

```
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSS            MMMMMM  MMMMMM  GGG            RRR      RRR      TTT      LLL
SSS            MMMMMM  MMMMMM  GGG            RRR      RRR      TTT      LLL
SSS            MMMMMM  MMMMMM  GGG            RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG            RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG            RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG            RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG            RRR      RRR      TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG            RRRRRRRRRRRR  TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG            RRRRRRRRRRRR  TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG            RRRRRRRRRRRR  TTT      LLL
SSS            MMM      MMM      GGG      GGGGGGGGGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGGGGGGGGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGGGGGGGGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGG      GGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGG      GGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGG      GGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGG      GGG  RRR      RRR      TTT      LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
```

```

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
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS

```



```

1 0001 0 %TITLE 'SMG$PUT_VIRTUAL_DISPLAY_ENCODED - Put to virtual display encoded'
2 0002 0 MODULE SMG$PUT_VIRTUAL_DISPLAY_ENCODED (
3 0003 0 IDENT = '1-006' ! File: SMGPUTENC.B32 Edit: RKR1006
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
11 0011 1 * ALL RIGHTS RESERVED.
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
18 0018 1 * TRANSFERRED.
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
22 0022 1 * CORPORATION.
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY: Screen Management
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1 This module contains a routine which accepts an encoded form
36 0036 1 of text and attribute information which needs to be placed in
37 0037 1 a virtual display buffer.
38 0038 1
39 0039 1 ENVIRONMENT: User mode, Shared library routines.
40 0040 1
41 0041 1 AUTHOR: R. Reichert, CREATION DATE: 19-APR-1983
42 0042 1
43 0043 1 MODIFIED BY:
44 0044 1
45 0045 1 1-001 - Original. Skeleton for future code. RKR 19-APR-1983
46 0046 1 1-002 - Start code. RKR 19-APR-1983.
47 0047 1 1-003 - Make it work. RKR 21-APR-1983.
48 0048 1 1-004 - Add $SMG$VALIDATE ROW COL after being stung. RKR 22-APR-1983
49 0049 1 1-005 - Adapt to change in SMG$PUT TEXT TO BUFFER. RKR 27-APR-1983
50 0050 1 1-006 - Remove external references to DD_ structures and count -- no
51 0051 1 longer needed (or available).
52 0052 1 RKR 20-May-1983.
53 0053 1 --
54 0054 1

```

```

56      0055 1 %SBTTL 'Declarations'
57      0056 1
58      0057 1 SWITCHES:
59      0058 1
60      0059 1
61      0060 1
62      0061 1 LINKAGES:
63      0062 1
64      0063 1 NONE
65      0064 1
66      0065 1 TABLE OF CONTENTS:
67      0066 1
68      0067 1
69      0068 1 FORWARD ROUTINE
70      0069 1
71      0070 1 ! Public entry points:
72      0071 1
73      0072 1 SMG$PUT_VIRTUAL_DISPLAY_ENCODED; ! Move text strings with
74      0073 1 ! specially-encoded video attr.
75      0074 1 ! information provided.
76      0075 1
77      0076 1 ! INCLUDE FILES
78      0077 1
79      0078 1
80      0079 1 REQUIRE 'RTLIN:SMGPROLOG'; ! defines psects, macros, tcb,
81      0157 1 ! wcb, & terminal symbols
82      0158 1 REQUIRE 'RTLIN:STRLNK'; ! JSB linkage
83      0343 1
84      0344 1
85      0345 1 ! EXTERNAL REFERENCES
86      0346 1
87      0347 1 EXTERNAL ROUTINE
88      0348 1 LIB$GET_VM, ! Allocate heap storage
89      0349 1
90      0350 1 LIB$FREE_VM, ! Deallocate heap storage
91      0351 1
92      0352 1 LIB$ANALYZE_SDESC_R2 : LIB$ANALYZE_SDESC_JSB_LINK,
93      0353 1
94      0354 1 SMG$$CHECK_FOR_OUTPUT_DCB, ! Force output if now is the time
95      0355 1
96      0356 1 SMG$$PUT_TEXT_TO_BUFFER, ! Move callers text to virtual
97      0357 1 ! display text buffer checking
98      0358 1 ! for characters that affect
99      0359 1 ! where characters land in
100     0360 1 ! buffer.
101     0361 1
102     0362 1 SMG$$SCROLL_AREA; ! Scroll a rectangular area
103     0363 1
104     0364 1
105     0365 1 EXTERNAL
106     0366 1 SMG$_FATERRLIB, ! Fatal error in SMG$ -- internal consistency
107     0367 1 ! check failed.
108     0368 1 SMG$_INVARG, ! Invalid argument
109     0369 1 SMG$_INVCOL, ! Invalid column number
110     0370 1 SMG$_INVDIS_ID, ! Invalid virtual display id
111     0371 1 SMG$_INVPAS_ID, ! Invalid pasteboard id
112     0372 1 SMG$_INVROW, ! Invalid row number

```


SMG\$PUT_VIRTUAL SMG\$PUT_VIRTUAL_DISPLAY_ENCODED - Put to virtua
1-006 Declarations

M 12
16-Sep-1984 01:11:42
14-Sep-1984 13:09:59

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGPUTENC.B32;1

Page 3
(2)

: 113 0373 1 SMG\$_WRONUMARG; ! Wrong number of arguments
: 114 0374 1
: 115 0375 1 !<BLF/PAGE>

```

117 0376 1 %SBTTL 'SMG$PUT_VIRTUAL_DISPLAY_ENCODED - Output encoded string'
118 0377 1 GLOBAL ROUTINE SMG$PUT_VIRTUAL_DISPLAY_ENCODED (
119 0378 1     DISPLAY_ID,
120 0379 1     ENCODED_LENGTH,
121 0380 1     ENCODED_TEXT : REF VECTOR [,BYTE],
122 0381 1     LINE_NO : REF VECTOR [,WORD],
123 0382 1     COL_NO : REF VECTOR [,WORD],
124 0383 1     WRAP_FLAG,
125 0384 1     CHAR_SET
126 0385 1 ) =
127 0386 1
128 0387 1 ++
129 0388 1 FUNCTIONAL DESCRIPTION:
130 0389 1
131 0390 1     This routine will output the encoded string to the explicit
132 0391 1     virtual display (if DISPLAY_ID is supplied) or to the default
133 0392 1     virtual display (if DISPLAY_ID is not supplied). If LINE_NO
134 0393 1     and COL_NO are omitted, the string is output at the current
135 0394 1     cursor position. The entire line is rewritten; if the text
136 0395 1     does not span the width of the display, blank fill is supplied.
137 0396 1
138 0397 1     This routine is useful to output strings where some but not all
139 0398 1     of the characters in a string will have video renditions, or
140 0399 1     where different video renditions are desired for characters in
141 0400 1     the same string.
142 0401 1
143 0402 1 CALLING SEQUENCE:
144 0403 1
145 0404 1     ret_status.wlc.v = SMG$PUT_VIRTUAL_DISPLAY_ENCODED (
146 0405 1         DISPLAY_ID.rl.r,
147 0406 1         ENCODED_LENGTH.rlu.r,
148 0407 1         ENCODED_TEXT.rab.v
149 0408 1         [,LINE_NO.rwu.r]
150 0409 1         [,COL_NO.rwu.r]
151 0410 1         [,WRAP_FLAG.rl.r]
152 0411 1         [,CHAR_SET.rl.r])
153 0412 1
154 0413 1 FORMAL PARAMETERS:
155 0414 1
156 0415 1     DISPLAY_ID.rl.r    The display id of the desired display.
157 0416 1
158 0417 1     ENCODED_LENGTH.rlu.r    Length of the encoded text
159 0418 1
160 0419 1     ENCODED_TEXT.rab.v The address of the encoded string to be output.
161 0420 1     Format of the encoding is:
162 0421 1
163 0422 1     text_of_string      | triplet | triplet | ... | length_of_postamble
164 0423 1     (encoded length = 5 bytes | 5 bytes | ... | 2 bytes
165 0424 1     length_of_postamble)
166 0425 1     bytes
167 0426 1
168 0427 1     Text_of_string      The actual text.
169 0428 1
170 0429 1     one or more triplets defining a field within the text
171 0430 1     consisting of:
172 0431 1
173 0432 1     Field_start_pos.rwu    Starting byte position of a
  
```



```

174 0433 1 field with the text (Numbering
175 0434 1 starts at 1).
176 0435 1
177 0436 1 Field_length.rwu Length of a field in bytes.
178 0437 1
179 0438 1 Field_attribute.rbu Bits representing video
180 0439 1 attributes to be applied to this
181 0440 1 field.
182 0441 1
183 0442 1 Fields have their video attributes applied on a field-by-
184 0443 1 field basis. Portions of the text not contained within a
185 0444 1 field definition have no video attributes applied. See
186 0445 1 examples below.
187 0446 1
188 0447 1 Length_of_postamble.rwu The length of the postamble --
189 0448 1 consisting of this byte count
190 0449 1 and the sum of the bytes
191 0450 1 occupied by the triplets.
192 0451 1 The number of triplets can be
193 0452 1 calculated as:
194 0453 1 (Length_of_postamble - 2) / 5
195 0454 1 Length_of_postamble must always
196 0455 1 be present - value will be 2 if
197 0456 1 there are no triplets.
198 0457 1
199 0458 1 Examples:
200 0459 1
201 0460 1 Length_of      Triplets      Text
202 0461 1 postamble
203 0462 1 -----
204 0463 1 7             1 3 code_for_underline      ABC
205 0464 1 is ABC underlined.
206 0465 1
207 0466 1 12            1 9 code_for_underline
208 0467 1                6 4 code_for_bold          ABCDEFGHI
209 0468 1 is ABCDEFGHI with ABCDE underlined and FGHI bolded (not underlined
210 0469 1 and bolded).
211 0470 1
212 0471 1 Each triplet is treated as an atomic unit; attributes are applied
213 0472 1 in the order specified. For instance, in the example above, the
214 0473 1 code for bold overrides the code for underline for letters FGHI.
215 0474 1 If the desired result was bold and underline for FGHI, the attribute
216 0475 1 would be specified as code_for_bold_OR_code_for_underline.
217 0476 1
218 0477 1 LINE_NO.rwu.r Optional. Line number at which to start output.
219 0478 1 If omitted (=0), the current line is used.
220 0479 1
221 0480 1 COL_NO.rwu.r Optional. Column number at which to start
222 0481 1 output. If omitted (=0), the current column is used.
223 0482 1
224 0483 1 WRAP_FLAG.rl.r Optional.
225 0484 1 = 0 means no wrap
226 0485 1 = 1 means wrap
227 0486 1 If omitted, no wrap is the default.
228 0487 1
229 0488 1 CHAR_SET.rl.r Optional. Character set to use.
230 0489 1 Choices are:
  
```



```

231 0490 1 | SMG$C_UNITED_KINGDOM
232 0491 1 | SMG$C_ASCII (default)
233 0492 1 | SMG$C_SPEC_GRAPHICS
234 0493 1 | SMG$C_ALT_CHAR
235 0494 1 | SMG$C_ALT_GRAPHICS
236 0495 1 |
237 0496 1 | IMPLICIT INPUTS:
238 0497 1 |
239 0498 1 |     NONE
240 0499 1 |
241 0500 1 | IMPLICIT OUTPUTS:
242 0501 1 |
243 0502 1 |     NONE
244 0503 1 |
245 0504 1 | COMPLETION STATUS:
246 0505 1 |
247 0506 1 |     SSS_NORMAL Normal successful completion
248 0507 1 |     LIB$INSVIRMEM Insufficient virtual memory.
249 0508 1 |     SMG$INVARG Invalid argument or combination of arguments.
250 0509 1 | For example, one of the attribute triplets may
251 0510 1 | specify a starting byte and number of bytes that
252 0511 1 | extends beyond the length of the string.
253 0512 1 |
254 0513 1 | SIDE EFFECTS:
255 0514 1 |
256 0515 1 |     NONE
257 0516 1 | --
258 0517 2 | BEGIN
259 0518 2 | BUILTIN
260 0519 2 | NULLPARAMETER;
261 0520 2 |
262 0521 2 | +
263 0522 2 | Definitions of part of a attribute triplet.
264 0523 2 | -
265 0524 2 | MACRO
266 0525 2 |     F_START = 0, 0, 16, 0%, ! Field starting position
267 0526 2 |     F_LEN = 2, 0, 16, 0%, ! Field length
268 0527 2 |     F_ATTR = 4, 0, 8, 0%, ! Field attribute bits
269 0528 2 |
270 0529 2 | LITERAL
271 0530 2 |     THRESHOLD = 240, ! If the size of the string
272 0531 2 | ! we're dealing with is greater
273 0532 2 | ! than this size, allocate
274 0533 2 | ! buffer from heap storage --
275 0534 2 | ! else use stack buffer.
276 0535 2 |     TRIP_SIZE = 5, ! Size of an attribute triplet
277 0536 2 |     POST_SIZE_FIELD = 2; ! Size of the postamble length
278 0537 2 | ! field
279 0538 2 | ! field.
280 0539 2 | LOCAL
281 0540 2 |     LOCAL_BUF: VECTOR [THRESHOLD, BYTE], ! Local buffer on stack
282 0541 2 |
283 0542 2 |     CSET, ! Character set being used.
284 0543 2 |
285 0544 2 |     ROW, ! Row in virtual display where next char goes.
286 0545 2 |
287 0546 2 |     COL, ! Col in virtual display where next char goes.
  
```



```

288      0547      2
289      0548      2      TRIP_START,      ! Offset in encoded string of
290      0549      2      ! 1st attribute triplet.
291      0550      2
292      0551      2      TRIP,      ! Index for which attribute
293      0552      2      ! triplet is currently being
294      0553      2      ! processed.
295      0554      2
296      0555      2      NO_TRIPS,      ! No. of attribute triplets in
297      0556      2      ! this string.
298      0557      2
299      0558      2      PAMBLE_LEN,      ! Length of the postamble
300      0559      2
301      0560      2      T_LEN,      ! Length of the text part of the
302      0561      2      ! string.
303      0562      2
304      0563      2      STATUS,      ! Status of subroutine calls
305      0564      2
306      0565      2      ABUF : REF VECTOR [,BYTE],      ! Address of a temporary buffer
307      0566      2      ! to hold the attributes while
308      0567      2      ! we apply the attributes
309      0568      2      ! indicated by the triplets
310      0569      2      ! in the input string.
311      0570      2
312      0571      2      DCB : REF BLOCK [,BYTE];      ! Addr of display control block
313      0572      2
314      0573      2
315      0574      2      $SMG$VALIDATE_ARGCOUNT (3, 7);
316      0575      2
317      0576      2      $SMG$GET_DCB ( .DISPLAY_ID, DCB);      ! Get addr of display control
318      0577      2      ! block
319      0578      2
320      0579      2      !+
321      0580      2      ! Calculate the lengths and counts of the pieces of this overall string.
322      0581      2      !-
323      0582      2      PAMBLE_LEN = (.ENCODED_TEXT [..ENCODED_LENGTH -POST_SIZE_FIELD])<0, POST_SIZE_FIELD*8>;
324      0583      2      T_LEN = ..ENCODED_LENGTH - .PAMBLE_LEN;
325      0584      2      TRIP_START = ENCODED_TEXT [..T_LEN];
326      0585      2      NO_TRIPS = (.PAMBLE_LEN - POST_SIZE_FIELD) / TRIP_SIZE;
327      0586      2
328      0587      2      !+
329      0588      2      ! Calc. where this text is supposed to go and see if it is a valid
330      0589      2      ! position. Do this before we potentially allocate a heap buffer
331      0590      2      ! since if $SMG$VALIDATE_ROW_COL exits with a error, we'll never
332      0591      2      ! get control back to free the buffer.
333      0592      2      !-
334      0593      2      ROW = (IF NOT NULLPARAMETER (4) THEN .LINE_NO [0]
335      0594      2      ! ELSE .DCB [DCB_W_CURSOR_ROW]);
336      0595      2
337      0596      2      COL = (IF NOT NULLPARAMETER (5) THEN .COL_NO [0]
338      0597      2      ! ELSE .DCB [DCB_W_CURSOR_COL]);
339      0598      2
340      0599      2      $SMG$VALIDATE_ROW_COL ( .ROW, .COL );
341      0600      2
342      0601      2      DCB [DCB_W_CURSOR_ROW] = .ROW;
343      0602      2      DCB [DCB_W_CURSOR_COL] = .COL;
344      0603      2      !+

```

```

345 0604 2 ! If needed, get a buffer to hold the attribute bytes.
346 0605 2 !-
347 0606 2 ABUF = LOCAL_BUF; ! assume stack space will suffice
348 0607 2 IF ..ENCODED_LENGTH GTR THRESHOLD
349 0608 2 THEN
350 0609 2 IF NOT ( STATUS = LIB$GET_VM ( T_LEN, ABUF))
351 0610 2 THEN
352 0611 2 RETURN (.STATUS);
353 0612 2
354 0613 2 !+
355 0614 2 Initialize the attribute bytes to be the defaults for the display.
356 0615 2 !-
357 0616 2 CH$FILL ( .DCB [DCB_B_DEF_VIDEO_ATTR], .T_LEN, .ABUF);
358 0617 2
359 0618 2 !+
360 0619 2 Apply each of the attribute triplets to our temporary attribute buffer
361 0620 2 !-
362 0621 2 TRIP = 0;
363 0622 2 WHILE .TRIP LSS .NO_TRIPS
364 0623 2 DO
365 0624 2 BEGIN ! For each triplet
366 0625 2 LOCAL
367 0626 2 PTR : REF BLOCK [,BYTE]; ! Pointer to current triplet
368 0627 2
369 0628 2 PTR = .TRIP_START + (.TRIP * TRIP_SIZE);
370 0629 2
371 0630 2 !+
372 0631 2 Make sure field start and field length make sense before
373 0632 2 using them.
374 0633 2 !-
375 0634 2 IF .PTR [F_LEN] LSS 0 OR
376 0635 2 .PTR [F_LEN] + .PTR [F_START] GTR .T_LEN
377 0636 2 THEN
378 0637 2 BEGIN ! Bailout
379 0638 2 IF ..ENCODED_LENGTH GTR THRESHOLD
380 0639 2 THEN
381 0640 2 BEGIN ! Give back heap first
382 0641 2 LIB$FREE_VM ( T_LEN, ABUF);
383 0642 2 RETURN (SMG$INVARG);
384 0643 2 END; ! Give back heap first
385 0644 2 END; ! Bailout
386 0645 2
387 0646 2 CH$FILL ( .PTR [F_ATTR], ! Attribute character
388 0647 2 .PTR [F_LEN], ! No. of bytes in field
389 0648 2 ABUF [ .PTR [F_START] -1 ] ); ! Start of attribute
390 0649 2 ! area to be changed.
391 0650 2 TRIP = .TRIP + 1;
392 0651 2 END; ! For each triplet
393 0652 2
394 0653 2 !+
395 0654 2 Now that we know the byte by byte attributes that need to go into
396 0655 2 the DCB's attribute buffer -- put the text and the attributes in
397 0656 2 in the DCB's buffers.
398 0657 2 NOTE: This chunk of code should be made smarter by having it
399 0658 2 call SMG$PUT_TEXT_TO_BUFFER with batches of characters that have the
400 0659 2 same attributes instead of character by character.
401 0660 2 !-
```



```

: 402      0661      3      CSET = (IF NOT NULLPARAMETER (7) THEN .CHAR_SET
: 403      0662      2      ELSE 0);
: 404      0663      2
: 405      0664      2      INCR I FROM 0 TO .T_LEN-1
: 406      0665      2      DO
: 407      0666      3      BEGIN ! Put text into buffer
: 408      0667      4      IF NOT ( STATUS = SMG$PUT_TEXT_TO_BUFFER (
: 409      0668      4          .DCB,                                ! DCB addr
: 410      0669      4          .ABUF [.I],                        ! Attr byte
: 411      0670      4          1,                                ! Text length
: 412      0671      4          ENCODED_TEXT [.I],                ! Text byte
: 413      0672      4          .CSET))                            ! Char set
: 414      0673      3      THEN
: 415      0674      3          RETURN (.STATUS);
: 416      0675      3      END;      ! Put text into buffer
: 417      0676      2
: 418      0677      2
: 419      0678      2      !+ Release the temporary buffer we've been using if in heap storage.
: 420      0679      2      !-
: 421      0680      2      IF ..ENCODED_LENGTH GTR THRESHOLD
: 422      0681      2      THEN
: 423      0682      2          IF NOT (STATUS = LIB$FREE_VM ( T_LEN, ABUF))
: 424      0683      2          THEN
: 425      0684      2              RETURN ( .STATUS);
: 426      0685      2
: 427      0686      2      RETURN (SMG$CHECK_FOR_OUTPUT_DCB ( .DCB,
: 428      0687      2          SMG$C_PUT_DISPLAY_ENCODED,
: 429      0688      2          .ROW));
: 430      0689      2
: 431      0690      1      END;      ! End of routine SMG$PUT_VIRTUAL_DISPLAY_ENCODED

```

```

.TITLE SMG$PUT_VIRTUAL_DISPLAY_ENCODED SMG$PUT_VIRTUAL_DISPLAY_ENCODE
D - Put to virt
ua

```

```

.IDENT \1-006\

```

```

.EXTRN LIB$GET_VM, LIB$FREE_VM
.EXTRN LIB$ANALYZE_SDESC_R2
.EXTRN SMG$CHECK_FOR_OUTPUT_DCB
.EXTRN SMG$PUT_TEXT_TO_BUFFER
.EXTRN SMG$SCROLL_AREA
.EXTRN SMG$_FATERR[IB, SMG$_INVARG
.EXTRN SMG$_INVCOL, SMG$_INVDIS_ID
.EXTRN SMG$_INVPAS_ID, SMG$_INVROW
.EXTRN SMG$_WRONUMARG

```

```

.PSECT _SMG$CODE,NOWRT, SHR, PIC,2

```

```

.ENTRY SMG$PUT_VIRTUAL_DISPLAY_ENCODED, Save R2,-      ; 0377
R3,R4,R5,R6,R7,R8,R9,R10,R11
MOVAB -252(SP), SP
SUBB3 #3, (AP), DIFF      ; 0574
CMPB DIFF, #4
BLEQU 1$
MOVL #SMG$_WRONUMARG, R0

```

```

OFFC 00000

```

```

50      5E      FF04      CE      9E      00002
        6C      03      83      00007
        04      50      91      0000B
        08      1B      0000E
50 00000000G 8F      D0      00010

```

			04	50	04	BC	D0	00017	1\$:	RET			
			04	BC	38	A0	D1	00018		MOVL	@DISPLAY_ID, R0		0576
				11	44	A0	12	00021		CMPL	56(R0), @DISPLAY_ID		
				50	00000000G	8F	13	00027		BNEQ	2\$		
				57	04	BC	91	00023		CMPB	68(R0), #17		
				50	00000000G	8F	D0	00029	2\$:	BEQL	3\$		
				50	00000000G	8F	D0	00030		MOVL	#SMG\$_INVDIS_ID, R0		
				50	00000000G	8F	D0	00031	3\$:	RET			
				50	00000000G	8F	D0	00035		MOVL	@DISPLAY_ID, DCB		0582
				50	00000000G	8F	D0	00038		ADDL3	@ENCODED_LENGTH, ENCODED_TEXT, R0		
				50	00000000G	8F	D0	0003F		MOVZBL	-2(R0), PAMBLE_LEN		0583
				50	00000000G	8F	D0	00045		SUBL3	PAMBLE_LEN, @ENCODED_LENGTH, T_LEN		0584
				50	00000000G	8F	D0	00048		ADDL3	T_LEN, ENCODED_TEXT, TRIP_START		0585
				50	00000000G	8F	D0	0004E		SUBL2	#2, R0		
				50	00000000G	8F	D0	00052		DIVL3	#5, R0, NO_TRIPS		
				50	00000000G	8F	D0	00055		CMPB	(AP), #4		0593
				50	00000000G	8F	D0	00057		BLSSU	4\$		
				50	00000000G	8F	D0	0005A		TSTL	16(AP)		
				50	00000000G	8F	D0	0005C		BEQL	4\$		
				50	00000000G	8F	D0	00060		MOVZWL	@LINE_NO, ROW		
				50	00000000G	8F	D0	00062	4\$:	BRB	5\$		0594
				50	00000000G	8F	D0	00066	5\$:	MOVZWL	40(DCB), ROW		0596
				50	00000000G	8F	D0	00069		CMPB	(AP), #5		
				50	00000000G	8F	D0	0006B		BLSSU	6\$		
				50	00000000G	8F	D0	0006E		TSTL	20(AP)		
				50	00000000G	8F	D0	00070		BEQL	6\$		
				50	00000000G	8F	D0	00074		MOVZWL	@COL_NO, COL		
				50	00000000G	8F	D0	00076	6\$:	BRB	7\$		0597
				50	00000000G	8F	D0	0007A	7\$:	MOVZWL	42(DCB), COL		0599
				50	00000000G	8F	D0	0007C		TSTL	ROW		
				50	00000000G	8F	D0	0007E		BLEQ	8\$		
				50	00000000G	8F	D0	00084		CMPZV	#0, #16, 2(DCB), ROW		
				50	00000000G	8F	D0	00086	8\$:	BGEQ	9\$		
				50	00000000G	8F	D0	0008D		MOVL	#SMG\$_INVROW, R0		
				50	00000000G	8F	D0	0008E	9\$:	RET			
				50	00000000G	8F	D0	00090		TSTL	COL		
				50	00000000G	8F	D0	00092		BLEQ	10\$		
				50	00000000G	8F	D0	00098		CMPZV	#0, #16, 6(DCB), COL		
				50	00000000G	8F	D0	0009A	10\$:	BGEQ	11\$		
				50	00000000G	8F	D0	000A1		MOVL	#SMG\$_INVCOL, R0		
				50	00000000G	8F	D0	000A2	11\$:	RET			
				50	00000000G	8F	D0	000A6		MOVW	ROW, 40(DCB)		0601
				50	00000000G	8F	D0	000AA		MOVW	COL, 42(DCB)		0602
				50	00000000G	8F	D0	000AF		MOVAB	LOCAL_BUF, ABUF		0606
				50	00000000G	8F	D0	000B1		CLRL	R10		0607
				50	00000000G	8F	D0	000B9		CMPL	@ENCODED_LENGTH, #240		
				50	00000000G	8F	D0	000BB		BLEQ	12\$		
				50	00000000G	8F	D0	000BD		INCL	R10		
				50	00000000G	8F	D0	000C0		PUSHAB	ABUF		0609
				50	00000000G	8F	D0	000C3		PUSHAB	T_LEN		
				50	00000000G	8F	D0	000CA		CALLS	#2, LIB\$GET_VM		
				50	00000000G	8F	D0	000CD		MOVL	R0, STATUS		
				50	00000000G	8F	D0	000D0		BLBS	STATUS, 12\$		
				50	00000000G	8F	D0	000D3	12\$:	BRW	20\$		
				50	00000000G	8F	D0	000DA		MOVCS	#0, (SP), 46(DCB), T_LEN, @ABUF		0616
				50	00000000G	8F	D0	000DC		CLRL	TRIP		0621

	5B	56	D1	000DE	13\$:	CMPL	TRIP, NO_TRIPS	0622
		44	18	000E1		BGEQ	15\$	
50	56	05	C5	000E3		MULL3	#5, TRIP, R0	0628
52	50	6E	C1	000E7		ADDL3	TRIP, START, R0, PTR	
	50	A2	3C	000EB		MOVZWL	2(PTR), R0	0635
	51	62	3C	000EF		MOVZWL	(PTR), R1	
	50	51	C0	000F2		ADDL2	R1, R0	
08	AE	50	D1	000F5		CMPL	R0, T_LEN	
	15	18	15	000F9		BLEQ	14\$	
		5A	E9	000FB		BLBC	R10, 14\$	0638
		AE	9F	000FE		PUSHAB	ABUF	0641
		OC	AE	9F	00101	PUSHAB	T_LEN	
00000000G	00	02	FB	00104		CALLS	#2, LIB\$FREE_VM	
	50	00	9E	0010B		MOVAB	SMG\$_INVARG, R0	0642
			04	00112		RET		
	50	62	3C	00113	14\$:	MOVZWL	(PTR), R0	0648
02	A2	50	AE	C0	00116	ADDL2	ABUF, R0	
		6E	00	2C	0011A	MOVCS	#0, (SP), 4(PTR), 2(PTR), -1(R0)	
			FF	A0	00121			
		56	D6	00123		INCL	TRIP	0650
		B7	11	00125		BRB	13\$	0622
07		6C	91	00127	15\$:	CMPB	(AP), #7	0661
		0B	1F	0012A		BLSSU	16\$	
		1C	AC	D5	0012C	TSTL	28(AP)	
		06	13	0012F		BEQL	16\$	
53		1C	BC	D0	00131	MOVL	@CHAR_SET, CSET	
		02	11	00135		BRB	17\$	
		53	D4	00137	16\$:	CLRL	CSET	
52		01	CE	00139	17\$:	MNEGL	#1, I	0669
		1C	11	0013C		BRB	19\$	
		53	DD	0013E	18\$:	PUSHL	CSET	0672
		OC	BC	42	9F	PUSHAB	@ENCODED_TEXT[I]	0671
			01	DD	00144	PUSHL	#1	
	7E	10	BE	42	9A	MOVZBL	@ABUF[I], -(SP)	
			57	DD	0014B	PUSHL	DCB	
00000000G	00	05	FB	0014D		CALLS	#5, SMG\$\$_PUT_TEXT_TO_BUFFER	
	58	50	D0	00154		MOVL	R0, STATUS	
	1B	58	E9	00157		BLBC	STATUS, 20\$	
DF	52	08	AE	F2	0015A	AOBLSS	T_LEN, I, 18\$	0664
	17		5A	E9	0015F	BLBC	R10, 21\$	0681
		04	AE	9F	00162	PUSHAB	ABUF	0683
		OC	AE	9F	00165	PUSHAB	T_LEN	
00000000G	00	02	FB	00168		CALLS	#2, LIB\$FREE_VM	
	58	50	D0	0016F		MOVL	R0, STATUS	
	04	58	E8	00172		BLBS	STATUS, 21\$	
	50	58	D0	00175	20\$:	MOVL	STATUS, R0	0685
			04	00178		RET		
		59	DD	00179	21\$:	PUSHL	ROW	0689
		13	DD	0017B		PUSHL	#19	0687
		57	DD	0017D		PUSHL	DCB	
00000000G	00	03	FB	0017F		CALLS	#3, SMG\$\$_CHECK_FOR_OUTPUT_DCB	
			04	00186		RET		0690

; Routine Size: 391 bytes, Routine Base: _SMG\$CODE + 0000

; 432 0691 1 !<BLF/PAGE>

SMG\$PUT_VIRTUAL
1-006

SMG\$PUT_VIRTUAL_DISPLAY_ENCODED - Put to virtua
SMG\$PUT_VIRTUAL_DISPLAY_ENCODED - Output encode

16-Sep-1984 01:11:42
14-Sep-1984 13:09:59

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGPUTENC.B32;1

Page 12
(4)

: 434
: 435
: 436

0692 1 END
0693 1
0694 0 ELUDOM

! End of module SMG\$PUT_VIRTUAL_DISPLAY_ENCODED

:

PSECT SUMMARY

:

Name

Bytes

Attributes

:

_SMG\$CODE

391 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

:

Library Statistics

:

File

Total

Symbols
Loaded

Percent

Pages
Mapped

Processing
Time

:

_\$255\$DUA28:[SYSLIB]STARLET.L32;1

9776

3

0

581

00:01.0

:

_\$255\$DUA28:[SMGRTL.OBJ]RTL.LIB.L32;1

36

0

0

8

00:00.1

:

_\$255\$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1

469

11

2

38

00:00.4

:

COMMAND QUALIFIERS

:

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

:

:

Size: 391 code + 0 data bytes

: Run Time: 00:11.3

: Elapsed Time: 00:38.1

: Lines/CPU Min: 3688

: Lexemes/CPU-Min: 14869

: Memory Used: 148 pages

: Compilation Complete

0360

**DIGITAL
CONFIDE**

EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY