NEQ DYN$C_WCB
        THEN BUG_CHECK (NOTWCWB, FATAL, 'Corrupted WCB list');
        POINTER[WCB$V_COMPLETE] = 0;
        POINTER = .POINTER[WCB$L_LINK];
      END
      UNTIL .POINTER EQL 0;
      SET IPL (0);
      RETURN;
    END;

    [DYN$C_FCB]: BEGIN
      POINTER = .FIRST_BLOCK;
      STARTP = POINTER[FCB$L_WLFL];
      SET IPL (IPL$ SYNCH);
      POINTER = .POINTER[FCB$L_WLFL];
      UNTIL .POINTER EQA .STARTP
      DO BEGIN
        IF .POINTER[WCB$B_TYPE] NEQ DYN$C_WCB
        THEN BUG_CHECK (NOTWCWB, FATAL, 'Corrupted WCB list');
        POINTER[WCB$V_COMPLETE] = 0;
        POINTER = .POINTER[WCB$L_WLFL];
      END
      SET IPL (0);
      RETURN;
    END;

    [OTHERWISE]: BUG_CHECK (WCBFCBMNG, FATAL, 'WCB/FCB correspondence broken');

    TES
  
```

! end of routine MARK\_INCOMPLETE

.EXTRN BUGS\_NOTWCWB

				.ENTRY	MARK_INCOMPLETE, Save R2	: 1456
51	04	0004 00000		MOVL	FIRST_BLOCK, R1	: 1507
52	0A	A1 9A 00006		MOVZBL	10(R1), R2	: SYS
12	52	91 0000A		CMPB	R2, #18	: SYS
	1C	12 0000D		BNEQ	3\$	: SYS
50	51	D0 0000F		MOVL	R1, POINTER	: 1511
12	08	DA 00012		MTPR	#8, #18	: 1512
12	0A	91 00015	1\$:	CMPB	10(POINTER), #18	: 1515
	04	13 00019		BEQL	2\$	: SYS
		FEFF 0001B		BUGW		: 1516
		0000* 0001D		.WORD	<BUGS_NOTWCWB!4>	: SYS
0B	A0	20 8A 0001F	2\$:	BICB2	#32, T1(POINTER)	: 1517
	50	20		MOVL	32(POINTER), POINTER	: 1518
		A0 D0 00023		BNEQ	1\$	: SYS
		EC 12 00027		BRB	6\$	: 1520
		2B 11 00029		CMPB	R2, #7	: 1521
07	52	91 0002B	3\$:	BNEQ	7\$	: 1525
	2A	12 0002E		MOVL	R1, POINTER	: 1526
	51	D0 00030				

51	10	A0	9E	00033		MOVAB	16(POINTER), STARTP	: 1527	Sym
12		08	DA	00037		MTPR	#8, #18	: 1528	---
50	10	A0	D0	0003A		MOVL	16(POINTER), POINTER	: 1529	SYS
51		50	D1	0003E	4\$:	CMPL	POINTER, STARTP	: 1530	SYS
		13	13	00041		BEQL	6\$	: 1531	SYS
12	0A	A0	91	00043		CMPB	10(POINTER), #18	: 1532	SYS
		04	13	00047		BEQL	5\$	: 1533	SYS
			FEFF	00049		BUGW		: 1534	SYS
			0000*	0004B		.WORD	<BUG\$, NOTWCBCB!4>	: 1535	SYS
OB	A0	20	8A	0004D	5\$:	BICB2	#32, T1(POINTER)	: 1536	SYS
		60	D0	00051		MOVL	(POINTER), POINTER	: 1537	SYS
		E8	11	00054		BRB	4\$	: 1538	SYS
OB		00	DA	00056	6\$:	MTPR	#0, #18	: 1539	XAB
		04	00059			RET		: 1540	
			FEFF	0005A	7\$::	BUGW		: 1541	
			0000*	0005C		.WORD	<BUG\$, WCBFCBMNG!4>	: 1542	
			04	0005E		RET		: 1543	
								: 1544	
								: 1545	
								: 1546	

; Routine Size: 95 bytes, Routine Base: \$LOCKEDC1\$ + 0330

```

559 1547 1 GLOBAL ROUTINE MARK_COMPLETE (WINDOW) : NOVALUE =
560 1548 1
561 1549 1 !++
562 1550 1
563 1551 1 ROUTINE DESCRIPTON:
564 1552 1
565 1553 1 This routine is used to mark the specified window as complete.
566 1554 1 It must be executed in kernel mode.
567 1555 1
568 1556 1 CALLING SEQUENCE:
569 1557 1     MARK_COMPLETE (ARG1)
570 1558 1
571 1559 1 INPUT PARAMETERS:
572 1560 1     ARG1: address of the window list to mark
573 1561 1
574 1562 1 IMPLICIT INPUTS:
575 1563 1     none
576 1564 1
577 1565 1 OUTPUT PARAMETERS:
578 1566 1     none
579 1567 1
580 1568 1 IMPLICIT OUTPUTS:
581 1569 1     none
582 1570 1
583 1571 1 ROUTINE VALUE:
584 1572 1     none
585 1573 1
586 1574 1 SIDE EFFECTS:
587 1575 1     none
588 1576 1
589 1577 1 --+
590 1578 1
591 1579 2 BEGIN
592 1580 2
593 1581 2 MAP
594 1582 2     WINDOW      : REF BBLOCK;           ! address of the window to mark
595 1583 2
596 1584 2 LOCAL
597 1585 2     NEXT_SEGMENT : REF BBLOCK;           ! address of the next segment
598 1586 2
599 1587 2 NEXT_SEGMENT = .WINDOW;
600 1588 2
601 1589 2 SET_IPL (IPL$_SYNCH);
602 1590 2
603 1591 2 DO
604 1592 3     BEGIN
605 1593 3     NEXT_SEGMENT[WCB$V_COMPLETE] = 1;       ! mark as complete
606 1594 3     NEXT_SEGMENT = .NEXT_SEGMENT[WCB$L_LINK];
607 1595 3     END
608 1596 2 UNTIL .NEXT_SEGMENT EQL 0;
609 1597 2
610 1598 2 SET_IPL (0);
611 1599 2
612 1600 2 RETURN;
613 1601 2
614 1602 1 END;

```

				.ENTRY	MARK COMPLETE, Save nothing	:	1547
	50	04	0000 00000	MOVL	WINDOW, NEXT_SEGMENT	:	1587
	12	08	DA 00006	MTPR	#8, #18	:	1589
OB	A0	20	88 00009	BISB2	#32, 11(NEXT SEGMENT)	:	1593
	50	20	A0 0000D	MOVL	32(NEXT_SEGMENT), NEXT_SEGMENT	:	1594
			F6 12 00011	BNEQ	18	:	1596
	12		00 DA 00013	MTPR	#0, #18	:	1598
			04 00016	RET		:	1602

: Routine Size: 23 bytes, Routine Base: \$LOCKEDC1\$ + 038F

: 615 1603 1  
: 616 1604 1 END  
: 617 1605 0 ELUDOM

#### PSECT SUMMARY

Name	Bytes	Attributes
\$LOCKEDC1\$	934	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

#### Library Statistics

File	----- Symbols -----			Pages Mapped	Processing Time
	Total	Loaded	Percent		
\$_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	41	0	1000	00:01.9

#### COMMAND QUALIFIERS

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

: Size: 934 code + 0 data bytes  
 : Run Time: 00:30.6  
 : Elapsed Time: 01:07.7  
 : Lines/CPU Min: 3143  
 : Lexemes/CPU-Min: 33776  
 : Memory Used: 359 pages

WITURN  
V04-000

; Compilation Complete

G 13  
16-Sep-1984 01:20:52 VAX-11 Bliss-32 v4.0-742

Page 19

0173 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

SNDSMB  
LTS

SNDER  
LIS

FAL

四  
八