


```

TTTTTTTTT1  BBBB8888  KK      KK  BBBB8888  AAAAAA  SSSSSSSS
TTTTTTTTTT  BBBB8888  KK      KK  BBBB8888  AAAAAA  SSSSSSSS
  TT        BB      BB  KK      KK  BB      BB  AA      AA  SS
  T1       BB      BB  KK      KK  BB      BB  AA      AA  SS
  TT       BB      BB  KK      KK  BB      BB  AA      AA  SS
  TT       BB      BB  KK      KK  BB      BB  AA      AA  SS
  TT       BBBB8888  KKKKKK  BBBB8888  AA      AA  SSSSSS
  TT       BBBB8888  KKKKKK  BBBB8888  AA      AA  SSSSSS
  TT       BB      BB  KK      KK  BB      BB  AAAAAAAAAA  SS
  TT       BB      BB  KK      KK  BB      BB  AAAAAAAAAA  SS
  TT       BB      BB  KK      KK  BB      BB  AA      AA  SS
  TT       BB      BB  KK      KK  BB      BB  AA      AA  SS
  TT       BBBB8888  KK      KK  BBBB8888  AA      AA  SSSSSSSS
  TT       BBBB8888  KK      KK  BBBB8888  AA      AA  SSSSSSSS

```

```

....
....
....
....

```

```

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 0001 0 MODULE TBKBAS ( IDENT = 'V04-000' ) =
2 0002 1 BEGIN
3 0003 1
4 0004 1 *****
5 0005 1 *
6 0006 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
7 0007 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
8 0008 1 * ALL RIGHTS RESERVED. *
9 0009 1 *
10 0010 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
11 0011 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
12 0012 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
13 0013 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
14 0014 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
15 0015 1 * TRANSFERRED. *
16 0016 1 *
17 0017 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
18 0018 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
19 0019 1 * CORPORATION. *
20 0020 1 *
21 0021 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
22 0022 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
23 0023 1 *
24 0024 1 *
25 0025 1 *****
26 0026 1
27 0027 1 FACILITY:      DEBUG (DBG)
28 0028 1
29 0029 1 ++
30 0030 1 FUNCTIONAL DESCRIPTION:
31 0031 1     Does TRACEback when a program causes an exception of
32 0032 1     sufficiently high severity level and the exception
33 0033 1     is not fielded by anyone else (namely by itself of
34 0034 1     by DEBUG).
35 0035 1
36 0036 1 Version:      1.01
37 0037 1
38 0038 1 History:
39 0039 1     Author:
40 0040 1         Carol Peters, 11 January 1978 : Version 01
41 0041 1
42 0042 1     Modified by:
43 0043 1         Dale Roedger, 15 June 1978: Version 17
44 0044 1         Victoria Holt, 21 December 1982
45 0045 1
46 0046 1 Revision history:
47 0047 1
48 0048 1     02      17-jan-78      KGP      -Changed call parameters, and
49 0049 1     03      18-jan-78      KGP      -added code to do the traceback
50 0050 1     04      19-JAN-78      KGP      -in non-symbolic form.
51 0051 1     05      26-jan-78      KGP      -Exception name is printed using
52 0052 1     06      26-jan-78      KGP      -system message formatter
53 0053 1     07      26-jan-78      KGP      -Output is now properly formatted.
54 0054 1     08      26-jan-78      KGP      -Now TRACEBACK doe: not $EXIT
55 0055 1     09      26-jan-78      KGP      -under any circumstances. This is
56 0056 1     10      26-jan-78      KGP      -so that we can $$$ RESIGNAL
57 0057 1     11      26-jan-78      KGP      -the original exception so that someone

```

58	0058	1							else (the 'catch-all' handler) has to ultimately decide what to do about it.
59	0059	1							
60	0060	1							
61	0061	1	06	22-feb-78	KGP				-Output is to SYSS..., not DBG\$... We put out a separate message for "traceback follows..."
62	0062	1							
63	0063	1							
64	0064	1	07	24-feb-78	KGP				-We always return the exception name now so that the EXIT/CONTINUE decision is made elsewhere.
65	0065	1							-Some diddling to make the severity level of the TRACEBACK message the same as the level of the exception.
66	0066	1							
67	0067	1							
68	0068	1							
69	0069	1							
70	0070	1	08	24-feb-78	KGP				-Formal parameter is now signal array address rather than exception name so we can PUT MESSAGE rather than doing all that ourselves.
71	0071	1							
72	0072	1							
73	0073	1							
74	0074	1	09	28-feb-78	KGP				-Changed all error returns in TRACE to be EXITS. Now, if TRACE returns at all to the assembly-language TRACE code, all must have gone well.
75	0075	1							-Added code to ensure that we don't try to 'trace' an overwritten stack.
76	0076	1							-We don't re-map the symbol table on successive TRACEbacks.
77	0077	1							-We now set up our output based on how PUTMSG did it since it looks after [not]creating the SYSS\$OUTPUT/SYSS\$ERROR files, etc.
78	0078	1							
79	0079	1							
80	0080	1							
81	0081	1							
82	0082	1							
83	0083	1							
84	0084	1							
85	0085	1							
86	0086	1	10	01-mar-78	KGP				-Took out all initialization of FABs/RABs, and we now don't do OPEN or CONNECT. We let PUTMSG setup everything.
87	0087	1							
88	0088	1							
89	0089	1	11	2-mar-78	KGP				-Beginning exception type for FTN PC correlation is now decided in BAS (is no longer local to DPC)
90	0090	1							
91	0091	1							
92	0092	1	12	7-mar-78	KGP				-Exception type is always forced to TRAP_EXC after the first stack frame symbolization.
93	0093	1							-We now call TBK\$PUTMSG to put out error messages.
94	0094	1							
95	0095	1							
96	0096	1							
97	0097	1	13	8-mar-78	KGP				-FIND DST now returns an indication of whether the traceback will be symbolic or not.
98	0098	1							
99	0099	1							
100	0100	1							
101	0101	1							
102	0102	1	14	13-mar-78	KGP				-All of TRACE is now separate from DEBUG. It has its own REQUIRE files.
103	0103	1							-Took out fake messages now that TRACMSG is installed into the system.
104	0104	1							-We now subtract 2 from the given signal arg count field so that PUTMSG doesn't try to print messages that don't exist.
105	0105	1							
106	0106	1							
107	0107	1	15	27-mar-78	KGP				-added IS EXCEPTION so that we can now start off TBK\$GL_EXC_TYPE correctly.
108	0108	1							-TBK\$PUTMSG changed to TBK\$FAKE_MSG, and DEBUG's DBG\$PUTMSG added, changed slightly, and renamed TBK\$PUT_MSG.
109	0109	1							
110	0110	1							
111	0111	1							
112	0112	1	16	26-APR-78	DAR				Modified require and library directives for native mode.
113	0113	1							
114	0114	1	17	15-JUN-78	DAR				Changed all DBG\$ symbols to TBK\$.

048

: R

115	0115	1	18	30-Oct-79	JBD	Removed the 'Unknown DST record' message
116	0116	1	19	3-DEC-79	JBD	Added stmt number support.
117	0117	1	1.01	30-Jan-80	JBD	Made module and routine names longer than 15
118	0118	1				characters appear on different lines
119	0119	1	3.01	03-Mar-82	RT	Passed in file channel number of image file
120	0120	1				so it doesn't have to be opened again to
121	0121	1				read the DST.
122	0122	1		21-Dec-82	VJH	Made corrections so that the original status is
123	0123	1				returned when the user turns off all the
124	0124	1				default message flags (SYSSPUTMSG does not
125	0125	1				define SYSSOUTPUT and SYSSERROR in that case).
126	0126	1		15-Aug-83	PS	Did general clean up to use updated files
127	0127	1				from DEBUG.
128	0128	1	--			

```

130 0129 1 ! TABLE OF CONTENTS
131 0130 1
132 0131 1 FORWARD ROUTINE
133 0132 1 IS_EXCEPTION, ! See if a given exception name
134 0133 1 ! is TRAP_EXC or FAULT_EXC type.
135 0134 1 out_traceback : NOVALUE, ! display line of traceback information
136 0135 1 TBK$DO_TRACEB; ! traceback user program after error
137 0136 1
138 0137 1
139 0138 1 ! REQUIRE FILES:
140 0139 1
141 0140 1 REQUIRE 'SRC$:TBKPROLOG.REQ';
142 0412 1
143 0413 1 EXTERNAL
144 0414 1 TBK$GL_EXC_TYPE, ! Initial FAULT/TRAP type for PC correlation.
145 0415 1 TBK$MODULE_CS : CS_POINTER,
146 0416 1 TBK$ROUTINE_CS : CS_POINTER,
147 0417 1 TBK$GL_STMT,
148 0418 1 TBK$GL_LINE,
149 0419 1 TBK$REC_PC,
150 0420 1 TBK$MODULE_DST : REF DST$RECORD,
151 0421 1
152 0422 1 tbk$gl_outprab: $RAB_DECL; ! RAB FOR 'OUTPUT'
153 0423 1
154 0424 1 EXTERNAL ROUTINE
155 0425 1 tbk$fake_msg : NOVALUE, ! write out fake traceback messages.
156 0426 1 tbk$put_msg, ! write out system-generated messages.
157 0427 1 tbk$fao_put, ! Format into output buffer.
158 0428 1 tbk$fao_out : NOVALUE,
159 0429 1 tbk$out_put : NOVALUE, ! Write out the output buffer.
160 0430 1 TBK$IO_SETUP, ! Set up for PUTMSG-type I/O.
161 0431 1 TBK$SYMBOLIZE : NOVALUE,
162 0432 1 tbk$find_dst; ! finds and maps in the DST for the image
163 0433 1
164 0434 1
165 0435 1 ! Diagnostic output control
166 0436 1
167 0437 1 LITERAL
168 0438 1 TBK_BAS1 = 0, ! print out input parameters
169 0439 1 TBK_BAS2 = 0, ! List off the entire DST.
170 0440 1 TBK_BAS3 = 0, ! Output during stack unwinding
171 0441 1 TBK_BAS4 = 0; ! Error messages.
172 0442 1
173 0443 1 %IF TBK_BAS2
174 0444 1 %THEN
175 0445 1 FORWARD ROUTINE
176 0446 1 pr_cs : novalue,
177 0447 1 LIST_DST;
178 0448 1
179 0449 1 EXTERNAL ROUTINE
180 0450 1 tbk$get_nxt_dst; ! Make successive DSTs available.
181 0451 1 %FI
182 0452 1
183 0453 1 MACRO
184 0454 1 CFP$L_HANDLER = 0, 0, 32, 0%,
185 0455 1 CFP$L_OLD_FP = 12, 0, 32, 0%,
186 0456 1 CFP$L_RETURN_PC = 16, 0, 32, 0%;

```

```
188 0457 1 GLOBAL ROUTINE tbk$do_traceb ( imgfilchan,  
189 0458 1 file_name,  
190 0459 1 img_header_blk,  
191 0460 1 symtab_sec_bnds,  
192 0461 1 signal_array,  
193 0462 1 first_fp,  
194 0463 1 current_fp,  
195 0464 1 current_pc) =  
196 0465 1  
197 0466 1 |++  
198 0467 1 | Functional description:  
199 0468 1 | Call PUTMSG to output the reason why TRACE was called.  
200 0469 1 | Then maps the DST into PO space and used it so  
201 0470 1 | give a symbolic stack dump of where the program  
202 0471 1 | was when it 'faulted'.  
203 0472 1 | We then return leaving ourselves and the DST mapped  
204 0473 1 | in so that on subsequent invocations of TRACE we can  
205 0474 1 | avoid the re-mapping overhead.  
206 0475 1 |  
207 0476 1 | All output is to SYSS$ERROR and SYSS$OUTPUT.  
208 0477 1 |  
209 0478 1 | Formal parameters:  
210 0479 1 | imgfilchan - the channel number that the image file is  
211 0480 1 | open on.  
212 0481 1 | file_name - a counted string to the file specification of  
213 0482 1 | the image file.  
214 0483 1 | img_header_blk - address of a byte block containing the image  
215 0484 1 | header data needed to find DST and GST data for  
216 0485 1 | the image.  
217 0486 1 | symtab_sec_bnds -address of a 2 longword vector (in the bootstrap)  
218 0487 1 | where the symbol table bounds are stored so that  
219 0488 1 | we don't need to map in the DST on successive TRACEbacks.  
220 0489 1 | signal_array -address of the 'signal array' generated for the  
221 0490 1 | exception that causes TRACEback.  
222 0491 1 | first_fp -FP of first frame NOT to be traced.  
223 0492 1 | (i.e. last frame we look at)  
224 0493 1 | current_fp - current value of user FP  
225 0494 1 | current_pc - current value of user PC  
226 0495 1 |  
227 0496 1 | Implicit inputs:  
228 0497 1 | PUTMSG creates a process logical name (SYSS$PUTMSG),  
229 0498 1 | the translation of which returns an encoding  
230 0499 1 | of the ISI numbers for SYSS$ERROR and SYSS$OUTPUT.  
231 0500 1 | We stuff these ISIs into our own RABs so that  
232 0501 1 | we don't worry about the SYSS$ERROR/SYSS$OUTPUT distinction  
233 0502 1 | and so that we avoid opening the channels on successive  
234 0503 1 | invocations.  
235 0504 1 |  
236 0505 1 | Output parameters:  
237 0506 1 | none  
238 0507 1 |  
239 0508 1 | Implicit outputs:  
240 0509 1 | The 2-longword vector in the bootstrap which points to the  
241 0510 1 | beginning and ending of the symbol table gets filled in  
242 0511 1 | with the mapped addresses of where we map the symbol table.  
243 0512 1 |  
244 0513 1 | Routine value:
```

```

245 0514 1 | Either an EXIT is done, or this routine returns
246 0515 1 | the exception name which caused TRACEback in the first place.
247 0516 1 |
248 0517 1 | Side effects:
249 0518 1 | The DST is mapped into P0 space. A number of lines are output to
250 0519 1 | logical device SYS$OUTPUT. If SYS$ERROR is different from
251 0520 1 | SYS$OUTPUT, the same output goes to SYS$ERROR.
252 0521 1 |
253 0522 1 | --
254 0523 1 |
255 0524 2 |     ;IN
256 0525 2 |     MAP
257 0526 2 |         syntab_sec_bnds : ref vector[,long],
258 0527 2 |         signal_array : ref vector[,long],
259 0528 2 |         FILE_NAME : REF VECTOR[,BYTE];
260 0529 2 |     MAP
261 0530 2 |         CURRENT_FP : REF BLOCK[,BYTE],
262 0531 2 |         CURRENT_PC : REF BLOCK[,BYTE];
263 0532 2 |     LOCAL
264 0533 2 |         symbolic,                ! Flag. 1 => symbolic traceback,
265 0534 2 |                                     !           0 => non-symbolic.
266 0535 2 |         exceptn_name,
267 0536 2 |         blank : CS_POINTER,
268 0537 2 |         status;
269 0538 2 |
270 0539 2 | ! Pick the exception name out of the signal array
271 0540 2 | ! so that we then use its severity to print the
272 0541 2 | ! standard TRACEback message. This is done so that
273 0542 2 | ! the levels of the first message and the "trace follows..."
274 0543 2 | ! message is the same - for consistency and so that
275 0544 2 | ! the two messages go to the same channel(s).
276 0545 2 | ! The message reflects the [non-]symbolic indication passed
277 0546 2 | ! back by FIND_DST.
278 0547 2 |
279 0548 2 | exceptn_name = .signal_array[1];
280 0549 2 |
281 0550 2 |
282 0551 2 | !++
283 0552 2 | ! Report on the cause of the exception, and
284 0553 2 | ! let PUTMSG open our output channel(s) for us.
285 0554 2 | ! If this fails, we must punt.
286 0555 2 | !--
287 0556 2 |
288 0557 2 | status = tbk$put_msg(.signal_array);
289 0558 2 |
290 0559 2 | IF NOT .status
291 0560 2 | THEN
292 0561 2 |     BEGIN
293 0562 2 |         $EXIT( code = .status);
294 0563 2 |     END;
295 0564 2 |
296 0565 2 | ! Set up to do I/O by relying on the fact that
297 0566 2 | ! PUTMSG has already sorted out the problems
298 0567 2 | ! of where SYS$OUTPUT and SYS$ERROR actually go to.
299 0568 2 |
300 0569 2 | status = tbk$io_setup();
301 0570 2 |

```



```
359      0628      BEGIN
360      0629
361      0630      ! Print the standard TRACE heading.
362      0631
363      0632      tbk$fao_put (uplit (%ascii
364      0633      'module name      routine name      line      rel PC      abs PC!/'));
365      0634      tbk$out_put();
366      0635      END;
367      0636
368      0637      ! For FORTRAN PC correlation, we need to set
369      0638      ! TBK$GL_EXC_TYPE to either FAULT_EXC or TRAP_EXC
370      0639      ! exception type so that DPC can come up with the best
371      0640      ! %LINE symbolization for the PC. We assume the latter
372      0641      ! and let IS_EXCEPTION cover the exceptions.
373      0642
374      0643      TBK$GL_EXC_TYPE = TRAP_EXC;
375      0644      IF( IS_EXCEPTION(.EXCEPTN_NAME) )
376      0645      THEN
377      0646          TBK$GL_EXC_TYPE = FAULT_EXC;
378      0647
379      0648      ! Loop printing out each active frame until we have
380      0649      ! 'unwound' to the frame set up by the DEBUG bootstrap
381      0650      ! when the user image was called in the first place.
382      0651
383      0652      WHILE (.first_fp GTRA .current_fp)
384      0653      DO      BEGIN
385      0654
386      0655      %IF TBK_BAS3
387      0656      %THEN
388      0657          $fao_tt_out('FP = !XL, PC = !XL',.CURRENT_FP,.CURRENT_PC);
389      0658      %FI
390      0659          TBK$SYMBOLIZE(.CURRENT_PC);
391      0660
392      0661          ! display current module/routine/line/PC
393      0662
394      0663          out_traceback (.tbk$module_cs,
395      0664                        .tbk$routine_cs,
396      0665                        .tbk$gl_line,
397      0666                        .tbk$gl_stmt,
398      0667                        .tbk$rel_pc,
399      0668                        .current_pc
400      0669                      );
401      0670
402      0671
403      0672      ! For FORTRAN pc-to-line symbolizations, it never
404      0673      ! makes sense for any frame other than the first
405      0674      ! to be of 'match' type FAULT_EXC.
406      0675
407      0676      TBK$GL_EXC_TYPE = TRAP_EXC;
408      0677
409      0678      ! Set FP and PC to that of previous frame, making
410      0679      ! sure not to get fooled by an overwritten stack.
411      0680      ! i.e. insist that the previous frame is 'above'
412      0681      ! the supposed current one.
413      0682
414      0683      IF( NOT .current_fp LSSA .current_fp[ CFP$L_OLD_FP ] )
415      0684      THEN
```

```

: 416      0685      4
: 417      0686      4
: 418      0687      4
: 419      0688      4
: 420      0689      4
: 421      0690      4
: 422      0691      4
: 423      0692      4
: 424      0693      4
: 425      0694      4
: 426      0695      4
: 427      0696      4
: 428      0697      4
: 429      0698      4
: 430      0699      4
: 431      0700      1

```

```

BEGIN
tbk$fake_msg(tbk$_badstack,0);
EXITLOOP;
END;

current_pc = .current_fp[ cfp$l_return_pc ];
CURRENT_FP = .CURRENT_FP[ CFP$l_OLD_FP ];
END;

! Only OK return point.
! We return the exception name we were passed
! so that the TRACE startup routine can
! decide what to do about it.

RETURN(.EXCEPTN_NAME);
END;

```

																.TITLE	TBKBAS								
																.IDENT	\V04-000\								
																.PSECT	TBK\$PLIT,NOWRT,	SHR,	PIC,0						
20	20	20	65	6D	61	6E	20	65	6C	75	64	6F	6D	4E	00000	P.AAA:	.ASCII	\Nmodule name	routine name						
20	65	6D	61	6E	20	65	6E	69	74	75	6F	72	20	20	0000F										
20	65	6E	69	6C	20	20	20	20	20	20	20	20	20	20	0001E										
20	20	20	43	50	20	6C	65	72	20	20	20	20	20	20	0002B		.ASCII	\	line	rel PC	abs PC!/-				
20	20	20	43	50	20	6C	65	72	20	20	20	20	20	20	00037			\<0>							
20	20	20	43	50	00	2F	21	43	50	20	73	62	61	20	00046										
																.EXTRN	TBK\$GL_EXC_TYPE								
																.EXTRN	TBK\$MODULE_CS, TBK\$ROUTINE_CS								
																.EXTRN	TBK\$GL_STMT, TBK\$GL_LINE								
																.EXTRN	TBK\$REL_PC, TBK\$MODULE_DST								
																.EXTRN	TBK\$GL_OUTPRAB, TBK\$FARE_MSG								
																.EXTRN	TBK\$PUT_MSG, TBK\$FAO_PUT								
																.EXTRN	TBK\$FAO_OUT, TBK\$OUT_PUT								
																.EXTRN	TBK\$IO_SETUP, TBK\$SYMBOLIZE								
																.EXTRN	TBK\$FIND_DST, SYS\$EXIT								
																.PSECT	TBK\$CODE,NOWRT,	SHR,	PIC,0						
																.ENTRY	TBK\$DO_TRACEB, Save R2,R3,R4,R5								
55	00000000G	00	9E	00002												MOVAB	TBK\$GL_EXC_TYPE, R5								
54	00000000G	00	9E	00009												MOVAB	TBK\$FARE_MSG, R4								
50	14	AC	DO	00010												MOVL	SIGNAL_ARRAY, R0								
53	04	A0	DO	00014												MOVL	4(R0), -EXCEPTN_NAME								
																PUSHL	R0								
00000000G	00	01	FB	0001A												CALLS	#1, TBK\$PUT_MSG								
52		50	DO	00021												MOVL	R0, STATUS								
09		52	E8	00024												BLBS	STATUS, 1\$								
																PUSHL	STATUS								
00000000G	00	01	FB	00029												CALLS	#1, SYS\$EXIT								
00000000G	00	00	FB	00030												CALLS	#0, TBK\$IO_SETUP								
52		50	DO	00037												MOVL	R0, STATUS								
00000629	8F	52	D1	0003A												CMPL	STATUS, #1577								
																BEQL	7\$								

	7E	0C	AC	7D	00043	MOVQ	IMG HEADER_BLK, -(SP)	: 0600
	7E	04	AC	7D	00047	MOVQ	IMGFILCHAN, -(SP)	: 0599
00000000G	00		04	FB	0004B	CALLS	#4, TBK\$FIND_DST	
	09		50	E9	00052	BLBC	SYMBOLIC, 2\$: 0606
	50	00098198	8F	DO	00055	MOVL	#623000, SYMBOLIC	
			07	11	0005C	BRB	3\$	
51	53		8F	DO	0005E	2\$: 3\$: MOVL	#623008, SYMBOLIC	
	03		00	EF	00065	EXTZV	#0, #3, EXCEPTN_NAME, R1	: 0607
	50		51	CO	0006A	ADDL2	R1, SYMBOLIC	
			7E	D4	0006D	CLRL	-(SP)	: 0611
			50	DD	0006F	PUSHL	SYMBOLIC	
	64		02	FB	00071	CALLS	#2, TBK\$FAKE_MSG	
1C	AC	18	AC	D1	00074	CML	FIRST_FP, CURRENT_FP	: 0624
			0D	1A	00079	BGTRU	4\$	
			7E	D4	0007B	CLRL	-(SP)	: 0626
	64	0009802B	8F	DD	0007D	PUSHL	#622635	
			02	FB	00083	CALLS	#2, TBK\$FAKE_MSG	
			12	11	00086	BRB	5\$	
		0000'	CF	9F	00088	4\$: PUSHAB	P.AAA	: 0632
00000000G	00		01	FB	0008C	CALLS	#1, TBK\$FAD_PUT	
00000000G	00		00	FB	00093	CALLS	#0, TBK\$OUT_PUT	: 0634
	65		01	DO	0009A	5\$: MOVL	#1, TBK\$GL_EXC_TYPE	: 0643
			53	DD	0009D	PUSHL	EXCEPTN_NAME	: 0644
0000V	CF		01	FB	0009F	CALLS	#1, IS_EXCEPTION	
	03		50	E9	000A4	BLBC	R0, 6\$	
	65		02	DO	000A7	MOVL	#2, TBK\$GL_EXC_TYPE	: 0646
	52	1C	AC	DO	000AA	6\$: MOVL	CURRENT_FP, R2	: 0652
	52	18	AC	D1	000AE	CML	FIRST_FP, R2	
			4D	1B	000B2	7\$: BLEQU	9\$	
		20	AC	DD	000B4	PUSHL	CURRENT_PC	: 0659
00000000G	00		01	FB	000B7	CALLS	#1, TBK\$SYMBOLIZE	
		20	AC	DD	000BE	PUSHL	CURRENT_PC	: 0668
		00000000G	00	DD	000C1	PUSHL	TBK\$REL_PC	: 0667
		00000000G	00	DD	000C7	PUSHL	TBK\$GL_STMT	: 0666
		00000000G	00	DD	000CD	PUSHL	TBK\$GL_LINE	: 0665
		00000000G	00	DD	000D3	PUSHL	TBK\$ROUTINE_CS	: 0664
		00000000G	00	DD	000D9	PUSHL	TBK\$MODULE_CS	: 0663
0000V	CF		06	FB	000DF	CALLS	#6, OUT_TRACEBACK	
	65		01	DO	000E4	MOVL	#1, TBK\$GL_EXC_TYPE	: 0676
0C	A2		52	D1	000E7	CML	R2, 12(R2)	: 0683
			0D	1F	000EB	BLSSU	8\$	
			7E	D4	000ED	CLRL	-(SP)	: 0686
	64	0009848C	8F	DD	000EF	PUSHL	#623804	
			02	FB	000F5	CALLS	#2, TBK\$FAKE_MSG	
			07	11	000F8	BRB	9\$: 0685
1C	AC	0C	A2	7D	000FA	8\$: MOVQ	12(R2), CURRENT_FP	: 0691
			A9	11	000FF	BRB	6\$: 0652
	50		53	DO	00101	9\$: MOVL	EXCEPTN_NAME, R0	: 0699
			04	00104	RET		: 0700	

; Routine Size: 261 bytes. Routine Base: TBK\$CODE + 0000

```

433 0701 1 ROUTINE out_traceback (mod_nam,
434 0702 1 lab_nam,
435 0703 1 line_num,
436 0704 1 stmt_num,
437 0705 1 rel_pc,
438 0706 1 abs_pc) : NOVALUE = ! outputs a line of traceback
439 0707 1 +-
440 0708 1
441 0709 1 Functional Description:
442 0710 1
443 0711 1 Outputs a line (or two) of traceback information.
444 0712 1
445 0713 1 Formal Parameters:
446 0714 1
447 0715 1 MOD_NAM: address of module name counted string
448 0716 1 LAB_NAM: address of label (routine) name CS
449 0717 1 LINE_NUM: line number matching the PC
450 0718 1 STMT_NUM: statement number within the line
451 0719 1 REL_PC: PC relative to label (routine)
452 0720 1 ABS_PC: PC matching the line number
453 0721 1
454 0722 1 Implicit Inputs:
455 0723 1
456 0724 1 File(s) have been opened already....
457 0725 1
458 0726 1 Implicit Outputs:
459 0727 1
460 0728 1 Output to file(s)...
461 0729 1
462 0730 1 Routine Value:
463 0731 1
464 0732 1 NOVALUE
465 0733 1
466 0734 1 Side Effects:
467 0735 1
468 0736 1 Output via TBK$FAO_PUT and TBK$OUT_PUT.
469 0737 1
470 0738 1 --
471 0739 1
472 0740 2 BEGIN MAP mod_nam : CS_POINTER,
473 0741 2 lab_nam : CS_POINTER;
474 0742 2
475 0743 2 LOCAL string_ptr : CS_POINTER;
476 0744 2
477 0745 2 BIND null_string = UPLIT BYTE (0);
478 0746 2
479 0747 2
480 0748 2 ! Print the module name, if we have one
481 0749 2
482 0750 2 string_ptr = (IF .mod_nam NEQ 0 THEN .mod_nam ELSE null_string);
483 0751 2
484 0752 2 tbk$fao_put (UPLIT (%ASCIC '!15AC '), .string_ptr);
485 0753 2
486 0754 2 string_ptr = (IF .lab_nam NEQ 0 THEN .lab_nam ELSE null_string);
487 0755 2
488 0756 2 IF .string_ptr[0] GTRU 31
489 0757 3 THEN BEGIN tbk$fao_put (UPLIT (%ASCIC '!63AC!/' ), .string_ptr);

```

```

: 490
: 491
: 492
: 493
: 494
: 495
: 496
: 497
: 498
: 499
: 500
: 501
: 502
: 503
: 504
: 505
: 506

```

```

0758
0759
0760
0761
0762
0763
0764
0765
0766
0767
0768
0769
0770
0771
0772
0773
0774
1 END;

```

```

tbk$fao_put (UPLIT (%ASCIC '!49* '));
END
ELSE
tbk$fao put (UPLIT (%ASCIC '!32AC'), .string_ptr);
IF
.line_num NEQ 0
THEN
tbk$fao_put (UPLIT (%ASCIC '!5UL'), .line_num)
ELSE
tbk$fao_put (UPLIT (%ASCIC '!5* '));
IF
.stmt_num NEQ 0
THEN
tbk$fao_put (UPLIT (%ASCIC '!4ZL'), .stmt_num)
ELSE
tbk$fao_put (UPLIT (%ASCIC '!5* '));
tbk$fao_put (UPLIT (%ASCIC '!9XL!10XL'), .rel_pc, .abs_pc);
tbk$out_put (); ! Cause the current buffer to be output.

```

											.PSECT TBK\$PLIT,NOWRT, SHR, PIC,0					
											00	00050	P.AAB:	.BYTE	0	
												00051		.BLKB	3	
	00	20	43	41	35	31	21	06			00054	P.AAC:	.ASCII	<6>\!15AC \<0>		
	2F	21	43	41	33	36	21	07			0005C	P.AAD:	.ASCII	<7>\!63AC!\		
	00	00	20	2A	39	34	21	05			00064	P.AAE:	.ASCII	<5>\!49* \<0><0>		
	00	00	43	41	32	33	21	05			0006C	P.AAF:	.ASCII	<5>\!32AC\<0><0>		
	00	00	00	4C	55	35	21	04			00074	P.AAG:	.ASCII	<4>\!5UL\<0><0><0>		
	00	00	00	2A	35	35	21	04			0007C	P.AAH:	.ASCII	<4>\!5* \<0><0><0>		
	00	00	4C	5A	34	21	2E	05			00084	P.AAI:	.ASCII	<5>\!4ZL\<0><0>		
	00	00	00	20	2A	35	21	04			0008C	P.AAJ:	.ASCII	<4>\!5* \<0><0><0>		
00	00	4C	58	30	31	21	4C	58	39	21	09	00094	P.AAK:	.ASCII	<9>\!9XL!10XL\<0><0>	

NULL_STRING= P.AAB

											.PSECT TBK\$CODE,NOWRT, SHR, PIC,0								
											001C	00000	OUT_TRACEBACK:						
													.WORD	Save R2,R3,R4		0701			
	54											0000'	CF	9E	000U2	MOVAB	NULL_STRING, R4		
	53											00000000G	00	9E	00007	MOVAB	TBK\$FAO_PUT, R3		
												04	AC	D5	0000E	TSTL	MOD_NAM		
													06	13	00011	BEQL	1\$	0750	
	52											04	AC	D0	00013	MOVL	MOD_NAM, STRING_PTR		
													03	11	00017	BRB	2\$		
	52												64	9E	00019	1\$: MOVAB	NULL_STRING, STRING_PTR		
													52	DD	0001C	2\$: PUSHL	STRING_PTR	0752	
												04	A4	9F	0001E	PUSHAB	P.AAC		
	63												02	FB	00021	CALLS	#2, TBK\$FAO_PUT		
													08	AC	D5	00024	TSTL	LAB_NAM	0754
													06	13	00027	BEQL	3\$		
	52												08	AC	D0	00029	MOVL	LAB_NAM, STRING_PTR	
													03	11	0002D	BRB	4\$		
	52												64	9E	0002F	3\$: MOVAB	NULL_STRING, STRING_PTR		
	1F												62	91	00032	4\$: CMPB	(STRING_PTR), #31	0756	

		10	1B	00035	BLEQU	5\$		
		52	DD	00037	PUSHL	STRING_PTR		0757
	0C	A4	9F	00039	PUSHAB	P.AAD		
63		02	FB	0003C	CALLS	#2, TBK\$FAO_PUT		
	14	A4	9F	0003F	PUSHAB	P.AAE		0758
63		01	FB	00042	CALLS	#1, TBK\$FAO_PUT		
		08	11	00045	BRB	6\$		0756
		52	DD	00047	PUSHL	STRING_PTR	5\$:	0760
	1C	A4	9F	00049	PUSHAB	P.AAF		
63		02	FB	0004C	CALLS	#2, TBK\$FAO_PUT		
	0C	AC	D5	0004F	TSTL	LINE_NUM	6\$:	0762
		08	13	00052	BEQL	7\$		
	0C	AC	DD	00054	PUSHL	LINE_NUM		0763
	24	A4	9F	00057	PUSHAB	P.AAG		
63		02	FB	0005A	CALLS	#2, TBK\$FAO_PUT		
		06	11	0005D	BRB	8\$		
	2C	A4	9F	0005F	PUSHAB	P.AAH	7\$:	0764
63		01	FB	00062	CALLS	#1, TBK\$FAO_PUT		
	10	AC	D5	00065	TSTL	STMT_NUM	8\$:	0766
		08	13	00068	BEQL	9\$		
	10	AC	DD	0006A	PUSHL	STMT_NUM		0767
	34	A4	9F	0006D	PUSHAB	P.AAI		
63		02	FB	00070	CALLS	#2, TBK\$FAO_PUT		
		06	11	00073	BRB	10\$		
	3C	A4	9F	00075	PUSHAB	P.AAJ	9\$:	0768
63		01	FB	00078	CALLS	#1, TBK\$FAO_PUT		
	7E	14	AC	7D 0007B	MOVQ	REL_PC, -(SP)	10\$:	0770
		44	A4	9F 0007F	PUSHAB	P.AAK		
	63	03	FB	00082	CALLS	#3, TBK\$FAO_PUT		
00000000G	00	00	FB	00085	CALLS	#0, TBK\$OUT_PUT		0772
		04	0008C	RET				0774

; Routine Size: 141 bytes, Routine Base: TBK\$CGDE + 0105

```

508 0775 1 ROUTINE IS_EXCEPTION( EXC_NAME ) =
509 0776 1
510 0777 1 ++
511 0778 1 Functional Description:
512 0779 1
513 0780 1 Given an exception name - the longword which encodes the
514 0781 1 type, etc, of an exception - deduce if this exception is
515 0782 1 the so-called FAULT_EXC type. This is for the PC_TO_LINE
516 0783 1 translation - we have to know if the PC is on the instruction
517 0784 1 which caused the exception, or if it is on the next instruction.
518 0785 1
519 0786 1 The answer to the question is simply whether
520 0787 1 the given EXC_NAME is in our table of exceptions. The only
521 0788 1 trickery is that this routine makes sure only to look at
522 0789 1 the part of the longword which encodes the error code - and
523 0790 1 not at the rest of it since that may change.
524 0791 1
525 0792 1 Formal Parameters:
526 0793 1
527 0794 1 EXC_NAME - the longword system-defined exception name.
528 0795 1
529 0796 1 Routine Value:
530 0797 1
531 0798 1 TRUE or FALSE. See above.
532 0799 1
533 0800 1 Side Effects:
534 0801 1 None.
535 0802 1 --
536 0803 1
537 0804 2 BEGIN
538 0805 2 MAP
539 0806 2 EXC_NAME : BLOCK [ XUPVAL, BYTE ];
540 0807 2 BIND
541 0808 2 ! The 0-ended list of exception codes.
542 0809 2 EXCEPTION_LIST = UPLIT WORD (
543 0810 2 SSS_ACCVIO,
544 0811 2 SSS_NOTRAN,
545 0812 2 SSS_RADRMOD,
546 0813 2 SSS_ROPRAND,
547 0814 2 SSS_OPCCUS,
548 0815 2 SSS_BREAK,
549 0816 2 SSS_FLTOVF_F,
550 0817 2 SSS_FLTUND_F,
551 0818 2 SSS_FLTDIV_F,
552 0819 2 SSS_TBIT,
553 0820 2 SSS_COMPAT,
554 0821 2 0
555 0822 2 )
556 0823 2 : VECTOR[ WORD ];
557 0824 2
558 0825 2 ! Simply loop thru the list checking each one,
559 0826 2 ! ending when the 0 one is encountered.
560 0827 2 INCR I FROM 0
561 0828 2 DO
562 0829 2 BEGIN
563 0830 2
564 0831 2

```



```

: 565      0832  3
: 566      0833  3
: 567      0834  3
: 568      0835  3
: 569      0836  3
: 570      0837  3
: 571      0838  3
: 572      0839  3
: 573      0840  3
: 574      0841  3
: 575      0842  3
: 576      0843  3
: 577      0844  3
: 578      0845  3
: 579      0846  3
: 580      0847  3
: 581      0848  3
: 582      0849  1 END;

```

```

LOCAL
LIST_ENTRY : BLOCK [ %UPVAL, BYTE ];

IF( (LIST_ENTRY = .EXCEPTION_LIST[ .I ]) EQL 0 )
THEN
EXITLOOP;

IF( .EXC_NAME[ ST$V_FAC_NO ] EQL 0 ) AND
( .EXC_NAME[ ST$V_MSG_NO ] EQL .LIST_ENTRY[ ST$V_MSG_NO ] )
THEN
RETURN(TRUE);

END;

! Entry not found in the exception list.
RETURN(FALSE);

```

```

                                .PSECT TBK$PLIT, NOWRT, SHR, PIC, 0
04BC 04C4 04B4 0414 0434 043C 0454 044C 0629 000C 000A0 P.AAL: .WORD 12, 1577, 1100, 1108, 1084, 1076, 1044, -
                                0000 042C 0464 000B4 1204, 1220, 1212, 1124, 1068, 0

```

EXCEPTION_LIST= P.AAL

```

                                .PSECT TBK$CODE, NOWRT, SHR, PIC, 0
                                0004 00000 IS_EXCEPTION:
                                .WORD Save R2
                                CLRL I
                                MOVZWL EXCEPTION_LIST[I], LIST_ENTRY
                                BEQL 3$
                                BITW EXC_NAME+2, #4095
                                BNEQ 2$
                                XORW3 EXC_NAME, LIST_ENTRY, R2
                                BITW R2, #65528
                                BNEQ 2$
                                MOVL #1, R0
                                RET
                                DB 50 7FFFFFFF 8F F3 00024 2$: AOBLEQ #2147483647, I, 1$
                                50 D4 0002C 3$: CLRL R0
                                04 0002E RET
                                : 0775
                                : 0839
                                : 0835
                                : 0839
                                : 0840
                                : 0842
                                : 0829
                                : 0848
                                : 0849

```

: Routine Size: 47 bytes, Routine Base: TBK\$CODE + 0192

```

584 L 0850 1 %IF TBK_BAS2
585 U 0851 1 %THEN
586 C 0852 1
587 U 0853 1 GLOBAL ROUTINE LIST_DST =
588 U 0854 1
589 U 0855 1 !++
590 U 0856 1 !--
591 U 0857 1 BEGIN
592 U 0858 1 LOCAL
593 U 0859 1     nt_count,
594 U 0860 1     DST_REC_ID,
595 U 0861 1     DST_REC_RD : REF DST$RECORD;
596 U 0862 1 $FAO TT OUT('listing off the DST');
597 U 0863 1 WHILE( (DST_REC_RD = TBK$GET_NXT_DST( DST_REC_ID )) NEQ 0 )
598 U 0864 1 DO
599 U 0865 1 BEGIN
600 U 0866 1
601 U 0867 1     ! Process each record depending on its DST type.
602 U 0868 1 %IF TBK_BAS2
603 U 0869 1 %THEN
604 U 0870 1     ! For diagnostic purposes we list out the entire record.
605 U 0871 1
606 U 0872 1 IF( .DST_REC_RD[DST$b_TYPE] EQL dst$k_modbeg)
607 U 0873 1 THEN
608 U 0874 1 BEGIN
609 U 0875 1     $FAO TT OUT('MC for module ');
610 U 0876 1     pr_cs(dst_recrd[dst$b_name]);
611 U 0877 1     end;
612 U 0878 1 $FAO TT_OUT( 'DST Rec Id=!XL, is at !XL, for !UD bytes.'
613 U 0879 1     .DST_REC_ID, .DST_REC_RD, .DST_REC_RD[dst$b_length] );
614 U 0880 1
615 U 0881 1     ! Dump the reocrd in bytes.
616 U 0882 1
617 U 0883 1 INCR I FROM 0 TO .DST_REC_RD[dst$b_length]
618 U 0884 1 DO
619 U 0885 1     $FAO TT_OUT('!XB ',.DST_REC_RD[ .I, 0, 8, 0 ] );
620 U 0886 1
621 U 0887 1 %FI
622 U 0888 1
623 U 0889 1 CASE .DST_REC_RD[dst$b_type] FROM dst$k_lowest TO dst$k_highest OF
624 U 0890 1
625 U 0891 1 SET
626 U 0892 1 [dst$k_modbeg]:      ! Module Begin Record.
627 U 0893 1 BEGIN
628 U 0894 1 LOCAL
629 U 0895 1     NEW_PTR : REF MC_RECORD;
630 U 0896 1 END;
631 U 0897 1
632 U 0898 1 [dst$k_modend]:      ! Module End Record.
633 U 0899 1
634 U 0900 1 BEGIN
635 U 0901 1
636 U 0902 1 END;
637 U 0903 1
638 U 0904 1 [dst$k_rtnbeg,
639 U 0905 1     ! Routine DSTs.
640 U 0906 1

```

```

641 U 0907 1
642 U 0908 1
643 U 0909 1
644 U 0910 1
645 U 0911 1
646 U 0912 1
647 U 0913 1
648 U 0914 1
649 U 0915 1
650 U 0916 1
651 U 0917 1
652 U 0918 1
653 U 0919 1
654 U 0920 1
655 U 0921 1
656 U 0922 1
657 U 0923 1
658 U 0924 1
659 U 0925 1
660 U 0926 1
661 U 0927 1
662 U 0928 1
663 U 0929 1
664 U 0930 1
665 U 0931 1
666 U 0932 1
667 U 0933 1
668 U 0934 1
669 U 0935 1
670 U 0936 1
671 U 0937 1
672 U 0938 1
673 U 0939 1
674 U 0940 1
675 U 0941 1
676 U 0942 1
677 U 0943 1 %IF tbk_bas2
678 U 0944 1 %THEN
679 U 0945 1
680 U 0946 1
681 U 0947 1
682 U 0948 1 %FI
683 U 0949 1
684 U 0950 1
685 U 0951 1
686 U 0952 1
687 U 0953 1
688 U 0954 1
689 U 0955 1
690 U 0956 1
691 U 0957 1
692 U 0958 1
693 U 0959 1
694 U 0960 1
695 U 0961 1
696 U 0962 1
697 U 0963 1

```

```

dst$k_label]:          ! Labels in FORTRAN and BLISS.
    BEGIN
    ! Just tally up the needed statistics
    ! so that we can build the other data
    ! structures later.
    NT_COUNT = .NT_COUNT +1;
    END;
[dst$k_rtnend,        ! BLISS-only End-of-Routine.
dst$k_blifld]:       ! BLISS-only FIELD records.
    ! We can safely ignore these for now.
    ;
[dst$k_lblorlit]:    ! Label or Literal DSTs. (MARS only)
    BEGIN
    NT_COUNT = .NT_COUNT +1;
    END;
[dst$k_psect]:       ! Psect DSTs.
    BEGIN
    BIND
    PSECT_LENGTH
    =
    ! Pick up the field length, which
    ! is after the NAME so must be
    ! dynamically located.
    (.DST_RECRD[dst$b_name]
    + DST_RECRD[dst$b_name]
    + 1 ): LONG;
    ! The symbol-name count,
    ! plus its address,
    ! addresses the LENGTH.
    $FAO TT_OUT('PSECT begins: !XL, ends !XL',
    .DST_RECRD[dst$l_value],
    .DST_RECRD[dst$l_value]+.PSECT_LENGTH+1 );
    nt_count = .nt_count +1;
    END;
[INRANGE, OVRANGE]:
    BEGIN
    ! The only reason for not making the 'SRM types'
    ! part of the above CASE is because of the huge
    ! case table which gets generated otherwise.
    IF( .DST_RECRD[dst$b_type] EQL DSC$K_DTYPE_Z )
    THEN
        BEGIN

```

```

: 698      U 0964 1      ! BLISS type ZERO records.
: 699      UU 0965 1 %IF TBK_bas2
: 700      UU 0966 1 %then
: 701      UU 0967 1
: 702      UU 0968 1 %FI
: 703      UU 0969 1
: 704      UU 0970 1
: 705      UU 0971 1
: 706      UU 0972 1
: 707      UU 0973 1
: 708      UU 0974 1
: 709      UU 0975 1
: 710      UU 0976 1
: 711      UU 0977 1
: 712      UU 0978 1
: 713      UU 0979 1
: 714      UU 0980 1
: 715      UU 0981 1
: 716      UU 0982 1
: 717      UU 0983 1
: 718      UU 0984 1
: 719      UU 0985 1
: 720      UU 0986 1
: 721      UU 0987 1
: 722      UU 0988 1
: 723      UU 0989 1
: 724      UU 0990 1
: 725      UU 0991 1
: 726      UU 0992 1
: 727      UU 0993 1
: 728      UU 0994 1
: 729      UU 0995 1
: 730      U 0996 1
: 731      U 0997 1

```

```

! BLISS type ZERO records.

$FAO TT_OUT('ignoring Z record');

! Whatever symbol this is, it contributes
! a name, for sure, and either a literal
! or a static. We assume the worst!

NT_COUNT = .NT_COUNT +1;
END
ELSE
IF( .DST_RECRD[dst$b_type] LEQ dsc$k_dtype_highest)
THEN
BEGIN

! These types are candidates for
! the LVT and NT tables only.

NT_COUNT = .NT_COUNT +1;
END;

END;
TES;

! Go back and process the next DST record.

END;

$FAO TT_OUT('DST listed OK');
RETURN(-1 );

END;
%FI

```

```

: 733      L 0998 1 %IF TBK_BAS2
: 734      UU 0999 1 %THEN
: 735      UU 1000 1      ! This routine is only used by DEBUGging output routines.
: 736      UU 1001 1
: 737      UU 1002 1 ROUTINE PR_CS( ADDR ) : NOVALUE =
: 738      UU 1003 1
: 739      UU 1004 1
: 740      UU 1005 1 !++
: 741      UU 1006 1 | Functional Description:
: 742      UU 1007 1 | Print out a counted string in an
: 743      UU 1008 1 | unambiguous way for debugging purposes.
: 744      UU 1009 1 |--
: 745      UU 1010 1
: 746      UU 1011 1 BEGIN
: 747      UU 1012 1     MAP
: 748      UU 1013 1         ADDR : REF VECTOR[,BYTE];
: 749      UU 1014 1
: 750      UU 1015 1     ! Don't get fooled!
: 751      UU 1016 1
: 752      UU 1017 1     IF( .ADDR EQL 0 )
: 753      UU 1018 1     THEN
: 754      UU 1019 1         $FAO_TT_OUT( '**** PR_CS AT 0 **** ' )
: 755      UU 1020 1     ELSE
: 756      UU 1021 1         $FAO_TT_OUT( 'Name(!UB.): '!AC'. ' , .ADDR[0], ADDR[0] );
: 757      UU 1022 1     END;
: 758      U 1023 1
: 759      1024 1 %FI

```

: 761 1025 1 END
: 762 1026 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
TBK\$PLIT	186	NOVEC,NOWRT, RD ; EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)
TBK\$CODE	449	NOVEC,NOWRT, RD ; EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	21	0	1000	00:01.9
_\$255\$DUA28:[TRACE.OBJ]TBKLIB.L32;1	157	5	3	14	00:00.2
_\$255\$DUA28:[TRACE.OBJ]STRUCDEF.L32;1	32	0	0	7	00:00.1
_\$255\$DUA28:[TRACE.OBJ]TBKDST.L32;1	414	103	24	30	00:00.3

COMMAND QUALIFIERS

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

: Size: 449 code + 186 data bytes
: Run Time: 00:15.7
: Elapsed Time: 00:56.1
: Lines/CPU Min: 3911
: Lexemes/CPU-Min: 21918
: Memory Used: 137 pages
: Compilation Complete

0401 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

The image displays a grid of 144 small terminal windows, each showing a different screen from the VAX/VMS V4.0 system. The screens contain various text-based data, including system status, error messages, and user prompts. Some windows are highlighted with larger text labels: TBKLIB LIS (top right), TBKBAS LIS (middle left), TBKST LIS (middle left), TBKINT LIS (middle right), and TBKOPC LIS (bottom left).