



```

EEEEEEEEEE XX XX TTTTTTTTTT HH HH DDDDDDDD RRRRRRRR
EEEEEEEEEE XX XX TTTTTTTTTT HH HH DDDDDDDD RRRRRRRR
EE XX XX TT HH HH DD DD RR RR
EE XX XX TT HH HH DD DD RR RR
EE XX XX TT HH HH DD DD RR RR
EEEEEEEE XX XX TT HHHHHHHHHH DD DD RRRRRRRR
EEEEEEEE XX XX TT HHHHHHHHHH DD DD RRRRRRRR
EE XX XX TT HH HH DD DD RR RR
EE XX XX TT HH HH DD DD RR RR
EE XX XX TT HH HH DD DD RR RR
EEEEEEEEEE XX XX TT HH HH DDDDDDDD RR RR
EEEEEEEEEE XX XX TT HH HH DDDDDDDD RR RR

```

```

LL IIIIII SSSSSSSS
LL IIIIII 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
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS

```



1  
2  
:001 :CDS0004  
4-1  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
:001 :CDS0004  
:002 :CDS0004  
:003 :CDS0004  
52  
53  
54

```

0001 0 MODULE EXTHDR (
0002 0
0003 0     LANGUAGE (BLISS32),
0004 0     IDENT = 'V04-001'
0005 1 ) =
0006 1 BEGIN
0007 1
0008 1 *****
0009 1 *
0010 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0011 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0012 1 *  ALL RIGHTS RESERVED.
0013 1 *
0014 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0015 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0016 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0017 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0018 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0019 1 *  TRANSFERRED.
0020 1 *
0021 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0022 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0023 1 *  CORPORATION.
0024 1 *
0025 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0026 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0027 1 *
0028 1 *
0029 1 *****
0030 1
0031 1 ++
0032 1
0033 1 FACILITY: F11ACP Structure Level 2
0034 1
0035 1 ABSTRACT:
0036 1
0037 1     This routine creates an extension file header for the given file
0038 1     header.
0039 1
0040 1 ENVIRONMENT:
0041 1
0042 1     STARLET operating system, including privileged system services
0043 1     and internal exec routines.
0044 1
0045 1 --
0046 1
0047 1
0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 27-Jul-1977 10:15
0049 1
0050 1 MODIFIED BY:
0051 1
0052 1     V04-001 CDS0004      Christian D. Saether    15-Nov-1984
0053 1     Set reference count for new extension fcb to 1.
0054 1
0055 1     V03-006 CDS0003      Christian D. Saether    14-Aug-1984
0056 1     Modify creation of extension fcbs.
0057 1

```

```
.. 55      0058 1 | V03-005 LMP0286      L. Mark Pilant,      26-Jul-1984  13:24
.. 56      0059 1 |      Correct a broken CALL offset.
.. 57      0060 1 |
.. 58      0061 1 | V03-004 ACG0415      Andrew C. Goldstein,  6-Apr-1984  20:59
.. 59      0062 1 |      Set CLF_NOBUILD in FCB initialization, since ACL is
.. 60      0063 1 |      already there.
.. 61      0064 1 |
.. 62      0065 1 | V03-003 ACG0408      Andrew C. Goldstein,  21-Mar-1984  11:40
.. 63      0066 1 |      Make APPLY_RVN and DEFAULT_RVN macros; also fix handling
.. 64      0067 1 |      of RVN in Back link.
.. 65      0068 1 |
.. 66      0069 1 | V03-002 CDS0002      Christian D. Saether  18-Jan-1984
.. 67      0070 1 |      Modify interface to DEFAULT_RVN.
.. 68      0071 1 |
.. 69      0072 1 | V03-001 CDS0001      Christian D. Saether  30-Dec-1983
.. 70      0073 1 |      Use L_NORM linkage and BIND_COMMON macro.
.. 71      0074 1 |
.. 72      0075 1 | V02-007 LMP0003      L. Mark Pilant,      11-Dec-1981  14:00
.. 73      0075 1 |      Add support for cathedral windows.
.. 74      0077 1 |
.. 75      0078 1 | V02-006 ACG0167      Andrew C. Goldstein,  16-Apr-1980  19:26
.. 76      0079 1 |      Previous revision history moved to f11B.REV
.. 77      0080 1 | ..
.. 78      0081 1 |
.. 79      0082 1 |
.. 80      0083 1 | LIBRARY 'SYSS:LIBRARY:LIB.L32';
.. 81      0084 1 | REQUIRE 'SRCS:FCPDEF.B32';
```

```

83 1075 1 GLOBAL ROUTINE EXTEND_HEADER (FIB, OLD_HEADER, FCB, NEW_VOLUME, BLOCKS_NEEDED) : L_NORM =
84 1076 1
85 1077 1 ++
86 1078 1
87 1079 1 FUNCTIONAL DESCRIPTION:
88 1080 1
89 1081 1     This routine creates an extension file header for the given file
90 1082 1     header.
91 1083 1
92 1084 1
93 1085 1 CALLING SEQUENCE:
94 1086 1     EXTEND_HEADER (ARG1, ARG2, ARG3, ARG4, ARG5)
95 1087 1
96 1088 1 INPUT PARAMETERS:
97 1089 1     ARG1: address of user FIB
98 1090 1     ARG2: address of present last file header
99 1091 1     ARG3: address of present last FCB or 0
100 1092 1     ARG4: if not present, stay on present volume
101 1093 1         if present:
102 1094 1             if non-zero, force switch to given volume
103 1095 1             if zero, switch to any other volume
104 1096 1     ARG5: number of blocks needed on new volume
105 1097 1
106 1098 1 IMPLICIT INPUTS:
107 1099 1     CURRENT_WINDOW: address of file window or 0
108 1100 1     PRIMARY_FCB: primary FCB of file
109 1101 1     CURRENT_VCB: address of VCB of volume
110 1102 1
111 1103 1 OUTPUT PARAMETERS:
112 1104 1     NONE
113 1105 1
114 1106 1 IMPLICIT OUTPUTS:
115 1107 1     NONE
116 1108 1
117 1109 1 ROUTINE VALUE:
118 1110 1     address of new file header
119 1111 1
120 1112 1 SIDE EFFECTS:
121 1113 1     file header created, window turned, FCB created
122 1114 1
123 1115 1 --
124 1116 1
125 1117 2 BEGIN
126 1118 2
127 1119 2 MAP
128 1120 2     FIB           : REF BBLOCK,   ! user FIB arg
129 1121 2     OLD_HEADER  : REF BBLOCK,   ! file header arg
130 1122 2     FCB         : REF BBLOCK;   ! FCB arg
131 1123 2
132 1124 2 LOCAL
133 1125 2     OLD_FID     : BBLOCK [FIDSC_LENGTH], ! file ID of old header
134 1126 2     FXT_FID     : BBLOCK [FIDSC_LENGTH], ! file ID of new header
135 1127 2     VBN        :          ! index file VBN of new header
136 1128 2     LBN        :          ! LBN of new header
137 1129 2     HEADER     : REF BBLOCK,   ! buffer address of current file header
138 1130 2     NEW_HEADER  : REF BBLOCK;   ! address of new file header
139 1131 2

```

```
140 1132 2 BIND_COMMON;
141 1133 2
142 1134 2 EXTERNAL ROUTINE
143 1135 2     INIT_FCB2      : L_NORM,      ! initialize FCB
144 1136 2     TURN_WINDOW   : L_NORM, ADDRESSING MODE (GENERAL), ! update window
145 1137 2     CHECKSUM      : L_NORM,      ! checksum file header
146 1138 2     SELECT_VOLUME  : L_NORM,      ! select new volume for use
147 1139 2     MARK_DIRTY    : L_NORM,      ! mark header for writeback
148 1140 2     CHARGE       : L_NORM,      ! charge user's disk quota
149 1141 2     CREATE_HEADER : L_NORM,      ! create a new file ID and header
150 1142 2     SWITCH_VOLUME  : L_NORM,      ! switch context to desired volume
151 1143 2     READ_HEADER    : L_NORM,      ! read file header
152 1144 2     CREATE_FCB     : L_NORM;      ! create an fcb.
153 1145 2
154 1146 2
155 1147 2 ! Save the file ID of the current last header. If the file is accessed, fix
156 1148 2 ! up the FCB if it is not the primary and turn the window to include blocks
157 1149 2 ! from the header if possible. Then prepare the old header for write-back.
158 1150 2
159 1151 2
160 1152 2 CHSMOVE (FID$C LENGTH, OLD_HEADER[FH2$W FID], OLD FID);
161 1153 2 OLD_FID[FID$W_RVN] = .OLD_FID[FID$W_RVN] + .CURRENT_RVN;
162 1154 2
163 1155 2 IF .FCB NEQ 0 AND .FCB NEQ .PRIMARY_FCB
164 1156 2 THEN
165 1157 2     BEGIN
166 1158 2     CLEANUP_FLAGS[CLF_NOBUILD] = 1;
167 1159 2     KERNEL_CALL (INIT_FCB2, .FCB, .OLD_HEADER);
168 1160 2     END;
169 1161 2
170 1162 2 IF .CURRENT_WINDOW NEQ 0
171 1163 2 THEN IF NOT .CURRENT_WINDOW[WCBSV_CATHEDRAL]
172 1164 2 THEN KERNEL_CALL (TURN_WINDOW, .CURRENT_WINDOW, .OLD_HEADER, .PRIMARY_FCB[FCB$L_FILESIZE], .FCB[FCB$L_STVBN])
173 1165 2
174 1166 2 CHECKSUM (.OLD_HEADER);
175 1167 2
176 1168 2 ! Now create the new file ID. Map and read the corresponding block in the
177 1169 2 ! index file to extract the file sequence number; then punt the buffer.
178 1170 2
179 1171 2
180 1172 2 IF ACTUALCOUNT GEQU 4
181 1173 2 THEN
182 1174 2     BEGIN
183 1175 2     IF .NEW_VOLUME NEQ 0
184 1176 2     AND .FIB[FIB$V_EXACT]
185 1177 2     THEN
186 1178 2     SWITCH_VOLUME (.NEW_VOLUME)
187 1179 2     ELSE
188 1180 2     SELECT_VOLUME (.FIB, .BLOCKS_NEEDED);
189 1181 2     END;
190 1182 2
191 1183 2 NEW_HEADER = CREATE_HEADER (EXT_FID);
192 1184 2 LBN = .HEADER_LBN;
193 1185 2
194 1186 2 ! Get back the old file header, which may or may not have been written out
195 1187 2 ! due to buffer pool thrashing. Check the segment number for overflow.
196 1188 2 ! Plug in the header extension linkage and write it.
```

```
197 1189 :
198 1190 :
199 1191 HEADER = READ HEADER (OLD_FID, .FCB);
200 1192 IF .HEADER[FH2$W_SEG_NUM] GEQU 65535
201 1193 THEN ERR_EXIT (SS$ HEADERFULL);
202 1194 CHSMOVE (FID$C_LENGTH, EXT_FID, HEADER[FH2$W_EXT_FID]);
203 1195 :
204 1196 CHECKSUM (.HEADER);
205 1197 MARK_DIRTY (.HEADER);
206 1198 IF ACTUALCOUNT GEQU 4
207 1199 THEN
208 1200 BEGIN
209 1201 SWITCH_VOLUME (.NEW_FID_RVN);
210 1202 END;
211 1203 :
212 1204 ! We now fabricate a buffer and copy the old header into it, thus
213 1205 ! keeping the attributes. Note that we truncate the Ident area to only the
214 1206 ! file name, to gain map pointer space.
215 1207 :
216 1208 :
217 1209 CHSMOVE (512, .HEADER, .NEW_HEADER);
218 1210 HEADER_LBN = .LBN;
219 1211 FILE_HEADER = .NEW_HEADER;
220 1212 :
221 1213 CHSMOVE (FID$C_LENGTH, EXT_FID, NEW_HEADER[FH2$W_FID]);
222 1214 NEW_HEADER[FH2$B_FID_RVN] = 0;
223 1215 :
224 1216 ! If this is the first extension header of the file, copy the file ID
225 1217 ! of the primary into the back link. Once there, it will propagate to
226 1218 ! succeeding extension headers.
227 1219 :
228 1220 :
229 1221 IF .NEW_HEADER[FH2$W_SEG_NUM] EQL 0
230 1222 THEN
231 1223 BEGIN
232 1224 CHSMOVE (FID$C_LENGTH, OLD_FID, NEW_HEADER[FH2$W_BACKLINK]);
233 1225 DEFAULT_RVN (NEW_HEADER[FH2$W_BK_FIDRVN], .CURRENT_RVN);
234 1226 END
235 1227 :
236 1228 ! If we're already in an extension header, the RVN may have to be fixed
237 1229 ! up, if the new extension header is on a different volume.
238 1230 :
239 1231 :
240 1232 ELSE
241 1233 BEGIN
242 1234 APPLY_RVN (NEW_HEADER[FH2$W_BK_FIDRVN], .OLD_FID[FID$B_RVN]);
243 1235 DEFAULT_RVN (NEW_HEADER[FH2$W_BK_FIDRVN], .CURRENT_RVN);
244 1236 END;
245 1237 NEW_FID = 0; ! header extension is complete
246 1238 :
247 1239 NEW_HEADER[FH2$W_SEG_NUM] = .NEW_HEADER[FH2$W_SEG_NUM] + 1;
248 1240 CH$FILL (0, FID$C_LENGTH, NEW_HEADER[FH2$W_EXT_FID]);
249 1241 NEW_HEADER[FH2$B_MAP_INUSE] = 0;
250 1242 NEW_HEADER[FH2$B_CONTIG] = 0;
251 1243 NEW_HEADER[FH2$B_LOCKED] = 0;
252 1244 NEW_HEADER[FH2$B_MPOFFSET] = .NEW_HEADER[FH2$B_IDOFFSET] +
253 1245 ($BYTEOFFSET (FID$W_REVISION)) / 2;
```

```

254 1246 2 NEW_HEADER[FH2$B_ACOFFSET] = ($BYTEOFFSET (FH2$W_CHECKSUM)) / 2;
255 1247 2 NEW_HEADER[FH2$B_RSOFFSET] = ($BYTEOFFSET (FH2$W_CHECKSUM)) / 2;
256 1248 2
257 1249 2 CHSFILL (0, 512 - .NEW_HEADER[FH2$B_MPOFFSET]*2, .NEW_HEADER + .NEW_HEADER[FH2$B_MPOFFSET]*2);
258 1250 2
259 1251 2 ! Now charge the user for the new extension header.
260 1252 2 !
261 1253 2
262 1254 2 CHARGE_QUOTA (.NEW_HEADER[FH2$L_FILEOWNER], 1, BITLIST (QUOTA_CHECK, QUOTA_CHARGE));
263 1255 2
264 1256 2 ! Finally create an extension FCB if the file is accessed.
265 1257 2 !
266 1258 2
267 1259 2 IF .FCB NEQ 0
268 1260 2 THEN
269 1261 2 BEGIN
270 1262 2 LOCAL
271 1263 2 NEWFCB : REF BBLOCK;
272 1264 2
273 1265 2 NEWFCB = CREATE_FCB (.NEW_HEADER);
274 1266 2
001 ! CDS0004 1267 2 NEWFCB [FCB$W_REFCNT] = 1;
275 1268 2 NEWFCB [FCB$L_LOCKBASIS] = .PRIMARY_FCB [FCB$L_LOCKBASIS];
276 1269 2 FCB [FCB$L_EXFCB] = .NEWFCB;
277 1270 2 CURRENT_VCB [VCB$W_TRANS] = .CURRENT_VCB [VCB$W_TRANS] + 1;
278 1271 2 END;
279 1272 2
280 1273 2 RETURN (.NEW_HEADER);
281 1274 2
282 1275 1 END;
! end of routine EXTEND_HEADER

```

```

.TITLE EXTHDR
.IDENT \V04-001\

.EXTRN INIT_FCB2, TURN_WINDOW
.EXTRN CHECKSUM, SELECT_VOLUME
.EXTRN MARK_DIRTY, CHARGE_QUOTA
.EXTRN CREATE_HEADER, SWITCH_VOLUME
.EXTRN READ_HEADER, CREATE_FCB

.PSECT $CODE$,NOWRT,2

.ENTRY EXTEND_HEADER, Save R2,R3,R4,R5,R6,R7,R8 : 1075
SUBL2 #16 SP
MOVL OLD_HEADER, R0 : 1152
MOVCL #6, 8(R0), OLD_FID
ADDW2 -96(BASE), OLD_FID+4 : 1153
TSTL FCB : 1155
BEQL 1$
CMPL FCB, 8(BASE)
BEQL 1$
BISB2 #8, 1(BASE) : 1158
PUSHL OLD_HEADER : 1159
PUSHL FCB
CALLS #2, INIT_FCB2
MOVL 12(BASE), R1 : 1162

```

```

01FC 0000
0B AE 0B 5E 10 C2 00002
0C AE 06 28 00009
0C AC 0A A0 0000F
0C AC 05 00014
0C AC 16 13 00017
0C AC 08 AA 0C AC D1 00019
0C AC 0F 13 0001E
0C AC 08 88 00020
0C AC 08 DD 00024
0C AC 01 AA 0C AC DD 00027
0000G CF 02 FB 0002A
51 0C AA D0 0002F 1$:

```



1A	0B	A1		1F	13	00033	BEQL	2\$			
		50	0C	06	E0	00035	BBS	#6, 11(R1), 2\$			1163
			2C	AC	DD	0003A	MOVL	FCB, R0			1164
		50	08	A0	DD	0003E	PUSHL	44(R0)			
			38	AA	DD	00041	MOVL	8(BASE), R0			
			08	A0	DD	00045	PUSHL	56(R0)			
				AC	DD	00048	PUSHL	OLD_HEADER			
	00000000G	00		51	DD	0004B	PUSHL	R1			
				04	FB	0004D	CALLS	#4, TURN_WINDOW			
	0000G	CF	08	AC	DD	00054	PUSHL	OLD_HEADER			1166
		04		01	FB	00057	CALLS	#1, CHECKSUM			
				6C	91	0005C	CMPB	(AP), #4			1172
				22	1F	0005F	BLSSU	4\$			
				10	AC	D5	00061	TSTL	NEW_VOLUME		1175
				12	13	00064	BEQL	3\$			
		50	04	AC	DD	00066	MOVL	FIB, R0			1176
		0A	20	A0	E9	0006A	BLBC	32(R0), 3\$			
	0000G	CF	10	AC	DD	0006E	PUSHL	NEW_VOLUME			1178
				01	FB	00071	CALLS	#1, SWITCH_VOLUME			
				0B	11	00076	BRB	4\$			
				14	AC	DD	00078	PUSHL	BLOCKS_NEEDED		1180
	0000G	CF	04	AC	DD	0007B	PUSHL	FIB			
				02	FB	0007E	CALLS	#2, SELECT_VOLUME			
	0000G	CF		5E	DD	00083	PUSHL	SP			1183
		56		01	FB	00085	CALLS	#1, CREATE_HEADER			
		58		50	DD	0008A	MOVL	R0, NEW_HEADER			
			B0	AA	DD	0008D	MOVL	-80(BASE), LBN			1184
			0C	AC	DD	00091	PUSHL	FCB			1191
	0000G	CF	0C	AE	9F	00094	PUSHAB	OLD_FID			
		57		02	FB	00097	CALLS	#2, READ_HEADER			
	FFFF	8F	04	50	DD	0009C	MOVL	R0, HEADER			
				A7	B1	0009F	CMPW	4(HEADER), #65535			1192
				05	1F	000A5	BLSSU	5\$			
			08C8	8F	BF	000A7	CHMU	#2248			1193
				04	000AB		RET				
0E	A7	6E		06	28	000AC	MOVCS	#6, EXT_FID, 14(HEADER)			1194
				57	DD	000B1	PUSHL	HEADER			1196
	0000G	CF		01	FB	000B3	CALLS	#1, CHECKSUM			
				57	DD	000B8	PUSHL	HEADER			1197
	0000G	CF		01	FB	000BA	CALLS	#1, MARK_DIRTY			
		04		6C	91	000BF	CMPB	(AP), #4			1198
				08	1F	000C2	BLSSU	6\$			
	0000G	CF	AC	AA	DD	000C4	PUSHL	-84(BASE)			1201
		67		01	FB	000C7	CALLS	#1, SWITCH_VOLUME			
	66	AA	0200	8F	28	000CC	MOVCS	#512, (HEADER), (NEW_HEADER)			1200
		04		58	DD	000D2	MOVL	LBN, -80(BASE)			1210
		AA		56	DD	000D6	MOVL	NEW_HEADER, 4(BASE)			1211
08	A6	6E		06	28	000DA	MOVCS	#6, EXT_FID, 8(NEW_HEADER)			1213
			0C	A6	94	000DF	CLRB	12(NEW_HEADER)			1214
			04	A6	B5	000E2	TSTW	4(NEW_HEADER)			1221
				08	12	000E5	BNEQ	7\$			
42	A6	08		06	28	000E7	MOVCS	#6, OLD_FID, 66(NEW_HEADER)			1224
		AE		18	11	000ED	BRB	9\$			1225
				46	A6	95	000EF	TSTB	70(NEW_HEADER)		1234
				05	12	000F2	BNEQ	8\$			
	46	A6	0C	AE	90	000F4	MOVB	OLD_FID+4, 70(NEW_HEADER)			
		01	46	A6	91	000F9	CMPB	70(NEW_HEADER), #T			

AO	AA	46	A6	03	OC	08	12	000FD	BNEQ	98	1235
						AE	95	000FF	TSTB	OLD_FID+4	
						03	12	00102	BNEQ	98	
					46	A6	94	00104	CLRB	70(NEW_HEADER)	
						00	ED	00107	CMPZV	#0, #8, 70(NEW_HEADER), -96(BASE)	1235
						03	12	0010E	BNEQ	108	
					46	A6	94	00110	CLRB	70(NEW_HEADER)	
					AB	AA	D4	00113	CLRL	-88(BASE)	1237
					04	A6	B6	00116	INCW	4(NEW_HEADER)	1239
	06		00	6E		00	2C	00119	MOVCS	#0, (SP), #0, #6, 14(NEW_HEADER)	1240
						0E	A6	0011E			
					3A	A6	94	00120	CLRB	58(NEW_HEADER)	1241
				34	A6	8F	8A	00123	BICB2	#192, 52(NEW_HEADER)	1243
					66	0A	81	00128	ADDB3	#10, (NEW_HEADER), 1(NEW_HEADER)	1244
	01	A6		02	A6	8F	B0	0012D	MOVW	#65535, 2(NEW_HEADER)	1246
					50	01	A6	00133	MOVZBL	1(NEW_HEADER), R0	1249
					51	50	CE	00137	MNEGL	R0, R1	
					51	02	C4	0013A	MULL2	#2, R1	
					51	C1	9E	0013D	MOVAB	512(R1), R1	
					57	6640	3E	00142	MOVAB	(NEW_HEADER)[R0], R7	
	51		00	6E		00	2C	00146	MOVCS	#0, (SP), #0, R1, (R7)	
						67		0014B			
						03	DD	0014C	PUSHL	#3	1254
						01	DD	0014E	PUSHL	#1	
					3C	A6	DD	00150	PUSHL	60(NEW_HEADER)	
		0000G	CF			03	FB	00153	CALLS	#3, CHARGE_QUOTA	
					0C	AC	D5	00158	TSTL	FCB	1259
						23	13	0015B	BEQL	118	
						56	DD	0015D	PUSHL	NEW_HEADER	1265
		0000G	CF			01	FB	0015F	CALLS	#1, CREATE_FCB	
		18	A0			01	B0	00164	MOVW	#1, 24(NEWFCB)	1267
			51		08	AA	D0	00168	MOVL	8(BASE), R1	1268
		4C	A0		4C	A1	D0	0016C	MOVL	76(R1), 76(NEWFCB)	
			51		0C	AC	D0	00171	MOVL	FCB, R1	1269
		0C	A1			50	D0	00175	MOVL	NEWFCB, 12(R1)	
			50		98	AA	D0	00179	MOVL	-104(BASE), R0	1270
					0C	A0	B6	0017D	INCW	12(R0)	
						56	D0	00180	MOVL	NEW_HEADER, R0	1273
						04	00183		RET		1275

: Routine Size: 388 bytes, Routine Base: \$CODE\$ + 0000

```

: 283      1276 1
: 284      1277 1 END
: 285      1278 0 ELUDOM

```

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	388	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
:_\$255\$DUA18:[SYSLIB]LIB.L32;1	18619	45	0	1000	00:01.8

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:EXTHDR/OBJ=OBJS:EXTHDR MSRCS:EXTHDR/UPDATE=(BUGS:EXTHDR)

: Size: 388 code + 0 data bytes  
: Run Time: 00:22.0  
: Elapsed Time: 00:33.8  
: Lines/CPU Min: 3479  
: Lexemes/CPU-Min: 43437  
: Memory Used: 284 pages  
: Compilation Complete

# 0443 AH-EF71A-SE VAX/VMS V4.1 SRC LST MCRF UPD

															
															
															
															
															
															
															
															
															
															
															
															
															