

RRRRRRRRRRRR	UUU	UUU	NNN	NNN	000000000	FFFFFFF	FFFFFFF
RRRRRRRRRRRR	UUU	UUU	NNN	NNN	000000000	FFFFFFF	FFFFFFF
RRRRRRRRRRRR	UUU	UUU	NNN	NNN	000000000	FFFFFFF	FFFFFFF
RRR	RRR	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	NNNNNN	NNN	000	FFF	FFF
RRR	RRR	UUU	NNNNNN	NNN	000	FFF	FFF
RRR	RRR	UUU	NNNNNN	NNN	000	FFF	FFF
RRRRRRRRRRRR	UUU	UUU	NNN	NNN	000	FFFFFFF	FFFFFFF
RRRRRRRRRRRR	UUU	UUU	NNN	NNN	000	FFFFFFF	FFFFFFF
RRRRRRRRRRRR	UUU	UUU	NNN	NNN	000	FFFFFFF	FFFFFFF
RRR	RRR	UUU	NNN	NNNNNN	000	FFF	FFF
RRR	RRR	UUU	NNN	NNNNNN	000	FFF	FFF
RRR	RRR	UUU	NNN	NNNNNN	000	FFF	FFF
RRR	RRR	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUU	NNN	NNN	000	FFF	FFF
RRR	RRR	UUUUUUUUUUUUUUUU	NNN	NNN	000000000	FFF	FFF
RRR	RRR	UUUUUUUUUUUUUUUU	NNN	NNN	000000000	FFF	FFF
RRR	RRR	UUUUUUUUUUUUUUUU	NNN	NNN	000000000	FFF	FFF

```
PPPPPPPP  UU      UU  TTTTTTTTTT  TTTTTTTTTT  000000  CCCCCCCC
PPPPPPPP  UU      UU  TTTTTTTTTT  TTTTTTTTTT  000000  CCCCCCCC
PP        PP  UU      UU      TT      TT      00      00  CC
PP        PP  UU      UU      TT      TT      00      00  CC
PP        PP  UU      UU      TT      TT      00      00  CC
PP        PP  UU      UU      TT      TT      00      00  CC
PPPPPPPP  UU      UU      TT      TT      00      00  CC
PPPPPPPP  UU      UU      TT      TT      00      00  CC
PP        UU      UU      TT      TT      00      00  CC
PP        UU      UU      TT      TT      00      00  CC
PP        UU      UU      TT      TT      00      00  CC
PP        UU      UU      TT      TT      00      00  CC
PP        UU      UU      TT      TT      00      00  CC
PP        UU      UU      TT      TT      00      00  CC
UUUUUUUUUU  TT      TT      000000  CCCCCCCC
UUUUUUUUUU  TT      TT      000000  CCCCCCCC
```

```
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 %TITLE 'Puttoc - routines to pass binary information to binary files'
2 0002 0 MODULE PUTTOC ( IDENT = 'V04-000'
3 P 0003 0 %BLISS32 [ , ADDRESSING_MODE ( EXTERNAL = LONG_RELATIVE,
4 0004 0 NONEXTERNAL = LONG_RELATIVE) ]
5 0005 0 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY: DSR (Digital Standard RUNOFF) / DSRPLUS
33 0033 1
34 0034 1 ABSTRACT: Routines to save table of contents information in a file.
35 0035 1
36 0036 1
37 0037 1 ENVIRONMENT: Transportable
38 0038 1
39 0039 1 AUTHOR: R.W.Friday CREATION DATE: May, 1979
40 0040 1
```

```

42 0041 1 %SBTTL 'Revision History'
43 0042 1
44 0043 1   MODIFIED BY:
45 0044 1
46 0045 1       014   KFA00014   Ken Alden   19-Jul-1983
47 0046 1           Conditionalized the internal logic errors to check for
48 0047 1           /AUTO conditions.
49 0048 1
50 0049 1       013   KAD00013   Keith Dawson 28-June-1983
51 0050 1           Fix bug in which .SEND TOC 5 would get an additional RGH
52 0051 1           written.
53 0052 1
54 0053 1       012   KAD00012   Keith Dawson 11-May-1983
55 0054 1           Disable sending .HEADER {example, figure, table} records to
56 0055 1           FLIP.
57 0056 1
58 0057 1       011   KAD00011   Keith Dawson 11-April-1983
59 0058 1           Added support for new termination error messages for
60 0059 1           information written to .BRN file. This involved adding
61 0060 1           another formal to the RGH routine; this third formal is
62 0061 1           TRUE or FALSE, as the caller determines whether or not to
63 0062 1           increment the count of information written to the .BRN file.
64 0063 1
65 0064 1       010   KAD00010   Keith Dawson 23-March-1983
66 0065 1           Changed GCA_FLIP bit to (.gca_op_dev EQL op_dev_flip).
67 0066 1
68 0067 1       009   KAD00009   Keith Dawson 20-Mar-1983
69 0068 1           Removed LN01 conditionals and all references to .BIX
70 0069 1           and .BTC files.
71 0070 1
72 0071 1       008   KAD00008   Keith Dawson 07-Mar-1983
73 0072 1           Global edit of all modules. Updated module names, idents,
74 0073 1           copyright dates. Changed require files to BLISS library.
75 0074 1
76 0075 1   --
```



```

: 78      0076 1 %SBTTL 'Module Level Declarations'
: 79      0077 1
: 80      0078 1
: 81      0079 1 | TABLE OF CONTENTS:
: 82      0080 1 |
: 83      0081 1 FORWARD ROUTINE
: 84      0082 1     PUTRTY : NOVALUE,      | Writes record-type information to TOC binary file
: 85      0083 1     PUTTXT : NOVALUE,      | " parsed text "
: 86      0084 1     PUTTPG : NOVALUE,      | " page number "
: 87      0085 1     PUTCNT : NOVALUE,      | " counter and display descriptor "
: 88      0086 1     PUTHLI : NOVALUE;      | " header level counters, display code "
: 89      0087 1
: 90      0088 1
: 91      0089 1 | EQUATED SYMBOLS:
: 92      0090 1 |
: 93      0091 1
: 94      0092 1
: 95      0093 1 | INCLUDE FILES:
: 96      0094 1 |
: 97      0095 1
: 98      0096 1 REQUIRE 'REQ:RNODEF.REQ';    ! RUNOFF definition
: 99      0227 1
: 100     0228 1 LIBRARY 'NXPORT:XPORT';      ! XPORT Library
: 101     0229 1
: 102     U 0230 1 %IF DSRPLUS %THEN
: 103     U 0231 1 LIBRARY 'REQ:DPLLIB';      ! DSRPLUS BLISS Library
: 104     0232 1 %ELSE
: 105     0233 1 LIBRARY 'REQ:DSRLIB';      ! DSR BLISS Library
: 106     0234 1 %FI
: 107     0235 1
: 108     0236 1
: 109     0237 1 | MACROS:
: 110     0238 1
: 111     0239 1 MACRO
: 112     M 0240 1     CHECK_OPEN =
: 113     M 0241 1 %IF FLIP %THEN
: 114     M 0242 1     (IF (.gca_op_dev EQL op_dev_flip)
: 115     M 0243 1     THEN
: 116     M 0244 1     .RNOIOB[IOB$V_OPEN]
: 117     M 0245 1     ELSE
: 118     M 0246 1     .BRNOOB[IOB$V_OPEN])
: 119     M 0247 1 %ELSE
: 120     M 0248 1     (.BRNOOB[IOB$V_OPEN])
: 121     M 0249 1 %FI
: 122     0250 1     %;
: 123     0251 1
: 124     0252 1
: 125     0253 1 | EQUATED SYMBOLS:
: 126     0254 1 |
: 127     0255 1
: 128     0256 1 | OWN STORAGE:
: 129     0257 1 |
: 130     0258 1
: 131     0259 1
: 132     0260 1 | EXTERNAL REFERENCES:
: 133     0261 1 |
: 134     0262 1
```

PUTTOC
V04-000

Puttoc - routines to pass binary information to
Module Level Declarations

B 12

16-Sep-1984 01:30:33
14-Sep-1984 13:07:49

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]PUTTOC.BLI;1

Page 4
(3)

```
: 135
: 136
: 137
: 138
: 139
: 140
: 141
: 142
: 143
: 144
: 145
: 146
: 147
: 148
: 149
: 150
: 151
: 152
: 153

U 0263 1 %IF FLIP %THEN
U 0264 1 EXTERNAL
U 0265 1 RNOIOB : REF $XPO_IOB ();
0266 1 %FI
0267 1
0268 1 EXTERNAL
0269 1 BRNOOB : $XPO_IOB ();
0270 1
0271 1 EXTERNAL
0272 1 GCA : GCA_DEFINITION,
0273 1 HLDSP : VECTOR [MAX_LEVELS],
0274 1 HLLIST : COUNTED_LIST [MAX_LEVELS];
0275 1
0276 1 EXTERNAL LITERAL
0277 1 RNFILE;
0278 1
0279 1 EXTERNAL ROUTINE
0280 1 RGH,
0281 1 ERMS;
```

!Error messages


```
155 0282 1 GLOBAL ROUTINE PUTRTY (MAJOR_CODE, MINOR_CODE) : NOVALUE =
156 0283 1
157 0284 1 !++
158 0285 1 FUNCTIONAL DESCRIPTION:
159 0286 1
160 0287 1 PUTRTY writes a record descriptor to the table of contents file
161 0288 1
162 0289 1 FORMAL PARAMETERS:
163 0290 1
164 0291 1 MAJOR_CODE is the major record type
165 0292 1 MINOR_CODE is the minor record type
166 0293 1
167 0294 1 IMPLICIT INPUTS: None
168 0295 1
169 0296 1 IMPLICIT OUTPUTS: None
170 0297 1
171 0298 1 ROUTINE VALUE:
172 0299 1 COMPLETION CODES: None
173 0300 1
174 0301 1 SIDE EFFECTS: None
175 0302 1
176 0303 1 --
177 0304 1
178 0305 2 BEGIN
179 0306 2
180 0307 2 LOCAL
181 0308 2 TEMP_RECORD : VECTOR [2];
182 U 0309 2 %IF FLIP %THEN
183 U 0310 2 LOCAL
184 U 0311 2 FLIP_RECORD : $FLIP_TOCRD;
185 0312 2 %FI
186 0313 2
187 0314 2 !Cursory check to make sure the file is opened.
188 0315 3 IF NOT check_open
189 0316 2 THEN
190 0317 3 BEGIN
191 0318 3 IF NOT .gca_black_box
192 0319 3 THEN
193 0320 3 ERMS (RNFILE, CH$PTR (UPLIT ('PUTRTY')), 6);
194 0321 3 RETURN;
195 0322 2 END;
196 0323 2
197 U 0324 2 %IF FLIP %THEN
198 U 0325 2 IF (.gca_op_dev EQL op_dev_flip)
199 U 0326 2 THEN
200 U 0327 2 BEGIN
201 U 0328 2 !Create a two-word descriptor record.
202 U 0329 2 FLIP_RECORD [TOCRD-MAJOR] = .MAJOR_CODE;
203 U 0330 2 FLIP_RECORD [TOCRD-MINOR] = .MINOR_CODE;
204 U 0331 2 FLIP_RECORD [TOCRD-CODE] = FLIP$K_TOCRD;
205 U 0332 2 $XPO-PUT (IOB=.RN$IOB, STRING=(FLIP$K_TOCRD_SIZE,CH$PTR(FLIP_RECORD)));
206 U 0333 2 RETURN
207 U 0334 2 END;
208 0335 2 %FI
209 0336 2
210 0337 2 !If this is an Index Record Group, write the Record Group Header for it.
211 0338 2 !The length of the described record is 2. This special case exists in
```

```
212 0339 2 !PUTRTY itself because it is called from outside this module -- namely,  
213 0340 2 !from DOPX.  
214 0341 2 ! Do count this record in the .BRN count.  
215 0342 2 IF (.major_code EQL maj_runoff ! Fix .STC 5 bug.  
216 0343 2 AND  
217 0344 2 .minor_code EQL min_index)  
218 0345 2 THEN  
219 0346 2 rgh (brn_contents, 2, true);  
220 0347 2  
221 0348 2 !Create a two-word descriptor record.  
222 0349 2 TEMP_RECORD [0] = .MAJOR_CODE;  
223 0350 2 TEMP_RECORD [1] = .MINOR_CODE;  
224 0351 2  
225 0352 2 !Now write the record type descriptor record.  
226 0353 2 $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (2, TEMP_RECORD));  
227 0354 2  
228 0355 2 RETURN;  
229 0356 2  
230 0357 1 END;
```

!End of PUTRTY

```
.TITLE PUTTOC Puttoc - routines to pass binary informa  
tion to  
.IDENT \V04-000\  
.PSECT $PLITS$,NOWRT,NOEXE,2  
00 00 59 54 52 54 55 50 00000 P.AAA: .ASCII \PUTRTY\<0><0>  
; .EXTRN BRNOOB, GCA, HLDSP  
; .EXTRN HLLIST, RNFILE, RGH  
; .EXTRN ERMS, XPO$PUT, XPO$FAILURE  
; .PSECT $CODE$,NOWRT,2  
; .ENTRY PUTRTY, Save R2  
MOVAB BRNOOB+48, R2  
SUBL2 #16, SP  
BLBS BRNOOB+50, 1$  
BLBS GCA+228, 3$  
PUSHL #6  
PUSHAB P.AAA  
PUSHL #RNFILE  
CALLS #3, ERMS  
RET  
CMPL MAJOR_CODE, #2  
BNEQ 2$  
CMPL MINOR_CODE, #5  
BNEQ 2$  
PUSHL #1  
PUSHL #2  
PUSHL #2  
CALLS #3, RGH  
MOVQ MAJOR_CODE, TEMP_RECORD  
MOVW #8, $IOB$OUTPUT  
MOVB #2, $IOB$OUTPUT+2  
MOVB #1, $IOB$OUTPUT+3  
; 0282  
; 0315  
; 0318  
; 0320  
; 0317  
; 0342  
; 0344  
; 0346  
; 0349  
; 0353
```



```
; Routine Size: 118 bytes,    Routine Base: $CODE$ + 0000
```

```
232 0358 1 GLOBAL ROUTINE PUTTXT (TEXT_LENGTH, TEXT_PTR, MAJOR_CODE, MINOR_CODE) : NOVALUE = !
233 0359 1
234 0360 1 ++
235 0361 1 FUNCTIONAL DESCRIPTION:
236 0362 1
237 0363 1 PUTTXT writes parsed text to the table of contents file.
238 0364 1
239 0365 1 FORMAL PARAMETERS:
240 0366 1
241 0367 1 TEXT_LENGTH is the number of bytes representing the text.
242 0368 1 TEXT_ADDRESS is the address of the text.
243 0369 1 MAJOR_CODE and MINOR code are the major and minor record descriptors
244 0370 1 associated with the text.
245 0371 1
246 0372 1 IMPLICIT INPUTS: None
247 0373 1
248 0374 1 IMPLICIT OUTPUTS: None
249 0375 1
250 0376 1 ROUTINE VALUE:
251 0377 1 COMPLETION CODES: None
252 0378 1
253 0379 1 SIDE EFFECTS: None
254 0380 1
255 0381 1 --
256 0382 1
257 0383 2 BEGIN
258 0384 2
259 0385 2 LOCAL
260 0386 2 TEMP_RECORD : VECTOR [1000];
261 U 0387 2 %IF FLIP %THEN
262 U 0388 2 LOCAL
263 U 0389 2 FLIP_RECORD : $FLIP_TOCTXT;
264 U 0390 2 %FI
265 0391 2
266 0392 2 !Cursory check to make sure the file is opened.
267 0393 2 IF NOT check_open
268 0394 2 THEN
269 0395 2 BEGIN
270 0396 2 IF NOT .gca_black_box
271 0397 2 THEN
272 0398 2 ERMS (RNFILE, CH$PTR (UPLIT ('PUTTXT')), 6);
273 0399 2 RETURN;
274 0400 2 END;
275 0401 2
276 U 0402 2 %IF FLIP %THEN
277 U 0403 2 IF (.gca_op_dev EQL op_dev_flip)
278 U 0404 2 THEN
279 U 0405 2 BEGIN
280 U 0406 2 FLIP_RECORD[TOCTXT_CODE] = FLIP$K TOCTXT;
281 U 0407 2 FLIP_RECORD[TOCTXT_MAJOR] = .MAJOR_CODE;
282 U 0408 2 FLIP_RECORD[TOCTXT_MINOR] = .MINOR_CODE;
283 U 0409 2 FLIP_RECORD[TOCTXT_LENGTH] = .TEXT_LENGTH;
284 U 0410 2 CH$MOVE (.TEXT_LENGTH, .TEXT_PTR, CH$PTR (FLIP_RECORD[TOCTXT_TEXT]));
285 U 0411 2 $XPO_PUT ( IOB=.RNOIOB, STRING=(.TEXT_LENGTH+FLIP$K_TOCTXT_BASISIZ,
286 U 0412 2 CH$PTR(FLIP_RECORD)));
287 U 0413 2 RETURN
288 U 0414 2 END;
```



```

: 289      0415 2 %FI
: 290      0416 2
: 291      0417 2 !Write the Record Group Header for this TOC record. The length of the
: 292      0418 2 !described record is 3 more than the allocation required for the text --
: 293      0419 2 !2 fullwords of TOC header information (PUTRTY) and 1 fullword of size.
: 294      0420 2 ! Do count this record in the .BRN count.
: 295      0421 2
: 296      0422 2 rgh (brn_contents, 3 + CH$ALLOCATION (.text_length), true);
: 297      0423 2
: 298      0424 2 !Write out the descriptor record
: 299      0425 2 PUTRTY (.MAJOR_CODE, .MINOR_CODE);
: 300      0426 2
: 301      0427 2 !Write out the number of characters of text.
: 302      0428 2 $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (1, TEXT_LENGTH));
: 303      0429 2
: 304      0430 2 !Only write text if there's some to write.
: 305      0431 2 IF .TEXT_LENGTH LEQ 0
: 306      0432 2 THEN
: 307      0433 2 !No text to write
: 308      0434 2 RETURN;
: 309      0435 2
: 310      0436 2 !Copy the text into an area guaranteed to start on a word boundary.
: 311      0437 2 CH$MOVE (.TEXT_LENGTH, .TEXT_PTR, CH$PTR (TEMP_RECORD));
: 312      0438 2
: 313      0439 2 !Write out the contents entry.
: 314      0440 2 $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (CH$ALLOCATION (.TEXT_LENGTH), TEMP_RECORD));
: 315      0441 2
: 316      0442 1 END;
                                !End of PUTTXT
```

.PSECT \$PLITS\$,NOWRT,NOEXE,2

00 00 54 58 54 54 55 50 00008 P.AAB: .ASCII \PUTTXT\<0><0>

.PSECT \$CODE\$,NOWRT,2

				03FC 00000	.ENTRY PUTTXT, Save R2,R3,R4,R5,R6,R7,R8,R9	: 0358
59	00000000G	EF	9E	00002	MOVAB XPOS\$PUT, R9	:
58	00000000G	EF	9E	00009	MOVAB XPOS\$FAILURE, R8	:
57	00000000G	EF	9E	00010	MOVAB IOB\$+68, R7	:
5E	F058	CE	9E	00017	MOVAB -4008(SP), SP	:
1E	EE	A7	E8	0001C	BLBS BRNOOB+50, 2\$	: 0393
01	00000000G	EF	E9	00020	BLBC GCA+228, 1\$	: 0396
				04 00027	RET	:
				06 DD 00028 1\$:	PUSHL #6	: 0398
	00000000'	EF	9F	0002A	PUSHAB P.AAB	:
	00000000G	8F	DD	00030	PUSHL #RNF FILE	:
00000000G	EF	03	FB	00036	CALLS #3, ERMS	:
				04 0003D	RET	: 0395
56	04	AC		01 DD 0003E 2\$:	PUSHL #1	: 0422
		56		03 C1 00040	ADDL3 #3, TEXT_LENGTH, R6	:
				04 C6 00045	DIVL2 #4, R6	:
		03	A6	9F 00048	PUSHAB 3(R6)	:
			02	DD 0004B	PUSHL #2	:

PUTTOC
V04-000

Puttoc - routines to pass binary information to
Module Level Declarations

H 12
16-Sep-1984 01:30:33
14-Sep-1984 13:07:49

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]PUTTOC.BLI;1

Page 10
(5)

00000000G	EF		03	FB	0004D	CALLS	#3, RGH		
	7E		AC	7D	00054	MOVQ	MAJOR CODE, -(SP)		0425
FF2D	CF		02	FB	00058	CALLS	#2, POTRTY		
	6E		04	B0	0005D	MOVW	#4, \$IOB\$OUTPUT		0428
02	AE		02	90	00060	MOVB	#2, \$IOB\$OUTPUT+2		
03	AE		01	90	00064	MOVB	#1, \$IOB\$OUTPUT+3		
04	AE		AC	9E	00068	MOVAB	TEXT_LENGTH, \$IOB\$OUTPUT+4		
	67		6E	9E	0006D	MOVAB	\$IOB\$OUTPUT, IOB\$+68		
E8	A7		07	90	00070	MOVB	#7, IOB\$+44		
			58	DD	00074	PUSHL	R8		
			7E	D4	00076	CLRL	-(SP)		
			A7	9F	00078	PUSHAB	IOB\$		
	69		03	FB	0007B	CALLS	#3, XPO\$PUT		
			AC	D5	0007E	TSTL	TEXT_LENGTH		0431
			29	15	00081	BLEQ	3\$		
08	AE	08	AC	28	00083	MOVW3	TEXT_LENGTH, @TEXT_PTR, TEMP_RECORD		0437
6E			04	A5	0008A	MULW3	#4, R6, \$IOB\$OUTPUT		0440
			02	90	0008E	MOVB	#2, \$IOB\$OUTPUT+2		
			01	90	00092	MOVB	#1, \$IOB\$OUTPUT+3		
			AE	9E	00096	MOVAB	TEMP_RECORD, \$IOB\$OUTPUT+4		
			6E	9E	0009B	MOVAB	\$IOB\$OUTPUT, IOB\$+68		
			07	90	0009E	MOVB	#7, IOB\$+44		
			58	DD	000A2	PUSHL	R8		
			7E	D4	000A4	CLRL	-(SP)		
			A7	9F	000A6	PUSHAB	IOB\$		
			03	FB	000A9	CALLS	#3, XPO\$PUT		
			04	000AC	3\$:	RET			0442

; Routine Size: 173 bytes, Routine Base: \$CODE\$ + 0076


```
318 0443 1 GLOBAL ROUTINE PUTTPG (PAGE_REF, REC_TYPE) : NOVALUE =
319 0444 1
320 0445 1 ++
321 0446 1 FUNCTIONAL DESCRIPTION:
322 0447 1
323 0448 1 PUTTPG writes a page number into the table of contents file
324 0449 1
325 0450 1 FORMAL PARAMETERS:
326 0451 1
327 0452 1 PAGE_REF is the address of a page-number block.
328 0453 1
329 0454 1 REC_TYPE is a record type, used by FLIP (only). This
330 0455 1 argument may be -1 if the call is made from DSR,
331 0456 1 not from DSRPLUS; in this case the argument is
332 0457 1 unused.
333 0458 1
334 0459 1 IMPLICIT INPUTS: None
335 0460 1
336 0461 1 IMPLICIT OUTPUTS: None
337 0462 1
338 0463 1 ROUTINE VALUE:
339 0464 1 COMPLETION CODES: None
340 0465 1
341 0466 1 SIDE EFFECTS: None
342 0467 1 --
343 0468 1
344 0469 2 BEGIN
345 0470 2
346 0471 2 MAP
347 0472 2 PAGE_REF : REF VECTOR [PAGE_SCT_SIZE];
348 0473 2
349 0474 2 LOCAL
350 0475 2 TEMP_RECORD : VECTOR [PAGE_SCT_SIZE];
351 U 0476 2 %IF FLIP %THEN
352 UU 0477 2 LOCAL
353 U 0478 2 FLIP_RECORD : $FLIP_TOCPAG;
354 0479 2 %FI
355 0480 2
356 0481 2 !Cursory check to make sure the file is opened.
357 0482 3 IF NOT CHECK_OPEN
358 0483 2 THEN
359 0484 3 BEGIN
360 0485 3 IF NOT .gca_black_box
361 0486 3 THEN
362 0487 3 ERMS (RNFILE, CH$PTR (UPLIT ('PUTTPG')), 6);
363 0488 3 RETURN;
364 0489 2 END;
365 0490 2
366 U 0491 2 %IF FLIP %THEN
367 UU 0492 2 IF (.gca_op_dev EQL op_dev_flip)
368 UU 0493 2 THEN
369 UU 0494 2 BEGIN
370 UU 0495 2 FLIP_RECORD [TOCPAG CODE] = .REC TYPE;
371 UU 0496 2 INCR I FROM 1 TO PAGE_SCT_SIZE DO
372 U 0497 2 VECTOR [FLIP_RECORD[TOCPAG PAGENO], I - 1] = .PAGE_REF [I - 1];
373 U 0498 2 $XPO_PUT( IOB=.RNOIOB, STRING=(FLIP$K_TOCPAG_SIZE,
374 U 0499 2 CH$PTR(FLIP_RECORD)));
```

```
: 375      U 0500 2      RETURN
: 376      U 0501 2      END;
: 377      0502 2      %FI
: 378      0503 2
: 379      0504 2      !Write the Record Group Header for this TOC record. The length of the
: 380      0505 2      !described record is 2 more than the allocation required for the page
: 381      0506 2      !number -- 2 fullwords of TOC header information (PUTRTY).
: 382      0507 2      ! Do not count this record in the .BRN count.
: 383      0508 2
: 384      0509 2      rgh (brn_contents, 2 + page_sct_size, false);
: 385      0510 2
: 386      0511 2      !Write a descriptor record identifying what's to come as a page number
: 387      0512 2      PUTRTY (MAJ_RUNOFF, MIN_PAGE);
: 388      0513 2
: 389      0514 2      !Copy the page number in too.
: 390      0515 2      INCR I FROM 1 TO PAGE_SCT_SIZE DO
: 391      0516 2      TEMP_RECORD [I - 1] = .PAGE_REF [I - 1];
: 392      0517 2
: 393      0518 2      $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (PAGE_SCT_SIZE, TEMP_RECORD));
: 394      0519 2
: 395      0520 1      END;                                !End of PUTTPG
```

```
.PSECT $PLIT$,NOWRT,NOEXE,2
00 00 47 50 54 54 55 50 00010 P.AAC: .ASCII \PUTTPG\<0><0>
```

```
.PSECT $CODE$,NOWRT,2
                                0004 00000
                                EF 9E 00002
52 00000000G 18 C2 00009
5E          02 A2 E8 0000C
1D          06 EF E8 00010
67 00000000G 06 DD 00017
                                EF 9F 00019
                                8F DD 0001F
00000000G EF 03 FB 00025
                                04 0002C
7E          06 7D 0002D 1$:
                                02 DD 00030
00000000G EF 03 FB 00032
                                01 DD 00039
                                02 DD 0003B
FE9B CF      02 FB 0003D
51      01 D0 00042
50      04 BC41 DE 00045 2$:
04 AE41 FC A0 D0 0004A
F1      51      04 F3 00050
6E      10 B0 00054
02 AE      02 90 00057
03 AE      01 90 0005B
04 AE      08 AE 9E 0005F
14 A2      6E 9E 00064

.ENTRY PUTTPG, Save R2
MOVAB BRNOOB+48, R2
SUBL2 #24, SP
BLBS BRNOOB+50, 1$
BLBS GCA+228, 3$
PUSHL #6
PUSHAB P.AAC
PUSHL #RNFILE
CALLS #3, ERMS
RET
MOVQ #6, -(SP)
PUSHL #2
CALLS #3, RGH
PUSHL #1
PUSHL #2
CALLS #2, PUTRTY
MOVL #1, I
MOVAL @PAGE_REF[I], R0
MOVL -4(R0), TEMP_RECORD-4[I]
AOBLEQ #4, I, 2$
MOVW #16, $IOB$OUTPUT
MOVB #2, $IOB$OUTPUT+2
MOVB #1, $IOB$OUTPUT+3
MOVAB TEMP_RECORD, $IOB$OUTPUT+4
MOVAB $IOB$OUTPUT, IOB$+68
```

0443
0482
0485
0487
0484
0509
0512
0516
0518

Puttoc - routines to pass binary information to
Module Level Declarations

K 12
16-Sep-1984 01:30:33
14-Sep-1984 13:07:49

```
VAX-11 Bliss-32 V4.0-742
[ RUNOFF.SRC ] PUT TOC.BLI;1
```

Page 13
(6)

```

FC      A2      07  90  00068
          00000000G EF  9F  0006C
                        7E  D4  00072
                        A2  9F  00074
00000000G EF      D0  03  FB  00077
                                04  0007E 3$:

```

MOV B #7, IOBS+44
PUSH AB
CLRL
PUSH AB
CALLS
BC

0520

```
; Routine Size: 127 bytes,    Routine Base: $CODE$ + 0123
```

```
397 0521 1 GLOBAL ROUTINE PUTCNT (
398 0522 1 !
399 0523 1 MAJOR_CODE, MINOR_CODE, COUNTER_VALUE, COUNTER_DISPLAY_CODE,
400 0524 1 PRE_LEN, PRE_PTR, POST_LEN, POST_PTR
401 0525 1 !
402 0526 1 ) : NOVALUE =
403 0527 1
404 0528 1 ++
405 0529 1 FUNCTIONAL DESCRIPTION:
406 0530 1
407 0531 1 PUTCNT writes a counter and its display descriptor to the table of
408 0532 1 contents file. In addition it writes pre-counter and post-counter
409 0533 1 strings if they are not null.
410 0534 1
411 0535 1 FORMAL PARAMETERS:
412 0536 1
413 0537 1 MAJOR_CODE and MINOR_CODE identify the counter to the table of contents program.
414 0538 1 COUNTER_VALUE and COUNTER_DISPLAY_CODE are the counter's value and display code.
415 0539 1 PRE_LEN and PRE_PTR describe the pre-counter string.
416 0540 1 POST_LEN and POST_PTR describe the post-counter string.
417 0541 1
418 0542 1 IMPLICIT INPUTS: None
419 0543 1
420 0544 1 IMPLICIT OUTPUTS: None
421 0545 1
422 0546 1 ROUTINE VALUE:
423 0547 1 COMPLETION CODES: None
424 0548 1
425 0549 1 SIDE EFFECTS: None
426 0550 1 --
427 0551 1
428 0552 2 BEGIN
429 0553 2
430 0554 2 LOCAL
431 0555 2 TEMP_RECORD : VECTOR [5];
432 0556 2
433 0557 2 !If this call is for a header-level, call the appropriate routine and quit.
434 0558 2 IF .MINOR_CODE EQL MIN_HL_INF
435 0559 2 THEN
436 0560 3 BEGIN
437 0561 3 PUTHLI ();
438 0562 3 RETURN;
439 0563 2 END;
440 0564 2
441 0565 2 !** Temporary restriction -- remove to enable sending DSRPLUS records to FLIP
442 0566 3 IF (.gca_op_dev EQL op_dev_flip)
443 0567 2 THEN
444 0568 2 RETURN;
445 0569 2 !** End of temporary restriction
446 0570 2
447 0571 2 !Cursor check to make sure the file is opened.
448 0572 3 IF NOT CHECK_OPEN
449 0573 2 THEN
450 0574 3 BEGIN
451 0575 3 IF NOT .gca_black_box
452 0576 3 THEN
453 0577 3 ERMS (RNFILE, CH$PTR (UPLIT ('PUTCNT')), 6);
```



```

454 0578 3      RETURN;
455 0579      END;
456 0580
457 0581      !
458 0582      !Write the Record Group Header for this TOC record. The length of the
459 0583      !described record is:
460 0584      ! 2 (TOC header, from PUTRTY) plus
461 0585      ! 2 (number of fullwords needed to save the parameters) plus
462 0586      ! 1 + ch$allocation (pre-counter string)
463 0587      ! 1 + ch$allocation (post-counter string)
464 0588      !
465 0589      ! Do not count this record in the .BRN count.
466 0590      rgh (brn contents, 2 + 2 + 2 +
467 0591          CH$ALLOCATION (.pre_len) +
468 0592          CH$ALLOCATION (.post_len)
469 0593          , false
470 0594      );
471 0595
472 0596      !
473 0597      !Write a descriptor record using the supplied descriptive information.
474 0598      PUTRTY (.MAJOR_CODE, .MINOR_CODE);
475 0599
476 0600      !First write out the counter value.
477 0601      $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (1, COUNTER_VALUE));
478 0602
479 0603      !Now write the display information
480 0604      $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (1, COUNTER_DISPLAY_CODE));
481 0605
482 0606      !
483 0607      !Write the length of the pre-counter string.
484 0608      $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (1, PRE_LEN));
485 0609
486 0610      !If that length is not zero, write the string too.
487 0611      IF .PRE_LEN GTR 0
488 0612      THEN
489 0613          BEGIN
490 0614              !Copy the string to an area guaranteed to start on a word boundary.
491 0615              CH$MOVE (.PRE_LEN, .PRE_PTR, CH$PTR (TEMP_RECORD));
492 0616              !Now write the string.
493 0617              $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (CH$ALLOCATION (.PRE_LEN), TEMP_RECORD));
494 0618              END;
495 0619
496 0620      !
497 0621      !Finally, write the length of the post-counter string.
498 0622      $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (1, POST_LEN));
499 0623
500 0624      !If that length is not zero, write the string too.
501 0625      IF .POST_LEN GTR 0
502 0626      THEN
503 0627          BEGIN
504 0628              !Copy the string to an area guaranteed to start on a word boundary.
505 0629              CH$MOVE (.POST_LEN, .POST_PTR, CH$PTR (TEMP_RECORD));
506 0630              !Now write the string.
507 0631              $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (CH$ALLOCATION (.POST_LEN), TEMP_RECORD));
508 0632              END;
509 0633
510 0634      RETURN

```

: 511
: 5120635 2
0636 1 END;

!End of PUTCNT

```
.PSECT $SPLIT$,NOWRT,NOEXE,2
00 00 54 4E 43 54 55 50 00018 P.AAD: .ASCII \PUTCNT\<0><0> ;

.PSECT $CODE$,NOWRT,2
.ENTRY PUTCNT, Save R2,R3,R4,R5,R6,R7,R8,R9,R10 : 0521
MOVAB XPOS$PUT, R10
MOVAB XPOS$FAILURE, R9
MOVAB IOB$+68, R8
SUBL2 #28, SP
TSTL MINOR_CODE : 0558
BNEQ 1$
CALLS #0, PUTHLI : 0561
RET : 0560
CMPZV #4, #4, GCA+208, #2 : 0566
BNEQ 2$
RET
BLBS BRNOOB+50, 4$ : 0572
BLBC GCA+228, 3$ : 0575
RET
PUSHL #6 : 0577
PUSHAB P.AAD
PUSHL #RNFILE
CALLS #3, ERMS
RET : 0574
CLRL -(SP) : 0590
ADDL3 #3, PRE_LEN, R6 : 0591
DIVL2 #4, R6
ADDL3 #3, POST_LEN, R7 : 0592
DIVL2 #4, R7
PUSHAB 6(R7)[R6] : 0591
PUSHL #2 : 0590
CALLS #3, RGH
MOVQ MAJOR_CODE, -(SP) : 0598
CALLS #2, PUTRTY
MOVW #4, $IOB$OUTPUT : 0601
MOVB #2, $IOB$OUTPUT+2
MOVB #1, $IOB$OUTPUT+3
MOVAB COUNTER_VALUE, $IOB$OUTPUT+4
MOVAB $IOB$OUTPUT, IOB$+68
MOVB #7, IOB$+44
PUSHL R9
CLRL -(SP)
PUSHAB IOB$
CALLS #3, XPOS$PUT
MOVW #4, $IOB$OUTPUT : 0604
MOVB #2, $IOB$OUTPUT+2
MOVB #1, $IOB$OUTPUT+3
MOVAB COUNTER_DISPLAY_CODE, $IOB$OUTPUT+4
```

5A	00000000G	EF	9E	00002	
59	00000000G	EF	9E	00009	
58	00000000G	EF	9E	00010	
5E		1C	C2	00017	
		08	AC	D5	0001A
			08	12	0001D
00000000V	EF	00	FB	0001F	
			04	00026	
02 00000000G	EF	04	ED	00027	1\$:
			01	12	00030
			04	00032	
	1E	EE	A8	E8	00033
01 00000000G	EF		E9	00037	2\$:
			04	0003E	
			06	DD	0003F
	00000000'	EF	9F	00041	3\$:
	00000000G		8F	DD	00047
00000000G	EF		03	FB	0004D
			04	00054	
			7E	D4	00055
56	14	AC	03	C1	00057
		56	04	C6	0005C
57	1C	AC	03	C1	0005F
		57	04	C6	00064
			06	A746	9F 00067
			02	DD	0006B
00000000G	EF		03	FB	0006D
	7E	04	AC	7D	00074
FDE1	CF		02	FB	00078
	6E		04	B0	0007D
02	AE		02	90	00080
03	AE		01	90	00084
04	AE	0C	AC	9E	00088
	68		6E	9E	0008D
E8	A8		07	90	00090
			59	DD	00094
			7E	D4	00096
		BC	A8	9F	00098
	6A		03	FB	0009B
	6E		04	B0	0009E
02	AE		02	90	000A1
03	AE		01	90	000A5
04	AE	10	AC	9E	000A9

			E8	68 A8		6E 07 59 7E A8 03 04 02 03 04 E8	9E 90 DD D4 9F FB B0 90 90 AC 9E 07 59 7E A8 03 14 29 AC 04 02 01 AE 6E 07 59 7E A8 03 14 29 AC 04 02 01 AE 6E 07 59 7E A8 03 1C 6E 07 59 7E A8 03 1C 20 57 02 03 04 E8 6A BC 03 04 0015D	000AE 000B1 000B5 000B7 000B9 000BC 000BF 000C2 000C6 000CA 000CF 000D2 000D6 000D8 000DA 000DD 000E0 000E3 000E5 000EC 000F0 000F4 000F8 000FD 00100 00104 00106 00108 0010B 0010E 00111 00115 00119 0011E 00121 00125 00127 00129 0012C 0012F 00132 00134 0013B 0013F 00143 00147 0014C 0014F 00153 00155 00157 0015A 0015D	5\$: 6\$:	MOVAB MOVB PUSHL CLRL PUSHAB CALLS MOVW MOVB MOVB MOVAB MOVAB MOVB PUSHL CLRL PUSHAB CALLS TSTL BLEQ MOVW3 MULW3 MOVB MOVB MOVAB MOVAB MOVB PUSHL CLRL PUSHAB CALLS MOVW MOVB MOVB MOVAB MOVAB MOVB PUSHL CLRL PUSHAB CALLS TSTL BLEQ MOVW3 MULW3 MOVB MOVB MOVAB MOVAB MOVB PUSHL CLRL PUSHAB CALLS RET	\$IOB\$OUTPUT, IOB\$+68 #7, IOB\$+44 R9 -(SP) IOB\$ #3, XPOS\$PUT #4, \$IOB\$OUTPUT #2, \$IOB\$OUTPUT+2 #1, \$IOB\$OUTPUT+3 PRE_LEN, \$IOB\$OUTPUT+4 \$IOB\$OUTPUT, IOB\$+68 #7, IOB\$+44 R9 -(SP) IOB\$ #3, XPOS\$PUT PRE_LEN 5\$ PRE_LEN, @PRE_PTR, TEMP_RECORD #4, R6, \$IOB\$OUTPUT #2, \$IOB\$OUTPUT+2 #1, \$IOB\$OUTPUT+3 TEMP_RECORD, \$IOB\$OUTPUT+4 \$IOB\$OUTPUT, IOB\$+68 #7, IOB\$+44 R9 -(SP) IOB\$ #3, XPOS\$PUT #4, \$IOB\$OUTPUT #2, \$IOB\$OUTPUT+2 #1, \$IOB\$OUTPUT+3 POST_LEN, \$IOB\$OUTPUT+4 \$IOB\$OUTPUT, IOB\$+68 #7, IOB\$+44 R9 -(SP) IOB\$ #3, XPOS\$PUT POST_LEN 6\$ POST_LEN, @POST_PTR, TEMP_RECORD #4, R7, \$IOB\$OUTPUT #2, \$IOB\$OUTPUT+2 #1, \$IOB\$OUTPUT+3 TEMP_RECORD, \$IOB\$OUTPUT+4 \$IOB\$OUTPUT, IOB\$+68 #7, IOB\$+44 R9 -(SP) IOB\$ #3, XPOS\$PUT	0608 0611 0615 0617 0622 0625 0629 0631 0636
--	--	--	----	----------	--	--	---	---	--------------	--	--	--

; Routine Size: 350 bytes, Routine Base: \$CODE\$ + 01A2


```
514 0637 1 GLOBAL ROUTINE PUTHLI : NOVALUE =
515 0638 1
516 0639 1 ++
517 0640 1 FUNCTIONAL DESCRIPTION:
518 0641 1
519 0642 1 PUTHLI write header level information into the table of contents file.
520 0643 1
521 0644 1 FORMAL PARAMETERS: None
522 0645 1
523 0646 1 IMPLICIT INPUTS: None
524 0647 1
525 0648 1 IMPLICIT OUTPUTS: None
526 0649 1
527 0650 1 ROUTINE VALUE:
528 0651 1 COMPLETION CODES: None
529 0652 1
530 0653 1 SIDE EFFECTS: None
531 0654 1
532 0655 1 --
533 0656 1
534 0657 2 BEGIN
535 0658 2
536 U 0659 2 %IF FLIP %THEN
537 U 0660 2 LOCAL
538 U 0661 2 FLIP_RECORD: $FLIP_TOCHLI;
539 0662 2 %FI
540 0663 2
541 0664 2 !Cursory check to make sure the file is opened.
542 0665 3 IF NOT CHECK_OPEN
543 0666 2 THEN
544 0667 3 BEGIN
545 0668 3 IF NOT .gca_black_box
546 0669 3 THEN
547 0670 3 ERMS (RNFILE, CH$PTR (UPLIT ('PUTHLI')), 6);
548 0671 3 RETURN;
549 0672 2 END;
550 0673 2
551 U 0674 2 %IF FLIP %THEN
552 U 0675 2 IF (.gca_op_dev EQL op_dev_flip)
553 U 0676 2 THEN
554 U 0677 2 BEGIN
555 U 0678 2 FLIP_RECORD[TOCHLI_CODE] = FLIP$K_TOCHLI; ! HLI record code
556 U 0679 2 INCR INDEX FROM 0 TO MAX_LEVELS+1 ! Copy HL info
557 U 0680 2 DO
558 U 0681 2 VECTOR[FLIP_RECORD[TOCHLI_HLLIST],.INDEX]=VECTOR[HLLIST,.INDEX];
559 U 0682 2 INCR INDEX FROM 0 TO MAX_LEVELS-1 ! Copy format info
560 U 0683 2 DO
561 U 0684 2 VECTOR[FLIP_RECORD[TOCHLI_HLDSP],.INDEX]=HLDSP[.INDEX];
562 U 0685 2 $XPO_PUT( IOB=.RNOIOB, STRING=(FLIP$K_TOCHLI_SIZE,
563 U 0686 2 CH$PTR(FLIP_RECORD)));
564 U 0687 2 RETURN
565 U 0688 2 END;
566 0689 2 %FI
567 0690 2
568 0691 2 !Write the Record Group Header for this TOC record. The length of the
569 0692 2 !described record is:
570 0693 2 ! 2 (TOC header, from PUTRTY) plus
```



```
: 571      0694 2      | 2 (header-level count) plus
: 572      0695 2      | twice the maximum header-level depth (MAX_LEVELS).
: 573      0696 2      |
: 574      0697 2      | ! Do not count this record in the .BRN count.
: 575      0698 2      |
: 576      0699 2      | rgh (brn_contents, 2 + 2 + max_levels + max_levels, false);
: 577      0700 2      |
: 578      0701 2      | !Write a descriptor record identifying what's to come as header level
: 579      0702 2      | ! information.
: 580      0703 2      | PUTRTY (MAJ_RUNOFF, MIN_HL_INF);
: 581      0704 2      |
: 582      0705 2      | !First write the header level numbers themselves
: 583      0706 2      | $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (2 + MAX_LEVELS, HLLIST));
: 584      0707 2      |
: 585      0708 2      | !Now write the display information
: 586      0709 2      | $XPO_PUT (IOB = BRNOOB, BINARY_DATA = (MAX_LEVELS, HLDSP));
: 587      0710 2      |
: 588      0711 2      | !And that's it for header levels.
: 589      0712 2      | RETURN
: 590      0713 2      |
: 591      0714 1      | END;
```

!End of PUTHLI

.PSECT \$SPLITS,NOWRT,NOEXE,2

00 00 49 4C 48 54 55 50 00020 P.AAE: .ASCII \PUTHLI\<0><0>

.PSECT \$CODE\$,NOWRT,2

			001C 00000	.ENTRY	PUTHLI, Save R2,R3,R4	: 0637
54	00000000G	EF	9E 00002	MOVAB	XPOS\$PUT, R4	
53	00000000G	EF	9E 00009	MOVAB	XPOS\$FAILURE, R3	
52	00000000G	EF	9E 00010	MOVAB	IOB\$+68, R2	
5E		08	C2 00017	SUBL2	#8, SP	
1D	EE	A2	E8 0001A	BLBS	BRNOOB+50, 1\$	: 0665
72	00000000G	EF	E8 0001E	BLBS	GCA+228, 2\$	: 0668
		06	DD 00025	PUSHL	#6	: 0670
	00000000'	EF	9F 00027	PUSHAB	P.AAE	
	00000000G	8F	DD 0002D	PUSHL	#RNFILE	
00000000G	EF	03	FB 00033	CALLS	#3, ERMS	
		04	0003A	RET		: 0667
7E		10	7D 0003B 1\$:	MOVQ	#16, -(SP)	: 0699
		02	DD 0003E	PUSHL	#2	
00000000G	EF	03	FB 00040	CALLS	#3, RGH	
7E		02	7D 00047	MOVQ	#2, -(SP)	: 0703
FCB1	CF	02	FB 0004A	CALLS	#2, PUTRTY	
6E		20	B0 0004F	MOVW	#32, \$IOB\$OUTPUT	: 0706
02	AE	02	90 00052	MOVB	#2, \$IOB\$OUTPUT+2	
03	AE	01	90 00056	MOVB	#1, \$IOB\$OUTPUT+3	
04	AE	EF	9E 0005A	MOVAB	HLLIST, \$IOB\$OUTPUT+4	
62		6E	9E 00062	MOVAB	\$IOB\$OUTPUT, IOB\$+68	
E8	A2	07	90 00065	MOVB	#7, IOB\$+44	
		53	DD 00069	PUSHL	R3	
		7E	D4 0006B	CLRL	-(SP)	

PUTTOC
V04-000

Puttoc - routines to pass binary information to
Module Level Declarations

E 13
16-Sep-1984 01:30:33
14-Sep-1984 13:07:49

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]PUTTOC.BLI;1

Page 20
(8)

		BC	A2	9F	0006D	PUSHAB	IOB\$
	64		03	FB	00070	CALLS	#3, XPOS\$PUT
	6E		18	B0	00073	MOVW	#24, \$IOB\$OUTPUT
02	AE		02	90	00076	MOVB	#2, \$IOB\$OUTPUT+2
03	AE		01	90	0007A	MOVB	#1, \$IOB\$OUTPUT+3
04	AE	00000000G	EF	9E	0007E	MOVAB	HLDSP, \$IOB\$OUTPUT+4
	62		6E	9E	00086	MOVAB	\$IOB\$OUTPUT, IOB\$+68
E8	A2		07	90	00089	MOVB	#7, IOB\$+44
			53	DD	0008D	PUSHL	R3
			7E	D4	0008F	CLRL	-(SP)
		BC	A2	9F	00091	PUSHAB	IOB\$
	64		03	FB	00094	CALLS	#3, XPOS\$PUT
			04	00097	2\$:	RET	

0709

0714

; Routine Size: 152 bytes, Routine Base: \$CODE\$ + 0300

: 592 0715 1
: 593 0716 1 END
: 594 0717 0 ELUDOM

!End of module

PSECT SUMMARY

Name	Bytes	Attributes
\$PLITS	40	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODE\$	920	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]XPORT.L32;1	590	99	16	252	00:00.1
\$255\$DUA28:[RUNOFF.SRC]DSRLIB.L32;1	1248	15	1	86	00:00.3

COMMAND QUALIFIERS

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

: Size: 920 code + 40 data bytes
: Run Time: 00:22.0
: Elapsed Time: 00:43.3

PUTTOC
V04-000

Puttoc - routines to pass binary information to
Module Level Declarations

F 13
16-Sep-1984 01:30:33

VAX-11 Bliss-32 V4.0-742

Page 21

; Lines/CPU Min: 1959
; Lexemes/CPU-Min: 51195
; Memory Used: 152 pages
; Compilation Complete

0347 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY