

1
1
C
C

1
1
C

100

.....

[illegible]

```

  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         S
  LL      II         S
  LL      II         S
  LLLLLLLLLL  IIIIII  SSSSSSSS
  LLLLLLLLLL  IIIIII  SSSSSSSS
  
```


[illegible]

TBKINT
V04-000

N 14

16-Sep-1984 02:15:19

14-Sep-1984 13:20:19

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[TRACE.SRC]TBKINT.B32;1

Page 2

(1)

58	0058	1			
59	0059	1			
60	0060	1			
61	0061	1	05	01-mar-78	KGP
62	0062	1			
63	0063	1			
64	0064	1	06	8-mar-78	KGP
65	0065	1			
66	0066	1			
67	0067	1			
68	0068	1			
69	0069	1			
70	0070	1	07	27-mar-78	KGP
71	0071	1	08	26-APR-78	DAR
72	0072	1			
73	0073	1	09	15-JUN-78	DAR
74	0074	1	1.01	08-DEC-78	DAR
75	0075	1			
76	0076	1	1.02	30-OCT-79	JBD
77	0077	1	3.01	03-Mar-82	RT
78	0078	1			
79	0079	1			
80	0080	1		23-Dec-82	VJH
81	0081	1			
82	0082	1		15-Aug-83	PS
83	0083	1			
84	0084	1			

produce a TRACEback).

- FIND DST finds the DST where it mapped the first time, on subsequent TRACES.
- FIND_DST is now NOVALUE - it either does its thing or leaves things so that a non-symbolic TRACE is done.
- Beefed up FIND_DST to return a symbolic indication, and to print proper error messages.
- Completed the separation of TRACE and DEBUG - even the REQUIRE files and P-SECT names are now separate.
- Renamed TBK\$PUTMSG to TBK\$FAKE_MSG

Modified require and library directives for native build.

Changed all DBG\$ symbols to TBK\$.

In TBK\$FIND_DST ask for the number of pages we need instead of 1 (\$EXPREG).

Changed CREMAPSEC to use EXPREG.

Passed in channel number to TBK\$RST_FIND so it doesn't have to open image file again.

Changed call to LIB\$_CREMAPSEC to call to SYSSCRMPSC.

Did general clean up to use updated files from DEBUG.


```

86      0085 1  ! Table of contents:
87      0086 1
88      0087 1 FORWARD ROUTINE
89      0088 1     tbk$find_dst,           ! Find the DST and make it available.
90      0089 1     tbk$get_dst_rec,       ! Make a certain DST record available.
91      0090 1     tbk$get_nxt_dst,       ! Make the next DST record available.
92      0091 1     tbk$positon_dst;       ! Make a certain DST record available
93      0092 1                               ! and set up for tbk$get_nxt_dst
94      0093 1
95      0094 1  ! INCLUDE FILES:
96      0095 1
97      0096 1 REQUIRE 'SRC$:TBKPROLOG.REQ';
98      0368 1
99      0369 1
100     0370 1
101     0371 1  ! MACROS:
102     0372 1
103     0373 1
104     0374 1
105     0375 1  ! EQUATED SYMBOLS:
106     0376 1
107     0377 1 LITERAL
108     0378 1     TBK_INT1                = 0;    ! Diagnostic variable for mapping the DST in.
109     0379 1
110     0380 1
111     0381 1  !
112     0382 1  ! OWN STORAGE:
113     0383 1
114     0384 1
115     0385 1 OWN
116     0386 1     dst_begin_addr : INITIAL(1),    ! virtual address where DST begins.
117     0387 1     ! 1 => the DST did not get mapped.
118     0388 1     dst_end_addr,                  ! virtual address of last byte in DST.
119     0389 1     dst_next_addr,                  ! virtual address where 'next' DST record begins.
120     0390 1     exe_file : $FAB (FAC = GET),
121     0391 1
122     0392 1     exe_input : $RAB (USZ = 512);
123     0393 1
124     0394 1
125     0395 1  ! EXTERNAL REFERENCES:
126     0396 1
127     0397 1 EXTERNAL ROUTINE
128     0398 1     TBK$FAO_OUT : NOVALUE,
129     0399 1     tbk$fake_msg : novalue,        ! print TRACEback messages.
130     0400 1     SY$CRMPSC: ADDRESSING_MODE (GENERAL); ! Creates and maps a global section.

```



```
132 0401 1 GLOBAL ROUTINE tbk$find_dst (imgfilchan, file_name,  
133 0402 1                                     img_header_blk, symtab_sec_bnds) =  
134 0403 1  
135 0404 1 ++  
136 0405 1 Functional description:  
137 0406 1     If a DST exists for the specified image, open the image file,  
138 0407 1     map in the DST, and set it up for get_nxt_dst,  
139 0408 1     get_dst_rec, and positon_dst.  
140 0409 1  
141 0410 1     If the DST cannot be mapped, for any reason, things are  
142 0411 1     left so that a non-symbolic TRACE happens along with the  
143 0412 1     appropriate (warning) error message.  
144 0413 1  
145 0414 1 Formal parameters:  
146 0415 1     imgfilchan      - the channel number that the image file is open on  
147 0416 1     file_name       - a counted string to the file specification of  
148 0417 1                   the image file.  
149 0418 1     img_header_blk  - the address of a byte block that contains the  
150 0419 1                   image header data needed to locate the DST.  
151 0420 1     symtab_sec_bnds - The address of a 2-longword vector where the  
152 0421 1                   beginning and end of where the DST was mapped to  
153 0422 1                   can be stored across TRACE invocations.  
154 0423 1  
155 0424 1 Implicit inputs:  
156 0425 1     The image activator has read in the image file header.  
157 0426 1  
158 0427 1 Output parameters:  
159 0428 1     none  
160 0429 1  
161 0430 1 Implicit outputs:  
162 0431 1     Three own variables are set up after the DST is mapped in.  
163 0432 1         dst_begin_addr - beginning of the DST  
164 0433 1         dst_end_addr   - end of the DST  
165 0434 1         dst_next_addr  - address of next DST record  
166 0435 1  
167 0436 1 Routine value:  
168 0437 1     TRUE, if we expect the TRACEback will be symbolic,  
169 0438 1     FALSE, otherwise.  
170 0439 1  
171 0440 1 Side effects:  
172 0441 1     The image file is opened and closed. The DST is mapped into the  
173 0442 1     top of P0 space. When this is done, the beginning and ending  
174 0443 1     addresses are 'stuffed' back into the 2-longword vector  
175 0444 1     we are passed a pointer to (symtab_sec_bnds).  
176 0445 1  
177 0446 1 --  
178 0447 1  
179 0448 2 BEGIN  
180 0449 2  
181 0450 2 MAP  
182 0451 2     symtab_sec_bnds : ref vector[,long],  
183 0452 2     file_name      : REF VECTOR [, BYTE],  
184 0453 2     img_header_blk : REF BLOCK [, BYTE];  
185 0454 2  
186 0455 2 BIND  
187 0456 2     sym_tbl_data    = .img_header_blk + .img_header_blk [ihd$w_syndbgoff] : BLOCK [, BYTE],  
188 0457 2     exe$ecnam       = UPLIT BYTE (%ASCII 'DST');
```



```

189      0458      2
190      0459      2
191      0460      2
192      0461      2
193      0462      2
194      0463      2
195      0464      2
196      0465      2
197      0466      2
198      0467      2
199      0468      2
200      L 0469      2
201      U 0470      2
202      U 0471      2
203      U 0472      2
204      U 0473      2
205      U 0474      2
206      U 0475      2
207      U 0476      2
208      U 0477      2
209      U 0478      2
210      U 0479      2
211      U 0480      2
212      U 0481      2
213      U 0482      2
214      U 0483      2
215      U 0484      2
216      U 0485      2
217      U 0486      2
218      U 0487      2
219      L 0488      2
220      U 0489      2
221      U 0490      2
222      U 0491      2
223      U 0492      2
224      U 0493      2
225      U 0494      2
226      U 0495      2
227      U 0496      2
228      U 0497      2
229      U 0498      2
230      U 0499      2
231      U 0500      2
232      U 0501      2
233      U 0502      2
234      L 0503      2
235      U 0504      2
236      U 0505      2
237      U 0506      2
238      U 0507      2
239      U 0508      2
240      U 0509      2
241      U 0510      2
242      U 0511      2
243      U 0512      2
244      U 0513      2
245      U 0514      2

      LITERAL      dst_end_address = 1;

      LOCAL
      local_buffer      : VECTOR [10],
      exe_sec_bounds      : VECTOR [2],
      exe_secnam_desc      : VECTOR [2],
      exefilnam_desc      : VECTOR [2],
      status;

      %if tbk_int1
      %then
      $fao_tt_out('!/symtab sec bnds vector is at !XL, begin = !XL',
      .symtab_sec_bnds,.symtab_sec_bnds[0]);
      %FI
      ! See if this is a second (or greater) invocation of
      ! TRACE so that we can just reuse the DST we mapped last time.
      IF( .SYMTAB_SEC_BNDS[0] NEQ 0 )
      THEN
      BEGIN
      ! Set up to use the DST in the same way as we used
      ! it during the first invocation of TRACE.
      dst_begin_addr = .symtab_sec_bnds [0];
      dst_end_addr = .symtab_sec_bnds [1];
      dst_next_addr = .dst_begin_addr;
      %IF TBK_INT1
      %THEN
      $fao_tt_out('!/DST exists at !XL and ends at !XL',
      .dst_begin_addr,.dst_end_addr);
      %FI
      ! The traceback will be symbolic.
      return(TRUE);
      END;
      !++
      ! First invocation - try to map in the DST.
      ! See whether the image header has valid data in it.
      !--
      %if tbk_int1
      %then
      $fao_tt_out('!/symdbgoff is !XL',.img_header_blk[ihd$w_symdbgoff]);
      %FI
      IF .img_header_blk [ihd$w_symdbgoff] EQL 0
      then
      begin
      TBK$fake_MSG(TBK$_BADHDR,0);
      ! traceback will be non-symbolic.
      return(false);

```



```
246      END;
247
248      ++
249      Now that the image header seems to be a valid one, see whether
250      it included DST data.
251      --
252      %if tbk_int1
253      %then
254          $fao_tt_out('!/dstblks = !XL, dstvbn = !XL',
255                    .sym_tbl_data[ ihs$w_dstblks ],
256                    .sym_tbl_data[ ihs$l_dstvbn ]);
257      %FI
258      IF .sym_tbl_data[ ihs$w_dstblks ] EQL 0
259      then
260          begin
261              TBK$fake_MSG(TBK$_BADDST,0);
262              ! traceback will be non-symbolic
263              return(false);
264          end;
265      ++
266      There appears to be a DST. Do one more consistency
267      check.
268      --
269      IF NOT (.sym_tbl_data [ihs$l_dstvbn] GTR 2)
270      then
271          begin
272              TBK$fake_MSG(TBK$_BADDSTVBN,.SYM_TBL_DATA[IHS$L_DSTVBN]);
273              ! TRACEback will be non-symbolic.
274              return(false);
275          end;
276      ++
277      Map the DST into P0 space.
278      --
279      exe_sec_bounds[0] = 200;
280      exe_sec_bounds[1] = 400;
281      exe$ecnam_desc [0] = 3;
282      exe$ecnam_desc [1] = exe$ecnam;
283      exe$filnam_desc [0] = .file_name [0];
284      exe$filnam_desc [1] = file_name[1];
285      status = SYS$CRMPSC(exe_sec_bounds, exe_sec_bounds, 0, SEC$M_EXPREG,
286                        exe$ecnam_desc, 0, 0, .imgfilchan, .sym_tbl_data[ihs$w_dstblks],
287                        .sym_tbl_data[ihs$l_dstvbn], 0, 0);
288      IF NOT .status
289      THEN
290          begin
291              TBK$fake_MSG(TBK$_BADDSTMAP,.status);
292              ! TRACEback will be non-symbolic.
293              return(false);
294          end;
295      ++
296      Map the DST into P0 space.
297      --
298      exe_sec_bounds[0] = 200;
299      exe_sec_bounds[1] = 400;
300      exe$ecnam_desc [0] = 3;
301      exe$ecnam_desc [1] = exe$ecnam;
302      exe$filnam_desc [0] = .file_name [0];
303      exe$filnam_desc [1] = file_name[1];
304      status = SYS$CRMPSC(exe_sec_bounds, exe_sec_bounds, 0, SEC$M_EXPREG,
305                        exe$ecnam_desc, 0, 0, .imgfilchan, .sym_tbl_data[ihs$w_dstblks],
306                        .sym_tbl_data[ihs$l_dstvbn], 0, 0);
307      IF NOT .status
308      THEN
309          begin
310              TBK$fake_MSG(TBK$_BADDSTMAP,.status);
311              ! TRACEback will be non-symbolic.
312              return(false);
313          end;
```



```

303      0572      2      |++
304      0573      2      | Now load up the addresses of the beginning and end of the DST.
305      0574      2      |--
306      0575      2      dst_begin_addr = .exe_sec_bounds [0];
307      0576      2      dst_end_addr = .exe_sec_bounds [1];
308      0577      2      dst_next_addr = .dst_begin_addr;
309      0578      2
310      0579      2      %IF TBK_INT1
311      0580      2      %THEN
312      0581      2      $fao_tt_out('!/DST mapped to begin at !XL and end at !XL',
313      0582      2      .dst_begin_addr,.dst_end_addr);
314      0583      2      %FI
315      0584      2      ! Save the mapped-to addresses so that if TRACE
316      0585      2      ! is invoked a second time we don't have to redo
317      0586      2      ! this mapping.
318      0587      2
319      0588      2      symtab_sec_bnds[0] = .dst_begin_addr;
320      0589      2      symtab_sec_bnds[1] = .dst_end_addr;
321      0590      2
322      0591      2      RETURN(true);
323      0592      1      END;
```

```

                                .TITLE TBKINT
                                .IDENT  \V04-000\
                                .PSECT  TBK$PLIT,NOWRT, SHR, PIC,0
54 53 44 03 00000 P.AAA: .ASCII <3>\DST\
                                .PSECT  TBK$OWN,NOEXE, PIC,2
00000001 00000 DST_BEGIN_ADDR:
                                .LONG   1
00004 DST_END_ADDR:
                                .BLKB   4
00008 DST_NEXT_ADDR:
                                .BLKB   4
03 0000C EXE_FILE:
00000000 .BYTE 3
50 0000D .BYTE 80
0000 0000E .WORD 0
00000000 00010 .LONG 0
00000000 00014 .LONG 0
00000000 00018 .LONG 0
00000000 0001C .LONG 0
0000 00020 .WORD 0
02 00022 .BYTE 2
00 00023 .BYTE 0
00000000 00024 .LONG 0
00 00028 .BYTE 0
00 00029 .BYTE 0
00 0002A .BYTE 0
02 0002B .BYTE 2
00000000 0002C .LONG 0
00000000 00030 .LONG 0
00000000 00034 .LONG 0
```



```

00000000 00038 .LONG 0
00000000 0003C .LONG 0
      00 00040 .BYTE 0
      00 00041 .BYTE 0
      0000 00042 .WORD 0
00000000 00044 .LONG 0
      0000 00048 .WORD 0
      00 0004A .BYTE 0
      00 0004B .BYTE 0
00000000 0004C .LONG 0
00000000 00050 .LONG 0
      0000 00054 .WORD 0
      00 00056 .BYTE 0
      00 00057 .BYTE 0
00000000 00058 .LONG 0
      01 0005C EXE_INPUT:
      44 0005D .BYTE 1
      0000 0005E .BYTE 68
00000000 00060 .WORD 0
00000000 00064 .LONG 0
00000000 00068 .LONG 0
      0000# 0006C .WORD 0[3]
      0000 00072 .WORD 0
00000000 00074 .LONG 0
      0000 00078 .WORD 0
      00 0007A .BYTE 0
      00 0007B .BYTE 0
      0200 0007C .WORD 512
      0000 0007E .WORD 0
00000000 00080 .LONG 0
00000000 00084 .LONG 0
00000000 00088 .LONG 0
00000000 0008C .LONG 0
      00 00090 .BYTE 0
      00 00091 .BYTE 0
      00 00092 .BYTE 0
      00 00093 .BYTE 0
00000000 00094 .LONG 0
00000000 00098 .LONG 0
00000000 0009C .LONG 0

```

```

EXESECNAM= P.AAA
      .EXTRN TBK$FAO OUT, TBK$FAKE_MSG
      .EXTRN SYSSCRMPSC
      .PSECT TBK$CODE,NOWRT, SHR, PIC,0
      .ENTRY TBK$FIND DST, Save R2,R3,R4
MOVAB DST_BEGIN_ADDR, R4
MOVAB -64(SP), SP
MOVL IMG_HEADER_BLK, R1
MOVZWL 4(RT), R0
ADDL3 R1, R0, R2
MOVL SYMTAB_SEC_BOUNDS, R3
TSTL (R3)
BEQL 1$

```

```

54 0000' CF 9E 00000 .ENTRY TBK$FIND DST, Save R2,R3,R4 : 0401
5E C0 AE 9E 00007 MOVAB DST_BEGIN_ADDR, R4 :
51 0C AC D0 0000B MOVAB -64(SP), SP :
50 04 A1 3C 0000F MOVL IMG_HEADER_BLK, R1 : 0456
50 51 C1 00013 MOVZWL 4(RT), R0 :
53 10 AC D0 00017 ADDL3 R1, R0, R2 :
63 D5 0001B MOVL SYMTAB_SEC_BOUNDS, R3 : 0477
0A 13 0001D TSTL (R3) :
      BEQL 1$ :

```


08	64	63	7D	0001F	MOVQ	(R3), DST_BEGIN_ADDR	0484			
	A4	64	D0	00022	MOVL	DST_BEGIN_ADDR, DST_NEXT_ADDR	0486			
		0090	31	00026	BRW	7\$	0495			
		04	A1	B5	00029	1\$: TSTW	4(R1)	0507		
			0A	12	0002C	BNEQ	2\$			
			7E	D4	0002E	CLRL	-(SP)	0510		
		00098023	8F	DD	00030	PUSHL	#622627			
			6D	11	00036	BRB	5\$			
		08	A2	B5	00038	2\$: TSTW	8(R2)	0527		
			0A	12	0003B	BNEQ	3\$			
			7E	D4	0003D	CLRL	-(SP)	0530		
		0009800B	8F	DD	0003F	PUSHL	#622603			
			5E	11	00045	BRB	5\$			
		02	62	D1	00047	3\$: CMPL	(R2), #2	0540		
			0A	14	0004A	BGTR	4\$			
			62	DD	0004C	PUSHL	(R2)	0543		
		00098013	8F	DD	0004E	PUSHL	#622611			
			4F	11	00054	BRB	5\$			
		10	AE	C8	8F	9A	00056	4\$: MOVZBL	#200, EXE_SEC_BOUNDS	0554
		14	AE	0190	8F	3C	0005B	MOVZWL	#400, EXE_SEC_BOUNDS+4	0555
		08	AE		03	D0	00061	MOVL	#3, EXESECNAM_DESC	0556
		0C	AE	0000	CF	9E	00065	MOVAB	EXESECNAM, EXESECNAM_DESC+4	0557
			6E	08	BC	9A	0006B	MOVZBL	@FILE_NAME, EXEFILNAM_DESC	0558
04	AE	08	AC		01	C1	0006F	ADDL3	#1, FILE_NAME, EXEFILNAM_DESC+4	0559
					7E	7C	00075	CLRQ	-(SP)	0560
					62	DD	00077	PUSHL	(R2)	0562
			7E	08	A2	3C	00079	MOVZWL	8(R2), -(SP)	0561
				04	AC	DD	0007D	PUSHL	IMGFILCHAN	
					7E	7C	00080	CLRQ	-(SP)	0560
				24	AE	9F	00082	PUSHAB	EXESECNAM_DESC	
				00020000	8F	DD	00085	PUSHL	#131072	
					7E	D4	0008B	CLRL	-(SP)	
				38	AE	9F	0008D	PUSHAB	EXE_SEC_BOUNDS	
				3C	AE	9F	00090	PUSHAB	EXE_SEC_BOUNDS	
00000000G	00				0C	FB	00093	CALLS	#12, SYSSCRMPSC	
	11				50	E8	0009A	BLBS	STATUS, 6\$	0563
					50	DD	0009D	PUSHL	STATUS	0566
				000981B0	8F	DD	0009F	PUSHL	#623024	
00000000G	00				02	FB	000A5	5\$: CALLS	#2, TBK\$FAKE_MSG	
					0F	11	000AC	BRB	8\$	0569
					AE	7D	000AE	6\$: MOVQ	EXE_SEC_BOUNDS, DST_BEGIN_ADDR	0575
08	64	10			64	D0	000B2	MOVL	DST_BEGIN_ADDR, DST_NEXT_ADDR	0577
	A4				64	7D	000B6	MOVQ	DST_BEGIN_ADDR, (R3)	0588
	63				01	D0	000B9	7\$: MOVL	#1, R0	0591
	50					04	000BC	RET		
					50	D4	000BD	8\$: CLRL	R0	0592
					04	000BF		RET		

; Routine Size: 192 bytes, Routine Base: TBK\$CODE + 0000


```

325 0593 1 GLOBAL ROUTINE tbk$get_dst_rec (rec_id) =
326 0594 1
327 0595 1 ++
328 0596 1 Functional description:
329 0597 1 Make the indicated DST record available.
330 0598 1
331 0599 1 Input parameters:
332 0600 1 rec_id - The ID of the record to be fetched.
333 0601 1 This ID must be one that was previously returned
334 0602 1 by a call to tbk$get_nxt_dst.
335 0603 1
336 0604 1 Implicit inputs:
337 0605 1 none
338 0606 1
339 0607 1 Output parameters:
340 0608 1 none
341 0609 1
342 0610 1 Implicit outputs:
343 0611 1 none
344 0612 1
345 0613 1 Routine value:
346 0614 1 0, if the indicated record does not exist;
347 0615 1 the address of where it can now be referenced, otherwise.
348 0616 1
349 0617 1 Side effects:
350 0618 1 The DST record is made available.
351 0619 1
352 0620 1 --
353 0621 1
354 0622 2 BEGIN
355 0623 2
356 0624 2 BIND
357 0625 2 dst_recrd = .rec_id : dst$record;
358 0626 2
359 0627 2 ++
360 0628 2 The record ID is the same as the virtual address at which it
361 0629 2 can be referenced. The next record, then, is simply the one that
362 0630 2 is virtually contiguous to this one, except for the last record.
363 0631 2 In that case, the convention is that the DST ended properly
364 0632 2 if a record is requested past the end marker, or if the count
365 0633 2 field for the supposed "next" record is 0.
366 0634 2 --
367 0635 2 IF .rec_id EQL .dst_end_addr + 1
368 0636 2 THEN RETURN 0;
369 0637 2
370 0638 2 ++
371 0639 2 Now that it is safe, check for zero length records.
372 0640 2 --
373 0641 2 IF .dst_recrd [dst$b_length] EQL 0
374 0642 2 THEN RETURN 0;
375 0643 2
376 0644 2 ++
377 0645 2 Check that the ID is valid.
378 0646 2 --
379 0647 2 IF .rec_id LSSA .dst_begin_addr OR .rec_id GTRA .dst_end_addr
380 0648 2 THEN RETURN 0
381 0649 2 ELSE RETURN .rec_id

```


TBKINT
V04-000

J 15
16-Sep-1984 02:15:19
14-Sep-1984 13:20:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[TRACE.SRC]TBKINT.B32;1
Page 11
(4)

; 382 0650 1 END;

51	0000'	50	04	0000	00000	.ENTRY	TBK\$GET_DST_REC, Save nothing	0593
		CF		AC	D0 00002	MOVL	REC_ID, R0	0625
		51		01	C1 00006	ADDL3	#1, DST_END_ADDR, R1	0635
				50	D1 0000C	CMPL	R0, R1	
				12	13 0000F	BEQL	1\$	
				60	95 00011	TSTB	(R0)	0641
				0E	13 00013	BEQL	1\$	
	0000'	CF		50	D1 00015	CMPL	R0, DST_BEGIN_ADDR	0647
	0000'	CF		07	1F 0001A	BLSSU	1\$	
				50	D1 0001C	CMPL	R0, DST_END_ADDR	
				02	1B 00021	BLEQU	2\$	
				50	D4 00023	CLRL	R0	0650
				04	00025	RET		

; Routine Size: 38 bytes, Routine Base: TBK\$CODE + 00C0


```
384 0651 1 GLOBAL ROUTINE tbk$get_nxt_dst (rec_id_ptr) =
385 0652 1
386 0653 1 ++
387 0654 1 Functional description:
388 0655 1 Make the next DST record available. Return a pointer by which it
389 0656 1 can be referenced, as well as an identifying code for it.
390 0657 1
391 0658 1 Input parameters:
392 0659 1 none
393 0660 1
394 0661 1 Implicit inputs:
395 0662 1 Current values of dst_begin_addr, dst_end_addr, and dst_next_addr.
396 0663 1
397 0664 1 Output parameters:
398 0665 1 rec_id_ptr - the identifier found for the DST record.
399 0666 1
400 0667 1 Implicit outputs:
401 0668 1 dst_next_addr is advanced by one DST record.
402 0669 1
403 0670 1 Routine value:
404 0671 1 0, if the indicated record does not exist; otherwise, the
405 0672 1 address of the record.
406 0673 1
407 0674 1 Side effects:
408 0675 1 none
409 0676 1
410 0677 1 --
411 0678 1
412 0679 2 BEGIN
413 0680 2
414 0681 2 MAP
415 0682 2 rec_id_ptr : REF VECTOR [,LONG];
416 0683 2
417 0684 2 ++
418 0685 2 Since record IDs are the same as their virtual addresses, the
419 0686 2 next can be obtained in the same way than ANY one can be
420 0687 2 obtained. The only detail to fill in is passing back the ID
421 0688 2 for this next one.
422 0689 2 --
423 0690 2 RETURN (rec_id_ptr [0] = tbk$posidon_dst (.dst_next_addr));
424 0691 1 END;
```

```
0000V CF 0000' 0000 DD 00000
04 BC 01 FB 00006
50 D0 0000B
04 0000F
```

```
.ENTRY TBK$GET NXT DST, Save nothing
PUSHL DST_NEXT_ADDR
CALLS #1, TBK$POSITON_DST
MOVL R0, @REC_ID_PTR
RET
```

```
: 0651
: 0690
:
: 0691
```

; Routine Size: 16 bytes, Routine Base: TBK\$CODE + 00E6


```
426 0692 1 GLOBAL ROUTINE tbk$posidon_dst (rec_id) =
427 0693 1
428 0694 1 !++
429 0695 1 Functional description:
430 0696 1 Position dst_next_addr to point to the DST record following the
431 0697 1 record whose DST identifying code is passed to this routine as
432 0698 1 an argument.
433 0699 1
434 0700 1 Input parameters:
435 0701 1 rec_id - the identifying code of the DST record.
436 0702 1 - 0 => position DST to the beginning.
437 0703 1
438 0704 1 Implicit inputs:
439 0705 1 dst_begin_addr and dst_end_addr
440 0706 1
441 0707 1 Output parameters:
442 0708 1 none
443 0709 1
444 0710 1 Implicit outputs:
445 0711 1 dst_next_addr is set to point to the DST record following the
446 0712 1 record whose identifying code is passed as an argument.
447 0713 1
448 0714 1 Routine value:
449 0715 1 FALSE - if the DST didn't get mapped.
450 0716 1
451 0717 1 0, if the indicated record does not exist; otherwise,
452 0718 1 the address of the DST record.
453 0719 1
454 0720 1 Side effects:
455 0721 1 'next' record is changed.
456 0722 1
457 0723 1 --
458 0724 1
459 0725 2 BEGIN
460 0726 2
461 0727 2 LOCAL
462 0728 2 rec_addr : REF dst$record;
463 0729 2
464 0730 2 !+
465 0731 2 Check for a 'rewind' command.
466 0732 2 -
467 0733 2
468 0734 2 IF( .REC_ID EQL 0 )
469 0735 2 THEN
470 0736 2 BEGIN
471 0737 2
472 0738 2 ! Reposition the 'dst' to the beginning.
473 0739 2 ! Note special return for when the
474 0740 2 ! DST didn't get mapped.
475 0741 2
476 0742 2 IF( (DST_NEXT_ADDR = .DST_BEGIN_ADDR) EQL 1 )
477 0743 2 THEN
478 0744 2 RETURN(FALSE);
479 0745 2 RETURN(TRUE);
480 0746 2 END;
481 0747 2
482 0748 2 !++
```


TBKINT
V04-000

M 15
16-Sep-1984 02:15:19
14-Sep-1984 13:20:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[TRACE.SRC]TBKINT.B32;1
Page 14
(6)

```

: 483      0749      2
: 484      0750      2
: 485      0751      2
: 486      0752      2
: 487      0753      2
: 488      0754      2
: 489      0755      2
: 490      0756      2
: 491      0757      2
: 492      0758      2
: 493      0759      2
: 494      0760      2
: 495      0761      1

```

```

! get_dst_rec does most of the work - this routine just
! includes the side effect described above.
!--
rec_addr = tbk$get_dst_rec (.rec_id);
IF .rec_addr EQL 0
THEN RETURN 0;

!++
! Re-initialize the notion of 'next' DST record.
!--
dst_next_addr = .rec_addr + .rec_addr [dst$b_length] + 1;
RETURN .rec_addr
END;
```

			0000	00000	.ENTRY	TBK\$POSITON_DST, Save nothing	: 0692
		04	AC	D5 00002	TSTL	REC_ID	: 0734
			13	12 00005	BNEQ	1\$	
0000'	50	0000'	CF	D0 00007	MOVL	DST_BEGIN_ADDR, R0	: 0742
	CF		50	D0 0000C	MOVL	R0, DST_NEXT_ADDR	
	01		50	D1 00011	CMPL	R0, #1	
			1A	13 00014	BEQL	2\$	
	50		01	D0 00016	MOVL	#1, R0	: 0745
				04 00019	RET		
		04	AC	DD 0001A 1\$:	PUSHL	REC_ID	: 0752
A9	AF		01	FB 0001D	CALLS	#1, TBK\$GET_DST_REC	: 0753
			50	D5 00021	TSTL	REC_ADDR	
			0B	13 00023	BEQL	2\$	
	51		60	9A 00025	MOVZBL	(REC_ADDR), R1	: 0759
0000'	CF	01	A140	9E 00028	MOVAB	1(R1)[REC_ADDR], DST_NEXT_ADDR	
				04 0002F	RET		: 0760
		50	D4	00030 2\$:	CLRL	R0	: 0761
			04	00032	RET		

; Routine Size: 51 bytes, Routine Base: TBK\$CODE + 00F6

```

: 497      0762 1 END
: 498      0763 0 ELUDOM

```

N 15
16-Sep-1984 02:15:19
14-Sep-1984 13:20:19

VAX-11 Bliss-32 V4.0-742 Page 15
DISK\$VMSMASTER:[TRACE.SRC]TBKINT.B32;1 (7)

```
!End of module
```

PSECT SUMMARY										
Name	Bytes	Attributes								
TBK\$OWN	160	NOVEC,	WRT,	RD	NOEXE,	NOSHR,	LCL,	REL,	CON,	PIC,ALIGN(2)
TBK\$PLIT	4	NOVEC,	NOWRT,	RD	EXE,	SHR,	LCL,	REL,	CON,	PIC,ALIGN(0)
TBK\$CODE	297	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	28	0	1000	00:01.8
-\$255\$DUA28:[TRACE.OBJ]TBKLIB.L32;1	157	2	1	14	00:00.3
-\$255\$DUA28:[TRACE.OBJ]STRUCDEF.L32;1	321	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$:TBKINT/OBJ=OBJ$:TBKINT MSRC$:TBKINT/UPDATE=(ENHS$:TBKINT)
;

```

```
: Size:          297 code + 164 data bytes
: Run Time:      00:15.1
: Elapsed Time:  00:55.5
: Lines/CPU Min: 3037
: Lexemes/CPU-Min: 28224
: Memory Used:   140 pages
: Compilation Complete
```

BCDEFGHIJKLMNOPQRSTUVWXYZ

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

