


```

MM      MM      AAAAAA      KK      KK      NN      NN      MM      MM      88898888
MM      MM      AAAAAA      KK      KK      NN      NN      MM      MM      88888888
MMMM    MMMM    AA      AA      KK      KK      NN      NN      MMMM    MMMM    88      88
MMMM    MMMM    AA      AA      KK      KK      NN      NN      MMMM    MMMM    88      88
MM  MM  MM  AA      AA      KK      KK      NNNN    NN      MM  MM  MM  88      88
MM  MM  MM  AA      AA      KK      KK      NNNN    NN      MM  MM  MM  88      88
MM      MM  AA      AA      KKKKKK      NN      NN      NN      MM      MM  88888888
MM      MM  AA      AA      KKKKKK      NN      NN      NN      MM      MM  88888888
MM      MM  AAAAAAAAAA      KK      KK      NN      NNNN    MM      MM  88      88
MM      MM  AAAAAAAAAA      KK      KK      NN      NNNN    MM      MM  88      88
MM      MM  AA      AA      KK      KK      NN      NN      MM      MM  88      88
MM      MM  AA      AA      KK      KK      NN      NN      MM      MM  88      88
MM      MM  AA      AA      KK      KK      NN      NN      MM      MM  88888888
MM      MM  AA      AA      KK      KK      NN      NN      MM      MM  88888888

```

```

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
LLLLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLLLL 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 MAKNMB (
0002 0
0003 0     LANGUAGE (BLISS32),
0004 0     IDENT = 'V04-000'
0005 0 ) =
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 1
0034 1
0035 1 ABSTRACT:
0036 1
0037 1     This routine converts a file name string into the
0038 1     RAD-50 name block format.
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: 2-Jan-1977 17:06
0049 1
0050 1 MODIFIED BY:
0051 1
0052 1     V03-004 CDS0003      Christian D. Saether    2-Jan-1984
0053 1     Use longword addressing on external FIL$ routine.
0054 1
0055 1     V03-003 CDS0002      Christian D. Saether    6-Dec-1983
0056 1     Change LIB$ references to FIL$.
0057 1

```

```
58 0058 1 : V03-002 ACG0302 Andrew C. Goldstein, 3-Dec-1982 13:55
59 0059 1 : Add $ and _ to file names, allow long names
60 0060 1 :
61 0061 1 : V03-001 CDS0001 C. Saether 1-Jul-1982
62 0062 1 : Don't force absolute addressing mode when declaring
63 0063 1 : external routine lib$cvtdtb.
64 0064 1 :
65 0065 1 : A0101 ACG0057 Andrew C. Goldstein, 10-Aug-1979 16:41
66 0066 1 : Wild card interface changes
67 0067 1 :
68 0068 1 : A0100 ACG00001 Andrew C. Goldstein, 10-Oct-1978 20:03
69 0069 1 : Previous revision history moved to F11A.REV
70 0070 1 : **
71 0071 1 :
72 0072 1 :
73 0073 1 : LIBRARY 'SYSS$LIBRARY:LIB.L32';
74 0074 1 : REQUIRE 'SRC$FCPDEF.B32';
75 1065 1 :
76 1066 1 :
77 1067 1 : Linkages to subroutines in this module.
78 1068 1 :
79 1069 1 :
80 1070 1 : LINKAGE
81 1071 1 : L_GETCHAR = JSB :
82 1072 1 : NOPRESERVE (5)
83 1073 1 : GLOBAL (COUNT = 6, STRINGP = 7, FCOUNT = 8),
84 1074 1 :
85 1075 1 : L_GETSTAR = JSB :
86 1076 1 : GLOBAL (COUNT = 6, STRINGP = 7),
87 1077 1 :
88 1078 1 : L_TYPE = JSB :
89 1079 1 : GLOBAL (COUNT = 6, STRINGP = 7);
90 1080 1 :
91 1081 1 : Routines in this module
92 1082 1 :
93 1083 1 :
94 1084 1 : FORWARD ROUTINE
95 1085 1 : MAKE_NAMEBLOCK : NOVALUE, : main routine
96 1086 1 : GETCHAR : L_GETCHAR, : get RAD-50 set character
97 1087 1 : GETSTAR : L_GETSTAR, : get star character, if any
98 1088 1 : TYPE : L_TYPE; : determine type of current character
```

```

100 1089 1 GLOBAL ROUTINE MAKE_NAMEBLOCK (LENGTH, STRING, NAMEBLOCK) : NOVALUE =
101 1090 1
102 1091 1 **
103 1092 1
104 1093 1 FUNCTIONAL DESCRIPTION:
105 1094 1
106 1095 1     This routine converts a file name string into the
107 1096 1     RAD-50 name block format.
108 1097 1
109 1098 1 CALLING SEQUENCE:
110 1099 1     MAKE_NAMEBLOCK (ARG1, ARG2, ARG3)
111 1100 1
112 1101 1 INPUT PARAMETERS:
113 1102 1     ARG1: length of file name string
114 1103 1     ARG2: address of file name string
115 1104 1
116 1105 1 IMPLICIT INPUTS:
117 1106 1     NONE
118 1107 1
119 1108 1 OUTPUT PARAMETERS:
120 1109 1     ARG3: address of file name block
121 1110 1
122 1111 1 IMPLICIT OUTPUTS:
123 1112 1     NONE
124 1113 1
125 1114 1 ROUTINE VALUE:
126 1115 1     NONE
127 1116 1
128 1117 1 SIDE EFFECTS:
129 1118 1     NONE
130 1119 1
131 1120 1 --
132 1121 1
133 1122 2 BEGIN
134 1123 2
135 1124 2 MAP
136 1125 2     NAMEBLOCK      : REF BBLOCK;      ! name block arg
137 1126 2
138 1127 2 GLOBAL REGISTER
139 1128 2     COUNT = 6,      ! characters remaining in string
140 1129 2     STRINGP = 7,   : REF VECTOR [,BYTE], ! string pointer
141 1130 2     FCOUNT = 8;   ! count of chars in current field
142 1131 2
143 1132 2 LOCAL
144 1133 2     VERSION,      ! file version number
145 1134 2     P,            ! string scan pointer
146 1135 2     BLOCKP       : REF VECTOR [,WORD]; ! pointer into name block
147 1136 2
148 1137 2 EXTERNAL ROUTINE
149 1138 2     FIL&CVT_DTB   : ADDRESSING_MODE (GENERAL); ! decimal to binary convert
150 1139 2
151 1140 2 ! Initialize all the locals.
152 1141 2
153 1142 2
154 1143 2 CHSFILL (0, NMBSC_LENGTH, .NAMEBLOCK); ! zero the entire block
155 1144 2 STRINGP = .STRING; ! set up string pointer
156 1145 2 P = CHSFIND_CH (.LENGTH, .STRINGP, ' '); ! look for a terminating space

```

SRJLJC

```

157 1146 COUNT = .P = .STRINGP;           ! compute count
158 1147 IF (MSFAIL (.P))
159 1148 THEN COUNT = .LENGTH;           ! use whole string if no space
160 1149 BLOCKP = NAMEBLOCK[NMBSW_NAME]; ! point to name field in block
161 1150 FCOUNT = 0;                   ! init chars in field count
162 1151
163 1152
164 1153 ! Build the name field, consisting of 3 words of 3 RAD-50 characters per word.
165 1154 !
166 1155
167 1156 DECR I FROM 3 TO 1 DO
168 1157 BEGIN
169 1158   DECR J FROM 3 TO 1 DO
170 1159     BLOCKP[0] = .BLOCKP[0] * 40 + GETCHAR ();
171 1160     BLOCKP = .BLOCKP + 2;
172 1161   END;
173 1162
174 1163 ! Eat remaining trailing name field characters.
175 1164 !
176 1165
177 1166 WHILE 1 DO
178 1167 BEGIN
179 1168   CASE TYPE () FROM 0 TO 6 OF
180 1169     SET
181 1170     [0,4,5,6]: EXITLOOP;
182 1171     [INRANGE, OVRANGE]:
183 1172     BEGIN
184 1173       COUNT = .COUNT - 1;
185 1174       STRINGP = .STRINGP + 1;
186 1175     END;
187 1176   TES;
188 1177 END;
189 1178
190 1179 IF GETSTAR ()           ! set wild card bits if star
191 1180 THEN
192 1181 BEGIN
193 1182   NAMEBLOCK[NMBSV_WILD] = 1;
194 1183   NAMEBLOCK[NMBSV_ALLNAM] = 1;
195 1184 END;
196 1185
197 1186 ! Pick up the name delimiter, which is either dot or end of string.
198 1187 !
199 1188
200 1189 CASE TYPE () FROM 1 TO 5 OF
201 1190 SET
202 1191 [1,2,3,4]: ERR_EXIT (SS$_BADFILENAME);
203 1192 [5]: BEGIN
204 1193   COUNT = .COUNT - 1;           ! pick up the character
205 1194   STRINGP = .STRINGP + 1;
206 1195 END;
207 1196 [OVRANGE]: 0;
208 1197 TES;
209 1198
210 1199 ! Now build the type field, consisting of 1 word of 3 RAD-50 characters.
211 1200 !
212 1201
213 1202 FCOUNT = 0;                   ! re-init chars in field count

```

```

214 1203
215 1204
216 1205
217 1206
218 1207
219 1208
220 1209
221 1210
222 1211
223 1212
224 1213
225 1214
226 1215
227 1216
228 1217
229 1218
230 1219
231 1220
232 1221
233 1222
234 1223
235 1224
236 1225
237 1226
238 1227
239 1228
240 1229
241 1230
242 1231
243 1232
244 1233
245 1234
246 1235
247 1236
248 1237
249 1238
250 1239
251 1240
252 1241
253 1242
254 1243
255 1244
256 1245
257 1246
258 1247
259 1248
260 1249
261 1250
262 1251
263 1252
264 1253
265 1254
266 1255
267 1256
268 1257
269 1258
270 1259

```

```

DECR J FROM 3 TO 1 DO
    BLOCKP[0] = .BLOCKP[0] + 40 + GETCHAR ();
; Eat remaining trailing type field characters.
;
WHILE 1 DO
    BEGIN
        CASE TYPE () FROM 0 TO 6 OF
            SET
                [0,4,5,6]: EXITLOOP;
                [INRANGE, OTRANGE]:
                    BEGIN
                        COUNT = .COUNT - 1;
                        STRINGP = .STRINGP + 1;
                    END;
            TES;
        END;
    IF GETSTAR () ! set wild card bits if star
    THEN
        BEGIN
            NAMEBLOCK[NMBSV_WILD] = 1;
            NAMEBLOCK[NMBSV_ALLTYP] = 1;
        END;
    ; Pick up the type delimiter, which may be dot, semicolon, or end of string.
    ;
    CASE TYPE () FROM 1 TO 6 OF
        SET
            [1,2,3,4]: ERR_EXIT (SS$_BADFILENAME);
            [5,6]:
                BEGIN
                    COUNT = .COUNT - 1; ! pick up the character
                    STRINGP = .STRINGP + 1;
                END;
        [OTRANGE]: 0;
        TES;
    ;
    ; If the version is not wild card and there are still characters present,
    ; get the binary version number.
    ;
    IF GETSTAR () ! set wild card bits if star
    THEN
        BEGIN
            NAMEBLOCK[NMBSV_WILD] = 1;
            NAMEBLOCK[NMBSV_ALLVER] = 1;
        END;
    ELSE IF .COUNT GTR 0
    THEN
        BEGIN
            BLOCKP = .BLOCKP + 2;
            IF NOT FIL$CVT DTB (.COUNT, .STRINGP, VERSION)
            THEN ERR_EXIT (SS$_BADFILENAME);
            IF .VERSION GTRU 32767

```

```

: 271      1260      3      THEN ERR_EXIT (SS$_BADFILEVER);
: 272      1261      3      (.BLOCKPT<0.16> = .VERSION;
: 273      1262      3      END;
: 274      1263      3
: 275      1264      3      RETURN 1;
: 276      1265      3
: 277      1266      3      END;

```

: end of routine MAKE_NAMEBLOCK

```

.TITLE  MAKNMB
.IDENT  \V04-000\
.EXTRN  FILSCVT_DTB
.PSECT  $CODE$,NOWRT,2
.ENTRY  MAKE_NAMEBLOCK, Save R2,R3,R4,R5,R6,R7,R8,- 1089
        OFFC 00000
        5B 0000V CF 9E 00002  MOVAB GETSTAR, R11
        5A 0000V CF 9E 00007  MOVAB TYPE, R10
        5E 04 C2 0000C  SUBL2 #4, SP
        28 00 6E 00 2C 0000F  MOVCS #0, (SP), #0, #40, @NAMEBLOCK 1143
        57 0C BC 00 00014
        67 04 AC 08 AC D0 00016  MOVL STRING, STRINGP 1144
        51 02 3A 0001A  LOCC #32, LENGTH, (STRINGP) 1145
        56 51 02 12 0001F  BNEQ 1$
        51 57 C3 00023 1$:  CLRL R1
        51 51 D5 00027  SUBL3 STRINGP, P, COUNT 1146
        04 04 12 00029  TSTL P 1147
        52 0C AC 04 AC D0 0002B  BNEQ 2$
        56 06 C1 0002F 2$:  MOVL LENGTH, COUNT 1148
        59 58 D4 00034  ADDL3 #6, NAMEBLOCK, BLOCKP 1149
        53 03 D0 00036  CLRL FCOUNT 1150
        54 03 D0 00039 3$:  MOVL #3, I 1156
        54 62 3C 0003C 4$:  MOVL #3, J 1158
        54 28 C4 0003F  MOVZWL (BLOCKP), R4
        62 54 0000V 30 00042  MULL2 #40, R4
        54 50 A1 00045  BSBW GETCHAR
        F0 53 F5 00049  ADDW3 R0, R4, (BLOCKP)
        52 02 C0 0004C  SOBGTR J, 4$
        E7 59 F5 0004F  ADDL2 #2, BLOCKP 1160
        6A 16 00052 5$:  SOBGTR I, 3$ 1156
        50 50 CF 00054  JSB TYPE 1168
        000E 06 00 50 CF 00054
        000E 000E 0014 0014 0014 00058 6$:  CASEL R0, #0, #6
        0014 0014 00060  .WORD 8$-6$,-
        7$-6$,-
        7$-6$,-
        7$-6$,-
        8$-6$,-
        8$-6$,-
        8$-6$,-
        56 D7 00066 7$:  DECL COUNT 1173
        57 D6 00068  INCL STRINGP 1174
        E6 11 0006A  BRB 5$ 1168
        6B 16 0006C 8$:  JSB GETSTAR 1179
        10 50 E9 0006E  BLBC ~0, 9$
        50 0C AC D0 00071  MOVL NAMEBLOCK, R0 1182

```


		11	A0		01	88	00075		BISB2	#1, 17(R0)		
			50	OC	AC	D0	00079		MOVL	NAMEBLOCK, R0		1183
		10	A0		20	88	0007D		BISB2	#32, 16(R0)		
					6A	16	00081	98	JSB	TYPE		1189
0098	04 0098		01		50	CF	00083		CASEL	R0, #1, #4		
			0098		0098		00087	108:	.WORD	238-108,-		
					000C		0008F			238-108,-		
										238-108,-		
										238-108,-		
										238-108,-		
										118-108		
					04	11	00091		BRB	128		
					56	D7	00093	118:	DECL	COUNT		1193
					57	D6	00095		INCL	STRINGP		1194
					58	D4	00097	128:	CLRL	FCOUNT		1202
			53		03	D0	00099		MOVL	#3, J		1204
			54		62	3C	0009C	138:	MOVZWL	(BLOCKP), R4		1205
			54		28	C4	0009F		MULL2	#40, R4		
					3000V	30	000A2		BSBW	GETCHAR		
					50	A1	000A5		ADDW3	R0, R4, (BLOCKP)		
	62		54		53	F5	000A9		SOBGTR	J, 138		
			FO		6A	16	000AC	148:	JSB	TYPE		1212
					50	CF	000AE		CASEL	R0, #0, #6		
000E	06 000E 0014		00		0014		000B2	158:	.WORD	178-158,-		
			0014		0014		000BA			168-158,-		
										168-158,-		
										168-158,-		
										178-158,-		
										178-158,-		
										178-158		
					56	D7	000C0	168:	DECL	COUNT		1217
					57	D6	000C2		INCL	STRINGP		1218
					E6	11	000C4		BRB	148		1212
					68	16	000C6	178:	JSB	GETSTAR		1223
			10		50	E9	000C8		BLBC	R0, 188		
			50	OC	AC	D0	000CB		MOVL	NAMEBLOCK, R0		1226
			11		01	88	000CF		BISB2	#1, 17(R0)		
			50	OC	AC	D0	000D3		MOVL	NAMEBLOCK, R0		1227
			10		10	88	000D7		BISB2	#16, 16(R0)		
					6A	16	000DB	188:	JSB	TYPE		1233
					50	CF	000DD		CASEL	R0, #1, #5		
003E	05 003E		01		003E		000E1	198:	.WORD	238-198,-		
			000E		000E		000E9			238-198,-		
										238-198,-		
										238-198,-		
										238-198,-		
										208-198,-		
										208-198		
					04	11	000ED		BRB	218		
					56	D7	000EF	208:	DECL	COUNT		1237
					57	D6	000F1		INCL	STRINGP		1238
					68	16	000F3	218:	JSB	GETSTAR		1247
					50	E9	JOF5		BLBC	R0, 228		
			11	OC	AC	D0	000F8		MOVL	NAMEBLOCK, R0		1250
			50		01	88	000FC		BISB2	#1, 17(R0)		
			11	OC	AC	D0	00100		MOVL	NAMEBLOCK, R0		1251
			50		08	88	00104		BISB2	#8, 16(R0)		
			10		04		00108		RET			1247
					56	D5	00109	228:	TSTL	COUNT		1253

		28	15	00108	BLEQ	26\$		
	52	02	C0	00100	ADDL2	#2, BLOCKP		1256
		5E	DD	00110	PUSHL	SP		1257
	7E	56	7D	00112	MOVQ	COUNT, -(SP)		
00000000G	00	03	FB	00115	CALLS	#3, FILECVT_DTB		
	05	50	E8	0011C	BLBS	R0, 24\$		
		0818	8F	BF 0011F	23\$: CHMU	#2072		1258
			04	00123	RET			
00007FFF	8F	6E	D1	00124	24\$: CMPL	VERSION, #32767		1259
		05	1B	0012B	BLEQU	25\$		
		0820	8F	BF 0012D	CHMU	#2080		1260
			04	00131	RET			
	62	6E	B0	00132	25\$: MOVW	VERSION, (BLOCKP)		1261
			04	00135	26\$: RET			1266

; Routine Size: 310 bytes. Routine Base: \$CODE\$ + 0000

```

279 1267 1 ROUTINE GETCHAR : L_GETCHAR =
280 1268 1
281 1269 1 : **
282 1270 1
283 1271 1 : FUNCTIONAL DESCRIPTION:
284 1272 1
285 1273 1 : This routine returns the RAD-50 code of the next character in the
286 1274 1 : input string if it is in the RAD-50 set. If it is not, or end of
287 1275 1 : string has been reached, it returns zero.
288 1276 1
289 1277 1 : CALLING SEQUENCE:
290 1278 1 : GETCHAR ( )
291 1279 1
292 1280 1 : INPUT PARAMETERS:
293 1281 1 : NONE
294 1282 1
295 1283 1 : IMPLICIT INPUTS:
296 1284 1 : COUNT: characters remaining in string
297 1285 1 : STRINGP: string pointer
298 1286 1 : FCOUNT: chars in current field
299 1287 1
300 1288 1 : OUTPUT PARAMETERS:
301 1289 1 : NONE
302 1290 1
303 1291 1 : IMPLICIT OUTPUTS:
304 1292 1 : NONE
305 1293 1
306 1294 1 : ROUTINE VALUE:
307 1295 1 : character code
308 1296 1
309 1297 1 : SIDE EFFECTS:
310 1298 1 : COUNT decremented and STRINGP advanced if legal character.
311 1299 1
312 1300 1 : --
313 1301 1
314 1302 2 BEGIN
315 1303 2
316 1304 2 REGISTER
317 1305 2 CHAR = 5; ! character in process
318 1306 2
319 1307 2 EXTERNAL REGISTER
320 1308 2 COUNT = 6; ! characters remaining in string
321 1309 2 STRINGP = 7; : REF VECTOR [,BYTE], ! string pointer
322 1310 2 FCOUNT = 8; ! count of chars in current field
323 1311 2
324 1312 2
325 1313 2 ! Get the next character from the string and dispatch in its type.
326 1314 2 !
327 1315 2
328 1316 2 CHAR = .STRINGP[0];
329 1317 2
330 1318 2 CASE TYPE ( ) FROM 0 TO 8 OF
331 1319 2 SET
332 1320 2 [0,5,6]: ! end, dot, or semicolon
333 1321 2 CHAR = 0;
334 1322 2 [1]: ! upper case alpha
335 1323 2 BEGIN

```

```

336      1324      CHAR = .CHAR - 'A' + 1;      ! convert to RAD-50 code
337      1325      COUNT = .COUNT - 1;      ! advance to next character
338      1326      STRINGP = .STRINGP + 1;
339      1327      FCOUNT = .FCOUNT + 1;      ! count character in field
340      1328      END;
341      1329      [2]:      ! lower case alpha
342      1330      BEGIN
343      1331      CHAR = .CHAR - 'a' + 1;      ! convert to RAD-50 code
344      1332      COUNT = .COUNT - 1;      ! advance to next character
345      1333      STRINGP = .STRINGP + 1;
346      1334      FCOUNT = .FCOUNT + 1;      ! count character in field
347      1335      END;
348      1336      [3]:      ! numeric
349      1337      BEGIN
350      1338      CHAR = .CHAR - '0' + 30;      ! convert to RAD-50 code
351      1339      COUNT = .COUNT - 1;      ! advance to next character
352      1340      STRINGP = .STRINGP + 1;
353      1341      FCOUNT = .FCOUNT + 1;      ! count character in field
354      1342      END;
355      1343      [7]:      ! dollar sign
356      1344      BEGIN
357      1345      CHAR = '.';      ! convert to RAD-50 code
358      1346      COUNT = .COUNT - 1;      ! advance to next character
359      1347      STRINGP = .STRINGP + 1;
360      1348      FCOUNT = .FCOUNT + 1;      ! count character in field
361      1349      END;
362      1350      [8]:      ! underscore
363      1351      BEGIN
364      1352      CHAR = '_';      ! convert to RAD-50 code
365      1353      COUNT = .COUNT - 1;      ! advance to next character
366      1354      STRINGP = .STRINGP + 1;
367      1355      FCOUNT = .FCOUNT + 1;      ! count character in field
368      1356      END;
369      1357      [4]:      ! star - legal as only char in field
370      1358      BEGIN
371      1359      CHAR = '*';
372      1360      IF .FCOUNT NEQ 0 THEN ERR_EXIT (SS$_BADFILENAME);
373      1361      END;
374      1362      TES;
375      1363
376      1364      RETURN .CHAR;
377      1365
378      1366      ! end of routine GETCHAR

```

		55	67	9A	00000	GETCHAR:MOVZBL	(STRINGP), CHAR	:	1316
			0000V	30	00003	BSBW	TYPE	:	1318
	08	00	50	CF	00006	CASEL	RO, #0, #8	:	
0022	001C	0016	0012		0000A	1\$:	.WORD	:	
0027	0012	0012	0037		00012		2\$-1\$,-	:	
			002C		0001A		3\$-1\$,-	:	
							4\$-1\$,-	:	
							5\$-1\$,-	:	
							9\$-1\$,-	:	
							2\$-1\$,-	:	

						2\$-1\$, -		
						6\$-1\$, -		
						7\$-1\$		
		55	D4	0001C	2\$:	CLRL	CHAR	1321
		2C	11	0001E		BRB	10\$	
55	CO	A5	9E	00020	3\$:	MOVAB	-64(R5), CHAR	1324
		13	11	00024		BRB	8\$	1325
55	AO	A5	9E	00026	4\$:	MOVAB	-96(R5), CHAR	1331
		0D	11	0002A		BRB	8\$	1332
55		12	C2	0002C	5\$:	SUBL2	#18, CHAR	1338
		08	11	0002F		BRB	8\$	1339
55		1B	D0	00031	6\$:	MOVL	#27, CHAR	1345
		03	11	00034		BRB	8\$	1346
55		1D	D0	00036	7\$:	MOVL	#29, CHAR	1352
		56	D7	00039	8\$:	DECL	COUNT	1353
		57	D6	0003B		INCL	STRINGP	1354
		58	D6	0003D		INCL	FCOUNT	1355
		0B	11	0003F		BRB	10\$	1318
		55	D4	00041	9\$:	CLRL	CHAR	1359
		58	D5	00043		TSTL	FCOUNT	1360
		05	13	00045		BEQL	10\$	
	0818	8F	BF	00047		CHMU	#2072	
		05	05	0004B		RSB		
50		55	D0	0004C	10\$:	MOVL	CHAR, R0	1364
		05	05	0004F		RSB		1366

; Routine Size: 80 bytes, Routine Base: \$CODE\$ + 0136

2A	00	15	00002	BLEQ	1\$:
	07	01	00004	CMPB	(STRINGP), #42	:
	08	12	00007	BNEQ	1\$:
	56	D7	00009	DECL	COUNT	: 1410
	57	D6	0000B	INCL	STRINGP	: 1411
50	01	D0	0000D	MOVL	#1, R0	: 1409
		05	00010	RSB		:
	50	D4	00011	CLRL	R0	: 1407
		05	00013	RSB		: 1417

; Routine Size: 20 bytes, Routine Base: \$CODE\$ + 0186

```

432 1418 1 ROUTINE TYPE : L_TYPE =
433 1419 1
434 1420 1
435 1421 1
436 1422 1 FUNCTIONAL DESCRIPTION:
437 1423 1
438 1424 1 This routine determines the type code of the current character
439 1425 1 in the string.
440 1426 1
441 1427 1 CALLING SEQUENCE:
442 1428 1 TYPE ( )
443 1429 1
444 1430 1 INPUT PARAMETERS:
445 1431 1 NONE
446 1432 1
447 1433 1 IMPLICIT INPUTS:
448 1434 1 COUNT: number of characters left in string
449 1435 1 STRINGP: string pointer
450 1436 1
451 1437 1 OUTPUT PARAMETERS:
452 1438 1 NONE
453 1439 1
454 1440 1 IMPLICIT OUTPUTS:
455 1441 1 NONE
456 1442 1
457 1443 1 ROUTINE VALUE:
458 1444 1 type code of character:
459 1445 1 0: end of string or non-RAD-50
460 1446 1 1: upper case alpha
461 1447 1 2: lower case alpha
462 1448 1 3: numeric
463 1449 1 4: star
464 1450 1 5: dot
465 1451 1 6: semicolon
466 1452 1 7: $
467 1453 1 8: -
468 1454 1
469 1455 1 SIDE EFFECTS:
470 1456 1 NONE
471 1457 1
472 1458 1 --
473 1459 1
474 1460 2 BEGIN
475 1461 2
476 1462 2 EXTERNAL REGISTER
477 1463 2 COUNT = 6, ! characters remaining in string
478 1464 2 STRINGP = 7 : REF VECTOR [,BYTE]; ! string pointer
479 1465 2
480 1466 2 ! Character match tables. First is low character of range, second is
481 1467 2 ! high character Type is table index of the matching range.
482 1468 2
483 1469 2
484 1470 2 BIND
485 1471 2 LOWCHAR = UPLIT BYTE (0, 'Aa0*.;$-') : VECTOR [,BYTE],
486 1472 2 HIGHCHAR = UPLIT BYTE (0, 'Zz9* $-') : VECTOR [,BYTE];
487 1473 2
488 1474 2 ! If the string is empty return 0 as the type. Else search the tables.

```



```

: 489 1475 2 :
: 490 1476
: 491 1477 IF .COUNT LEQ 0 THEN RETURN 0;
: 492 1478
: 493 1479 INCR I FROM 1 TO 8 DO
: 494 1480 IF .STRINGP[0] GEQU .LOWCHAR[I]
: 495 1481 AND .STRINGP[0] LEQU .HIGHCHAR[I]
: 496 1482 THEN RETURN .I;
: 497 1483
: 498 1484 ERR_EXIT (SS$_BADFILENAME);      ! other characters are illegal
: 499 1485
: 500 1486
: 501 1487 1 END;                      ! end of routine TYPE

```

```

5F 24 3B 2E 2A 30 61 00 0019A P.AAA: .BYTE 0
41 0019B .ASCII \Aa0*.;$_\
00 001A3 P.AAB: .BYTE 0
5F 24 3B 2E 2A 39 7A 5A 001A4 .ASCII \Zz9*.;$_\

```

```

LOWCHAR= P.AAA
HIGHCHAR= P.AAB

```

		56 D5 0000	TYPE:	TSTL	COUNT		1477
		1A 15 0002		BLEQ	3\$		
	50	01 D0 0004		MOVL	#1, I		1480
	E2 AF40	67 91 0007	1\$:	CMPB	(STRINGP), LOWCHAR[I]		
		07 1F 000C		BLSSU	2\$		
	E4 AF40	67 91 000E		CMPB	(STRINGP), HIGHCHAR[I]		1481
		0B 1B 0013		BLEQU	4\$		
EE	50	08 F3 0015	2\$:	AOBLEQ	#8, I, 1\$		1480
		08 1B 0013		CHMU	#2072		1484
		05 001D		RSB			
		50 D4 001E	3\$:	CLRL	R0		1487
		05 00020	4\$:	RSB			

```
; Routine Size: 33 bytes, Routine Base: $CODE$ + 01AC
```

```

: 502 1488 1
: 503 1489 1 END
: 504 1490 0 ELUDOM

```

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	461	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	25	0	1000	00:02.0

COMMAND QUALIFIERS

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

: Size: 443 code + 18 data bytes
: Run Time: 00:15.7
: Elapsed Time: 00:33.1
: Lines/CPU Min: 5694
: Lexemes/CPU-Min: 21183
: Memory Used: 176 pages
: Compilation Complete

0171 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

