

BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL

```

BBBBBBBBB      AAAAAA      SSSSSSSS      FFFFFFFFFF      SSSSSSSS      PPPPPPPP
BBBBBBBBB      AAAAAA      SSSSSSSS      FFFFFFFFFF      SSSSSSSS      PPPPPPPP
BB           BB  AA           AA  SS           FF           SS           PP           PP
BB           BB  AA           AA  SS           FF           SS           PP           PP
BB           BB  AA           AA  SS           FF           SS           PP           PP
BB           BB  AA           AA  SS           FF           SS           PP           PP
BBBBBBBBB      AA           AA  SSSSSS      FFFFFFFFFF      SSSSSS      PPPPPPPP
BBBBBBBBB      AA           AA  SSSSSS      FFFFFFFFFF      SSSSSS      PPPPPPPP
BB           BB  AAAAAAAAAA      SS           FF           SS           PP
BB           BB  AAAAAAAAAA      SS           FF           SS           PP
BB           BB  AA           AA  SS           FF           SS           PP
BB           BB  AA           AA  SS           FF           SS           PP
BBBBBBBBB      AA           AA  SSSSSSSS      FF           SSSSSSSS      PP
BBBBBBBBB      AA           AA  SSSSSSSS      FF           SSSSSSSS      PP
...
LL           II           SSSSSSSS
LL           II           SSSSSSSS
LL           II
LL           II
LL           II
LL           II
LL           II
LL           II
LL           II
LL           II
LL           II
LL           II
LLLLLLLLLLLL  II           SSSSSSSS
LLLLLLLLLLLL  II           SSSSSSSS

```

```
1 0001 0 MODULE BAS$FSP ( IDENT = '1-006' ! File: BAS$FSP.B32 EDIT:FM1006
2 0002 0 ) =
3 0003 0
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: BASIC-PLUS-2 Miscellaneous I/O
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains the BASIC FSP function,
36 0036 1 Which returns information about the open file.
37 0037 1
38 0038 1 ENVIRONMENT: VAX-11 User Mode
39 0039 1
40 0040 1 AUTHOR: John Sauter, CREATION DATE: 18-APR-1979
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 1-001 - Original.
45 0045 1 1-002 - Change LIB$$ and OTS$$ to STR$. JBS 21-MAY-1979
46 0046 1 1-003 - Change calls to STR$COPY. JBS 16-JUL-1979
47 0047 1 1-004 - Set up ISB$A_USER_FP. JBS 25-JUL-1979
48 0048 1 1-005 - Correct an error in a comment. JBS 26-OCT-1979
49 0049 1 1-006 - Return LUB$ALQ. We had to redefine a lub location for this so
50 0050 1 bas$open can retain this value. FM 22-SEP-80
51 0051 1 --
52 0052 1
53 0053 1 !<BLF/PAGE>
```

```
55 0054 1 |
56 0055 1 | SWITCHES:
57 0056 1 |
58 0057 1 |
59 0058 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
60 0059 1 |
61 0060 1 |
62 0061 1 | LINKAGES:
63 0062 1 |
64 0063 1 |
65 0064 1 | REQUIRE 'RTLIN:OTSLNK';           ! Define linkages
66 0493 1 |
67 0494 1 |
68 0495 1 | TABLE OF CONTENTS:
69 0496 1 |
70 0497 1 |
71 0498 1 | FORWARD ROUTINE
72 0499 1 |     BASSFSP : NOVALUE;           ! Return info about the file
73 0500 1 |
74 0501 1 |
75 0502 1 | INCLUDE FILES:
76 0503 1 |
77 0504 1 |
78 0505 1 | REQUIRE 'RTLML:OTSLUB';           ! Get LUB definitions
79 0645 1 |
80 0646 1 | REQUIRE 'RTLML:OTSISB';           ! Get ISB definitions
81 0814 1 |
82 0815 1 | REQUIRE 'RTLIN:RTLPSECT';         ! Macros for defining psects
83 0910 1 |
84 0911 1 | LIBRARY 'RTLSTARLE';             ! System symbols
85 0912 1 |
86 0913 1 |
87 0914 1 | MACROS:
88 0915 1 |
89 0916 1 |     NONE
90 0917 1 |
91 0918 1 | EQUATED SYMBOLS:
92 0919 1 |
93 0920 1 |
94 0921 1 | LITERAL
95 0922 1 |     BASSK_ORG_SEQ = 0,           ! Organization is sequential
96 0923 1 |     BASSK_ORG_REL = 16,          ! Relative
97 0924 1 |     BASSK_ORG_IND = 32,          ! Indexed
98 0925 1 |     BASSK_RFM_FIX = 1,           ! Record format is fixed
99 0926 1 |     BASSK_RFM_VAR = 2,           ! variable
100 0927 1 |     BASSK_RFM_VFC = 3,           ! variable with fixed control area
101 0928 1 |     BASSK_RFM_NONE = 0,          ! (special code for virtual files)
102 0929 1 |     BASSK_RAT_FTN = 1,           ! Record attribute FORTRAN
103 0930 1 |     BASSK_RAT_CR = 2,            ! Carriage Return
104 0931 1 |     BASSK_RAT_PRN = 4,           ! Print File Format
105 0932 1 |     BASSK_RAT_NONE = 0;          ! (special code for virtual and stream)
106 0933 1 |
107 0934 1 |
108 0935 1 | PSECTS:
109 0936 1 |
110 0937 1 | DECLARE_PSECTS (BAS);           ! Declare psects for BASS facility
111 0938 1 |
```

```

: 112 0939 1 ! OWN STORAGE:
: 113 0940 1 !
: 114 0941 1 !     NONE
: 115 0942 1 !
: 116 0943 1 ! EXTERNAL REFERENCES:
: 117 0944 1 !
: 118 0945 1 !
: 119 0946 1 ! EXTERNAL ROUTINE
: 120 0947 1 !     BAS$$CB_PUSH : JSB CB_PUSH NOVALUE,      ! Load register CCB
: 121 0948 1 !     BAS$$CB_POP  : JSB CB_POP NOVALUE,        ! Done with register CCB
: 122 0949 1 !     BAS$$STOP_IO : NOVALUE,                  ! Signal fatal I/O error
: 123 0950 1 !     BAS$$STOP   : NOVALUE,                  ! Signal fatal error
: 124 0951 1 !     STR$COPY_R;                               ! Copy a string by reference
: 125 0952 1 !
: 126 0953 1 ! +
: 127 0954 1 ! The following are the error codes used in this module.
: 128 0955 1 ! -
: 129 0956 1 !
: 130 0957 1 ! EXTERNAL LITERAL
: 131 0958 1 !     BAS$K_IO_CHANOT : UNSIGNED (8),          ! Channel not open.
: 132 0959 1 !     BAS$K_MAXMEMEXC : UNSIGNED (8),          ! Max memory exceeded
: 133 0960 1 !     BAS$K_PROLOSSOR : UNSIGNED (8),          ! Program lost, sorry.
: 134 0961 1 !     BAS$K_ILLIO_CHA : UNSIGNED (8);          ! Illegal I/O channel.
: 135 0962 1 !
```

```
137 0963 1 GLOBAL ROUTINE BASFSP (
138 0964 1     RESULT,
139 0965 1     CHAN
140 0966 1 ) : NOVALUE =
141 0967 1
142 0968 1 ++
143 0969 1 FUNCTIONAL DESCRIPTION:
144 0970 1
145 0971 1     Return information about the file which is open on this
146 0972 1     channel. This is especially useful if the channel was
147 0973 1     opened with organization UNDEFINED. The result is returned
148 0974 1     as a 32-character string, laid out as follows:
149 0975 1
150 0976 1     BYTE    MEANING
151 0977 1
152 0978 1     1       The record format and file organization, see below
153 0979 1
154 0980 1     2       The record attributes, see below.
155 0981 1
156 0982 1     3-4     FAB$W__MRS.
157 0983 1
158 0984 1     5-8     FAB$L_ALQ.
159 0985 1
160 0986 1     9-10    FAB$W_BLS if this is a mag tape, otherwise FAB$B_BKS.
161 0987 1
162 0988 1     11      Number of keys in the file (indexed only).
163 0989 1
164 0990 1     12      Zero.
165 0991 1
166 0992 1     13-16   FAB$L_MRN.
167 0993 1
168 0994 1     17-20   RAB$L_BKT.
169 0995 1
170 0996 1     21-32   Zero.
171 0997 1
172 0998 1
173 0999 1     The first byte is organization + record format, encoded as:
174 1000 1     sequential = 0, relative = 16, indexed = 32; fixed = 1,
175 1001 1     variable = 2, VFC = 3. The second byte is the record
176 1002 1     attribute, encoded as: FORTRAN = 1, CR = 2, PRN = 4,
177 1003 1     none = 0. Virtual files have both bytes 0. Channel 0 is
178 1004 1     sequential, variable, RAT=none, all other bytes zero.
179 1005 1
180 1006 1 FORMAL PARAMETERS:
181 1007 1
182 1008 1     RESULT.wb.dx  Where to put the 32-byte result.
183 1009 1     CHAN.rl.v     The channel to do this to.
184 1010 1
185 1011 1 IMPLICIT INPUTS:
186 1012 1
187 1013 1     NONE
188 1014 1
189 1015 1 IMPLICIT OUTPUTS:
190 1016 1
191 1017 1     NONE
192 1018 1
193 1019 1 ROUTINE VALUE:
```

```
194 1020 1 1 COMPLETION CODES:
195 1021 1 1
196 1022 1 1 NONE
197 1023 1 1
198 1024 1 1 SIDE EFFECTS:
199 1025 1 1
200 1026 1 1 Signals if an error is encountered.
201 1027 1 1 BAS$$CB_PUSH will signal if the channel number is invalid.
202 1028 1 1 We signal BAS$K_IO_CHANOT if the channel is not open.
203 1029 1 1
204 1030 1 1 --
205 1031 1 1
206 1032 2 1 BEGIN
207 1033 2 1
208 1034 2 1 BUILTIN
209 1035 2 1 FP;
210 1036 2 1
211 1037 2 1 GLOBAL REGISTER
212 1038 2 1 CCB = K_CCB_REG : REF BLOCK [, BYTE];
213 1039 2 1
214 1040 2 1 LOCAL
215 1041 2 1 FMP : REF BLOCK [, BYTE],
216 1042 2 1 FSP_STRING : VECTOR [32, BYTE];
217 1043 2 1
218 1044 2 1 FMP = .FP;
219 1045 2 1 +
220 1046 2 1 Set all positions to zero, so any not stored in below will come
221 1047 2 1 out zero.
222 1048 2 1 -
223 1049 2 1
224 1050 2 1 INCR COUNTER FROM 0 TO 31 DO
225 1051 2 1 FSP_STRING [.COUNTER] = 0;
226 1052 2 1
227 1053 2 1 +
228 1054 2 1 If this is the user's terminal, return the same information as
229 1055 2 1 the PDP-11.
230 1056 2 1 -
231 1057 2 1
232 1058 2 1 IF (.CHAN EQL 0)
233 1059 2 1 THEN
234 1060 2 1 BEGIN
235 1061 2 1 FSP_STRING [0] = BAS$K_ORG_SEQ + BAS$K_RFM_VAR;
236 1062 2 1 FSP_STRING [1] = BAS$K_RAT_NONE;
237 1063 2 1 +
238 1064 2 1 Copy our string back to the caller.
239 1065 2 1 -
240 1066 2 1 STR$COPY_R (.RESULT, %REF (32), FSP_STRING);
241 1067 2 1 END
242 1068 2 1 ELSE
243 1069 2 1 BEGIN
244 1070 2 1 +
245 1071 2 1 Get the CCB for the channel.
246 1072 2 1 -
247 1073 2 1 BAS$$CB_PUSH (.CHAN, LUB$K_LUN_MIN);
248 1074 2 1 CCB [ISB$A_USER_FP] = .FMP[SF$L_SAVE_FP];
249 1075 2 1 +
250 1076 2 1 Fail if the channel is not open.
```

```
251 1077 3 !-
252 1078
253 1079 IF (.CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
254 1080
255 1081 !+
256 1082 Extract the organization
257 1083 !-
258 1084 FSP_STRING [0] =
259 1085 BEGIN
260 1086
261 1087 IF (.CCB [LUB$B_ORGAN] EQL LUB$K_ORG_VIRTU)
262 1088 THEN
263 1089 0
264 1090 ELSE
265 1091 BEGIN
266 1092
267 1093 CASE .CCB [LUB$B_ORGAN] FROM LUB$K_ORG_SEQUE TO LUB$K_ORG_TERMI OF
268 1094 SET
269 1095
270 1096 [LUB$K_ORG_SEQUE] :
271 1097 BAS$K_ORG_SEQ;
272 1098
273 1099 [LUB$K_ORG_RELAT] :
274 1100 BAS$K_ORG_REL;
275 1101
276 1102 [LUB$K_ORG_INDEX] :
277 1103 BAS$K_ORG_IND;
278 1104
279 1105 [LUB$K_ORG_TERMI] :
280 1106 BAS$K_ORG_SEQ;
281 1107
282 1108 [OUTRANGE] :
283 1109 BEGIN
284 1110 BAS$$STOP_IO (BAS$K_PROLOSSOR);
285 1111 0
286 1112 END;
287 1113 TES
288 1114
289 1115 END
290 1116 +
291 1117 BEGIN
292 1118
293 1119 IF (.CCB [LUB$V_FIXED])
294 1120 THEN
295 1121 BAS$K_RFM_FIX
296 1122 ELSE
297 1123
298 1124 IF (.CCB [RAB$L_RHB] EQLA 0) THEN BAS$K_RFM_VAR ELSE BAS$K_RFM_VFC
299 1125
300 1126 END
301 1127
302 1128 END;
303 1129 FSP_STRING [1] =
304 1130 BEGIN
305 1131
306 1132 IF (.CCB [LUB$B_ORGAN] EQL LUB$K_ORG_VIRTU)
307 1133 THEN
```

```

308      1134 4      BASS$K_RAT_NONE
309      1135 4      ELSE
310      1136 4
311      1137 5      IF ((.CCB [LUB$B_RAT] AND FAB$M_FTN) NEQ 0)
312      1138 4      THEN
313      1139 4      BASS$K_RAT_FTN
314      1140 4      ELSE
315      1141 4
316      1142 5      IF ((.CCB [LUB$B_RAT] AND FAB$M_CR) NEQ 0)
317      1143 4      THEN
318      1144 4      BASS$K_RAT_CR
319      1145 4      ELSE
320      1146 4
321      1147 4      IF ((.CCB [LUB$B_RAT] AND FAB$M_PRN) NEQ 0) THEN BASS$K_RAT_PRN ELSE BASS$K_RAT_NONE
322      1148 4
323      1149 3      END;
324      1150 3      +
325      1151 3      Store the maximum record size, FAB$W_MRS. (somewhat fudged)
326      1152 3      -
327      1153 3      BLOCK [FSP_STRING [2], 0, 0, 16, 0] = .CCB [LUB$W_RBUF_SIZE];
328      1154 3      +
329      1155 3      Store the allocation quantity.
330      1156 3      -
331      1157 3      BLOCK [FSP_STRING [4], 0, 0, 32, 0] = .CCB [LUB$L_ALQ];
332      1158 3      +
333      1159 3      Store the block size or bucket size. This is currently not
334      1160 3      supported, since we do not support non-file mag tape.
335      1161 3      Always return zero.
336      1162 3      -
337      1163 3      BLOCK [FSP_STRING [8], 0, 0, 16, 0] = 0;
338      1164 3      +
339      1165 3      Store the number of keys in an indexed file. We don't support
340      1166 3      this, either. Always return zero.
341      1167 3      -
342      1168 3      FSP_STRING [10] = 0;
343      1169 3      +
344      1170 3      Store the number of the highest record that can be written to
345      1171 3      the file. Since we don't retain this information either, just
346      1172 3      return the highest record number.
347      1173 3      -
348      1174 3      BLOCK [FSP_STRING [12], 0, 0, 32, 0] = .CCB [LUB$L_REC_MAX];
349      1175 3      +
350      1176 3      Store the current block/record number.
351      1177 3      -
352      1178 3      BLOCK [FSP_STRING [16], 0, 0, 32, 0] = .CCB [RAB$L_BKT];
353      1179 3      +
354      1180 3      Copy our string back to the caller.
355      1181 3      -
356      1182 3      STR$COPY_R (.RESULT, %REF (32), FSP_STRING);
357      1183 3      +
358      1184 3      We are done with register CCB.
359      1185 3      -
360      1186 3      BASS$CB_POP ();
361      1187 3      END;
362      1188 2
363      1189 2      RETURN;
364      1190 1      END;
                                     ! end of BASS$FSP
```

				.TITLE	BASSFSP			
				.IDENT	\1-006\			
				.EXTRN	BASS\$CB_PUSH, BASS\$CB_POP			
				.EXTRN	BASS\$STOP_IO, BASS\$STOP			
				.EXTRN	STR\$COPY_R, BASSK_IO_CHANOT			
				.EXTRN	BASSK_MAXMEMEXC			
				.EXTRN	BASSK_PROLOSSOR			
				.EXTRN	BASSK_ILLIO_CHA			
				.PSECT	_BASS\$CODE, NOWRT, SHR, PIC, 2			
				.ENTRY	BASSFSP, Save R2, R3, R4, R5, R11	0963		
		55	00000000G	00	9E 00002	MOVAB STR\$COPY_R, R5		
		54	00000000G	00	9E 00009	MOVAB BASS\$STOP_IO, R4		
		5E		24	C2 00010	SUBL2 #36, SP		
		53		5D	D0 00013	MOVL FP, FMP	1044	
				50	D4 00016	CLRL COUNTER	1050	
		04	AE40	94	00018	CLRB FSP_STRING[COUNTER]	1051	
F8		50		1F	F3 0001C	AOBLEQ #31, COUNTER, 1\$		
			08	AC	D5 00020	TSTL CHAN	1058	
			15	12	00023	BNEQ 2\$		
	04	AE		02	B0 00025	MOVW #2, FSP_STRING	1061	
			04	AE	9F 00029	PUSHAB FSP_STRING	1066	
	04	AE		20	D0 0002C	MOVL #32, 4(SP)		
			04	AE	9F 00030	PUSHAB 4(SP)		
			04	AC	DD 00033	PUSHL RESULT		
		65		03	FB 00036	CALLS #3, STR\$COPY_R		
				04	00039	RET	1058	
			50	D4	0003A	CLRL R0	1073	
		52		08	AC	D0 0003C	MOVL CHAN, R2	
			00000000G	00	16 00040	JSB BASS\$CB_PUSH		
FF4C		CB		0C	A3	D0 00046	MOVL 12(FMP), -180(CCB)	1074
		07		FC	AB	E8 0004C	BLBS -4(CCB), 3\$	1079
		7E		00G	8F	9A 00050	MOVZBL #BASSK_IO_CHANOT, -(SP)	
		64		01	FB	00054	CALLS #1, BASS\$STOP_IO	
				52	D4	00057	CLRL R2	1087
		05		C4	AB	91 00059	CMPB -60(CCB), #5	
				06	12	0005D	BNEQ 4\$	
				52	D6	0005F	INCL R2	
				50	D4	00061	CLRL R0	
				3F	11	00063	BRB 13\$	
		01		C4	AB	8F 00065	CASEB -60(CCB), #1, #3	1093
001B	03	0016	0011	001B	0006A	5\$: .WORD 8\$-5\$,-		
						6\$-5\$,-		
						7\$-5\$,-		
						8\$-5\$		
		7E		00G	8F	9A 00072	MOVZBL #BASSK_PROLOSSOR, -(SP)	1110
		64		01	FB	00076	CALLS #1, BASS\$STOP_IO	
				0A	11	00079	BRB 8\$	1093
		51		10	D0	0007B	MOVL #16, R1	
				07	11	0007E	BRB 9\$	
		51		20	D0	00080	MOVL #32, R1	
				02	11	00083	BRB 9\$	
				51	D4	00085	CLRL R1	

05	FD	AB		02	E1	00087	9\$:	BBC	#2, -3(CCB), 10\$	1119
		50		01	D0	0008C		MOVL	#1, R0	
			2C	0D	11	0008F		BRB	12\$	
				AB	D5	00091	10\$:	TSTL	44(CCB)	1124
		50		05	12	00094		BNEQ	11\$	
				02	D0	00096		MOVL	#2, R0	
		50		03	11	00099		BRB	12\$	
		50		03	D0	0009B	11\$:	MOVL	#3, R0	1117
		51		50	C0	0009E	12\$:	ADDL2	R0, R1	1116
		50		51	D0	000A1		MOVL	R1, R0	
	04	AE		50	90	000A4	13\$:	MOVB	R0, FSP_STRING	1085
		1D		52	E8	000A8		BLBS	R2, 16\$	1132
		05	F6	AB	E9	000AB		BLBC	-10(CCB), 14\$	1137
		50		01	D0	000AF		MOVL	#1, R0	
				16	11	000B2		BRB	17\$	
05	F6	AB		01	E1	000B4	14\$:	BBC	#1, -10(CCB), 15\$	1142
		50		02	D0	000B9		MOVL	#2, R0	
				0C	11	000BC		BRB	17\$	
05	F6	AB		02	E1	000BE	15\$:	BBC	#2, -10(CCB), 16\$	1147
		50		04	D0	000C3		MOVL	#4, R0	
				02	11	000C6		BRB	17\$	
				50	D4	000C8	16\$:	CLRL	R0	
	05	AE		50	90	000CA	17\$:	MOVB	R0, FSP_STRING+1	1130
	06	AE	D2	AB	B0	000CE		MOVW	-46(CCB), FSP_STRING+2	1153
	08	AE	DC	AB	D0	000D3		MOVL	-36(CCB), FSP_STRING+4	1157
			0C	AE	B4	000D8		CLRW	FSP_STRING+8	1163
			0E	AE	94	000DB		CLRB	FSP_STRING+10	1168
	10	AE	E4	AB	D0	000DE		MOVL	-28(CCB), FSP_STRING+12	1174
	14	AE	38	AB	D0	000E3		MOVL	56(CCB), FSP_STRING+16	1178
			04	AE	9F	000E8		PUSHAB	FSP_STRING	1182
	04	AE		20	D0	000EB		MOVL	#32, 4(SP)	
			04	AE	9F	000EF		PUSHAB	4(SP)	
			04	AC	DD	000F2		PUSHL	RESULT	
		65		03	FB	000F5		CALLS	#3, STR\$COPY_R	
			00000000G	00	16	000F8		JSB	BAS\$CB_POP	1186
				04	000FE			RET		1190

; Routine Size: 255 bytes, Routine Base: _BAS\$CODE + 0000

: 365 1191 1
: 366 1192 1 END
: 367 1193 1
: 368 1194 0 ELUDOM

! end of module BAS\$FSP

PSECT SUMMARY

Name	Bytes	Attributes
_BAS\$CODE	255	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
;\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	6	0	581	00:01.2

COMMAND QUALIFIERS

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

; Size: 255 code + 0 data bytes
; Run Time: 00:11.8
; Elapsed Time: 00:24.4
; Lines/CPU Min: 6076
; Lexemes/CPU-Min: 37633
; Memory Used: 159 pages
; Compilation Complete

0023 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

