

FFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFF	111	111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFFFFFFFF,FFF	111	111		
FFFFFFFFFFFFF	111	111	XXX	
FFFFFFFFFFFFF	111	111	XXX	
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111111111	111111111	XXX	XXX
FFF	111111111	111111111	XXX	XXX
FFF	111111111	111111111	XXX	XXX

Synt

 10C9
 10-C
 10-C
 10-D
 10-F
 10-S
 KICL

 KILL
 KILL
 LB_E
 LB_C
 LB_F
 LB_H
 LB_L
 LOCA
 LOCA
 LOCK

 LOCK
 LOCK
 LOCK
 LOC-
 LOC-
 L-CC
 L-CC
 L-DA
 L-DA
 MAIN
 MAKE
 MAKE
 MAKE
 MAKE
 MAKE
 MAKE

 MAKE
 MAKE
 MAP-
 MAP-

 MAP
 MAP
 MAP
 MAP
 MAP

```
EEEEEEEEEE XX XX TTTTTTTTTT EEEEEEEEEE NN NN DDDDDDDD
EEEEEEEEEE XX XX TTTTTTTTTT EEEEEEEEEE NN NN DDDDDDDD
EE XX XX TTT TT DD DD
EE XX XX TTT TT DD DD
EE XX XX TTT TT DD DD
EEEEEEEEE XX XX TTT TT DD DD
EEEEEEEEE XX XX TTT TT DD DD
EE XX XX TTT TT DD DD
EE XX XX TTT TT DD DD
EE XX XX TTT TT DD DD
EEEEEEEEEE XX XX TTT TT DD DD
EEEEEEEEEE XX XX TTT TT DD DD
.....
.....
.....
.....
```

```
LL IIIII SSSSSSSS
LL IIIII SSSSSSSS
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 IIIII SSSSSSSS
LLLLLLLLLL IIIII SSSSSSSS
```



```
1 0001 0 MODULE EXTEND (
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 1
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This routine extends a file by the requested number of blocks.
38 0038 1
39 0039 1 ENVIRONMENT:
40 0040 1
41 0041 1 STARLET operating system, including privileged system services
42 0042 1 and internal exec routines.
43 0043 1
44 0044 1 --
45 0045 1
46 0046 1
47 0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 24-Feb-1977 15:42
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1 V03-011 CDS0008 Christian D. Saether 14-Aug-1984
52 0052 1 Set CLF_MARKFCBSTALE. Modify handling of new
53 0053 1 extension fcbs.
54 0054 1
55 0055 1 V03-010 CDS0007 Christian D. Saether 31-July-1984
56 0056 1 Declare registers notused in get_map_pointer linkage.
57 0057 1
```


EXTEND
V04-000

I 4
16-Sep-1984 00:25:07
14-Sep-1984 12:30:22

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[CF11X.SRC]EXTEND.B32;1 Page 2
(1)

58	0058	1	V03-009	CDS0006	Christian D. Saether	25-Apr-1984	
59	0059	1		Long addressing mode for certain routines.			
60	0060	1					
61	0061	1	V03-008	CDS0005	Christian D. Saether	1-Mar-1984	
62	0062	1		Remove FLUSH_FID call now that caching is restored.			
63	0063	1					
64	0064	1	V03-007	CDS0004	Christian D. Saether	30-Dec-1983	
65	0065	1		Use L_NORM linkage and BIND_COMMON macro.			
66	0066	1					
67	0067	1	V03-006	CDS0003	Christian D. Saether	16-Jun-1983	
68	0068	1		Don't release allocation lock before returning.			
69	0069	1		This is because index file extension calls this routine			
70	0070	1		in secondary context and must retain the allocation			
71	0071	1		lock. Deadlock with header creation is avoided			
72	0072	1		by releasing the allocation lock in create_header.			
73	0073	1					
74	0074	1	V03-005	CDS0002	Christian D. Saether	10-Jun-1983	
75	0075	1		Write the modified storage bitmap blocks before			
76	0076	1		releasing the allocation synchronization lock.			
77	0077	1					
78	0078	1	V03-004	CDS0001	Christian D. Saether	16-May-1983	
79	0079	1		Release allocation lock before returning.			
80	0080	1		Don't check VCB\$\$_FREE.			
81	0081	1					
82	0082	1	V03-003	STJ3071	Steven T. Jeffreys,	23-Mar-1983	
83	0083	1		Use the ERASE_REQUESTED parameter of RETURN_BLOCKS.			
84	0084	1					
85	0085	1	V03-002	ACG0300	Andrew C. Goldstein,	12-Oct-1982	17:26
86	0086	1		Clear header under merged map pointer			
87	0087	1					
88	0088	1	V03-001	ACG0270	Andrew C. Goldstein,	23-Mar-1982	9:53
89	0089	1		Shut off placement RVN after first loop iteration			
90	0090	1					
91	0091	1	V02-009	LMP0003	L. Mark Pilant,	24-Nov-1981	10:30
92	0092	1		Added support for extending a file with cathedral windows.			
93	0093	1					
94	0094	1	V02-008	STJ34965	Steven T. Jeffreys	28-Feb-1981	
95	0095	1		Temporary fix to not charge for diskquota			
96	0096	1		if FIB\$V_NOCHARGE is set.			
97	0097	1					
98	0098	1	V02-007	ACG0162	Andrew C. Goldstein,	21-Mar-1980	14:10
99	0099	1		Fix NOTVOLSET status in extend failure			
100	0100	1					
101	0101	1	B0106	ACG26369	Andrew C. Goldstein,	28-Dec-1979	15:46
102	0102	1		Fix multi-header interlock bug			
103	0103	1					
104	0104	1	B0105	ACG0082	Andrew C. Goldstein,	14-Nov-1979	0:11
105	0105	1		Add write-back cacheing of file headers			
106	0106	1					
107	0107	1	B0104	ACG0053	Andrew C. Goldstein,	20-Jul-1979	13:59
108	0108	1		Flag while blocks allocated but not charged			
109	0109	1					
110	0110	1	B0103	ACG0044	Andrew C. Goldstein,	14-Jun-1979	14:03
111	0111	1		Add disk quota support			
112	0112	1					
113	0113	1	B0102	ACG0008	Andrew C. Goldstein,	26-Dec-1978	18:07
114	0114	1		Add placement control support			

EXTEND
V04-

EXTEND
V04-000

J 4
16-Sep-1984 00:25:07
14-Sep-1984 12:30:22

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]EXTEND.B32;1 Page 3 (1)

:	115	0115	1	:	
:	116	0116	1	:	
:	117	0117	1	:	
:	118	0118	1	:	
:	119	0119	1	:	
:	120	0120	1	:	
:	121	0121	1	:	!!
:	122	0122	1	:	
:	123	0123	1	:	
:	124	0124	1	:	LIBRARY 'SYSS\$LIBRARY:LIB.L32';
:	125	0125	1	:	REQUIRE 'SRC\$:FCPDEF.B32';
:	126	1116	1	:	

B0101 ACG0003 Andrew C. Goldstein, 27-Nov-1978 18:57
Add multi-volume support

B0100 ACG00001 Andrew C. Goldstein, 10-Oct-1978 20:00
Previous revision history moved to [F11B.SRC]F11B.REV

EXTE
V04-

```
128 1117 1 GLOBAL ROUTINE EXTEND (USER_FIB, FILEHEADER) : L_NORM NOVALUE =
129 1118 1
130 1119 1 ++
131 1120 1
132 1121 1 FUNCTIONAL DESCRIPTION:
133 1122 1
134 1123 1 This routine extends the given file by the amount and in the
135 1124 1 mode given in the FIB. The blocks are allocated from the storage
136 1125 1 bitmap and retrieval pointers are constructed in the header.
137 1126 1
138 1127 1 CALLING SEQUENCE:
139 1128 1 EXTEND (ARG1, ARG2)
140 1129 1
141 1130 1 INPUT PARAMETERS:
142 1131 1 ARG1: address of FIB for operation
143 1132 1 ARG2: address of file header
144 1133 1
145 1134 1 IMPLICIT INPUTS:
146 1135 1 CURRENT_WINDOW: window of file, if open
147 1136 1 LOC_RVN: placement RVN or 0
148 1137 1 LOC_LBN: placement LBN or 0
149 1138 1
150 1139 1 OUTPUT PARAMETERS:
151 1140 1 NONE
152 1141 1
153 1142 1 IMPLICIT OUTPUTS:
154 1143 1 NONE
155 1144 1
156 1145 1 ROUTINE VALUE:
157 1146 1 NONE
158 1147 1
159 1148 1 SIDE EFFECTS:
160 1149 1 blocks allocated, file header altered
161 1150 1
162 1151 1 --
163 1152 1
164 1153 2 BEGIN
165 1154 2
166 1155 2 LINKAGE
167 1156 2
168 1157 2 This local declaration of the get_map_pointer linkage is done here
169 1158 2 so that different names can be used for the global registers from
170 1159 2 the names used in the fcpdef version.
171 1160 2
172 1161 2
173 1162 2 L_MAP_POINTER_ALT = JSB :
174 1163 2 GLOBAL (OLD COUNT = 6, OLD LBN = 7, MAP_POINTER = 8)
175 1164 2 NOTUSED (2,3,4,5,9,10,11),
176 1165 2
177 1166 2 L_MAKE_POINTER = CALL :
178 1167 2 GLOBAL (BUILD_POINTER = 9);
179 1168 2
180 1169 2 MAP
181 1170 2 USER_FIB : REF BBLOCK, ! FIB of operation
182 1171 2 FILEHEADER : REF BBLOCK; ! file header to extend
183 1172 2
184 1173 2 LABEL
```



```
185      1174 2      ALLOC_LOOP;                ! Block allocation and recording loop
186      1175 2
187      1176 2 GLOBAL REGISTER
188      1177 2      OLD_COUNT      = 6,          ! count of previous retrieval pointer
189      1178 2      OLD_LBN       = 7,          ! LBN of previous retrieval pointer
190      1179 2      MAP_POINTER   = 8 : REF BBLOCK, ! pointer to retrieve map entries
191      1180 2      BUILD_POINTER = 9 : REF BBLOCK; ! pointer to build map entries
192      1181 2
193      1182 2 LOCAL
194      1183 2      FIB           : REF BBLOCK,    ! address of FIB
195      1184 2      HEADER       : REF BBLOCK,    ! address of current file header
196      1185 2      FCB          : REF BBLOCK,    ! FCB of header being extended
197      1186 2      NEW_HEADER   : REF BBLOCK,    ! next extension file header
198      1187 2      WINDOW_SEGMENT : REF BBLOCK,    ! address of the next window segment
199      1188 2      PLACEMENT    : BBLOCK [4],    ! placement control pointer to build
200      1189 2      REREAD       :                ! flag to re-read primary file header
201      1190 2      MAP_END      :                ! end of map area pointers
202      1191 2      BLOCKS_NEEDED :                ! number of blocks to be allocated
203      1192 2      CBT_COUNT    :                ! count of bitmap scans
204      1193 2      EXTEND_VBN   :                ! starting VBN of extend
205      1194 2      LBN         :                ! LBN of blocks allocated
206      1195 2      ALLOC_COUNT  :                ! number of blocks allocated
207      1196 2      COUNT       :                ! count of blocks for map pointers
208      1197 2
209      1198 2 BIND_COMMON;
210      1199 2
211      1200 2 EXTERNAL ROUTINE
212      1201 2      PMS_START_SUB : L_NORM,        ! start subfunction metering
213      1202 2      PMS_END_SUB   : L_NORM,        ! end subfunction metering
214      1203 2      CHARGE_QUOTA  : L_NORM,        ! charge user's disk quota
215      1204 2      NEXT_HEADER   : L_NORM,        ! read next extension header
216      1205 2      MARK_DIRTY    : L_NORM,        ! mark buffer for write-back
217      1206 2      MARK_INCOMPLETE : L_NORM ADDRESSING_MODE (GENERAL), ! mark the windows incomplete
218      1207 2      GET_MAP_POINTER : L_MAP_POINTER ALT, ! get contents of map pointer
219      1208 2      MAKE_POINTER  : L_MAKE_POINTER, ! build new map pointer
220      1209 2      ALLOC_BLOCKS  : L_NORM,        ! allocate blocks from storage map
221      1210 2      EXTEND_HEADER : L_NORM,        ! create extension header
222      1211 2      RETURN_BLOCKS : L_NORM,        ! return blocks to storage map
223      1212 2      CHECKSUM      : L_NORM,        ! compute file header checksum
224      1213 2      TURN_WINDOW   : L_NORM ADDRESSING_MODE (GENERAL), ! update file window
225      1214 2      INIT_FCB2     : L_NORM,        ! initialize FCB
226      1215 2      READ_HEADER   : L_NORM;        ! read file header
227      1216 2
228      1217 2
229      1218 2 ! Start metering for this subfunction.
230      1219 2 !
231      1220 2
232      1221 2 PMS_START_SUB (PMS_ALLOC);
233      1222 2
234      1223 2 ! Check the allocation control bits for validity. Then get the block count
235      1224 2 ! and set up pointers. Check the amount requested against the user's
236      1225 2 ! disk quota. We check first and charge after to (1) simplify error recovery
237      1226 2 ! and (2) avoid penalizing the user for cluster roundup.
238      1227 2 !
239      1228 2
240      1229 2 FIB = .USER FIB;
241      1230 3 IF (NOT .FIB[FIB$V_ALCON] AND .FIB[FIB$V_FILCON])
```

```
242 1231 2 OR .FIB[FIB$L_EXSZ] LSS 0
243 1232 2 THEN ERR_EXIT (SS$_BADPARAM);
244 1233 2
245 1234 3 BLOCKS_NEEDED = (
246 1235 3 IF .FIB[FIB$V_ALDEF]
247 1236 3 THEN MAXU (.CURRENT_VCB[VCB$W_EXTEND], .FIB[FIB$L_EXSZ])
248 1237 3 ELSE .FIB[FIB$L_EXSZ]
249 1238 3 );
250 1239 2
251 1240 2 HEADER = .FILEHEADER;
252 1241 2 FCB = .PRIMARY_FCB;
253 1242 2 EXTEND_VBN = 1;
254 1243 2
255 1244 2 ! If the NOCHARGE bit is set, do not charge the file for diskquota.
256 1245 2
257 1246 2 IF NOT .FIB [FIB$V_NOCHARGE]
258 1247 2 THEN
259 1248 2 CHARGE_QUOTA (.HEADER[FH2$L_FILEOWNER], .BLOCKS_NEEDED, BITLIST (QUOTA_CHECK));
260 1249 2
261 1250 2 ! If the file is marked contiguous best effort, make the extend so.
262 1251 2
263 1252 2
264 1253 2 IF .HEADER[FH2$V_CONTIGB]
265 1254 2 THEN
266 1255 2 IF NOT .FIB[FIB$V_ALCON] THEN FIB[FIB$V_ALCONB] = 1;
267 1256 2
268 1257 2 ! Scan through this header's map area and through the map area of all
269 1258 2 ! extension headers to compute the current file size and find the end of file
270 1259 2 ! to start extension.
271 1260 2
272 1261 2
273 1262 2 REREAD = 0;
274 1263 2 WHILE 1 DO
275 1264 3 BEGIN
276 1265 3 MAP_POINTER = .HEADER + .HEADER[FH2$B_MPOFFSET]*2;
277 1266 3 MAP_END = .MAP_POINTER + .HEADER[FH2$B_MAP_INUSE]*2;
278 1267 3 BUILD_POINTER = .MAP_POINTER;
279 1268 3
280 1269 3 IF .HEADER[FH2$B_MAP_INUSE] NEQ 0 AND .FIB[FIB$V_FILCON]
281 1270 3 THEN ERR_EXIT (SS$_BADPARAM);
282 1271 3
283 1272 3 UNTIL .MAP_POINTER GEQA .MAP_END DO
284 1273 4 BEGIN
285 1274 4 BUILD_POINTER = .MAP_POINTER;
286 1275 4 GET_MAP_POINTER ();
287 1276 4 EXTEND_VBN = .EXTEND_VBN + .OLD_COUNT;
288 1277 4 END;
289 1278 3
290 1279 3 NEW_HEADER = NEXT_HEADER (.HEADER, .FCB);
291 1280 3 IF .NEW_HEADER EQ 0 THEN EXITLOOP;
292 1281 3 HEADER = .NEW_HEADER;
293 1282 3 REREAD = 1;
294 1283 3
295 1284 3 FCB = .FCB[FCB$L_EXFCB];
296 1285 3 END;
297 1286 2
298 1287 2 ! Check the remaining parameters and set the relevant cleanup action flags.
```



```
299 1288 2 !
300 1289 2 !
301 1290 2 IF .FIB[FIB$L_EXVBN] NEQ 0 AND .FIB[FIB$L_EXVBN] NEQ .EXTEND_VBN
302 1291 2 THEN ERR_EXIT('SS$BADPARAM');
303 1292 2
304 1293 2 MARK DIRTY (.HEADER);
305 1294 2 CLEANUP_FLAGS[CLF_TRUNCATE] = 1;
306 1295 2 CLEANUP_FLAGS[CLF_FIXFCB] = 1;
307 1296 2 CLEANUP_FLAGS[CLF_NOTCHARGED] = 1;
308 1297 2
309 1298 2 CBT_COUNT = 0; ! init count of bitmap scans
310 1299 2 FIB[FIB$L_EXSZ] = 0;
311 1300 2 FIB[FIB$L_EXVBN] = .EXTEND_VBN;
312 1301 2 MAP_POINTER = .BUILD_POINTER; ! point to last entry in map
313 1302 2
314 1303 2 ! Now loop, allocating blocks from the storage map and building retrieval
315 1304 2 ! pointers in the header. Accumulate blocks allocated in the I/O
316 1305 2 ! status block. Note that blocks may be preallocated - pick them up if so.
317 1306 2 !
318 1307 2
319 1308 2 ALLOC_LOOP:
320 1309 3 BEGIN
321 1310 3 UNTIL .BLOCKS_NEEDED EQL 0 DO
322 1311 4 BEGIN
323 1312 4
324 1313 4 WHILE 1 DO ! loop to allocate space
325 1314 5 BEGIN
326 1315 5 IF .UNREC_COUNT NEQ 0
327 1316 5 THEN
328 1317 6 BEGIN
329 1318 6 COUNT = .UNREC_COUNT;
330 1319 6 LBN = .UNREC_LBN;
331 1320 6 EXITLOOP;
332 1321 6 END
333 1322 6
334 1323 6 ! If the volume is totally full, or if placement control is directing us to
335 1324 6 ! another volume, continue the file on another volume in the set if this is
336 1325 6 ! a volume set. If this is not a volume set then it's all over but the shouting.
337 1326 6 !
338 1327 6
339 1328 5 ELSE
340 1329 6 BEGIN
341 1330 7 IF (IF .LOC_RVN EQL 0
342 1331 7 OR .LOC_RVN EQL .CURRENT_RVN
343 1332 8 OR (.LOC_RVN EQL 1 AND .CURRENT_RVN EQL 0)
344 1333 7 THEN ALLOC_BLOCKS(.FIB, .BLOCKS_NEEDED, LBN, ALLOC_COUNT)
345 1334 7 ELSE 0
346 1335 7 )
347 1336 6 THEN
348 1337 7 BEGIN
349 1338 7 COUNT = .ALLOC_COUNT;
350 1339 7 EXITLOOP;
351 1340 7 END
352 1341 7
353 1342 7 ! For whatever reason, we do not want to continue allocating on the current
354 1343 7 ! volume. The decision tree runs as follows: If this is not a volume set,
355 1344 7 ! we lose - return either device full or not volume set status, depending
```


on whether volume placement was or was not specified. If this is a volume set and placement wants us on this volume, we also lose. If exact placement was specified, give up, else throw away placement data and try elsewhere. If placement wants us elsewhere (or no volume placement is specified), go do it.

```
ELSE
  BEGIN
    IF .CURRENT_RVN EQL 0
    THEN
      BEGIN
        IF .LOC_RVN LEQU 1
        THEN ERR_EXIT (SS$_DEVICEFULL)
        ELSE ERR_EXIT (SS$_NOTVOLSET);
      END
    ELSE IF .CURRENT_RVN EQL .LOC_RVN
    THEN
      BEGIN
        IF .FIB[FIB$V_EXACT]
        THEN ERR_EXIT (SS$_DEVICEFULL)
        ELSE
          BEGIN
            LOC_RVN = 0;
            LOC_LBN = 0;
          END;
      END;

      HEADER = EXTEND HEADER (.FIB, .HEADER, .FCB, .LOC_RVN, .BLOCKS_NEEDED);
      FCB = .FCB[FCB$_EXFCB];
      FCB [FCB$_STVBN] = .FCB [FCB$_STVBN]
        + .PRIMARY_FCB [FCB$_FILESIZE]
        + .FIB [FIB$_EXSZ];

      MAP_POINTER = .HEADER + .HEADER[FH2$B_MPOFFSET]*2;
      BUICD_POINTER = .MAP_POINTER;
      REREAD = 1;
    END;
  END;

  ! end of allocation loop

  FIB[FIB$_EXSZ] = .FIB[FIB$_EXSZ] + .COUNT;
  BLOCKS_NEEDED = .BLOCKS_NEEDED - MINU (.BLOCKS_NEEDED, .COUNT);

  ! If this is a placed allocation, construct a suitable placement pointer.

  PLACEMENT = 0;
  IF .FIB[FIB$_ALALIGN] NEQ 0
  THEN
    BEGIN
      PLACEMENT<0,32> = .FIB[FIB$_ALOPTS];
      IF .LOC_LBN NEQ 0
      THEN PLACEMENT[FM2$V_LBN] = 1;
      IF .LOC_RVN NEQ 0
      THEN PLACEMENT[FM2$V_RVN] = 1;
    END;
```



```

413 1402 4
414 1403 4 ! Build the map pointer. If the new area allocated is contiguous with
415 1404 4 ! the last pointer in the header, merge the pointers.
416 1405 4
417 1406 4
418 1407 4 IF .HEADER[FH2$B_MAP_INUSE] NEQ 0
419 1408 4 THEN
420 1409 5 BEGIN
421 1410 5 BUILD_POINTER = .MAP_POINTER; ! save pointer position
422 1411 5 GET_MAP_POINTER ();
423 1412 5 IF .OLD_LBN + .OLD_COUNT EQL .LBN
424 1413 5 THEN
425 1414 6 BEGIN
426 1415 6 HEADER[FH2$B_MAP_INUSE] = .HEADER[FH2$B_MAP_INUSE]
427 1416 6 - (.MAP_POINTER - .BUILD_POINTER) / 2;
428 1417 6 COUNT = .COUNT + .OLD_COUNT;
429 1418 6 LBN = .OLD_LBN;
430 1419 6 IF .BUILD_POINTER[FM2$V_FORMAT] EQL FM2$C_PLACEMENT
431 1420 6 THEN PLACEMENT = .PLACEMENT OR .BUILD_POINTER[FM2$W_WORD0];
432 1421 6
433 1422 6 CH$FILL (0, .MAP_POINTER - .BUILD_POINTER, .BUILD_POINTER);
434 1423 6 MAP_POINTER = .BUILD_POINTER;
435 1424 6 END
436 1425 5 ELSE
437 1426 5 BUILD_POINTER = .MAP_POINTER;
438 1427 4 END;
439 1428 4
440 1429 4 ! Now build a retrieval pointer to map the allocated blocks. If the map fills
441 1430 4 ! up, store the unrecorded blocks in common so they can be returned
442 1431 4 ! by the extend cleanup, and create an extension header. If header
443 1432 4 ! extension is inhibited, return the unrecorded blocks and get out quietly.
444 1433 4 ! We first attempt to create a smaller pointer to map at least some of
445 1434 4 ! the blocks that were allocated. We return header full status only if
446 1435 4 ! no new blocks were recorded.
447 1436 4
448 1437 4
449 1438 4 IF NOT MAKE_POINTER (.COUNT, .LBN, .HEADER, .PLACEMENT)
450 1439 4 THEN
451 1440 5 BEGIN
452 1441 5 IF .FIB[FIB$V_NOHDREXT]
453 1442 5 THEN
454 1443 6 BEGIN
455 1444 6 IF MAKE_POINTER (1^14, .LBN, .HEADER, .PLACEMENT)
456 1445 6 THEN
457 1446 7 BEGIN
458 1447 7 COUNT = .COUNT - 1^14;
459 1448 7 LBN = .LBN + 1^14;
460 1449 7 END
461 1450 6 ELSE IF MAKE_POINTER (256, .LBN, .HEADER, .PLACEMENT)
462 1451 6 THEN
463 1452 7 BEGIN
464 1453 7 COUNT = .COUNT - 256;
465 1454 7 LBN = .LBN + 256;
466 1455 6 END;
467 1456 6 RETURN_BLOCKS (.LBN, .COUNT, DO_NOT_ERASE);
468 1457 6 UNREC_COUNT = 0;
469 1458 6 FIB[FIB$L_EXSZ] = .FIB[FIB$L_EXSZ] - .COUNT;
```



```

: 470      1459 6      IF .FIB[FIB$L_EXSZ] EQL 0
: 471      1460 6      THEN ERR_EXIT(SS$_HEADERFULL);
: 472      1461 6      LEAVE ALLOC_LOOP;
: 473      1462 6      END
: 474      1463 5      ELSE
: 475      1464 6      BEGIN
: 476      1465 6      UNREC_LBN = .LBN;
: 477      1466 6      UNREC_COUNT = .COUNT;
: 478      1467 6      UNREC_RVN = .CURRENT_RVN;
: 479      1468 6      HEADER = EXTEND_HEADER(.FIB, .HEADER, .FCB);
: 480      1469 6      FCB = .FCB[FCB$[EXFCB];
: 481      1470 6      FCB[FCB$L_STVBN] = .FCB[FCB$L_STVBN]
: 482      1471 6      + .PRIMARY_FCB[FCB$L_FILESIZE]
: 483      1472 6      + .FIB[FIB$L_EXSZ] - .COUNT;
: 484      1473 6
: 485      1474 6      MAP_POINTER = .HEADER + .HEADER[FH2$B_MPOFFSET]*2;
: 486      1475 6      BUILD_POINTER = .MAP_POINTER;
: 487      1476 6      REREAD = 1;
: 488      1477 6
: 489      1478 6      IF NOT MAKE_POINTER(.COUNT, .LBN, .HEADER, .PLACEMENT)
: 490      1479 6      THEN BUG_CHECK(EXHFUL, FATAL, 'File extension header has no room');
: 491      1480 5      END;
: 492      1481 4      END;
: 493      1482 4
: 494      1483 4      UNREC_COUNT = 0;
: 495      1484 4      ! all blocks are now recorded
: 496      1485 4      ! If this was a contiguous allocation, we are done. Else count the pass
: 497      1486 4      through the allocator. After 3 passes, shut off the contiguous best try
: 498      1487 4      bit to avoid taking forever (since each CBT try is a full sweep of the map).
: 499      1488 4      Also shut off the placement LBN and RVN,
: 500      1489 4      in case this is a placed, non-contiguous allocation, so that it will
: 501      1490 4      simply search upwards.
: 502      1491 4
: 503      1492 4
: 504      1493 4      IF .FIB[FIB$V_ALCON] THEN EXITLOOP;
: 505      1494 4      CBT_COUNT = .CBT_COUNT + 1;
: 506      1495 4      IF .CBT_COUNT GEQ 3
: 507      1496 4      THEN FIB[FIB$V_ALCONB] = 0;
: 508      1497 4      LOC_LBN = 0;
: 509      1498 4      LOC_RVN = 0;
: 510      1499 4      FIB[FIB$V_ONCYL] = 0;
: 511      1500 3      END;
: 512      1501 2      END;
: 513      1502 2      ! end of block ALLOC_LOOP
: 514      1503 2      ! If the file is open by the caller, turn the window to the last VBN
: 515      1504 2      that previously existed as a friendly gesture. Then, if the current header
: 516      1505 2      is an extension header, write it and read back the primary header. Also
: 517      1506 2      set the contiguous bit in the header appropriately and return the extend
: 518      1507 2      data in the FIB. Update the file size in the primary FCB.
: 519      1508 2
: 520      1509 2
: 521      1510 2      IF .CURRENT_WINDOW NEQ 0
: 522      1511 2      THEN
: 523      1512 3      BEGIN
: 524      1513 3      IF NOT .CURRENT_WINDOW[WCB$V_CATHEDRAL]
: 525      1514 4      THEN KERNEL_CALL(TURN_WINDOW, .CURRENT_WINDOW, .HEADER, .FIB[FIB$L_EXVBN]-1, .FCB[FCB$L_STVBN])
: 526      1515 3      ELSE
```


EXTEND
V04-000

E 5
16-Sep-1984 00:25:07
14-Sep-1984 12:30:22

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]EXTEND.B32;1 Page 11
(2)

```

527 1516 4      BEGIN
528 1517 4      KERNEL_CALL (MARK_INCOMPLETE, .PRIMARY_FCB);
529 1518 3      END;
530 1519 2      END;
531 1520 2
532 1521 2      IF .REREAD
533 1522 2      THEN
534 1523 2      BEGIN
535 1524 2      CHECKSUM (.HEADER);
536 1525 2      IF .FCB NEQ 0 THEN KERNEL_CALL (INIT_FCB2, .FCB, .HEADER);
537 1526 2      HEADER = READ_HEADER (FIB[FIB$W_FID], .PRIMARY_FCB);
538 1527 2      END;
539 1528 2
540 1529 2      ! Update the HIBLK field in the record attributes to reflect the new file
541 1530 2      ! size.
542 1531 2      !
543 1532 2
544 1533 2      CLEANUP_FLAGS [CLF_MARKFCBSTALE] = 1;
545 1534 2      MARK_DIRTY (.HEADER);
546 1535 2      BBLOCK [HEADER[FH2$W_RECATTR], FAT$HIBLK] = ROT (.FIB[FIB$L_EXVBN] + .FIB[FIB$L_EXSZ] - 1, 16);
547 1536 2      HEADER[FH2$V_CONTIG] = .FIB[FIB$V_FICON];
548 1537 2      HEADER[FH2$V_CONTIGB] = .FIB[FIB$V_ALCONB];
549 1538 2      USER_STATUS[T] = .FIB[FIB$L_EXSZ];
550 1539 2      PRIMARY_FCB [FCB$L_FILESIZE] = .FIB[FIB$L_EXVBN] + .FIB[FIB$L_EXSZ] - 1;
551 1540 2
552 1541 2      ! Finally charge the blocks allocated to the user.
553 1542 2      ! If the NOCHARGE bit is set, do not charge the file for diskquota.
554 1543 2      !
555 1544 2      IF NOT .FIB [FIB$V_NOCHARGE]
556 1545 2      THEN
557 1546 2      CHARGE_QUOTA (.HEADER[FH2$L_FILEOWNER], .FIB[FIB$L_EXSZ], BITLIST (QUOTA_CHARGE));
558 1547 2      CLEANUP_FLAGS[CLF_NOTCHARGED] = 0;
559 1548 2
560 1549 2      ! Stop metering of this subfunction
561 1550 2      !
562 1551 2
563 1552 2      PMS_END_SUB ();
564 1553 2
565 1554 1      END;

```

! end of routine EXTEND

```

.TITLE  EXTEND
.IDENT  \V04-000\

.EXTRN  PMS_START_SUB, PMS_END_SUB
.EXTRN  CHARGE_QUOTA, NEXT_HEADER
.EXTRN  MARK_DIRTY, MARK_INCOMPLETE
.EXTRN  GET_MAP_POINTER
.EXTRN  MAKE_POINTER, ALLOC_BLOCKS
.EXTRN  EXTEND_HEADER, RETURN_BLOCKS
.EXTRN  CHECKSUM, TURN_WINDOW
.EXTRN  INIT_FCB2, READ_HEADER
.EXTRN  BUG$_EXHFUL

.PSECT  $CODE$,NOWRT,2

.ENTRY  EXTEND, Save R2,R3,R4,R5,R6,R7,R8,R9,R11 ; 1117

```

OBFC 00000

EXTF
V04-

: Ro

: 1
: 1
: 1

		5E	80	24	C2	00002	SUBL2	#36, SP		
			08	AA	9F	00005	PUSHAB	-128(BASE)	1196	
			1C	AA	9F	00008	PUSHAB	8(BASE)		
				AA	9F	0000B	PUSHAB	28(BASE)		
				08	DD	0000E	PUSHL	#8	1221	
	0000G	CF		01	FB	00010	CALLS	#1, PMS_START_SUB		
		5B	04	AC	D0	00015	MOVL	USER_FIB, FIB	1229	
		08	16	AB	E8	00019	BLBS	22(FIB), 2\$	1230	
03	16	AB		02	E1	0001D	BBC	#2, 22(FIB), 2\$		
				00C6	31	00022	BRW	12\$		
			18	AB	D5	00025	TSTL	24(FIB)	1231	
				F8	19	00028	BLSS	1\$		
18	16	AB		03	E1	0002A	BBC	#3, 22(FIB), 4\$	1235	
		50	98	AA	D0	0002F	MOVL	-104(BASE), R0	1236	
		50	3E	A0	3C	00033	MOVZWL	62(R0), R0		
		AB		50	D1	00037	CMPL	R0, 24(FIB)		
				04	1E	0003B	BGEQU	3\$		
		50	18	AB	D0	0003D	MOVL	24(FIB), R0		
	1C	AE		50	D0	00041	MOVL	R0, BLOCKS_NEEDED		
				05	11	00045	BRB	5\$		
	1C	AE	18	AB	D0	00047	MOVL	24(FIB), BLOCKS_NEEDED	1237	
	0C	AE	08	AC	D0	0004C	MOVL	FILEHEADER, HEADER	1240	
	10	AE	04	BE	D0	00051	MOVL	@4(SP), FCB	1241	
		52		01	D0	00056	MOVL	#1, EXTEND_VBN	1242	
			17	AB	95	00059	TSTB	23(FIB)	1246	
				11	19	0005C	BLSS	6\$		
				01	DD	0005E	PUSHL	#1	1248	
53	14	AE	20	AE	DD	00060	PUSHL	BLOCKS_NEEDED		
				3C	C1	00063	ADDL3	#60, HEADER, R3		
				63	DD	00068	PUSHL	(R3)		
	0000G	CF		03	FB	0006A	CALLS	#3, CHARGE_QUOTA		
50	0C	AE		34	C1	0006F	ADDL3	#52, HEADER, R0	1253	
08		60		05	E1	00074	BBC	#5, (R0), 7\$		
		04	16	AB	E8	00078	BLBS	22(FIB), 7\$	1255	
		AB		02	88	0007C	BISB2	#2, 22(FIB)		
			20	AE	D4	00080	CLRL	REREAD	1262	
51	0C	AE		01	C1	00083	ADDL3	#1, HEADER, R1	1265	
		50		61	9A	00088	MOVZBL	(R1), R0		
		58	0C	BE40	3E	0008B	MOVAW	@HEADER[R0], MAP_POINTER		
51	0C	AE		3A	C1	00090	ADDL3	#58, HEADER, R1	1266	
		50		61	9A	00095	MOVZBL	(R1), R0		
		53		6840	3E	00098	MOVAW	(MAP_POINTER)[R0], MAP_END		
		59		58	D0	0009C	MOVL	MAP_POINTER, BUILD_POINTER	1267	
50	0C	AE		3A	C1	0009F	ADDL3	#58, HEADER, R0	1269	
				60	95	000A4	TSTB	(R0)		
				05	13	000A6	BEQL	9\$		
3E	16	AB		02	E0	000A8	BBS	#2, 22(FIB), 12\$		
		53		58	D1	000AD	CMPL	MAP_POINTER, MAP_END	1272	
				0B	1E	000B0	BGEQU	10\$		
		59		58	D0	000B2	MOVL	MAP_POINTER, BUILD_POINTER	1274	
				0000G	30	000B5	BSBW	GET_MAP_POINTER	1275	
		52		56	C0	000B8	ADDL2	OLD_COUNT, EXTEND_VBN	1276	
				F0	11	000BB	BRB	9\$	1272	
			10	AE	DD	000BD	PUSHL	FCB	1279	
			10	AE	DD	000C0	PUSHL	HEADER		
	0000G	CF		02	FB	000C3	CALLS	#2, NEXT_HEADER		
		54		50	D0	000C8	MOVL	R0, NEW_HEADER		

50	OC	AE	13	13	000CB	BEQL	11\$	:	1280		
	20	AE	54	D0	000CD	MOVL	NEW_HEADER, HEADER	:	1281		
	10	AE	01	D0	000D1	MOVL	#1, REREAD	:	1282		
	10	AE	0C	C1	000D5	ADDL3	#12, FCB, R0	:	1284		
			60	D0	000DA	MOVL	(R0), FCB	:			
			A3	11	000DE	BRB	8\$	:	1263		
		1C	AB	D5	000E0	TSTL	28(FIB)	:	1290		
			09	13	000E3	BEQL	13\$	:			
	52	1C	AB	D1	000E5	CMPL	28(FIB), EXTEND_VBN	:			
			03	13	000E9	BEQL	13\$	:			
			14	BF	000EB	CHMU	#20	:	1291		
				04	000ED	RET		:			
	0000G	CF	0C	AE	DD	000EE	13\$:	PUSHL	HEADER	:	1293
		6A	20040002	01	FB	000F1		CALLS	#1, MARK DIRTY	:	
			24	8F	C8	000F6		BISL2	#537133058, (BASE)	:	1296
			18	AE	D4	000FD		CLRL	CBT_COUNT	:	1298
	1C	AB		AB	D4	00100		CLRL	24(FIB)	:	1299
		58		52	D0	00103		MOVL	EXTEND_VBN, 28(FIB)	:	1300
				59	D0	00107		MOVL	BUILD_POINTER, MAP_POINTER	:	1301
		1C		AE	D5	0010A	14\$:	TSTL	BLOCKS_NEEDED	:	1310
				03	12	0010D		BNEQ	15\$	:	
			0258	31	0010F			BRW	38\$	:	
		28	AA	D5	00112	15\$:	TSTL	40(BASE)	:	1315	
			OC	13	00115		BEQL	16\$	:		
	14	AE	28	AA	D0	00117		MOVL	40(BASE), COUNT	:	1318
	2C	AE	24	AA	D0	0011C		MOVL	36(BASE), LBN	:	1319
				2F	11	00121		BRB	18\$	:	1317
			00	BE	D5	00123	16\$:	TSTL	@0(SP)	:	1330
				12	13	00126		BEQL	17\$	:	
	A0	AA	00	BE	D1	00128		CMPL	@0(SP), -96(BASE)	:	1331
		01	00	0B	13	0012D		BEQL	17\$	:	
				BE	D1	0012F		CMPL	@0(SP), #1	:	1332
				1F	12	00133		BNEQ	19\$	:	
		A0	AA	D5	00135		TSTL	-96(BASE)	:		
			1A	12	00138		BNEQ	19\$	:		
		28	AE	9F	0013A	17\$:	PUSHAB	ALLOC_COUNT	:	1333	
		30	AE	9F	0013D		PUSHAB	LBN	:		
		24	AE	DD	00140		PUSHL	BLOCKS_NEEDED	:		
			5B	DD	00143		PUSHL	FIB	:		
	0000G	CF		04	FB	00145		CALLS	#4, ALLOC_BLOCKS	:	
		07		50	E9	0014A		BLBC	R0, 19\$	:	
	14	AE	28	AE	D0	0014D		MOVL	ALLOC_COUNT, COUNT	:	1338
				75	11	00152	18\$:	BRB	24\$	:	1337
			A0	AA	D5	00154	19\$:	TSTL	-96(BASE)	:	1354
				0B	12	00157		BNEQ	20\$	:	
		01	00	BE	D1	00159		CMPL	@0(SP), #1	:	1357
				10	1B	0015D		BLEQU	21\$	:	
			0998	8F	BF	0015F		CHMU	#2456	:	1359
				04	00163		RET		:		
	00	BE	A0	AA	D1	00164	20\$:	CMPL	-96(BASE), @0(SP)	:	1361
				0F	12	00169		BNEQ	23\$	:	
		05	20	AB	E9	0016B		BLBC	32(FIB), 22\$	:	1364
			0850	8F	BF	0016F	21\$:	CHMU	#2128	:	1365
				04	00173		RET		:		
			00	BE	D4	00174	22\$:	CLRL	@0(SP)	:	1368
			20	AA	D4	00177		CLRL	32(BASE)	:	1369
			1C	AE	DD	0017A	23\$:	PUSHL	BLOCKS_NEEDED	:	1373

			04	BE	DD	0017D	PUSHL	@4(SP)		
			18	AE	DD	00180	PUSHL	FCB		
			18	AE	DD	00183	PUSHL	HEADER		
				5B	DD	00186	PUSHL	FIB		
	0000G	CF		05	FB	00188	CALLS	#5, EXTEND_HEADER		
50	OC	AE		50	DO	0018D	MOVL	R0, HEADER		
	10	AE		OC	C1	00191	ADDL3	#12, FCB, R0		1374
	10	AE		60	DO	00196	MOVL	(R0), FCB		
51	10	AE	04	BE	DO	0019A	MOVL	@4(SP), R0		1376
50		61		2C	C1	0019E	ADDL3	#44, FCB, R1		
52	10	AE	38	A0	C1	001A3	ADDL3	56(R0), (R1), R0		
		62		2C	C1	001A8	ADDL3	#44, FCB, R2		1377
51	OC	AE	18	BB40	9E	001AD	MOVAB	@24(FIB)(R0), (R2)		
		50		01	C1	001B2	ADDL3	#1, HEADER, R1		1379
		58		61	9A	001B7	MOVZBL	(R1), R0		
		59	OC	BE40	3E	001BA	MOVAB	@HEADER(R0), MAP_POINTER		
	20	AE		58	DO	001BF	MOVL	MAP_POINTER, BUILD_POINTER		1380
				01	DO	001C2	MOVL	#1, REREAD		1381
	18	AB		FF49	31	001C6	BRW	15\$		1313
	50		14	AE	CO	001C9	ADDL2	COUNT, 24(FIB)		1386
	14	AE	1C	AE	DO	001CE	MOVL	BLOCKS_NEEDED, R0		1387
		50		50	D1	001D2	CMPL	R0, COUNT		
	1C	AE		04	1B	001D6	BLEQU	25\$		
			14	AE	DO	001D8	MOVL	COUNT, R0		
				50	C2	001DC	SUBL2	R0, BLOCKS_NEEDED		
			18	AE	D4	001E0	CLRL	PLACEMENT		1392
			21	AB	95	001E3	TSTB	33(FIB)		1393
				17	13	001E6	BEQL	27\$		
	18	AE	20	AB	9A	001E8	MOVZBL	32(FIB), PLACEMENT		1396
			20	AA	D5	001ED	TSTL	32(BASE)		1397
				04	13	001F0	BEQL	26\$		
	19	AE		10	88	001F2	BISB2	#16, PLACEMENT+1		1398
			00	BE	D5	001F6	TSTL	@0(SP)		1399
				04	13	001F9	BEQL	27\$		
50	19	AE		20	88	001FB	BISB2	#32, PLACEMENT+1		1400
	OC	AE		3A	C1	001FF	ADDL3	#58, HEADER, R0		1407
				60	95	00204	TSTB	(R0)		
		59		47	13	00206	BEQL	30\$		
				58	DO	00208	MOVL	MAP_POINTER, BUILD_POINTER		1410
			0000G	30	0020B	BSBW	GET_MAP_POINTER			1411
50		57		56	C1	0020E	ADDL3	OLD_COUNT, OLD_LBN, R0		1412
	2C	AE		50	D1	00212	CMPL	R0, LBN		
				34	12	00216	BNEQ	29\$		
51		59		58	C3	00218	SUBL3	MAP_POINTER, BUILD_POINTER, R1		1416
50		51		02	C7	0021C	DIVL3	#2, R1, R0		
52	OC	AE		3A	C1	00220	ADDL3	#58, HEADER, R2		
		62		50	80	00225	ADDB2	R0, (R2)		
	14	AE		56	CO	00228	ADDL2	OLD_COUNT, COUNT		1417
	2C	AE		57	DO	0022C	MOVL	OLD_LBN, LBN		1418
	CO	8F	01	A9	93	00230	BITB	1(BUILD_POINTER), #192		1419
				07	12	00235	BNEQ	28\$		
		50		69	3C	00237	MOVZWL	(BUILD_POINTER), R0		1420
	18	AE		50	C8	0023A	BISL2	R0, PLACEMENT		
		50		51	CE	0023E	MNEGL	R1, R0		1422
50	00	6E		00	2C	00241	MOVC5	#0, (SP), #0, R0, (BUILD_POINTER)		
				69		00246				
		58		59	DO	00247	MOVL	BUILD_POINTER, MAP_POINTER		1423

			03	11	0024A	BRB	30\$		1412	
	59		58	DO	0024C	29\$:	MOVL	MAP_POINTER, BUILD_POINTER	1426	
		18	AE	DD	0024F	30\$:	PUSHL	PLACEMENT	1438	
		10	AE	DD	00252		PUSHL	HEADER		
		34	AE	DD	00255		PUSHL	LBN		
		20	AE	DD	00258		PUSHL	COUNT		
	0000G	CF	04	FB	0025B		CALLS	#4, MAKE_POINTER		
		03	50	E9	00260		BLBC	R0, 31\$		
6D	17	AB	00E3	31	00263		BRW	36\$		
			01	E1	00266	31\$:	BBC	#1, 23(FIB), 35\$	1441	
		18	AE	DD	0026B		PUSHL	PLACEMENT	1444	
		10	AE	DD	0026E		PUSHL	HEADER		
		34	AE	DD	00271		PUSHL	LBN		
	0000G	7E	4000	8F	3C	00274	MOVZWL	#16384, -(SP)		
		CF		04	FB	00279	CALLS	#4, MAKE_POINTER		
		12		50	E9	0027E	BLBC	R0, 32\$		
	14	AE	00004000	8F	C2	00281	SUBL2	#16384, COUNT	1447	
	2C	AE	00004000	8F	C0	00289	ADDL2	#16384, LBN	1448	
				26	11	00291	BRB	33\$	1444	
		18	AE	DD	00293	32\$:	PUSHL	PLACEMENT	1450	
		10	AE	DD	00296		PUSHL	HEADER		
		34	AE	DD	00299		PUSHL	LBN		
	0000G	7E	0100	8F	3C	0029C	MOVZWL	#256, -(SP)		
		CF		04	FB	002A1	CALLS	#4, MAKE_POINTER		
		10		50	E9	002A6	BLBC	R0, 33\$		
	14	AE	00000100	8F	C2	002A9	SUBL2	#256, COUNT	1453	
	2C	AE	00000100	8F	C0	002B1	ADDL2	#256, LBN	1454	
				7E	D4	002B9	33\$:	CLRL	-(SP)	1456
		18	AE	DD	002BB		PUSHL	COUNT		
		34	AE	DD	002BE		PUSHL	LBN		
	0000G	CF		03	FB	002C1	CALLS	#3, RETURN_BLOCKS		
		28	AA	D4	002C6		CLRL	40(BASE)	1457	
	18	AB	14	AE	C2	002C9	SUBL2	COUNT, 24(FIB)	1458	
				03	13	002CE	BEQL	34\$	1459	
			0097	31	002D0		BRW	38\$		
		08C8	8F	BF	002D3	34\$:	CHMU	#2248	1460	
				04	002D7		RET			
	24	AA	2C	AE	DO	002D8	35\$:	MOVL	LBN, 36(BASE)	1465
	28	AA	14	AE	DO	002DD		MOVL	COUNT, 40(BASE)	1466
	2C	AA	A0	AA	DO	002E2		MOVL	-96(BASE), 44(BASE)	1467
			10	AE	DD	002E7		PUSHL	FCB	1468
			10	AE	DD	002EA		PUSHL	HEADER	
				5B	DD	002ED		PUSHL	FIB	
	0000G	CF		03	FB	002EF	CALLS	#3, EXTEND_HEADER		
		OC		50	DO	002F4	MOVL	R0, HEADER		
50		10		OC	C1	002F8	ADDL3	#12, FCB, R0	1469	
		10		60	DO	002FD	MOVL	(R0), FCB		
		50	04	BE	DO	00301	MOVL	@4(SP), R0	1471	
51		10		2C	C1	00305	ADDL3	#44, FCB, R1		
50		61	38	A0	C1	0030A	ADDL3	56(R0), (R1), R0		
		50	18	AB	C0	0030F	ADDL2	24(FIB), R0	1472	
51		10		2C	C1	00313	ADDL3	#44, FCB, R1		
61		50	14	AE	C3	00318	SUBL3	COUNT, R0, (R1)		
51		OC		01	C1	0031D	ADDL3	#1, HEADER, R1	1474	
				61	9A	00322	MOVZBL	(R1), R0		
			OC	BE	40	3E	00325	MOVAV	@HEADER[R0], MAP_POINTER	
		58		58	DO	0032A	MOVL	MAP_POINTER, BUILD_POINTER	1475	

EXTEND
V04-000

J 5
16-Sep-1984 00:25:07
14-Sep-1984 12:30:22

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]EXTEND.B32;1

Page 16
(2)

EXTH
V04-

20	AE	01	DO	0032D	MOVL	#1, REREAD	1476
		18	AE	DD 00331	PUSHL	PLACEMENT	1478
		10	AE	DD 00334	PUSHL	HEADER	
		34	AE	DD 00337	PUSHL	LBN	
		20	AE	DD 0033A	PUSHL	COUNT	
0000G	CF	04	FB	0033D	CALLS	#4, MAKE_POINTER	
	04	50	E8	00342	BLBS	R0, 36\$	
			FEFF	00345	BUGW		1479
			0000*	00347	.WORD	<BUG\$ EXHFUL!4>	
		28	AA	D4 00349	CLRL	40(BASE)	1483
	1A	16	AB	E8 0034C	BLBS	22(FIB), 38\$	1493
		24	AE	D6 00350	INCL	CBT_COUNT	1494
	03	24	AE	D1 00353	CMPL	CBT_COUNT, #3	1495
			04	1F 00357	BLSSU	37\$	
16	AB	02	8A	00359	BICB2	#2, 22(FIB)	1496
		20	AA	D4 0035D	CLRL	32(BASE)	1497
		00	BE	D4 00360	CLRL	@0(SP)	1498
20	AB	02	8A	00363	BICB2	#2, 32(FIB)	1499
			FDA0	31 00367	BRW	14\$	1310
	50	0C	AA	DO 0036A	MOVL	12(BASE), R0	1510
			29	13 0036E	BEQL	40\$	
1A	OB	A0	06	E0 00370	BBS	#6, 11(R0), 39\$	1513
52	10	AE	2C	C1 00375	ADDL3	#44, FCB, R2	1514
			62	DD 0037A	PUSHL	(R2)	
7E	1C	AB	01	C3 0037C	SUBL3	#1, 28(FIB), -(SP)	
		14	AE	DD 00381	PUSHL	HEADER	
			50	DD 00384	PUSHL	R0	
00000000G	00		04	FB 00386	CALLS	#4, TURN_WINDOW	
			0A	11 0038D	BRB	40\$	
		04	BE	DD 0038F	PUSHL	@4(SP)	1517
00000000G	00		01	FB 00392	CALLS	#1, MARK_INCOMPLETE	
	27	20	AE	E9 00399	BLBC	REREAD, 42\$	1521
		0C	AE	DD 0039D	PUSHL	HEADER	1524
0000G	CF		01	FB 003A0	CALLS	#1, CHECKSUM	
		10	AE	D5 003A5	TSTL	FCB	1525
			0B	13 003A8	BEQL	41\$	
		0C	AE	DD 003AA	PUSHL	HEADER	
		14	AE	DD 003AD	PUSHL	FCB	
0000G	CF		02	FB 003B0	CALLS	#2, INIT_FCB2	
		04	BE	DD 003B5	PUSHL	@4(SP)	1526
		04	AB	9F 003B8	PUSHAB	4(FIB)	
0000G	CF		02	FB 003BB	CALLS	#2, READ_HEADER	
	OC	50	DO	003C0	MOVL	R0, HEADER	
	01	AA	8F	88 003C4	BISB2	#64, 1(BASE)	1533
		40	AE	DD 003C9	PUSHL	HEADER	1534
0000G	CF		01	FB 003CC	CALLS	#1, MARK_DIRTY	
	1C	AB	AB	C1 003D1	ADDL3	24(FIB), -28(FIB), R1	1535
		50	A1	9E 003D7	MOVAB	-1(R1), R0	
51	OC	AE	18	C1 003DB	ADDL3	#24, HEADER, R1	
61		50	10	9C 003E0	RTL	#16, R0, (R1)	
51	OC	AE	02	EF 003E4	EXTZV	#2, #1, 22(FIB), R0	1536
			34	C1 003EA	ADDL3	#52, HEADER, R1	
61		07	50	F0 003EF	INSV	R0, #7, #1, (R1)	
50	16	AB	01	EF 003F4	EXTZV	#1, #1, 22(FIB), R0	1537
			34	C1 003FA	ADDL3	#52, HEADER, R1	
61		05	50	F0 003FF	INSV	R0, #5, #1, (R1)	
		AE	04	C1 00404	ADDL3	#4, 8(SP), R0	1538

EXTEND
V04-000

K 5
16-Sep-1984 00:25:07
14-Sep-1984 12:30:22

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]EXTEND.B32;1

Page 17
(2)

51	1C 38	60 50 AB A0	18 04 18 FF 17	AB BE AB A1 AB 11 02	D0 00409 D0 0040D C1 00411 9E 00417 95 0041C 19 0041F DD 00421	MOVL 24(FIB), (R0) MOVL 24(SP), R0 ADDL3 24(FIB), 28(FIB), R1 MOVAB -1(R1), 56(R0) TSTB 23(FIB) BLSS 43\$ PUSHL #2	1539
52	14	AE	18	AB 3C 62 03 20 00	DD 00423 C1 00426 DD 0042B FB 0042D 8A 00432 FB 00436 04 0043B	PUSHL 24(FIB) ADDL3 #60, HEADER, R2 PUSHL (R2) CALLS #3, CHARGE_QUOTA BICB2 #32, 3(BASE) CALLS #0, PMS_END_SUB RET	1544 1546 1547 1552 1554

; Routine Size: 1084 bytes, Routine Base: \$CODE\$ + 0000

: 566 1555 1
: 567 1556 1 END
: 568 1557 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	1084	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	51	0	1000	00:01.9

COMMAND QUALIFIERS

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

: Size: 1084 code + 0 data bytes
: Run Time: 00:33.6
: Elapsed Time: 01:13.4
: Lines/CPU Min: 2783

EXTEND
V04-000

16-Sep-1984 00:25:07

VAX-11 Bliss-32 V4.0-742

Page 18

```
: Lexemes/CPU-Min: 32132
: Memory Used: 437 pages
: Compilation Complete
```

EXTH
V04-

.....

0170

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY