


```

EEEEEEEEEE  NN      NN  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
EEEEEEEEEE  NN      NN  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
EE          NN      NN      TT          EE          RR      RR
EE          NN      NN      TT          EE          RR      RR
EE          NNNN     NN      TT          EE          RR      RR
EE          NNNN     NN      TT          EE          RR      RR
EEEEEEEEEE  NN  NN  NN      TT          EEEEEEEEE  RRRRRRRR
EEEEEEEEEE  NN  NN  NN      TT          EEEEEEEEE  RRRRRRRR
EE          NN      NNNN     TT          EE          RR  RR
EE          NN      NNNN     TT          EE          RR  RR
EE          NN      NN      TT          EE          RR      RR
EEEEEEEEEE  NN      NN      TT          EEEEEEEEEE  RR      RR
EEEEEEEEEE  NN      NN      TT          EEEEEEEEEE  RR      RR

```

```

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

```

1
2
3
4
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
52
53
54
55
56
57

```

0001 0 MODULE ENTER (
0002 0
0003 0     LANGUAGE (BLISS32),
0004 0     IDENT = 'V04-001'
0005 1 BEGIN
0006 1
0007 1
0008 1
0009 1
0010 1 *
0011 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0012 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
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 a new directory entry for the given file.
0038 1
0039 1 ENVIRONMENT:
0040 1
0041 1     STARLET operating system, including privileged system services
0042 1     and internal exec routines.
0043 1
0044 1 --
0045 1
0046 1
0047 1 AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  11-Jan-1978  16:50
0048 1
0049 1 MODIFIED BY:
0050 1
0051 1     V04-001  CWH4001      CW Hobbs      9-Aug-1984
0052 1           Fix truncation error.
0053 1
0054 1     V03-006  CDS0003    Christian D. Saether  3-Aug-1984
0055 1           Modify handling of directory index updating.
0056 1
0057 1     V03-005  LMP0252    L. Mark Pilant,      25-Jun-1984  10:03

```

```

58 0058 1  Make sure that the test for removing a directory entry on an
59 0059 1  an auto-purge uses the relative position of the entry, not the
60 0060 1  address. This corrects a problem with corrupted directories
61 0061 1  for large directories with a version limit of 1.
62 0062 1
63 0063 1  V03-004 ACG0419 Andrew C. Goldstein, 19-Apr-1984 15:27
64 0064 1  Add NOSAFE switch to MAKE_ENTRY to circumvent CSE bug
65 0065 1
66 0066 1  V03-003 ACG0408 Andrew C. Goldstein, 20-Mar-1984 17:39
67 0067 1  Make APPLY_RVN and DEFAULT_RVN macros
68 0068 1
69 0069 1  V03-002 CDS0002 Christian D. Saether 18-Jan-1984
70 0070 1  Modify interface to DEFAULT_RVN and APPLY_RVN.
71 0071 1
72 0072 1  V03-001 CDS0001 Christian D. Saether 19-Dec-1983
73 0073 1  Use BIND_COMMON macro to reduce number of
74 0074 1  external COMMON declarations.
75 0075 1
76 0076 1  V02-012 ACG0264 Andrew C. Goldstein, 12-Feb-1982 15:21
77 0077 1  Allow negative version numbers (and treat like zero)
78 0078 1
79 0079 1  V02-011 ACG0259 Andrew C. Goldstein, 26-Jan-1982 19:17
80 0080 1  Fix autopurge when version limit is 1
81 0081 1
82 0082 1  V02-010 ACG0238 Andrew C. Goldstein, 10-Dec-1981 14:22
83 0083 1  Invalidate RMS directory cache when dir is superseded
84 0084 1
85 0085 1  V02-009 ACG0208 Andrew C. Goldstein, 28-Oct-1981 20:44
86 0086 1  Add segmented directory record support
87 0087 1
88 0088 1  V02-008 ACG34341 Andrew C. Goldstein, 3-Mar-1981 17:06
89 0089 1  Fix remove cleanup when first block has been squished
90 0090 1
91 0091 1  V02-007 ACG0167 Andrew C. Goldstein, 16-Apr-1980 19:26
92 0092 1  Previous revision history moved to F11B.REV
93 0093 1  **
94 0094 1
95 0095 1
96 0096 1  LIBRARY 'SYSS$LIBRARY:LIB.L32';
97 0097 1  REQUIRE 'SRC$:FCPDEF.B32';
98 1088 1
99 1089 1
100 1090 1  FORWARD ROUTINE
101 1091 1  ENTER : L_NORM NOVALUE, ! main ENTER routine
102 1092 1  MAKE_ENTRY : L_NORM NOVALUE, ! build new directory entry
103 1093 1  RESTORE_DIR : L_NORM NOVALUE; ! restore directory context

```

```

105 1094 1 GLOBAL ROUTINE ENTER (ABD, FIB, RESULT_LENGTH, RESULT) : L_NORM NOVALUE =
106 1095 1
107 1096 1 !++
108 1097 1
109 1098 1 FUNCTIONAL DESCRIPTION:
110 1099 1
111 1100 1     This routine enters the given file name in the specified directory.
112 1101 1
113 1102 1 CALLING SEQUENCE:
114 1103 1     ENTER (ARG1, ARG2, ARG3, ARG4)
115 1104 1
116 1105 1 INPUT PARAMETERS:
117 1106 1     ARG1: address of buffer descriptor packet
118 1107 1     ARG2: address of FIB of operation
119 1108 1
120 1109 1 IMPLICIT INPUTS:
121 1110 1     NONE
122 1111 1
123 1112 1 OUTPUT PARAMETERS:
124 1113 1     ARG3: address of longword to receive length of result string
125 1114 1     ARG4: address of result string buffer
126 1115 1
127 1116 1 IMPLICIT OUTPUTS:
128 1117 1     DIR_BUFFER: buffer address of current directory block
129 1118 1     DIR_ENTRY: address of directory record
130 1119 1     DIR_VERSION: address of directory version entry
131 1120 1     DIR_END: end of directory data
132 1121 1     PREV_VERSION: version number of superseded entry
133 1122 1
134 1123 1 ROUTINE VALUE:
135 1124 1     NONE
136 1125 1
137 1126 1 SIDE EFFECTS:
138 1127 1     directory altered, result string and length written into buffer packet
139 1128 1
140 1129 1 --
141 1130 1
142 1131 2 BEGIN
143 1132 2
144 1133 2 MAP
145 1134 2     ABD           : REF BBLOCKVECTOR [,ABD$C_LENGTH],
146 1135 2                   ! descriptor list arg
147 1136 2     FIB           : REF BBLOCK,           ! FIB argument
148 1137 2     RESULT        : REF VECTOR [,BYTE]; ! result string arg
149 1138 2
150 1139 2 LOCAL
151 1140 2     NAME_DESC       : BBLOCK [FND_LENGTH], ! file name descriptor block
152 1141 2     NAME_BUFFER     : VECTOR [FILENAME_LENGTH, BYTE], ! buffer for file name
153 1142 2     STATUS,         :                               ! result of directory search
154 1143 2     P,              :                               ! string pointer
155 1144 2     VBN,            :                               ! current VBN of directory file
156 1145 2     NAME_LENGTH;   :                               ! length of file name, rounded even
157 1146 2
158 1147 2 BIND_COMMON;
159 1148 2
160 1149 2 DIR_CONTEXT_DEF;
161 1150 2

```

```

: 162 1151 2 EXTERNAL ROUTINE
: 163 1152 2     PMS_START SUB   : L_NORM,      | start subfunction metering
: 164 1153 2     PMS_END SUB    : L_NORM,      | end subfunction metering
: 165 1154 2     DIR_ACCESS  : L_NORM,      | access the directory
: 166 1155 2     PARSE_NAME  : L_NORM,      | parse file string
: 167 1156 2     DIR_SCAN    : L_NORM,      | search the directory
: 168 1157 2     UPDATE_DIRSEQ : L_NORM,      | update directory sequence count in UCB
: 169 1158 2     ADDRESSING_MODE (LONG_RELATIVE),
: 170 1159 2     NEXT_DIR_REC  : L_NORM,      | advance to next matching record
: 171 1160 2     RETURN_DIR   : L_NORM,      | return data to buffer packet
: 172 1161 2     MARK_DIRTY   : L_NORM,      | mark buffer for rewrite
: 173 1162 2
: 174 1163 2
: 175 1164 2 ! Start metering for this subfunction.
: 176 1165 2 !
: 177 1166 2
: 178 1167 2 PMS_START_SUB (PMS_ENTER);
: 179 1168 2
: 180 1169 2 ! The file ID to be entered must be non-zero.
: 181 1170 2 !
: 182 1171 2
: 183 1172 2 IF .FIB[FIB$W_FID_NUM] EQL 0
: 184 1173 2 THEN ERR_EXIT (SS$_BADPARAM);
: 185 1174 2
: 186 1175 2 ! Find the name string in the buffer packet. Parse the string into the
: 187 1176 2 ! name descriptor block. Mask out the wild card bits, since they are
: 188 1177 2 ! not permitted. Check the version number; it must be positive.
: 189 1178 2 !
: 190 1179 2
: 191 1180 2 PARSE_NAME (NAME_DESC, NAME_BUFFER, .ABD[ABD$C_NAME, ABD$W_COUNT],
: 192 1181 2     .ABD[ABD$C_NAME, ABD$W_TEXT] + ABD[ABD$C_NAME, ABD$W_TEXT] + 1,
: 193 1182 2     .FIB[FIB$W_NMCTL] AND FIB$M_NEWVER);
: 194 1183 2 IF .NAME_DESC[FND_WILD]
: 195 1184 2 THEN ERR_EXIT (SS$_BADFILENAME);
: 196 1185 2 IF .NAME_DESC[FND_VERSION] LSS 0
: 197 1186 2 THEN NAME_DESC[FND_VERSION] = 0;
: 198 1187 2 NAME_LENGTH = .NAME_DESC[FND_COUNT] + 1 AND NOT 1;
: 199 1188 2
: 200 1189 2 ! Access the directory.
: 201 1190 2 !
: 202 1191 2
: 203 1192 2 DIR_ACCESS (.FIB, 1);
: 204 1193 2
: 205 1194 2 ! Search the directory for the indicated name. If the search succeeds,
: 206 1195 2 ! we have a duplicate entry. If supersede is enabled, do it; otherwise,
: 207 1196 2 ! take an error exit. If the search failed, make a new entry.
: 208 1197 2 !
: 209 1198 2
: 210 1199 2 DIR_RECORD = 0;
: 211 1200 2 LAST_ENTRY[0] = 0;
: 212 1201 2 STATUS = DIR_SCAN (NAME_DESC, 0, 0, 0, 0, 0, -1);
: 213 1202 2 IF .DIR_VERSION NEQ 0
: 214 1203 2 THEN FIB[FIB$W_VERLIMIT] = .VERSION_LIMIT;
: 215 1204 2
: 216 1205 2 IF .STATUS
: 217 1206 2 AND NOT .NAME_DESC[FND_MAX_VER]
: 218 1207 2 AND .NAME_DESC[FND_VERSION] NEQ 0

```

: R

```

219 1208 2 THEN
220 1209 3 BEGIN
221 1210 3 IF NOT .FIB[FIB$V SUPERSEDE]
222 1211 3 THEN ERR_EXIT (SS$_DUPFILENAME);
223 1212 3
224 1213 3 DIR_END = .DIR_VERSION;
225 1214 3 PREV_VERSION = .DIR_VERSION[DIR$W VERSION];
226 1215 3 CH$MOVE (FIB$S FID, DIR_VERSION[DIR$W FID], SUPER_FID);
227 1216 3 APPLY_RVN (SUPER_FID[FIB$W RVN], CURRENT_RVN);
228 1217 3 CH$MOVE (FIB$S FID, FIB[FIB$W FID], DIR_VERSION[DIR$W FID]);
229 1218 3 DEFAULT_RVN (DIR_VERSION[DIR$W FID_RVN], CURRENT_RVN);
230 1219 3 USER_STATUS[0] = SS$ SUPERSEDE;
231 1220 3 CLEANUP_FLAGS[CLF_SUPERSEDE] = 1;
232 1221 3
233 1222 3 ! Determine if the entry being superseded is of the form xxx.DIR;1. If
234 1223 3 ! so, bump the directory sequence count in the UCB to invalidate RMS caches.
235 1224 3 !
236 1225 3
237 1226 3 IF .DIR_VERSION[DIR$W_VERSION] EQL 1 ! simple tests first
238 1227 3 THEN
239 1228 4 BEGIN
240 1229 4 P = CH$FIND_SUB (.DIR_ENTRY[DIR$B_NAMECOUNT], DIR_ENTRY[DIR$T_NAME],
241 1230 4 4, UPLIT_BYTE ('.DIR'));
242 1231 4 IF NOT CH$FAIL (.P)
243 1232 4 AND .P + 4 EQL DIR_ENTRY[DIR$T_NAME] + .DIR_ENTRY[DIR$B_NAMECOUNT]
244 1233 4 THEN KERNEL_CALL (UPDATE_DIRSEQ);
245 1234 4 END;
246 1235 3 END
247 1236 3
248 1237 3 ! The operation is not a supersede. Create the new directory entry.
249 1238 3 !
250 1239 3
251 1240 2 ELSE
252 1241 2 MAKE_ENTRY (NAME_DESC, .FIB);
253 1242 2
254 1243 2 ! Determine whether higher or lower versions exist of the new entry.
255 1244 2 ! This is deduced from the relative position of the new version in the record.
256 1245 2 !
257 1246 2
258 1247 2 IF .VERSION_COUNT GTRU 0
259 1248 2 THEN FIB[FIB$V_HIGHVER] = 1;
260 1249 2
261 1250 2 IF .DIR_VERSION LSSA .DIR_ENTRY + .DIR_ENTRY[DIR$W_SIZE] + 2 - DIR$C_VERSION
262 1251 2 OR
263 1252 3 BEGIN
264 1253 3 VBN = .DIR_VBN;
265 1254 3 NEXT_DIR_REC (.DIR_ENTRY, VBN) NEQ 0
266 1255 3 END
267 1256 2 THEN FIB[FIB$V_LOWVER] = 1;
268 1257 2
269 1258 2 FIB[FIB$W_VERLIMIT] = .VERSION_LIMIT;
270 1259 2
271 1260 2 ! Write out the modified directory block and
272 1261 2 ! return the resultant directory string to the caller.
273 1262 2 !
274 1263 2
275 1264 2 KERNEL_CALL (RETURN_DIR, .RESULT_LENGTH, .RESULT, .ABD);

```

ENTER
V04-001

F 1
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2 Page 6 (2)

ENT
V04

```

: 276      1265  2
: 277      1266  2 MARK_DIRTY (.DIR_BUFFER);
: 278      1267  2
: 279      1268  2 PMS_END_SUB ();
: 280      1269  2
: 281      1270  1 END;

```

! end of routine ENTER

```

.TITLE ENTER
.IDENT  \V04-001\
.PSECT  $CODE$,NOWPT,2
        52 49 44 2E 0000 P.AAA: .ASCII  \.DIR\
.EXTRN  PMS_START_SUB, PMS_END_SUB
.EXTRN  DIR_ACCESS, PARSE_NAME
.EXTRN  DIR_SCAN, UPDATE_DIRSEQ
.EXTRN  NEXT_DIR_REC, RETURN_DIR
.EXTRN  MARK_DIRTY

        .ENTRY  ENTER, Save R2,R3,R4,R5,R6,R7,R8
        SE      9C  AE 9E 00002  MOVAB  -100(SP), SP
        56      00DC CA 9E 00006  MOVAB  220(BASE), R6
        57      01FE CA 9E 0000B  MOVAB  510(BASE), R7
        58      0C   A6 9E 00010  MOVAB  12(R6), R8
        0000G   CF   07 DD 00014  PUSHL  #7
        50      08   01 FB 00016  CALLS  #1, PMS_START_SUB
        04      A0 B5 0001F  MOVL   FIB, R0
        03      12 00022  TSTW  4(R0)
        14      BF 00024  BNEQ  1$
        04      00026  CHMU  #20
        50      08  AC 00 00027  1$:  MOVL   FIB, R0
        51      14  A0 3C 0002B  MOVZWL 20(R0), R1
        7E      51 FFFFDF 8F CB 0002F  BICL3  #-513, R1, -(SP)
        51      04  AC 00 00037  MOVL   ABD, R1
        52      10  A1 9E 0003B  MOVAB  16(R1), R2
        50      62 3C 0003F  MOVZWL (R2), R0
        01 A240 9F 00042  PUSHAB 1(R2)[R0]
        7E      12  A1 7F 00046  MOVZWL 18(R1), -(SP)
        10      AE 7F 0004A  PUSHAB NAME_BUFFER
        64      AE 9F 0004D  PUSHAB NAME_DESC
        0000G   CF 05 05 FB 00050  CALLS  #5, PARSE_NAME
        05      55  AE E9 00055  BLBC  NAME_DESC+1, 2$
        0818    8F BF 00059  CHMU  #2072
        04      0005D  RET
        60      AF B5 0005E  2$:  TSTW  NAME_DESC+12
        03      18 00061  BGEQ  3$
        60      AE B4 00063  CLRW  NAME_DESC+12
        50      58  AE 01 C1 00066  3$:  ADDL3  #1, NAME_DESC+4, R0
        50      01 01 8A 0006B  BICB2  #1, NAME_LENGTH
        01      01 DD 0006E  PUSHL  #1
        0000G   CF 08  AC DD 00070  PUSHL  FIB
        00D8    CA D4 00073  CALLS  #2, DIR_ACCESS
        1C      A6 94 0007C  CLRL  216(BASE)
        CLRB  28(R6)

```

```

: 1094
: 1145
: 1147
: 1167
: 1172
: 1173
: 1182
: 1181
: 1180
: 1183
: 1184
: 1185
: 1186
: 1187
: 1192
: 1199
: 1200

```


			7E		01	CE	0007F		MNEGL	#1, -(SP)	1201
					7E	7C	00082		CLRQ	-(SP)	
					7E	7C	00084		CLRQ	-(SP)	
					7E	D4	00086		CLRL	-(SP)	
				6C	AE	9F	00088		PUSHAB	NAME_DESC	
		0000G	CF		07	FB	0008B		CALLS	#7, DIR_SCAN	
					68	D5	00090		TSTL	(R8)	1202
					09	13	00092		BEQL	4\$	
			51	08	AC	D0	00094		MOVL	FIB, R1	1203
		2C	A1	18	A6	B0	00098		MOVW	24(R6), 44(R1)	
			03		50	E8	0009D	4\$:	BLBS	STATUS, 6\$	1205
					00A5	31	000A0	5\$:	BRW	12\$	
F8		55	AE		01	E0	000A3	6\$:	BBS	#1, NAME_DESC+1, 5\$	1206
				60	AE	B5	000AB		TSTW	NAME_DESC+12	1207
					F3	13	000AB		BEQL	5\$	
			50	08	AC	D0	000AD		MOVL	FIB, R0	1210
05		15	A0		02	E0	000B1		BBS	#2, 21(R0), 7\$	
				0868	8F	BF	000B6		CHMU	#2152	1211
					04	000BA			RET		
		10	A6		68	D0	000BB	7\$:	MOVL	(R8), 16(R6)	1213
		0152	CA	00	B8	32	000BF		CVTWL	20(R8), 338(BASE)	1214
			50		68	D0	000C5		MOVL	(R8), R0	1215
67		02	A0		06	28	000C8		MOVC3	#6, 2(R0), (R7)	
				04	A7	95	000CD		TSTB	4(R7)	1216
					05	12	000D0		BNEQ	8\$	
		04	A7	A0	AA	90	000D2		MOVB	-96(BASE), 4(R7)	
			01	04	A7	91	000D7	8\$:	CMPB	4(R7), #1	
					08	12	000DB		BNEQ	9\$	
				A0	AA	D5	000DD		TSTL	-96(BASE)	
					03	12	000E0		BNEQ	9\$	
				04	A7	94	000E2		CLRB	4(R7)	
			51	08	AC	D0	000E5	9\$:	MOVL	FIB, R1	1217
			50		68	D0	000E9		MOVL	(R8), R0	
		02	A0	04	A1	06	000EC		MOVC3	#6, 4(R1), 2(R0)	
					50	68	000F2		MOVL	(R8), R0	1218
A0	AA	06	A0		00	ED	000F5		CMPZV	#0, #8, 6(R0), -96(BASE)	
					03	12	000FC		BNEQ	10\$	
				06	A0	94	000FE		CLRB	6(R0)	
				80	AA	31	00101	10\$:	MOVZWL	#1585, -128(BASE)	1219
					6A	20	00107		BISB2	#32, (BASE)	1220
					01	00	0010A		CMPW	20(R8), #1	1226
					43	12	0010E		BNEQ	13\$	
			50	08	A6	D0	00110		MOVL	8(R6), R0	1229
			51	05	A0	9A	00114		MOVZBL	5(R0), R1	
0G	A0		51	FEDF	CF	04	00118		MATCHC	#4, P.AAA, R1, 6(R0)	1230
					03	13	00120		BEQL	11\$	
			53		04	D0	00122		MOVL	#4, R3	
			53		04	C2	00125	11\$:	SUBL2	#4, R3	
					29	13	00128		BEQL	13\$	1231
			53		04	C0	0012A		ADDL2	#4, R3	1232
			51	08	A6	D0	0012D		MOVL	8(R6), R1	
			50	05	A1	9A	00131		MOVZBL	5(R1), R0	
			50	06	A140	9E	00135		MOVAB	6(R1)[R0], R0	
			50		53	D1	0013A		CMP	R3, R0	
					14	12	0013D		BNEQ	13\$	
00000000G			EF		00	FB	0013F		CALLS	#0, UPDATE_DIRSEQ	1233
					0B	11	00146		BRB	13\$	1205

: R

: .

: .

: .

: .

: .

ENTER
V04-001

H 1
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2

Page 8
(2)

		08	AC	DD	00148	12\$:	PUSHL	FIB		1241
		58	AE	9F	0014B		PUSHAB	NAME_DESC		
0000V	CF		02	FB	0014E		CALLS	#2, MAKE_ENTRY		
		1A	A6	B5	00153	13\$:	TSTW	26(R6)		1247
			09	13	00156		BEQL	14\$		
	50	08	AC	DD	00158		MOVL	FIB, R0		1248
15	A0	80	8F	88	0015C		BISB2	#128, 21(R0)		
	50	08	B6	3C	00161	14\$:	MOVZWL	@8(R6), R0		1250
	50	08	A6	C0	00165		ADDL2	8(R6), R0		
	50		06	C2	00169		SUBL2	#6, R0		
	50		68	D1	0016C		CMPL	(R8), R0		
			11	1F	0016F		BLSSU	15\$		
	6E		66	DD	00171		MOVL	(R6), VBN		1253
			5E	DD	00174		PUSHL	SP		1254
		08	A6	DD	00176		PUSHL	8(R6)		
0000G	CF		02	FB	00179		CALLS	#2, NEXT_DIR_REC		
			50	D5	0017E		TSTL	R0		
			09	13	00180		BEQL	16\$		
	50	08	AC	DD	00182	15\$:	MOVL	FIB, R0		1256
15	A0	40	8F	88	0018C		BISB2	#64, 21(R0)		
	50	08	AC	DD	0018B	16\$:	MOVL	FIB, R0		1258
2C	A0	18	A6	B0	0018F		MOVW	24(R6), 44(R0)		
		04	AC	DD	00194		PUSHL	ABD		1264
	7E	0C	AC	7D	00197		MOVQ	RESULT_LENGTH, -(SP)		
0000G	CF		03	FB	0019B		CALLS	#3, RETURN_DIR		
		04	A6	DD	001A0		PUSHL	4(R6)		1266
0000G	CF		01	FB	001A3		CALLS	#1, MARK_DIRTY		
0000G	CF		00	FB	001A8		CALLS	#0, PMS_END_SUB		1268
			04	001AD			RET			1270

; Routine Size: 430 bytes, Routine Base: \$CGDE\$ + 0004

ENTR
V04-

S
R
E
L
M
C

```

283 1271 1 GLOBAL ROUTINE MAKE_ENTRY (NAME_DESC, FIB) : L_NORM NOVALUE =
284 1272 1
285 1273 1 ++
286 1274 1
287 1275 1 FUNCTIONAL DESCRIPTION:
288 1276 1
289 1277 1     This routine makes a new directory entry, either for a new file
290 1278 1     name or for a new version of an existing file.
291 1279 1
292 1280 1
293 1281 1 CALLING SEQUENCE:
294 1282 1     MAKE_ENTRY (ARG1, ARG2)
295 1283 1
296 1284 1 INPUT PARAMETERS:
297 1285 1     ARG1: address of file name descriptor block
298 1286 1     ARG2: address of user FIB
299 1287 1
300 1288 1 IMPLICIT INPUTS:
301 1289 1     DIR_VBN: VBN of current directory block
302 1290 1     DIR_BUFFER: buffer address of current directory block
303 1291 1     DIR_ENTRY: address of directory record
304 1292 1     DIR_VERSION: address of directory version entry
305 1293 1
306 1294 1 OUTPUT PARAMETERS:
307 1295 1     NONE
308 1296 1
309 1297 1 IMPLICIT OUTPUTS:
310 1298 1     DIR_END: end of directory data
311 1299 1     PREV_VERSION: version number of superseded entry
312 1300 1
313 1301 1 ROUTINE VALUE:
314 1302 1     NONE
315 1303 1
316 1304 1 SIDE EFFECTS:
317 1305 1     directory buffer altered, directory may be extended
318 1306 1
319 1307 1 --
320 1308 1
321 1309 2 BEGIN
322 1310 2
323 1311 2 !*****
324 1312 2 SWITCHES NOSAFE;
325 1313 2 ! To prevent incorrect formation of a value CSE on DIR_VERSION across the
326 1314 2 ! call to DIR_SCAN in the auto-purge logic. Compiler: T4.0-741.
327 1315 2 !*****
328 1316 2
329 1317 2 MAP
330 1318 2     NAME_DESC      : REF BBLOCK,    ! file name descriptor block
331 1319 2     FIB            : REF BBLOCK;    ! user FIB
332 1320 2
333 1321 2 LOCAL
334 1322 2     STATUS,        ! misc status value
335 1323 2     EOF,           ! entering at end of file flag
336 1324 2     DN             : REF BBLOCK,    ! local pointer to name descriptor
337 1325 2     NAME_LENGTH,   ! length of file name, rounded even
338 1326 2     NEW_SIZE,      ! size in bytes of new record
339 1327 2     VERSIONS,     ! version limit for new directory record

```

```

340 1328 2          SAVE_VERSION          ! saved version number of new entry
341 1329 2          SAVE_DIR_CONTEXT : BBLOCK [DCX_LENGTH]; ! saved context of enter
342 1330 2
343 1331 2 BIND_COMMON;
344 1332 2
345 1333 2 DIR_CONTEXT_DEF;
346 1334 2
347 1335 2 EXTERNAL ROUTINE
348 1336 2     DIR_SCAN          : L_NORM,          ! scan directory file
349 1337 2     READ_BLOCK     : L_NORM,          ! read a disk block
350 1338 2     REMOVE         : L_NORM,          ! remove a directory entry
351 1339 2     SHUFFLE_DIR     : L_NORM,          ! extend the directory
352 1340 2     NEXT_REC       : L_NORM,          ! get next directory record
353 1341 2     UPDATE_INDXX   : NOVALUE;        ! update directory FCB index
354 1342 2
355 1343 2
356 1344 2 ! Set up the position for the insert. If DIR_ENTRY came back zero, either
357 1345 2 ! we scanned off the end of the directory, or this is a cleanup from a REMOVE
358 1346 2 ! in which the block DIR_VBN got squished out. We read back the previous
359 1347 2 ! block, and set up to append the entry, unless it was the first, in which
360 1348 2 ! case we read the first and enter at its beginning. Scan to the end of the
361 1349 2 ! records in the block.
362 1350 2
363 1351 2
364 1352 2 EOF = 0;
365 1353 2 DN = .NAME DESC;
366 1354 2 NAME_LENGTH = .DN[.FND_COUNT] + 1 AND NOT 1;
367 1355 2 DIR_END = .DIR_ENTRY;
368 1356 2 IF .DIR_ENTRY EQL 0
369 1357 2 THEN
370 1358 3     BEGIN
371 1359 3     IF .DIR_VBN GTRU 1
372 1360 3     THEN
373 1361 4         BEGIN
374 1362 4         EOF = .EOF + 1;
375 1363 4         DIR_VBN = .DIR_VBN - 1;
376 1364 3         END;
377 1365 3     DIR_ENTRY = DIR_END = DIR_BUFFER = READ_BLOCK (.DIR_VBN+.DIR_FCB[FCB$SL_STLBN]-1,
378 1366 3     1, DIRECTORY_TYPE);
379 1367 2     END;
380 1368 2
381 1369 2 UNTIL .DIR_END[DIR$W_SIZE] EQL 65535
382 1370 2 DO DIR_END = NEXT_REC (.DIR_END);
383 1371 2 IF .EOF
384 1372 2 THEN DIR_ENTRY = .DIR_END;
385 1373 2 DIR_END = .DIR_END + 2;
386 1374 2
387 1375 2 ! If there was not a name match, we are constructing a whole new record.
388 1376 2 ! Compute the record size and see if there is enough space. If not, extend
389 1377 2 ! the directory. Then shuffle down the rest of the records and build the
390 1378 2 ! new entry. We update the directory index if this is a new last record
391 1379 2 ! in the block.
392 1380 2
393 1381 2
394 1382 2 IF .DIR_VERSION EQL 0
395 1383 2 THEN
396 1384 3     BEGIN

```

```

397 1385 3
398 1386
399 1387
400 1388
401 1389
402 1390
403 1391
404 1392
405 1393
406 1394
407 1395
408 1396
409 1397
410 1398
411 1399
412 1400
413 1401
414 1402
415 1403
416 1404
417 1405
418 1406
419 1407
420 1408
421 1409
422 1410
423 1411
424 1412
425 1413
426 1414
427 1415
428 1416
429 1417
430 1418
431 1419
432 1420
433 1421
434 1422
435 1423
436 1424
437 1425
438 1426
439 1427
440 1428
441 1429
442 1430
443 1431
444 1432
445 1433
446 1434
447 1435
448 1436
449 1437
450 1438
451 1439
452 1440
453 1441

```

```

IF .DN[FND_VERSION] EQL 0
THEN DN[FND_VERSION] = .DN[FND_VERSION] + 1;

NEW_SIZE = DIR$C_LENGTH + DIR$C_VERSION + .NAME_LENGTH;
IF .NEW_SIZE GTRO .DIR_BUFFER + 512 - .DIR_END
THEN SHUFFLE_DIR (1);

IF .DIR_ENTRY[DIR$W_SIZE] EQL 65535
THEN
    UPDATE_INDX (.DIR_VBN-1, .DN [FND_COUNT], .DN [FND_STRING],
                .DIR_FCB);

CH$MOVE (.DIR_END-.DIR_ENTRY, .DIR_ENTRY, .DIR_ENTRY+.NEW_SIZE);

DIR_ENTRY[DIR$W_SIZE] = .NEW_SIZE - 2;
VERSIONS = .FIB[FIB$W_VERLIMIT];
IF .VERSIONS EQL 0
THEN VERSIONS = .DIR_FCB[FCB$W_VERSIONS];
IF .VERSIONS EQL 0
THEN VERSIONS = 32767;
DIR_ENTRY[DIR$W_VERLIMIT] = .VERSIONS;
VERSION_LIMIT = .VERSIONS;

DIR_ENTRY[DIR$B_FLAGS] = DIR$C_FID;
DIR_ENTRY[DIR$B_NAMECOUNT] = .DN[FND_COUNT];
CH$COPY (.DN[FND_COUNT], .DN[FND_STRING],
        0, .NAME_LENGTH, DIR_ENTRY[DIR$T_NAME]);

DIR_VERSION = .DIR_ENTRY + .NEW_SIZE - DIR$C_VERSION;
END

! Otherwise we are adding a new version to an existing entry. If the
! version limit is less than maximal, check for version overflow. If
! so, move out the oldest version to be superseded. We do this by
! saving the current directory position and scanning to the lowest
! version of the file. If the scan for lowest version comes back with
! an error, this indicates that we were already at the lowest version;
! if so, give an error.

ELSE
BEGIN
IF .DN[FND_VERSION] EQL 0
OR (.DN[FND_MAX_VER] AND .DN[FND_VERSION] LEQU .DIR_VERSION[DIR$W_VERSION])
THEN DN[FND_VERSION] = .DIR_VERSION[DIR$W_VERSION] + 1;
IF .DN[FND_VERSION] LSS 0
THEN ERR_EXIT (SS$_BADFILEVER);

IF .VERSION_LIMIT LSSU 32767
THEN
BEGIN
SAVE_VERSION = .DN[FND_VERSION];
DN[FND_FLAGS] = 0;
DN[FND_VERSION] = 32768;
CH$MOVE (DCX_LENGTH, DIR_CONTEXT, SAVE_DIR_CONTEXT);
STATUS = DIR_SCAN (.DN, 0, .DIR_VBN-1, .DIR_ENTRY, .DIR_VERSION, .DIR_PRED, -1);

```

```

454 1442 4 IF .STATUS
455 1443 4 THEN
456 1444 5 BEGIN
457 1445 5 IF .VERSION_COUNT + 1 GEQU .VERSION_LIMIT
458 1446 5 THEN
459 1447 6 BEGIN
460 1448 6 PREV_VERSION = .DIR_VERSION[DIR$W_VERSION];
461 1449 6 CH$MOVE (.DIR_ENTRY[DIR$B_NAMECOUNT]+1,
462 1450 6 DIR_ENTRY[DIR$B_NAMECOUNT], PREV_NAME);
463 1451 6 CH$MOVE (FIB$S_FID, DIR_VERSION[DIR$W_FID], SUPER_FID);
464 1452 6 APPLY_RVN (SUPER_FID[FID$W_RVN], .CURRENT_RVN);
465 1453 6 USER_STATUS[0] = "SS$ FILEPORGED";
466 1454 6 CLEANUP_FLAGS[CLF_SUPERSEDE] = 1;
467 1455 6 REMOVE ? .DIR_VBN EQL .SAVE_DIR_CONTEXT[DCX_VBN]
468 1456 6 AND .DIR_ENTRY = .DIR_BUFFER EQL
469 1457 6 .SAVE_DIR_CONTEXT[DCX_ENTRY] - .SAVE_DIR_CONTEXT[DCX_BUFFER]);
470 1458 5 END;
471 1459 5 ELSE
472 1460 4 END
473 1461 4 IF .VERSION_COUNT GEQU .VERSION_LIMIT
474 1462 4 THEN EPR_EXIT (SS$_TOOMANYVER);
475 1463 4
476 1464 4 ! Now reposition to the point of the enter to make the new entry.
477 1465 4 !
478 1466 4
479 1467 4 DN[FND_VERSION] = .SAVE_VERSION;
480 1468 4 RESTORE_DIR (SAVE_DIR_CONTEXT);
481 1469 4 END;
482 1470 3
483 1471 3 ! Check available space and extend if necessary. If we are out of space
484 1472 3 ! in the block, the version is going to the front of the record, the
485 1473 3 ! record is the only one in its block, and the record has a
486 1474 3 ! predecessor, we backspace to the previous record and try to append
487 1475 3 ! the entry. This prevents pathological directory fragmentation when
488 1476 3 ! new versions are inserted in descending order, as might happen during
489 1477 3 ! a wild card copy. If this gambit fails or is not applicable, we have
490 1478 3 ! to split the block.
491 1479 3
492 1480 3
493 1481 3 IF DIR$C_VERSION GTRU .DIR_BUFFER + 512 - .DIR_END
494 1482 3 THEN
495 1483 4 BEGIN
496 1484 4 IF .DIR_ENTRY EQL .DIR_BUFFER
497 1485 4 AND .DIR_VBN GTRU 1
498 1486 4 AND .DIR_VERSION EQL .DIR_ENTRY + .NAME_LENGTH + DIR$C_LENGTH
499 1487 4 AND .BBLOCK [NEXT REC (.DIR_ENTRY), DIR$W_SIZE] EQL 65535
500 1488 4 AND CH$EQL (.DIR_ENTRY[DIR$B_NAMECOUNT]+1,
501 1489 4 DIR_ENTRY[DIR$B_NAMECOUNT],
502 1490 4 .DIR_ENTRY[DIR$B_NAMECOUNT]+1,
503 1491 4 LAST_ENTRY)
504 1492 4 THEN
505 1493 5 BEGIN
506 1494 5 DIR_VBN = .DIR_VBN - 1;
507 1495 5 DIR_BUFFER = DIR_END = READ_BLOCK (.DIR_VBN-1+.DIR_FCB[FCB$S_STLBN], 1, DIRECTORY_TYPE);
508 1496 5 DO
509 1497 6 BEGIN
510 1498 6 DIR_ENTRY = .DIR_END;

```


			01BB	31	00121		BRW	29\$		1382		
			60	B5	00124	10\$:	TSTW	(R0)		1428		
			0D	13	00126		BEQL	11\$				
11		69	09	E1	00128		BBC	#9, (DN), 12\$		1429		
		51	04	BE	D0	0012C	MOVL	@4(SP), R1				
		61	60	B1	00130		CMPW	(R0), (R1)				
			08	1A	00133		BGTRU	12\$				
60		51	04	BE	D0	00135	11\$:	MOVL	@4(SP), R1	1430		
		61	01	A1	00139		ADDW3	#1, (R1), (R0)				
			0C	A9	B5	0013D	12\$:	TSTW	12(DN)	1431		
			05	18	00140		BGEQ	13\$				
			0820	8F	BF	00142	CHMU	#2080		1432		
				04	00146		RET					
		7FFF	8F	18	A8	B1	00147	13\$:	CMPW	24(R8), #32767	1434	
				03	1F	0014D	BLSSU	14\$				
				00D7	31	0014F	BRW	23\$				
				56	0C	A9	32	00152	14\$:	CVTWL	12(DN), SAVE_VERSION	1437
				69	B4	00156	CLRW	(DN)		1438		
14	AE	OC	A9	8000	8F	B0	00158	MOVW	#-32768, 12(DN)	1439		
			68	0070	8F	28	0015E	MOVC3	#112, (R8), SAVE_DIR_CONTEXT	1440		
			7E		01	CE	00165	MNEGL	#1, -(SP)	1441		
				14	A8	DD	00168	PUSHL	20(R8)			
				0C	BE	DD	0016B	PUSHL	@12(SP)			
					6B	DD	0016E	PUSHL	(R11)			
		7E			01	C3	00170	SUBL3	#1, (R8), -(SP)			
					7E	D4	00174	CLRL	-(SP)			
					59	DD	00176	PUSHL	DN			
		0000G	CF		07	FB	00178	CALLS	#7, DIR_SCAN			
			03		50	E8	0017D	BLBS	STATUS, -15\$	1442		
					008E	31	00180	BRW	21\$			
				50	1A	A8	3C	00183	15\$:	MOVZWL	26(R8), R0	1445
						50	D6	00187	INCL	R0		
50	18	A8		10	00	ED	00189	CMPZV	#0, #16, 24(R8), R0			
					7E	1A	0018F	BGTRU	20\$			
				50	04	BE	D0	00191	MOVL	@4(SP), R0	1448	
		0152	CA		60	32	00195	CVTWL	(R0), 338(BASE)			
			50		6B	D0	0019A	MOVL	(R11), R0	1449		
			51		05	A0	9A	0019D	MOVZBL	5(R0), R1		
					51	D6	001A1	INCL	R1			
0156	CA	05	A0		51	28	001A3	MOVC3	R1, 5(R0), 342(BASE)	1450		
			50		04	BE	D0	001AA	MOVL	@4(SP), R0	1451	
	OC	BE	02	A0	06	28	001AE	MOVC3	#6, 2(R0), @12(SP)			
		50	OC	AE	04	C1	001B4	ADDL3	#4, 12(SP), R0	1452		
					60	95	001B9	TSTB	(R0)			
					09	12	001BB	BNEQ	16\$			
		50	OC	AE	04	C1	001BD	ADDL3	#4, 12(SP), R0			
				60	A0	AA	90	001C2	MOVB	-96(BASE), (R0)		
		50	OC	AE	04	C1	001C6	16\$:	ADDL3	#4, 12(SP), R0		
				01	60	91	001CB	CMPB	(R0), #1			
					0C	12	001CE	BNEQ	17\$			
					A0	AA	D5	001D0	TSTL	-96(BASE)		
					07	12	001D3	BNEQ	17\$			
		50	OC	AE	04	C1	001D5	ADDL3	#4, 12(SP), R0			
					60	94	001DA	CLRB	(R0)			
			80	AA	0679	8F	3C	001DC	17\$:	MOVZWL	#1657, -128(BASE)	1453
				6A		20	88	001E2	BISB2	#32, (BASE)	1454	
						51	D4	001E5	CLRL	R1	1455	

	14	AE		68	D1	001E7		CMPL	(R8), SAVE_DIR_CONTEXT	
				02	12	001EB		BNEQ	18\$	
50		6B	08	51	D6	001ED		INCL	R1	
52	1C	AE	18	BE	C3	001EF	18\$:	SUBL3	@8(SP), (R11), R0	1456
				53	C3	001F4		SUBL3	SAVE_DIR_CONTEXT+4, SAVE_DIR_CONTEXT+8, R2	1457
		52		50	D4	001FA		CLRL	R3	
				02	D1	001FC		CMPL	R0, R2	
				53	12	001FF		BNEQ	19\$	
		54		51	D6	00201		INCL	R3	
7E		53		54	D2	00203	19\$:	MCOML	R1, R4	1456
	0000G	CF		01	CB	00206		BICL3	R4, R3, -(SP)	
	18	A8	1A	0C	FB	0020A		CALLS	#1, REMOVE	
				05	11	0020F	20\$:	BRB	22\$	1442
				0990	B1	00211	21\$:	CMPW	26(R8), 24(R8)	1461
					1F	00216		BLSSU	22\$	
					BF	00218		CHMU	#2448	1462
					04	0021C		RET		
	0C	A9		56	B0	0021D	22\$:	MOVW	SAVE_VERSION, 12(DN)	1467
			14	AE	9F	00221		PUSHAB	SAVE_DIR_CONTEXT	1468
50	0000V	CF		01	FB	00224		CALLS	#1, RESTORE DIR	
	08	BE	00	BE	C3	00229	23\$:	SUBL3	@0(SP), @8(SP), R0	1481
		50	0200	C0	9E	0022F		MOVAB	512(R0), R0	
		08		50	D1	00234		CMPL	R0, #8	
				03	1F	00237		BLSSU	24\$	
				0091	31	00239		BRW	28\$	
	08	BE		6B	D1	0023C	24\$:	CMPL	(R11), @8(SP)	1484
				2F	12	00240		BNEQ	25\$	
		01		68	D1	00242		CMPL	(R8), #1	1485
				7F	1B	00245		BLEQU	27\$	
50		6B		57	C1	00247		ADDL3	NAME_LENGTH, (R11), R0	1486
		50		06	C0	0024B		ADDL2	#6, R0	
		50	04	BE	D1	0024E		CMPL	@4(SP), R0	
				72	12	00252		BNEQ	27\$	
				6B	DD	00254		PUSHL	(R11)	1487
	0000G	CF		01	FB	00256		CALLS	#1, NEXT_REC	
	FFFF	8F		60	B1	0025B		CMPW	(R0), #65535	
				64	12	00260		BNEQ	27\$	
		50		6B	D0	00262		MOVL	(R11), R0	1488
		51	05	A0	9A	00265		MOVZBL	5(R0), R1	
				51	D6	00269		INCL	R1	
1C	A8	05	A0	51	29	0026B		CMPC3	R1, 5(R0), 28(R8)	1489
				53	12	00271	25\$:	BNEQ	27\$	
				68	D7	00273		DECL	(R8)	1494
				02	DD	00275		PUSHL	#2	1495
				01	DD	00277		PUSHL	#1	
		50	18	BE	D0	00279		MOVL	@24(SP), R0	
58		68	30	A0	C1	0027D		ADDL3	48(R0), (R8), R8	
			FF	A8	9F	00282		PUSHAB	-1(R8)	
	0000G	CF		03	FB	00285		CALLS	#3, READ_BLOCK	
	00	BE		50	D0	0028A		MOVL	R0, @0(SP)	
	08	BE		50	D0	0028E		MOVL	R0, @8(SP)	
		6B	00	BE	D0	00292	26\$:	MOVL	@0(SP), (R11)	1498
			00	BE	DD	00296		PUSHL	@0(SP)	1499
	0000G	CF		01	FB	00299		CALLS	#1, NEXT_REC	
	00	BE		50	D0	0029E		MOVL	R0, @0(SP)	
		50	00	BE	D0	002A2		MOVL	@0(SP), R0	1501
	FFFF	8F		60	B1	002A6		CMPW	(R0), #65535	

		04	BE	00	E5	12	002AB	BNEQ	26\$:	1502	
		00	BE		BE	D0	002AD	MOVL	@0(SP), @4(SP)	:	1503	
	50	08	BE	00	02	C0	002B2	ADDL2	#2, @0(SP)	:	1504	
			50	0200	BE	C3	002B6	SUBL3	@0(SP), @8(SP), R0	:		
			08		C0	9E	002BC	MOVAB	512(R0), R0	:		
					50	D1	002C1	CMPL	R0, #8	:		
					07	1E	002C4	BGEQU	28\$:		
					01	DD	002C6	PUSHL	#1	:	1508	
		0000G	CF		01	FB	002C8	CALLS	#1, SHUFFLE_DIR	:		
		00	BB		08	A0	002CD	ADDW2	#8, @0(R11)	:	1511	
			50	04	BE	D0	002D1	MOVL	@4(SP), R0	:	1512	
	08	51	BE		50	C3	002D5	SUBL3	R0, @0(SP), R1	:		
		A0	60		51	28	002DA	MOV3	R1, (R0), 8(R0)	:		
			02		8F	88	002DF	BISB2	#64, 2(BASE)	:	1518	
			50	04	BE	D0	002E4	MOVL	@4(SP), R0	:	1519	
			60		0C	A9	002E8	MOVW	12(DN), (R0)	:		
			51		0A	AC	002EC	MOVL	FIB, R1	:	1520	
			50	04	BE	D0	002F0	MOVL	@4(SP), R0	:		
	02	A0	A1		06	28	002F4	MOV3	#6, 4(R1), 2(R0)	:		
			50	04	BE	D0	002FA	MOVL	@4(SP), R0	:	1521	
A0	AA	06	A0		00	ED	002FE	CMPZV	#0, #8, 6(R0), -96(BASE)	:		
					03	12	00305	BNEQ	30\$:		
					06	A0	94	00307	CLRB	6(R0)	:	
						04	0030A	RET		:	1523	

: Routine Size: 779 bytes, Routine Base: \$CODE\$ + 01B2

```

537 1524 1 GLOBAL ROUTINE RESTORE_DIR (CONTEXT) : L_NORM NOVALUE =
538 1525 1
539 1526 1 !++
540 1527 1
541 1528 1 FUNCTIONAL DESCRIPTION:
542 1529 1
543 1530 1 This routine repositions a directory to the location specified
544 1531 1 in the context block and copies the context back into the
545 1532 1 main context block.
546 1533 1
547 1534 1 CALLING SEQUENCE:
548 1535 1 RESTORE_DIR (ARG1)
549 1536 1
550 1537 1 INPUT PARAMETERS:
551 1538 1 ARG1: address of context block to use
552 1539 1
553 1540 1 IMPLICIT INPUTS:
554 1541 1 NONE
555 1542 1
556 1543 1 OUTPUT PARAMETERS:
557 1544 1 NONE
558 1545 1
559 1546 1 IMPLICIT OUTPUTS:
560 1547 1 DIR_CONTEXT: receives contents of supplied context
561 1548 1
562 1549 1 ROUTINE VALUE:
563 1550 1 NONE
564 1551 1
565 1552 1 SIDE EFFECTS:
566 1553 1 directory block read
567 1554 1
568 1555 1 --
569 1556 1
570 1557 2 BEGIN
571 1558 2
572 1559 2 MAP
573 1560 2 CONTEXT : REF BBLOCK; ! directory context arg
574 1561 2
575 1562 2 BIND_COMMON;
576 1563 2
577 1564 2 DIR_CONTEXT_DEF;
578 1565 2
579 1566 2 EXTERNAL ROUTINE
580 1567 2 READ_BLOCK : L_NORM; ! read a disk block
581 1568 2
582 1569 2
583 1570 2 ! We have to reread the block if either the supplied context is the
584 1571 2 ! main context (indicating a required reposition) or the supplied
585 1572 2 ! context points to a different block than the main context (indicating
586 1573 2 ! that we have actually moved).
587 1574 2
588 1575 2
589 1576 2 IF .CONTEXT EQL DIR_CONTEXT
590 1577 2 OR .CONTEXT[DCX_VBN] NEQ .DIR_VBN
591 1578 2 OR .CONTEXT[DCX_BUFFER] NEQ .DIR_BUFFER
592 1579 2 THEN
593 1580 3 BEGIN

```

```

594 1581 3 DIR_VBN = .CONTEXT[DCX_VBN];
595 1582 3 DIR_ENTRY = .CONTEXT[DCX_ENTRY] - .CONTEXT[DCX_BUFFER];
596 1583 3 IF .CONTEXT[DCX_VERSION] NEQ 0
597 1584 3 THEN DIR_VERSION = .CONTEXT[DCX_VERSION] - .CONTEXT[DCX_BUFFER];
598 1585 3 IF .CONTEXT[DCX_END] NEQ 0
599 1586 3 THEN DIR_END = .CONTEXT[DCX_END] - .CONTEXT[DCX_BUFFER];
600 1587 3 IF .CONTEXT[DCX_PRED] NEQ 0
601 1588 3 THEN DIR_PRED = .CONTEXT[DCX_PRED] - .CONTEXT[DCX_BUFFER];
602 1589 3 DIR_BUFFER = READ_BLOCK (.DIR_VBN+.DIR_FCB[FCBSL_STLBN]-1, 1, DIRECTORY_TYPE);
603 1590 3 DIR_ENTRY = .DIR_ENTRY + .DIR_BUFFER;
604 1591 3 IF .DIR_VERSION NEQ 0
605 1592 3 THEN DIR_VERSION = .DIR_VERSION + .DIR_BUFFER;
606 1593 3 IF .DIR_END NEQ 0
607 1594 3 THEN DIR_END = .DIR_END + .DIR_BUFFER;
608 1595 3 IF .DIR_PRED NEQ 0
609 1596 3 THEN DIR_PRED = .DIR_PRED + .DIR_BUFFER;
610 1597 3 VERSION_LIMIT = .CONTEXT[DCX_VERSION_LIMIT];
611 1598 3 VERSION_COUNT = .CONTEXT[DCX_VERSION_COUNT];
612 1599 3 CH$MOVE (FILENAME_LENGTH+1, .CONTEXT[DCX_NAME], LAST_ENTRY);
613 1600 3 END
614 1601 2 ELSE
615 1602 2 CH$MOVE (DCX_LENGTH, .CONTEXT, DIR_CONTEXT);
616 1603 2
617 1604 1 END;

```

! End of routine RESTORE_DIR

				003C 00000	.ENTRY	RESTORE DIR, Save R2,R3,R4,R5		1524
		52	00DC	CA 9E 00002	MOVAB	220(BASE), R2		1560
		53	04	A2 9E 00007	MOVAB	4(R2), R3		1562
		50	04	AC D0 0000B	MOVL	CONTEXT, R0		1576
		52		50 D1 0000F	CMPL	R0, R2		
				0E 13 00012	BEQL	1\$		
		62		60 D1 00014	CMPL	(R0), (R2)		1577
				09 12 00017	BNEQ	1\$		
		63	04	A0 D1 00019	CMPL	4(R0), (R3)		1578
				03 12 0001D	BNEQ	1\$		
				0095 31 0001F	BRW	8\$		
		62		60 D0 00022	MOVL	(R0), (R2)		1581
		50	04	AC D0 00025	MOVL	CONTEXT, R0		1582
08	A2	08	A0	04 C3 00029	SUBL3	4(R0), 8(R0), 8(F2)		
		50	04	AC D0 00030	MOVL	CONTEXT, R0		1583
			0C	A0 D5 00034	TSTL	12(R0)		
				07 13 00037	BEQL	2\$		
0C	A2	0C	A0	04 C3 00039	SUBL3	4(R0), 12(R0), 12(R2)		1584
		50	04	AC D0 00040	MOVL	CONTEXT, R0		1585
			10	A0 D5 00044	TSTL	16(R0)		
				07 13 00047	BEQL	3\$		
10	A2	10	A0	04 C3 00049	SUBL3	4(R0), 16(R0), 16(R2)		1586
		50	04	AC D0 00050	MOVL	CONTEXT, R0		1587
			14	A0 D5 00054	TSTL	20(R0)		
				07 13 00057	BEQL	4\$		
14	A2	14	A0	04 C3 00059	SUBL3	4(R0), 20(R0), 20(R2)		1588
				02 DD 00060	PUSHL	#2		1589
				01 DD 00062	PUSHL	#1		

ENTER
V04-001

G 2
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[FIX.SRC]ENTER.B32;2

Page 20
(4)

ERA
V04

50	50	00D0	CA	D0	00064	MOVL	208(BASE), R0	:			
	62	30	A0	C1	00069	ADDL3	48(R0), (R2), R0	:			
		FF	A0	9F	0006E	PUSHAB	-1(R0)	:			
0000G	CF		03	FB	00071	CALLS	#3, READ_BLOCK	:			
	63		50	D0	00076	MOVL	R0, (R3)	:			
08	A2		63	C0	00079	ADDL2	(R3), 8(R2)	:	1590		
		0C	A2	D5	0007D	TSTL	12(R2)	:	1591		
			04	13	00080	BEQL	5\$:			
	0C	A2	63	C0	00082	ADDL2	(R3), 12(R2)	:	1592		
		10	A2	D5	00086	TSTL	16(R2)	:	1593		
			04	13	00089	BEQL	6\$:			
	10	A2	63	C0	0008B	ADDL2	(R3), 16(R2)	:	1594		
		14	A2	D5	0008F	TSTL	20(R2)	:	1595		
			04	13	00092	BEQL	7\$:			
	14	A2	63	C0	00094	ADDL2	(R3), 20(R2)	:	1596		
		04	AC	D0	00098	MOVL	CONTEXT, R0	:	1597		
	18	A2	18	A0	B0	0009C	MOVW	24(R0), 24(R2)	:		
		50	04	AC	D0	000A1	MOVL	CONTEXT, R0	:	1598	
	1A	A2	1A	A0	B0	000A5	MOVW	26(R0), 26(R2)	:		
		50	04	AC	D0	000AA	MOVL	CONTEXT, R0	:	1599	
1C	A2	1C	A0	0051	8F	28	000AE	MOV3	#81, 28(R0), 28(R2)	:	
					04	000B6	RET	:	1576		
	62		60	0070	8F	28	000B7	MOV3	#112, (R0), (R2)	:	1602
					04	000BD	RET	:	1604		

: Routine Size: 190 bytes, Routine Base: \$CODE\$ + 048D

```

: 618      1605 1
: 619      1606 1 END
: 620      1607 0 ELUDOM

```

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	1403	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

SR
C

ENTER
V04-001

H 2
16-Sep-1984 00:21:45
14-Sep-1984 12:30:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]ENTER.B32;2 Page 21
(4)

**F

COMMAND QUALIFIERS

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

: Size: 1399 code + 4 data bytes
: Run Time: 00:51.8
: Elapsed Time: 01:35.9
: Lines/CPU Min: 1861
: Lexemes/CPU-Min: 47008
: Memory Used: 377 pages
: Compilation Complete

ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE
ELITE	ELITE	ELITE	DEACCS LIS	ELITE	DELETE LIS	ELITE	DIRSCH LIS	ELITE	ELITE
ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	DISPAT LIS	ELITE
ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE
ELITE	ELITE	CREHDR LIS	ELITE	ELITE	ELITE	DIRACC LIS	ELITE	ELITE	ELITE
ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE
ELITE	ELITE	ELITE	ELITE	DELBAD LIS	ELITE	ELITE	ELITE	ELITE	ELITE
ELITE	CREFCB LIS	ELITE	CREWIN LIS	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE
ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE
ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE
ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	DISPATCH LIS	ENTER LIS
ELITE	ELITE	ELITE	ELITE	ELITE	DELFT LIS	ELITE	ELITE	ELITE	ELITE
ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE	ELITE

