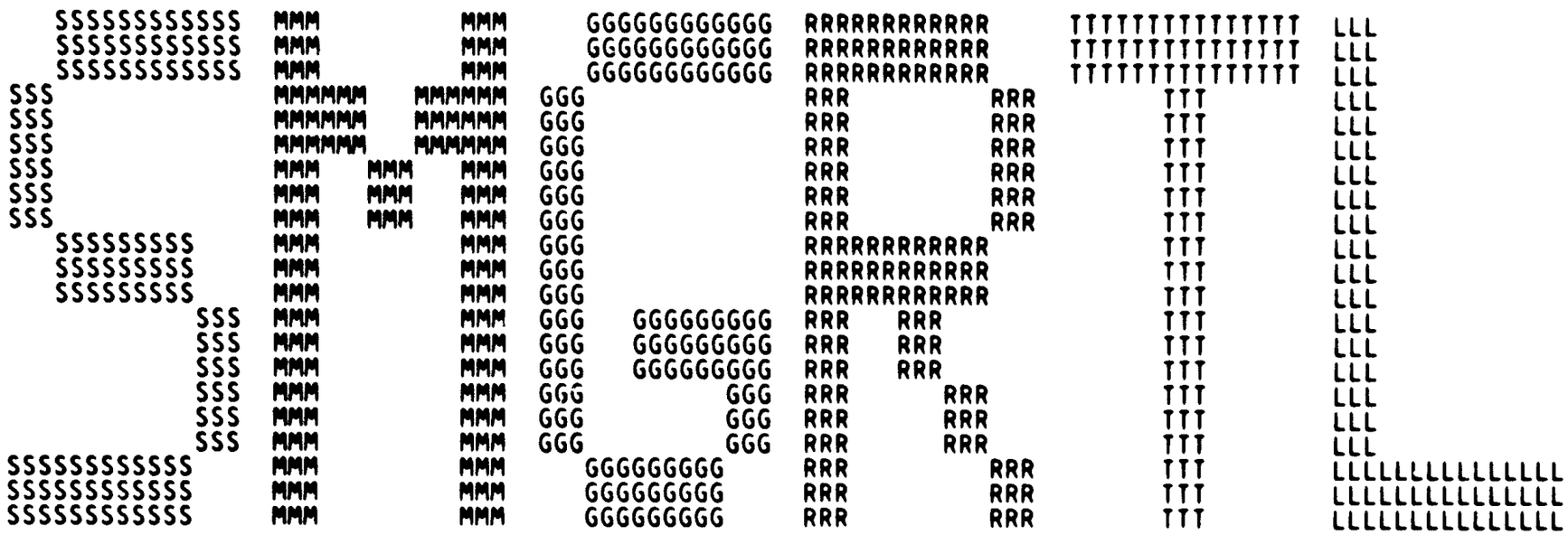
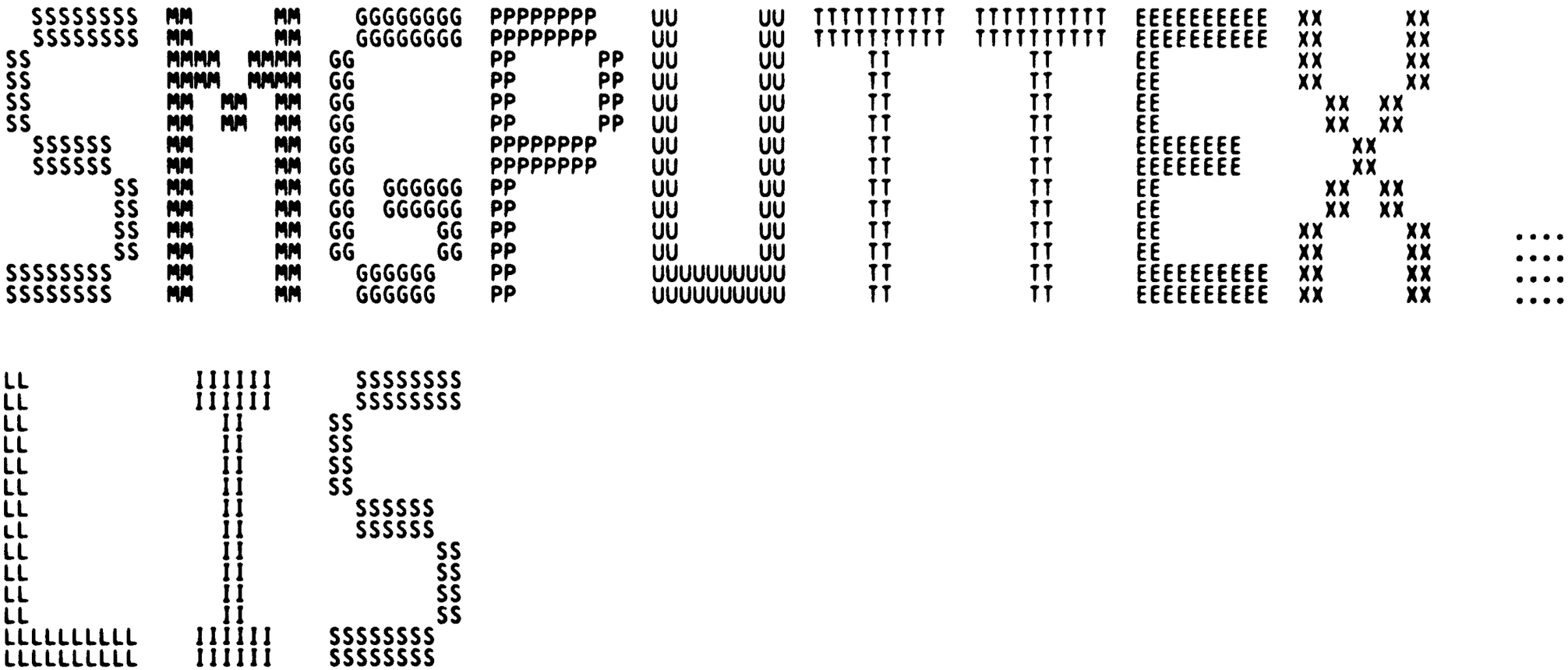


_$2

Val

001
001
001
001
001
001
001
001
001
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF

FILEID**SMGPUTTEX



```
;     1          0001  0 MODULE SMG$$PUT_TEXT_TO_BUFFER ( %TITLE 'Put text to display buffer'
;     2          0002  0                  IDENT = '1-012'          ! File: SMGPUTTEX.B32 Edit: PLL1012
;     3          0003  0                  ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                      *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;    17          0017  1 !*  TRANSFERRED.                                                              *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    21          0021  1 !*  CORPORATION.                                                              *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:     Screen Management
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This is an internal routine used by screen management procedures to
;    36          0036  1 !       place user's text into a display buffer.  The text is spanned for
;    37          0037  1 !       special characters.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:  User mode - AST reentrant
;    40          0040  1 !
;    41          0041  1 ! AUTHOR: P. Levesque, CREATION DATE: 14-Apr-1983
;    42          0042  1 !
;    43          0043  1 ! MODIFIED BY:
;    44          0044  1 !
;    45          0045  1 ! 1-001 - Original.  PLL 14-Apr-1983
;    46          0046  1 ! 1-002 - Finish coding.  PLL 20-Apr-1983
;    47          0047  1 ! 1-003 - Add error message, character set buffer allocation.  PLL 4-May-1983
;    48          0048  1 ! 1-004 - Fix second half of the scan table to agree with actions for
;    49          0049  1 !         DEC Multinational.  PLL 5-May-1983
;    50          0050  1 ! 1-005 - If on the last line and we have found a line feed, scroll.  PLL 11-May-1983
;    51          0051  1 ! 1-006 - If a bell character is found, call SMG$RING_BELL instead of setting
;    52          0052  1 !         a bell bit.  PLL 20-May-1983
;    53          0053  1 ! 1-007 - If a LF is found, scroll according to the new dcb top & bottom of
;    54          0054  1 !         scrolling region fields.  PLL 26-May-1983
;    55          0055  1 ! 1-008 - If an ESC is detected, call the terminal simulator routine to
;    56          0056  1 !         interpret the sequence and perform the correct SMG$ function.
;    57          0057  1 !         PLL 7-Jul-1983
```

```
;    58          0058  1 ! 1-009 - Allow 2 'reserved' positions in upper half of table to pass thru
;    59          0059  1 !           as printable characters.  PLL 17-Aug-1983
;    60          0060  1 ! 1-010 - SMG$$SIM_TERM may set the graphics bit in the DCB's default
;    61          0061  1 !           attributes byte.  Take this into account when copying the attribute
;    62          0062  1 !           bytes for characters into the buffer.  PLL 29-Aug-1983
;    63          0063  1 ! 1-011 - Call SMG$$SIM_TERM when DCB_V_ALLOW_ESC is set.  PLL 2-Sept-1983
;    64          0064  1 ! 1-012 - In order to print carriage control characters instead of execute
;    65          0065  1 !           them, check the DCB_V_DISPLAY_CONTROLS bit and move the ascii rep
;    66          0066  1 !           into the text buffer in a different way.  PLL 23-Sep-1983
;    67          0067  1 !--
;    68          0068  1
```

```
;    70          0069  1 %SBTTL 'Declarations'
;    71          0070  1 !
;    72          0071  1 ! SWITCHES:
;    73          0072  1 !
;    74          0073  1
;    75          0074  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    76          0075  1
;    77          0076  1 !
;    78          0077  1 ! LINKAGES:
;    79          0078  1 !
;    80          0079  1 !       NONE
;    81          0080  1 !
;    82          0081  1 ! TABLE OF CONTENTS:
;    83          0082  1 !
;    84          0083  1
;    85          0084  1 FORWARD ROUTINE
;    86          0085  1     SMG$$PUT_TEXT_TO_BUFFER;
;    87          0086  1
;    88          0087  1 !
;    89          0088  1 ! INCLUDE FILES:
;    90          0089  1 !
;    91          0090  1
;    92          0091  1 REQUIRE 'RTLIN:SMGPROLOG';              ! defines Psects, macros, data base
;    93          0169  1
;    94          0170  1 !
;    95          0171  1 ! MACROS:
;    96          01    1 !
;    97          0173  1 !       NONE
;    98          0174  1 !
;    99          0175  1 ! EQUATED SYMBOLS:
;   100          0176  1 !
;   101          0177  1 !       NONE
;   102          0178  1 !
;   103          0179  1 ! FIELDS:
;   104          0180  1 !
;   105          0181  1 !       NONE
;   106          0182  1 !
;   107          0183  1 ! PSECTS:
;   108          0184  1 !
;   109          0185  1
;   110          0186  1 !
;   111          0187  1 ! EXTERNAL REFERENCES:
;   112          0188  1 !
;   113          0189  1
;   114          0190  1 EXTERNAL ROUTINE
;   115          0191  1     SMG$$SIM_TERM,
;   116          0192  1     SMG$$SCROLL_AREA,
;   117          0193  1     SMG$RING_BELL;
;   118          0194  1
;   119          0195  1 EXTERNAL LITERAL
;   120          0196  1     SMG$_FATERRLIB,
;   121          0197  1     SMG$_STRTERESC;
;   122          0198  1
;   123          0199  1 ! Some constants needed by reference.
;   124          0200  1 OWN
;   125          0201  1     ALLONES     : BYTE INITIAL (-1);
;   126          0202  1
```

```
;  127          0203  1 ! The following macro is used to move a control character into the
;  128          0204  1 ! text buffer in such a way that output will later convert to the
;  129          0205  1 ! appropriate device dependent graphic character.
;  130          0206  1
;  131          0207  1 MACRO
;  132        M 0208  1         $INSERT_CTRL_CHAR (CHAR) =
;  133        M 0209  1         BEGIN
;  134        M 0210  1         LOCAL
;  135        M 0211  1             INDEX,
;  136        M 0212  1             REMAINING_COLS;
;  137        M 0213  1
;  138        M 0214  1         REMAINING_COLS = .DCB [DCB_W_NO_COLS] - .DCB [DCB_W_CURSOR_ROW];
;  139        M 0215  1         INDEX = $SMG$LINEAR (.DCB [DCB_W_CURSOR_ROW], .DCB [DCB_W_CURSOR_COL]);
;  140        M 0216  1
;  141        M 0217  1         IF 1 GTR .REMAINING_COLS
;  142        M 0218  1         THEN
;  143        M 0219  1             WORK_OVERFLOW = .BYTES_REMAINING
;  144        M 0220  1         ELSE
;  145        M 0221  1             BEGIN                   ! move the low nibble into the high nibble
;  146        M 0222  1             LOCAL
;  147        M 0223  1                 SHIFT_NIBBLE : BYTE,
;  148        M 0224  1                 WORK_ATTR;
;  149        M 0225  1             SHIFT_NIBBLE = (CHAR <0,4>) ^ 4;
;  150        M 0226  1             CH$MOVE (1, SHIFT_NIBBLE, TEXT_BUF [.INDEX]);
;  151        M 0227  1             WORK_ATTR = ATTR_M_USER_GRAPHIC OR .ATTR_CODE;
;  152        M 0228  1             CH$MOVE (1, WORK_ATTR, ATTR_BUF [.INDEX]);
;  153        M 0229  1             END;
;  154        M 0230  1
;  155        M 0231  1         DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_CURSOR_COL] + 1;
;  156        M 0232  1         IF .DCB [DCB_W_CURSOR_COL] EQL .DCB [DCB_W_NO_COLS]
;  157        M 0233  1         THEN
;  158        M 0234  1             DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_NO_COLS];
;  159          0235  1         END%;
;  160          0236  1
;  161          0237  1 !<BLF/PAGE>
```

```
;  163     0238  1 !+
;  164     0239  1 ! The table below (CHAR_TABLE) is used with a SCANC instruction to
;  165     0240  1 ! detect characters that have an impact on how text needs to be
;  166     0241  1 ! positioned in a text buffer that models what is on a portion of the
;  167     0242  1 ! screen.  Each character position is occupied by a code indicating
;  168     0243  1 ! the kind of action that this character has on text placement.
;  169     0244  1 ! Characters are grouped into 10 categories based on their impact on
;  170     0245  1 ! the terminal and hence on their impact on what should be placed in
;  171     0246  1 ! the buffer at what position.
;  172     0247  1 !
;  173     0248  1 ! These categories (codes) are:
;  174     0249  1 !
;  175     0250  1 !       Action Code     Action
;  176     0251  1 !       -----------     ------
;  177     0252  1 !            0          Normal processing.  Character occupies next
;  178     0253  1 !                       available slot in buffer.  Cursor column is
;  179     0254  1 !                       advanced by 1 after placement.
;  180     0255  1 !
;  181     0256  1 !            1          Character can be discarded.  Cursor is not
;  182     0257  1 !                       advanced.
;  183     0258  1 !
;  184     0259  1 !            2          Character can be discarded.  Cursor is not
;  185     0260  1 !                       modified, but a note must be made that the
;  186     0261  1 !                       bell needs to be sounded.
;  187     0262  1 !
;  188     0263  1 !            3          Character can be discarded, but cursor must be
;  189     0264  1 !                       backed up one column.  Be careful about cursor
;  190     0265  1 !                       already being in column 1.
;  191     0266  1 !
;  192     0267  1 !            4          Character can be discarded, but cursor must be
;  193     0268  1 !                       advanced to next TAB stop and intervening
;  194     0269  1 !                       character positions in the buffer are
;  195     0270  1 !                       undisturbed.
;  196     0271  1 !
;  197     0272  1 !                       TAB stops are assumed to be set in the following
;  198     0273  1 !                       columns with column numbering starting at 1:
;  199     0274  1 !                       9, 17, 25, 33, 41, 49, 57, 65, 73 ( width=80)
;  200     0275  1 !
;  201     0276  1 !                       9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97,
;  202     0277  1 !                       105, 113, 121, 129 ( width=132)
;  203     0278  1 !
;  204     0279  1 !            5          Character can be discarded.  Cursor must be
;  205     0280  1 !                       advanced by one line.
;  206     0281  1 !
;  207     0282  1 !            6          Character can be discarded.  Cursor must be
;  208     0283  1 !                       advanced by one line.  (VT treated the same
;  209     0284  1 !                       as #5, FF.)
;  210     0285  1 !
;  211     0286  1 !            7          Character can be discarded.  Effect is
;  212     0287  1 !                       to clear the buffer and reset the cursor to
;  213     0288  1 !                       line 1 column 1.
;  214     0289  1 !
;  215     0290  1 !            8          Character can be discarded.  Effect is to set
;  216     0291  1 !                       cursor to column 1 of current line.
;  217     0292  1 !
;  218     0293  1 !            9          Character can be discarded.  For this version,
;  219     0294  1 !                       ESC terminates the string.  Eventually, subsequent
```

```
;  220          0295  1 !                      characters need to be inspected to see if they
;  221          0296  1 !                      constitute a recognized escape sequence whose
;  222          0297  1 !                      effect must be simulated-- E.g., cursor setting,
;  223          0298  1 !                      rendition setting.
;  224          0299  1 !
;  225          0300  1 !                      Some problems with this are:
;  226          0301  1 !                          1. What to do about sequences that we don't
;  227          0302  1 !                             recognize ?
;  228          0303  1 !                          2. What to do about sequences that we
;  229          0304  1 !                             recognize as ones that can cause
;  230          0305  1 !                             confusion later is allowed to be
;  231          0306  1 !                             sent to terminal -- E.g. select graphics
;  232          0307  1 !                             rendition, etc ?
;  233          0308  1 !
;  234          0309  1 !               10     Character can be discarded.  Character is
;  235          0310  1 !                      treated as a no-op.  It is broken out separately
;  236          0311  1 !                      in case we ever need to do something special
;  237          0312  1 !                      with it.
;  238          0313  1 !
;  239          0314  1 ! In summary:
;  240          0315  1 !
;  241          0316  1 !   Hex Character Codes   ASCII Character              Action Code
;  242          0317  1 !   ------------------   ---------------              -----------
;  243          0318  1 !       00 to 06         NUL to ACK                       1
;  244          0319  1 !          07            BEL                              2
;  245          0320  1 !          08            BS                               3
;  246          0321  1 !          09            HT                               4
;  247          0322  1 !          0A            LF                               5
;  248          0323  1 !          0B            VT                               6
;  249          0324  1 !          0C            FF                               7
;  250          0325  1 !          0D            CR                               8
;  251          0326  1 !       0E to 0F         SO to SI                         9
;  252          0327  1 !       10 to 1A         DLE to SUB                       1
;  253          0328  1 !          1B            ESC                              9
;  254          0329  1 !       1C to 1F         FS to US                         1
;  255          0330  1 !       20 to 7E         SP to ~                          0
;  256          0331  1 !          7F            DEL                             10
;  257          0332  1 !
;  258          0333  1 !       80 to 9F         control chars                    1
;  259          0334  1 !          A0            reserved                         1
;  260          0335  1 !       A1 to FE         printing chars                   0
;  261          0336  1 !          FF            reserved                         1
;  262          0337  1 !
```

```
;  264          0338  1 GLOBAL
;  265          0339  1     CHAR_TABLE : VECTOR [256, BYTE] INITIAL ( BYTE (
;  266          0340  1                                             ! 1st half is US ASCII
;  267          0341  1                                             ! for DEC Multinational set (default)
;  268          0342  1          1, 1, 1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9,  ! 00 to 0F
;  269          0343  1          1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 9, 1, 1, 1, 1,  ! 10 to 1F
;  270          0344  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! 20 to 2F
;  271          0345  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! 30 to 3F
;  272          0346  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! 40 to 4F
;  273          0347  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! 50 to 5F
;  274          0348  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! 60 to 6F
;  275          0349  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,10,  ! 70 to 7F
;  276          0350  1                                             ! 2nd half is DEC Supplemental Graphics
;  277          0351  1                                             ! for DEC Multinational set (default)
;  278          0352  1          1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  ! 80 to 8F
;  279          0353  1          1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  ! 90 to 9F
;  280          0354  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! A0 to AF
;  281          0355  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! B0 to BF
;  282          0356  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! C0 to CF
;  283          0357  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! D0 to DF
;  284          0358  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ! E0 to EF
;  285          0359  1          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0   ! F0 to FF
;  286          0360  1                                                 ));
;  287          0361  1
;  288          0362  1
;  289          0363  1
;  290          0364  1
;  291          0365  1 !<BLF/PAGE>
```

```
;  293          0366  1 %SBTTL 'SMG$$PUT_TEXT_TO_BUFFER - Put text to buffer'
;  294          0367  1 GLOBAL ROUTINE SMG$$PUT_TEXT_TO_BUFFER (
;  295          0368  1                                       DCB : REF BLOCK [,BYTE],
;  296          0369  1                                       ATTR_CODE : BYTE,
;  297          0370  1                                       TEXT_LEN,
;  298          0371  1                                       TEXT_ADDR,
;  299          0372  1                                       CHAR_SET,
;  300          0373  1                                       OVERFLOW
;  301          0374  1                                           ) =
;  302          0375  1 !++
;  303          0376  1 ! FUNCTIONAL DESCRIPTION:                                                                    ;
;  304          0377  1 !
;  305          0378  1 !       This procedure places a text string into a buffer given the
;  306          0379  1 !       current row and column in the buffer where output is to go.
;  307          0380  1 !       The input text string is scanned for special characters that
;  308          0381  1 !       prohibit simply moving the text into the buffer.  For example,
;  309          0382  1 !       TABs reposition the maintained cursor position and the text
;  310          0383  1 !       must be deposited at the appropriate tab boundaries as a
;  311          0384  1 !       function of current position in the line.  Escape sequences
;  312          0385  1 !       are not handled; an escape character is treated as a terminator,
;  313          0386  1 !       and a qualified success status will be returned to indicate
;  314          0387  1 !       that truncation occurred.
;  315          0388  1 !
;  316          0389  1 !       Positions in BUFFER that are modified have the corresponding
;  317          0390  1 !       positions in ATTR_BUFFER and CHAR_BUFFER set.
;  318          0391  1 !
;  319          0392  1 !
;  320          0393  1 !
;  321          0394  1 ! CALLING SEQUENCE:
;  322          0395  1 !
;  323          0396  1 !       ret_status.wlc.v = SMG$$PUT_TEXT_TO_BUFFER (
;  324          0397  1 !                                   DCB.mab.r,
;  325          0398  1 !                                   ATTR_CODE.rb.v,
;  326          0399  1 !                                   TEXT_LEN.rl.v,
;  327          0400  1 !                                   TEXT_ADDR.rl.v,
;  328          0401  1 !                                   CHAR_SET.rl.v
;  329          0402  1 !                                   [,OVERFLOW.wl.r])
;  330          0403  1 !
;  331          0404  1 ! FORMAL PARAMETERS:
;  332          0405  1 !
;  333          0406  1 !       DCB.mab.r       Address of virtual display control block.
;  334          0407  1 !                       Various fields from within in this block are
;  335          0408  1 !                       are interrogated and/or updated.
;  336          0409  1 !
;  337          0410  1 !       ATTR_CODE.rb.v  Video rendition attribute code.
;  338          0411  1 !                       Bit 0   Bold
;  339          0412  1 !                       Bit 1   Reverse video
;  340          0413  1 !                       Bit 2   Blinking
;  341          0414  1 !                       Bit 3   Underscored
;  342          0415  1 !
;  343          0416  1 !       TEXT_LEN.rl.v   Length of text string
;  344          0417  1 !
;  345          0418  1 !       TEXT_ADDR.rl.v  Address of text string
;  346          0419  1 !
;  347          0420  1 !       CHAR_SET.rl.v   Character set to use.
;  348          0421  1 !                               SMG$C_UNITED_KINGDOM
;  349          0422  1 !                               SMG$C_ASCII
```

```
;  350          0423  1 !                                  SMG$C_SPEC_GRAPHICS
;  351          0424  1 !                                  SMG$C_ALT_CHAR
;  352          0425  1 !                                  SMG$C_ALT_GRAPHICS
;  353          0426  1 !
;  354          0427  1 !       OVERFLOW.wl.r   Optional.  Address of longword in which
;  355          0428  1 !                       to return the number of characters that
;  356          0429  1 !                       did not fit on the line.
;  357          0430  1 !
;  358          0431  1 ! IMPLICIT INPUTS:
;  359          0432  1 !
;  360          0433  1 !       NONE
;  361          0434  1 !
;  362          0435  1 ! IMPLICIT OUTPUTS:
;  363          0436  1 !
;  364          0437  1 !       NONE
;  365          0438  1 !
;  366          0439  1 ! COMPLETION STATUS:
;  367          0440  1 !
;  368          0441  1 !       SS$_NORMAL      Normal successful completion
;  369          0442  1 !
;  370          0443  1 ! SIDE EFFECTS:
;  371          0444  1 !
;  372          0445  1 !       NONE
;  373          0446  1 !--
;  374          0447  1
;  375          0448  2     BEGIN
;  376          0449  2
;  377          0450  2     BUILTIN
;  378          0451  2         SCANC,
;  379          0452  2         NULLPARAMETER;
;  380          0453  2
;  381          0454  2     LOCAL
;  382          0455  2         TEXT_BUF : REF VECTOR [,BYTE],  ! Addr of text buffer
;  383          0456  2         ATTR_BUF : REF VECTOR [,BYTE],  ! Addr of attr buffer
;  384          0457  2         CHAR_BUF : REF VECTOR [,BYTE],  ! Addr of char set buffer
;  385          0458  2         STATUS,             ! status of subroutine calls
;  386          0459  2         WORK_OVERFLOW : INITIAL (0),    ! no. of overflow chars
;  387          0460  2         BYTES_REMAINING,! No. of bytes in input string yet to be
;  388          0461  2                         ! processed.
;  389          0462  2         IN_POINTER;     ! Current pointer into input string
;  390          0463  2
;  391          0464  2     LITERAL
;  392          0465  2         K_OVERFLOW_ARG = 6;
;  393          0466  2
;  394          0467  2     TEXT_BUF = .DCB [DCB_A_TEXT_BUF];
;  395          0468  2     ATTR_BUF = .DCB [DCB_A_ATTR_BUF];
;  396          0469  2     CHAR_BUF = .DCB [DCB_A_CHAR_SET_BUF];
;  397          0470  2
;  398          0471  2     BYTES_REMAINING = .TEXT_LEN;
;  399          0472  2     IN_POINTER = .TEXT_ADDR;
;  400          0473  2
;  401          0474  2     WHILE .BYTES_REMAINING NEQ 0
;  402          0475  2     DO
;  403          0476  3         BEGIN   ! Overall loop
;  404          0477  3         LOCAL
;  405          0478  3             CHARS_TO_MOVE,          ! No. of characters to move on this
;  406          0479  3                                     ! iteration
```

```
;  407          0480  3              PLACE_TO_MOVE,          ! Place to move from on this iteration
;  408          0481  3              NEW_BYTES_REMAINING,    ! No. of bytes remaining as returned
;  409          0482  3                                      ! by SCANC
;  410          0483  3              ADDR_DIFF;              ! Adur of char in input stream whose
;  411          0484  3                                      ! index into scanc table yields
;  412          0485  3                                      ! non-zero code.
;  413          0486  3          !+
;  414          0487  3          ! See if any of the remaining input characters require special
;  415          0488  3          ! treatment.
;  416          0489  3          !-
;  417          0490  3          SCANC ( BYTES_REMAINING,        ! No. of bytes remaining
;  418          0491  3                  .IN_POINTER,            ! Current pointer to source
;  419          0492  3                  CHAR_TABLE,             ! Address of SCANC table
;  420          0493  3                  ALLONES;                ! Mask for ANDing
;  421          0494  3                  NEW_BYTES_REMAINING,    ! New remaining no. of bytes
;  422          0495  3                                          ! including the byte which
;  423          0496  3                                          ! caused the instruction to
;  424          0497  3                                          ! halt.  Is zero only if all
;  425          0498  3                                          ! bytes did not satify search.
;  426          0499  3                  ADDR_DIFF);             ! Addr of char in input stream
;  427          0500  3                                          ! whose index into scanc table
;  428          0501  3                                          ! yields non-zero code.
;  429          0502  3
;  430          0503  3          CHARS_TO_MOVE = .BYTES_REMAINING - .NEW_BYTES_REMAINING;
;  431          0504  3          PLACE_TO_MOVE = .IN_POINTER;
;  432          0505  3          IN_POINTER = .IN_POINTER + .CHARS_TO_MOVE;
;  433          0506  3          BYTES_REMAINING = .NEW_BYTES_REMAINING;
;  434          0507  3
;  435          0508  3          !+
;  436          0509  3          ! Copy the appropriate number of characters into the text buffer
;  437          0510  3          ! and the appropriate number of copies of the attribute code
;  438          0511  3          ! into the attribute buffer.
;  439          0512  3          !-
;  440          0513  3          IF .CHARS_TO_MOVE NEQ 0
;  441          0514  3          THEN
;  442          0515  4              BEGIN
;  443          0516  4              LOCAL
;  444          0517  4                  INDEX,  ! 0-based index into BUFFER and ATTR_BUFFER.
;  445          0518  4                  REMAINING_COLS;
;  446          0519  4
;  447          0520  4              INDEX = $SMG$LINEAR ( .DCB [DCB_W_CURSOR_ROW], .DCB [DCB_W_CURSOR_COL]);
;  448          0521  4
;  449          0522  4              REMAINING_COLS = .DCB [DCB_W_NO_COLS] - .DCB [DCB_W_CURSOR_COL] + 1;
;  450          0523  4              IF .CHARS_TO_MOVE GTR .REMAINING_COLS
;  451          0524  4              THEN                                ! chars will overflow line
;  452          0525  5                  BEGIN
;  453          0526  5                  WORK_OVERFLOW = .BYTES_REMAINING +
;  454          0527  5                                  (.CHARS_TO_MOVE - .REMAINING_COLS);
;  455          0528  5                  CHARS_TO_MOVE = .REMAINING_COLS;
;  456          0529  4                  END;
;  457          0530  4
;  458          0531  4              !+
;  459          0532  4              ! Move text into buffer.
;  460          0533  4              !-
;  461          0534  4              CH$MOVE (.CHARS_TO_MOVE,            ! No. of chars
;  462          0535  4                       .PLACE_TO_MOVE,            ! From
;  463          0536  4                       TEXT_BUF [ .INDEX ] );     ! To
```

```
;  464          0537  4
;  465          0538  4                !+
;  466          0539  4                ! Rewrite attribute bytes.  Normally the attributes are
;  467          0540  4                ! passed to us, but for the 'autobended' case where escape
;  468          0541  4                ! sequences are used, we should look at the default attributes
;  469          0542  4                ! which may have been altered by SMG$$SIM_TERM.
;  470          0543  4                !-
;  471          0544  5                BEGIN
;  472          0545  5                LOCAL
;  473          0546  5                    WORK_ATTR;
;  474          0547  5                WORK_ATTR = .ATTR_CODE;
;  475          0548  5                IF .DCB [DCB_V_ALLOW_ESC]
;  476          0549  5                THEN
;  477          0550  5                    WORK_ATTR = .DCB [DCB_B_DEF_VIDEO_ATTR];
;  478          0551  5                CH$FILL (.WORK_ATTR,                    ! Char. to replicate
;  479          0552  5                         .CHARS_TO_MOVE,                ! No. of times
;  480          0553  5                         ATTR_BUF [ .INDEX ] );         ! Destination
;  481          0554  4                END;
;  482          0555  4
;  483          0556  4                !+
;  484          0557  4                ! Write the character set bytes, if necessary.
;  485          0558  4                !-
;  486          0559  4                IF .CHAR_BUF EQL 0 AND
;  487          0560  4                   .CHAR_SET NEQ SMG$C_ASCII
;  488          0561  4                THEN
;  489          0562  4                    0;      ! first char set - alloc buffer
;  490          0563  4
;  491          0564  4                IF .CHAR_BUF NEQ 0
;  492          0565  4                THEN
;  493          0566  4                    CH$FILL (.CHAR_SET,
;  494          0567  4                             .CHARS_TO_MOVE,
;  495          0568  4                             CHAR_BUF [.INDEX]);
;  496          0569  4
;  497          0570  4                !+
;  498          0571  4                ! Adjust resulting cursor position.  Check for overflow.
;  499          0572  4                !-
;  500          0573  4                DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_CURSOR_COL] +
;  501          0574  4                                         .CHARS_TO_MOVE;
;  502          0575  4                IF .DCB [DCB_W_CURSOR_COL] GTR .DCB [DCB_W_NO_COLS]
;  503          0576  4                THEN
;  504          0577  4                    DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_NO_COLS];
;  505          0578  4
;  506          0579  4                IF .WORK_OVERFLOW NEQ 0
;  507          0580  4                THEN
;  508          0581  4                    EXITLOOP;
;  509          0582  3                END;
;  510          0583  3
;  511          0584  3
;  512          0585  3            IF .NEW_BYTES_REMAINING EQL 0
;  513          0586  3            THEN
;  514          0587  3                EXITLOOP;           ! Break out of loop -- we're done
;  515          0588  3
;  516          0589  3            !+
;  517          0590  3            ! Dispatch on the non-zero code located to see what special
;  518          0591  3            ! action is needed.
;  519          0592  3            !-
;  520          0593  3            CASE .CHAR_TABLE [.(.ADDR_DIFF) <0,8>] FROM 1 TO 10 OF
```

```
;  521          0594  3          SET
;  522          0595  3              [1]:
;  523          0596  3              !+
;  524          0597  3              !    Hex Character Codes       ASCII Character
;  525          0598  3              !    ------------------        ---------------
;  526          0599  3              !    00 to 06                   NUL to ACK
;  527          0600  3              !    10 to 1A                   DLE to SUB
;  528          0601  3              !    1C to 1F                   FS to US
;  529          0602  3              !
;  530          0603  3              ! Character can be discarded.  Cursor is not advanced.
;  531          0604  3              !
;  532          0605  3              ! Special case if the user_graphic bit is set.  That indicates
;  533          0606  3              ! a device-independent code which should be placed in the buffer
;  534          0607  3              ! for later interpretation by output.  Notice that we are guaranteed
;  535          0608  3              ! that TEXT_ADDR contains only 1 character since only we call this
;  536          0609  3              ! routine.
;  537          0610  3              !-
;  538          0611  3              IF (.ATTR_CODE AND ATTR_M_USER_GRAPHIC) NEQ 0
;  539          0612  3              THEN
;  540          0613  3                  $INSERT_CTRL_CHAR ( TEXT_ADDR);
;  541          0614  3
;  542          0615  3              [2]:
;  543          0616  3              !+
;  544          0617  3              !    Hex Character Codes       ASCII Character
;  545          0618  3              !    ------------------        ---------------
;  546          0619  3              !       07                       BEL
;  547          0620  3              !
;  548          0621  3              ! Character can be discarded.  Cursor is not modified, and we
;  549          0622  3              ! call a routine to ring the bell now.  (Note that if we had
;  550          0623  3              ! stored the bell in the attribute buffer, the bell would've
;  551          0624  3              ! been rung every time the screen was repainted.)
;  552          0625  3              !-
;  553          0626  3              SMG$RING_BELL (.DCB [DCB_L_DID]);
;  554          0627  3
;  555          0628  3              [3]:
;  55           0629  3              !+
;  557          0630  3              !    Hex Character Codes       ASCII Character
;  558          0631  3              !    ------------------        ---------------
;  559          0632  3              !       08                       BS
;  560          0633  3              !
;  561          0634  3              ! Character can be discarded, but cursor must be backed up
;  562          0635  3              ! one column.  Be careful about cursor already being in
;  563          0636  3              ! column 1.
;  564          0637  3              !-
;  565          0638  4              BEGIN
;  566          0639  4              IF .DCB [DCB_W_CURSOR_COL] NEQ 1
;  567          0640  4              THEN
;  568          0641  4                  DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_CURSOR_COL] -1;
;  569          0642  4
;  570          0643  3              END;
;  571          0644  3
;  572          0645  3              [4]:
;  573          0646  3              !+
;  574          0647  3              !    Hex Character Codes       ASCII Character
;  575          0648  3              !    ------------------        ---------------
;  576          0649  3              !       09                       HT
;  577          0650  3              !
```

```
;  578     0651  3              ! Character can be discarded, but cursor must be advanced to
;  579     0652  3              ! next TAB stop and intervening character positions in the
;  580     0653  3              ! buffer must be left undisturbed.
;  581     0654  3              !
;  582     0655  3              ! TAB stops are assumed to be set in the following columns:
;  583     0656  3              ! 9, 17, 25, 33, 41, 49, 57, 65, 73 ( width=80)
;  584     0657  3              !
;  585     0658  3              ! 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 105, 113,
;  586     0659  3              ! 121, 129 ( width=132)
;  587     0660  3              !-
;  588     0661  4              BEGIN
;  589     0662  4              !+
;  590     0663  4              ! Be careful about tabbing off the end of the line or beyond
;  591     0664  4              ! the end of the virtual display line.
;  592     0665  4              !-
;  593     0666  4              IF NOT .DCB [DCB_V_DISPLAY_CONTROLS]
;  594     0667  4              THEN
;  595     0668  5                  BEGIN
;  596     0669  5                  DCB [DCB_W_CURSOR_COL] =
;  597     0670  5                                  (1+(.DCB [DCB_W_CURSOR_COL]-1)/8+1)*8+1;
;  598     0671  5                  IF .DCB [DCB_W_CURSOR_COL] GTR .DCB [DCB_W_NO_COLS]
;  599     0672  5                  THEN
;  600     0673  5                      DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_NO_COLS];
;  601     0674  5                  END
;  602     0675  4              ELSE
;  603     0676  4                  $INSERT_CTRL_CHAR (TAB);
;  604     0677  3              END;
;  605     0678  3
;  606     0679  3              [5,6]:
;  607     0680  3              !+
;  608     0681  3              !    Hex Character Codes       ASCII Character
;  609     0682  3              !    -------------------       ---------------
;  610     0683  3              !       0A                       LF
;  611     0684  3              !       0B                       VT
;  612     0685  3              !
;  613     0686  3              ! Character can be discarded.  Cursor must be advanced by
;  614     0687  3              ! one line.  Don't advance beyond last line of display.
;  615     0688  3              !-
;  616     0689  4              BEGIN
;  617     0690  4              !+
;  618     0691  4              ! If cursor not at bottom, advance DCB [DCB_W_CURSOR_ROW]
;  619     0692  4              ! by one.
;  620     0693  4              !-
;  621     0694  4              IF NOT .DCB [DCB_V_DISPLAY_CONTROLS]
;  622     0695  4              THEN
;  623     0696  5                  BEGIN
;  624     0697  5                  IF .DCB [DCB_W_CURSOR_ROW] + 1 LEQ .DCB [DCB_W_BOTTOM_OF_SCRREG]
;  625     0698  5                  THEN
;  626     0699  5                      DCB [DCB_W_CURSOR_ROW] = .DCB [DCB_W_CURSOR_ROW] + 1
;  627     0700  5                  ELSE
;  628     0701  5                      SMG$$SCROLL_AREA (.DCB,
;  629     0702  5                                        .DCB [DCB_W_TOP_OF_SCRREG],
;  630     0703  5                                        .DCB [DCB_W_COL_START],
;  631     0704  6                                        (.DCB [DCB_W_BOTTOM_OF_SCRREG] -
;  632     0705  5                                         .DCB [DCB_W_TOP_OF_SCRREG] + 1),
;  633     0706  5                                        .DCB [DCB_W_NO_COLS],
;  634     0707  5                                        SMG$M_UP,
```

```
;  635     0708  5                                        1);
;  636     0709  5                    END
;  637     0710  4                ELSE
;  638     0711  5                    BEGIN
;  639     0712  5                    LOCAL
;  640     0713  5                        CHAR;
;  641     0714  5                    CHAR = .(.ADDR_DIFF)<0,8>;
;  642     0715  5                    $INSERT_CTRL_CHAR (.CHAR);
;  643     0716  4                    END;
;  644     0717  3                END;
;  645     0718  3
;  646     0719  3                [7]:
;  647     0720  3                !+
;  648     0721  3                !   Hex Character Codes       ASCII Character
;  649     0722  3                !   ------------------        ---------------
;  650     0723  3                !      0C                       FF
;  651     0724  3                !
;  652     0725  3                ! Character can be discarded.  Effect is to clear the buffer
;  653     0726  3                ! and reset the cursor to line 1 column 1.
;  654     0727  3                !-
;  655     0728  4                BEGIN
;  656     0729  4                IF NOT .DCB [DCB_V_DISPLAY_CONTROLS]
;  657     0730  4                THEN
;  658     0731  5                    BEGIN
;  659     0732  5                    IF .DCB [DCB_W_CURSOR_ROW] + 1 LEQ .DCB [DCB_W_BOTTOM_OF_SCRREG]
;  660     0733  5                    THEN
;  661     0734  5                        DCB [DCB_W_CURSOR_ROW] = .DCB [DCB_W_CURSOR_ROW] + 1
;  662     0735  5                    ELSE
;  663     0736  5                        SMG$$SCROLL_AREA (.DCB,
;  664     0737  5                                          .DCB [DCB_W_TOP_OF_SCRREG],
;  665     0738  5                                          .DCB [DCB_W_COL_START],
;  666     0739  6                                          (.DCB [DCB_W_BOTTOM_OF_SCRREG] -
;  667     0740  5                                           .DCB [DCB_W_TOP_OF_SCRREG] + 1),
;  668     0741  5                                          .DCB [DCB_W_NO_COLS],
;  669     0742  5                                          SMG$M_UP,
;  670     0743  5                                          1);
;  671     0744  5                    END
;  672     0745  4                ELSE
;  673     0746  4                    $INSER _CTRL_CHAR (FF);
;  674     0747  3                END;
;  675     0748  3
;  676     0749  3                [8]:
;  677     0750  3                !+
;  678     0751  3                !   Hex Character Codes       ASCII Character
;  679     0752  3                !   ------------------        ---------------
;  680     0753  3                !      0D                       CR
;  681     0754  3                !
;  682     0755  3                ! Character can be discarded.  Effect is to set cursor to
;  683     0756  3                ! column 1 of current line.
;  684     0757  3                !-
;  685     0758  4                BEGIN
;  686     0759  4                IF NOT .DCB [DCB_V_DISPLAY_CONTROLS]
;  687     0760  4                THEN
;  688     0761  4                    DCB [DCB_W_CURSOR_COL] = 1
;  689     0762  4                ELSE
;  690     0763  4                    $INSERT_CTRL_CHAR (CR);
;  691     0764  3                END;
```

```
;   692          0765  3
;   693          0766  3                [9]:
;   694          0767  3                !+
;   695          0768  3                !    Hex Character Codes       ASCII Character
;   696          0769  3                !    ------------------        ---------------
;   697          0770  3                !       1B                      ESC
;   698          0771  3                !       0E                      SO
;   699          0772  3                !       0F                      SI
;   700          0773  3                !
;   701          0774  3                ! Character can be discarded.  Subsequent characters need
;   702          0775  3                ! to be inspected to see if they constitute a recognized
;   703          0776  3                ! escape sequence whose effect must be simulated-- E.g.,
;   704          0777  3                ! cursor setting, rendition setting.
;   705          0778  3                !
;   706          0779  3                ! SMG$$SIM_TERM processes the escape sequence, then returns
;   707          0780  3                ! here to allow any remaining characters to be processed.
;   708          0781  3                !-
;   709          0782  4                BEGIN
;   710          0783  4                IF NOT .DCB [DCB_V_ALLOW_ESC]
;   711          0784  4                THEN
;   712          0785  5                    RETURN (SMG$_STRTERESC) ! error from true SMG$
;   713          0786  4                ELSE
;   714          0787  5                    BEGIN                   ! autobended - attempt to interpret
;   715          0788  5                    LOCAL
;   716          0789  5                        LEN_OF_SEQUENCE,
;   717          0790  5                        STATUS;
;   718          0791  5                    STATUS = SMG$$SIM_TERM (.DCB,
;   719          0792  5                                            .BYTES_REMAINING,
;   720          0793  5                                            .IN_POINTER,  ! pass ptr to esc char
;   721          0794  5                                            LEN_OF_SEQUENCE);
;   722          0795  5                    IF NOT .STATUS THEN RETURN (.STATUS);
;   723          0796  5
;   724          0797  5                    !+
;   725          0798  5                    ! Update the number of bytes processed.  Since there is
;   726          0799  5                    ! an automatic update (by 1 character) at the end of this
;   727          0800  5                    ! loop, don't count the ESC now.
;   728          0801  5                    !-
;   729          0802  5                    BYTES_REMAINING = .BYTES_REMAINING - .LEN_OF_SEQUENCE + 1;
;   730          0803  5                    IN_POINTER = .IN_POINTER + .LEN_OF_SEQUENCE - 1;
;   731          0804  4                    END;                    ! autobended - attempt to interpret
;   732          0805  4
;   733          0806  3                END;
;   734          0807  3
;   735          0808  3                [10]:
;   736          0809  3                !+
;   737          0810  3                !    Hex Character Codes       ASCII Character
;   738          0811  3                !    ------------------        ---------------
;   739          0812  3                !       7F                      DEL
;   740          0813  3                !
;   741          0814  3                ! Character can be discarded.
;   742          0815  3                !-
;   743          0816  3                ;    ! no special action
;   744          0817  3
;   745          0818  3                [INRANGE, OUTRANGE]:
;   746          0819  3                !+
;   747          0820  3                ! Should never get here -- there are no other codes in
;   748          0821  3                ! CHAR_TABLE.  If we do, we've got a problem.
```

```
;  749          0822  3              !-
;  750          0823  4              BEGIN
;  751          0824  4              RETURN SMG$_FATERRLIB;
;  752          0825  3              END;
;  753          0826  3          TES;
;  754          0827  3
;  755          0828  3
;  756          0829  3          !+
;  757          0830  3          ! Re-adjust pointer and count of bytes left to account for
;  758          0831  3          ! the special character(s) just processed.
;  759          0832  3          !-
;  760          0833  3          IN_POINTER = .IN_POINTER + 1;
;  761          0834  3          BYTES_REMAINING = .BYTES_REMAINING -1;
;  762          0835  2          END;    ! Overall loop
;  763          0836  2
;  764          0837  2      IF .DCB [DCB_W_CURSOR_COL] EQL .DCB [DCB_W_NO_COLS]
;  765          0838  2      THEN
;  766          0839  2          DCB [DCB_V_COL_80] = 1;
;  767          0840  2
;  768          0841  2      IF NOT NULLPARAMETER (K_OVERFLOW_ARG)
;  769          0842  2      THEN
;  770          0843  2          .OVERFLOW = .WORK_OVERFLOW;
;  771          0844  2                                  ! ret overflow chars if requested
;  772          0845  2      RETURN (SS$_NORMAL);
;  773          0846  1      END;                        ! End of routine SMG$$PUT_TEXT_TO_BUFFER


                                                        .TITLE  SMG$$PUT_TEXT_TO_BUFFER Put text to display buf
                                                                                        fer
;                                                       .IDENT  \1-012\

                                                        .PSECT  _SMG$DATA,NOEXE,  PIC,2

                                                    FF  00000 ALLONES:.BYTE   -1                                ;
                                                        00001         .BLKB   3
09  08  07  06  05  04  03  02  01  01  01  01  01  01  01  00004 CHAR_TABLE::
                                                        .BYTE   1, 1, 1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 7, 8, -     ;
01  01  09  01  01  01  01  01  01  01  01  01  01  01  09  00013           9, 9, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 9, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  01  01  00022           1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00031           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00040           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  0004F           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  0005E           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  0006D           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
01  01  01  01  01  01  01  0A  00  00  00  00  00  00  00  0007C           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
01  01  01  01  01  01  01  01  01  01  01  01  01  01  01  0008B           0, 10, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -       ;
00  00  00  00  00  01  01  01  01  01  01  01  01  01  01  0009A           1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  000A9           1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  000B8           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  000C7           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  000D6           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  000E5           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  000F4           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
                                                    00  00103           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -     ;
                                                                        0, 0, 0, 0, 0                               ;

                                                        .EXTRN  SMG$$SIM_TERM, SMG$$SCROLL_AREA
```

```
                                                        .EXTRN  SMG$RING_BELL, SMG$_FATERRLIB
                                                        .EXTRN  SMG$_STRTERESC

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

                                        0FFC 00000      .ENTRY  SMG$$PUT_TEXT_TO_BUFFER, Save R2,R3,R4,R5,- ; 0367
                                                                R6,R7,R8,R9,R10,R11
                                    5E      20  C2 00002   SUBL2   #32, SP
                                        18  AE  D4 00005   CLRL    WORK_OVERFLOW                             ; 0448
                                    59  04  AC  D0 00008   MOVL    DCB, R9                                   ; 0467
                                    5B  10  A9  D0 0000C   MOVL    16(R9), TEXT_BUF
                                    5A  14  A9  D0 00010   MOVL    20(R9), ATTR_BUF                          ; 0468
                                14  AE  18  A9  D0 00014   MOVL    24(R9), CHAR_BUF                          ; 0469
                                    57  0C  AC  7D 00019   MOVQ    TEXT_LEN, BYTES_REMAINING                 ; 0471
                                            57  D5 0001D 1$:    TSTL    BYTES_REMAINING                       ; 0474
                                            03  12 0001F   BNEQ    2$
                                        02CE    31 00021   BRW     41$
00000000'  EF 00000000'  EF     68          57  2A 00024 2$:    SCANC   BYTES_REMAINING, (IN_POINTER), CHAR_TABLE, -; 0490
                                                                ALLONES
                                10  AE      50  D0 00031   MOVL    R0, 16(SP)
                                0C  AE      51  D0 00035   MOVL    R1, 12(SP)
                            56      57  10  AE  C3 00039   SUBL3   NEW_BYTES_REMAINING, BYTES_REMAINING, -  ; 0503
                                                                CHARS_TO_MOVE
                                    52      58  D0 0003E   MOVL    IN_POINTER, PLACE_TO_MOVE                 ; 0504
                                    58      56  C0 00041   ADDL2   CHARS_TO_MOVE, IN_POINTER                ; 0505
                                    57  10  AE  D0 00044   MOVL    NEW_BYTES_REMAINING, BYTES_REMAINING      ; 0506
                                            56  D5 00048   TSTL    CHARS_TO_MOVE                             ; 0513
                                            7C  13 0004A   BEQL    8$
                                    50  28  A9  3C 0004C   MOVZWL  40(R9), R0                               ; 0520
                                            50  D7 00050   DECL    R0
                                    51  06  A9  3C 00052   MOVZWL  6(R9), R1
                                    50      51  C4 00056   MULL2   R1, R0
                                08  AE  2A  A9  9E 00059   MOVAB   42(R9), 8(SP)
                                    51  08  BE  3C 0005E   MOVZWL  @8(SP), R1
                                04  AE  FF A140 9E 00062   MOVAB   -1(R1)[R0], INDEX
                                    50  06  A9  3C 00068   MOVZWL  6(R9), R0                               ; 0522
                                    50      51  C2 0006C   SUBL2   R1, R0
                                            50  D6 0006F   INCL    REMAINING_COLS
                                    50      56  D1 00071   CMPL    CHARS_TO_MOVE, REMAINING_COLS             ; 0523
                                            0C  15 00074   BLEQ    3$
                                51          56  C3 00076   SUBL3   REMAINING_COLS, CHARS_TO_MOVE, R1       ; 0527
                            18  AE          51  C1 0007A   ADDL3   BYTES_REMAINING, R1, WORK_OVERFLOW
                                    56      50  D0 0007F   MOVL    REMAINING_COLS, CHARS_TO_MOVE             ; 0528
                        04 BE4B             62          56  28 00082 3$:    MOVC3   CHARS_TO_MOVE, (PLACE_TO_MOVE), @INDEX- ; 0536
                                                                [TEXT_BUF]
                                    50  08  AC  9A 00088   MOVZBL  ATTR_CODE, WORK_ATTR                     ; 0547
                                04      34  A9  05 E1 0008C BBC     #5, 52(R9), 4$                          ; 0548
                                    50  2E  A9  9A 00091   MOVZBL  46(R9), WORK_ATTR                        ; 0550
                    56              50      6E  00  2C 00095 4$:    MOVC5   #0, (SP), WORK_ATTR, CHARS_TO_MOVE, @INDEX- ; 0553
                                                04 BE4A     0009A   [ATTR_BUF]
                                            14  AE  D5 0009D   TSTL    CHAR_BUF                              ; 0559
                                            0E  13 000A0   BEQL    5$                                        ; 0564
                                6E      04  AE  14  AE  C1 000A2   ADDL3   CHAR_BUF, INDEX, (SP)             ; 0568
                    56      14  AC          6E  00  2C 000A8   MOVC5   #0, (SP), CHAR_SET, CHARS_TO_MOVE, @0(SP)
                                                00  BE 000AE
                                08  BE          56  A0 000B0 5$:    ADDW2   CHARS_TO_MOVE, @8(SP)             ; 0574
                                06  A9      08  BE  B1 000B4   CMPW    @8(SP), 6(R9)                         ; 0575
```

```
                                    05   1B 000B9          BLEQU   6$                                ;
              08   BE          06   A9   B0 000BB          MOVW    6(R9), @8(SP)                     ; 0577
                               18   AE   D5 000C0 6$:      TSTL    WORK_OVERFLOW                     ; 0579
                                    03   13 000C3          BEQL    8$                                ;
                                  022A   31 000C5 7$:      BRW     41$                               ;
                               10   AE   D5 000C8 8$:      TSTL    NEW_BYTES_REMAINING               ; 0585
                                    F8   13 000CB          BEQL    7$                                ;
                      52        0C  BE   9A 000CD          MOVZBL  @ADDR_DIFF, R2                    ; 0593
          09          01 00000000'EF42   8F 000D1          CASEB   CHAR_TABLE[R2], #1, #9            ;
  006C  0061        0055          001C      000DA 9$:      .WORD   10$-9$,-                          ;
  0186  0118        00DA          00DA      000E2                  12$-9$,-                          ;
                    0211          01E0      000EA                  13$-9$,-                          ;
                                                                   15$-9$,-                          ;
                                                                   21$-9$,-                          ;
                                                                   21$-9$,-                          ;
                                                                   23$-9$,-                          ;
                                                                   31$-9$,-                          ;
                                                                   38$-9$,-                          ;
                                                                   40$-9$                            ;
                      50 00000000G  8F   D0 000EE          MOVL    #SMG$_FATERRLIB, R0               ; 0624
                                         04 000F5          RET                                       ;
          49     08   AC            06   E1 000F6 10$:     BBC     #6, ATTR_CODE, 14$                ; 0611
                      53        06  A9   3C 000FB          MOVZWL  6(R9), REMAINING_COLS             ; 0613
                      50        28  A9   3C 000FF          MOVZWL  40(R9), R0                        ;
                      53            50   C2 00103          SUBL2   R0, REMAINING_COLS                ;
                      50        28  A9   3C 00106          MOVZWL  40(R9), R0                        ;
                                    50   D7 0010A          DECL    R0                                ;
                      51        06  A9   3C 0010C          MOVZWL  6(R9), R1                         ;
                      50            51   C4 00110          MULL2   R1, R0                            ;
                      52        2A  A9   9E 00113          MOVAB   42(R9), R2                        ;
                      51            62   3C 00117          MOVZWL  (R2), R1                          ;
                      51      FF A140    9E 0011A          MOVAB   -1(R1)[R0], INDEX                 ;
                                    53   D5 0011F          TSTL    REMAINING_COLS                    ;
                                    03   14 00121          BGTR    11$                               ;
                                  016C   31 00123          BRW     34$                               ;
    53      10   AC           04    00   EF 00126 11$:     EXTZV   #0, #4, TEXT_ADDR, R3             ;
                                  00BA   31 0012C          BRW     22$                               ;
                               38   A9   DD 0012F 12$:     PUSHL   56(R9)                            ; 0626
               00000000G  00        01   FB 00132          CALLS   #1, SMG$RING_BELL                 ;
                                    77   11 00139          BRB     20$                               ;
                      01        2A  A9   B1 0013B 13$:     CMPW    42(R9), #1                        ; 0639
                                    71   13 0013F          BEQL    20$                               ;
                                2A  A9   B7 00141          DECW    42(R9)                            ; 0641
                                    6C   11 00144 14$:     BRB     20$                               ; 0593
                      53        2A  A9   9E 00146 15$:     MOVAB   42(R9), R3                        ; 0669
          18     2F   A9            02   E0 0014A          BBS     #2, 47(R9), 16$                   ; 0666
                      50            63   3C 0014F          MOVZWL  (R3), R0                          ; 0670
                                    50   D7 00152          DECL    R0                                ;
                      50            08   C6 00154          DIVL2   #8, R0                            ;
          51          50            03   78 00157          ASHL    #3, R0, R1                        ;
          63          51            09   A1 0015B          ADDW3   #9, R1, (R3)                      ;
                 06   A9            63   B1 0015F          CMPW    (R3), 6(R9)                       ; 0671
                                    4D   1B 00163          BLEQU   20$                               ;
                                    47   11 00165          BRB     19$                               ; 0673
                      52        06  A9   3C 00167 16$:     MOVZWL  6(R9), REMAINING_COLS             ; 0676
                      50        28  A9   3C 0016B          MOVZWL  40(R9), R0                        ;
                      52            50   C2 0016F          SUBL2   R0, REMAINING_COLS                ;
```

```
                                50        28    A9  3C 00172          MOVZWL  40(R9), R0                                      ;
                                          50    D7 00176           DECL    R0                                              ;
                                51        06    A9  3C 00178           MOVZWL  6(R9), R1                                       ;
                                50              51  C4 0017C           MULL2   R1, R0                                          ;
                                51              63  3C 0017F           MOVZWL  (R3), R1                                        ;
                                51        FF A140   9E 00182           MOVAB   -1(R1)[R0], INDEX                               ;
                                                52  D5 00187           TSTL    REMAINING_COLS                                  ;
                                                06  14 00189           BGTR    17$                                             ;
                             18   AE              57  D0 0018B           MOVL    BYTES_REMAINING, WORK_OVERFLOW                  ;
                                                15  11 0018F           BRB     18$                                             ;
                                50        90    8F  90 00191 17$:      MOVB    #-112, SHIFT_NIBBLE                             ;
                              614B              50  90 00195           MOVB    SHIFT_NIBBLE, (INDEX)[TEXT_BUF]                 ;
                                50        08    AC  9A 00199           MOVZBL  ATTR_CODE, WORK_ATTR                            ;
      50        01                        06        01  F0 0019D           INSV    #1, #6, #1, WORK_ATTR                           ;
                              614A              50  90 001A2           MOVB    WORK_ATTR, (INDEX)[ATTR_BUF]                    ;
                                                63  B6 001A6 18$:      INCW    (R3)                                            ;
                             06   A9              63  B1 001A8           CMPW    (R3), 6(R9)                                     ;
                                                5A  12 001AC           BNEQ    25$                                             ;
                                63        06    A9  B0 001AE 19$:      MOVW    6(R9), (R3)                                     ;
                                                7D  11 001B2 20$:      BRB     27$                                             ; 0593
                   3E       2F   A9              02  E1 001B4 21$:      BBC     #2, 47(R9), 24$                                 ; 0694
                                54              52  D0 001B9           MOVL    R2, CHAR                                        ; 0714
                                53        06    A9  3C 001BC           MOVZWL  6(R9), REMAINING_COLS                           ; 0715
                                50        28    A9  3C 001C0           MOVZWL  40(R9), R0                                      ;
                                53              50  C2 001C4           SUBL2   R0, REMAINING_COLS                              ;
                                50        28    A9  3C 001C7           MOVZWL  40(R9), R0                                      ;
                                                50  D7 001CB           DECL    R0                                              ;
                                51        06    A9  3C 001CD           MOVZWL  6(R9), R1                                       ;
                                50              51  C4 001D1           MULL2   R1, R0                                          ;
                                52        2A    A9  9E 001D4           MOVAB   42(R9), R2                                      ;
                                51              62  3C 001D8           MOVZWL  (R2), R1                                        ;
                                51        FF A140   9E 001DB           MOVAB   -1(R1)[R0], INDEX                               ;
                                                53  D5 001E0           TSTL    REMAINING_COLS                                  ;
                                                75  15 001E2           BLEQ    29$                                             ;
      53                54                    04        00  EF 001E4           EXTZV   #0, #4, CHAR, R3                        ;
                        53              53        04  78 001E9 22$:      ASHL    #4, R3, R3                                      ;
                                50              53  90 001ED           MOVB    R3, SHIFT_NIBBLE                                ;
                                                6C  11 001F0           BRB     30$                                             ;
                        3C       2F   A9              02  E0 001F2 23$:      BBS     #2, 47(R9), 28$                         ; 0729
                                50        28    A9  3C 001F7 24$:      MOVZWL  40(R9), R0                                      ; 0732
                                                50  D6 001FB           INCL    R0                                              ;
      50        4A    A9                    10        00  ED 001FD           CMPZV   #0, #16, 74(R9), R0                     ;
                                                05  19 00203           BLSS    26$                                             ;
                                          28    A9  B6 00205           INCW    40(R9)                                          ; 0734
                                                62  11 00208 25$:      BRB     32$                                             ;
                                                01  DD 0020A 26$:      PUSHL   #1                                              ; 0736
                                                01  DD 0020C           PUSHL   #1                                              ;
                                7E        06    A9  3C 0020E           MOVZWL  6(R9), -(SP)                                    ; 0741
                                50        4A    A9  3C 00212           MOVZWL  74(R9), R0                                      ; 0740
                                51        48    A9  3C 00216           MOVZWL  72(R9), R1                                      ;
                                50              51  C2 0021A           SUBL2   R1, R0                                          ;
                                          01    A0  9F 0021D           PUSHAB  1(R0)                                           ; 0739
                                7E        04    A9  3C 00220           MOVZWL  4(R9), -(SP)                                    ; 0738
                                7E        48    A9  3C 00224           MOVZWL  72(R9), -(SP)                                   ; 0737
                                                59  DD 00228           PUSHL   R9                                              ; 0736
                     00000000G  00              07  FB 0022A           CALLS   #7, SMG$$SCROLL_AREA                            ;
                                                39  11 00231 27$:      BRB     32$                                             ; 0729
```

```
                                53       06   A9   3C 00233 28$:    MOVZWL  6(R9), REMAINING_COLS                    ; 0746
                                50       28   A9   3C 00237         MOVZWL  40(R9), R0
                                53            50   C2 0023B         SUBL2   R0, REMAINING_COLS
                                50       28   A9   3C 0023E         MOVZWL  40(R9), R0
                                             50   D7 00242         DECL    R0
                                51       06   A9   3C 00244         MOVZWL  6(R9), R1
                                50            51   C4 00248         MULL2   R1, R0
                                52       2A   A9   9E 0024B         MOVAB   42(R9), R2
                                51            62   3C 0024F         MOVZWL  (R2), R1
                                51       FF A140   9E 00252         MOVAB   -1(R1)[R0], INDEX
                                             53   D5 00257         TSTL    REMAINING_COLS
                                             37   15 00259 29$:    BLEQ    34$
                                50           3F   92 0025B         MCOMB   #63, SHIFT_NIBBLE
                                             3B   11 0025E 30$:    BRB     36$
                                52       2A   A9   9E 00260 31$:    MOVAB   42(R9), R2                              ; 0761
                          05    2F   A9        02   E0 00264         BBS     #2, 47(R9), 33$                         ; 0759
                                62             01   B0 00269         MOVW    #1, (R2)                                ; 0761
                                             7D   11 0026C 32$:    BRB     40$
                                53       06   A9   3C 0026E 33$:    MOVZWL  6(R9), REMAINING_COLS                   ; 0763
                                50       28   A9   3C 00272         MOVZWL  40(R9), R0
                                53            50   C2 00276         SUBL2   R0, REMAINING_COLS
                                50       28   A9   3C 00279         MOVZWL  40(R9), R0
                                             50   D7 0027D         DECL    R0
                                51       06   A9   3C 0027F         MOVZWL  6(R9), R1
                                50            51   C4 00283         MULL2   R1, R0
                                51            62   3C 00286         MOVZWL  (R2), R1
                                51       FF A140   9E 00289         MOVAB   -1(R1)[R0], INDEX
                                             53   D5 0028E         TSTL    REMAINING_COLS
                                             06   14 00290         BGTR    35$
                           18   AE             57   D0 00292 34$:    MOVL    BYTES_REMAINING, WORK_OVERFLOW
                                             14   11 00296         BRB     37$
                                50           30   8E 00298 35$:    MNEGB   #48, SHIFT_NIBBLE
                              614B           50   90 0029B 36$:    MOVB    SHIFT_NIBBLE, (INDEX)[TEXT_BUF]
                                50       08   AC   9A 0029F         MOVZBL  ATTR_CODE, WORK_ATTR
                50        01              06   01   F0 002A3         INSV    #1, #6, #1, WORK_ATTR
                              614A           50   90 002A8         MOVB    WORK_ATTR, (INDEX)[ATTR_BUF]
                                             62   B6 002AC 37$:    INCW    (R2)
                           06   A9             62   B1 002AE         CMPW    (R2), 6(R9)
                                             37   12 002B2         BNEQ    40$
                                62       06   A9   B0 002B4         MOVW    6(R9), (R2)
                                             31   11 002B8         BRB     40$                                     ; 0593
                          08    34   A9        05   E0 002BA 38$:    BBS     #5, 52(R9), 39$                         ; 0783
                                50 00000000G   8F   D0 002BF         MOVL    #SMG$_STRTERESC, R0                     ; 0785
                                             04   002C6         RET
                                       1C    AE   9F 002C7 39$:    PUSHAB  LEN_OF_SEQUENCE                         ; 0791
                                7E           57   7D 002CA         MOVQ    BYTES_REMAINING, -(SP)                  ; 0792
                                             59   DD 002CD         PUSHL   R9                                      ; 0791
                     00000000G  00           04   FB 002CF         CALLS   #4, SMG$$SIM_TERM
                                36           50   E9 002D6         BLBC    STATUS, 44$                             ; 0795
                50              57       1C   AE   C3 002D9         SUBL3   LEN_OF_SEQUENCE, BYTES_REMAINING, R0    ; 0802
                                57       01   A0   9E 002DE         MOVAB   1(R0), BYTES_REMAINING
                50              58       1C   AE   C1 002E2         ADDL3   LEN_OF_SEQUENCE, IN_POINTER, R0         ; 0803
                                58       FF   A0   9E 002E7         MOVAB   -1(R0), IN_POINTER
                                             58   D6 002EB 40$:    INCL    IN_POINTER                              ; 0833
                                             57   D7 002ED         DECL    BYTES_REMAINING                         ; 0834
                                           FD2B   31 002EF         BRW     1$                                      ; 0474
                           06   A9        2A   A9   B1 002F2 41$:    CMPW    42(R9), 6(R9)                           ; 0837
```

```
                                 04  12 002F7         BNEQ    42$                          ;
                34   A9          02  88 002F9         BISB2   #2, 52(R9)                   ; 0839
                     06          6C  91 002FD 42$:    CMPB    (AP), #6                     ; 0841
                                 0A  1F 00300         BLSSU   43$                          ;
                          18     AC  D5 00302         TSTL    24(AP)                       ;
                                 05  13 00305         BEQL    43$                          ;
                18   BC   18     AE  D0 00307         MOVL    WORK_OVERFLOW, @OVERFLOW     ; 0843
                     50          01  D0 0030C 43$:    MOVL    #1, R0                       ; 0845
                                     04 0030F 44$:    RET                                  ; 0846

; Routine Size:  784 bytes,    Routine Base:  _SMG$CODE + 0000


;  774          0847  1 !<BLF/PAGE>
```

```
;  776          0848  1 END                       ! End of module SMG$$PUT_TEXT_TO_BUFFER
;  777          0849  1
;  778          0850  0 ELUDOM
```

```
;                                   PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  _SMG$DATA                     260  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _SMG$CODE                     784  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
```

```
;                               Library Statistics

;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time

;  _$255$DUA28:[SYSLIB]STARLET.L32;1      9776         5         0         581       00:01.0
;  _$255$DUA28:[SMGRTL.OBJ]RTLLIB.L32;1     36         0         0           8       00:00.1
;  _$255$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1    469        19         4          38       00:00.4
```

```
;                                   COMMAND QUALIFIERS

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

; Size:          784 code + 260 data bytes
; Run Time:          00:23.5
; Elapsed Time:      01:25.0
; Lines/CPU Min:      2170
; Lexemes/CPU-Min: 18446
; Memory Used:   354 pages
; Compilation Complete
```

SMGNUMTAB
LIS

SMGMSGPTR
LIS

SMGSCROLL
LIS

SMGMISC
LIS

SMGMSGTXT
LIS

SMGPUTENC
LIS

SMGPUTTEX
LIS

SMGSIMTRM
LIS

SMGNUMPAR
LIS

SMGPRVINP
LIS