```
RRRRRRRRRRR      UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF   FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF   FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF   FFFFFFFFFFFFFFF
RRR          RRR UUU          UUU  NNN         NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNN         NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNN         NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000          000 FFF               FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000          000 FFF               FFF
RRRRRRRRRRR      UUU          UUU  NNN   NNN   NNN  000          000 FFFFFFFFFFF       FFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN   NNN   NNN  000          000 FFFFFFFFFFF       FFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN   NNN   NNN  000          000 FFFFFFFFFFF       FFFFFFFFFFF
RRR   RRR        UUU          UUU  NNN       NNNNNN 000          000 FFF               FFF
RRR   RRR        UUU          UUU  NNN       NNNNNN 000          000 FFF               FFF
RRR   RRR        UUU          UUU  NNN       NNNNNN 000          000 FFF               FFF
RRR      RRR     UUU          UUU  NNN          NNN 000          000 FFF               FFF
RRR      RRR     UUU          UUU  NNN          NNN 000          000 FFF               FFF
RRR      RRR     UUU          UUU  NNN          NNN 000          000 FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
RRR         RRR  UUUUUUUUUUUUUUU  NNN          NNN    000000000     FFF               FFF
```

```
_$2

Sym
---
NDX
NDX
NUM
NUM
OPE
OUT
PAC
PAC
PAC
PAC
PAC
PAC
PAC
PAD
PAG
PAG
PAG
PAG
PAG
PAG
PAG
PER
PUT
RCO
RIN

RLI
RNO
RNO
RTY
SAV
STR
STR
STR
STR
STR
STR

STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
```

FILEID**STYLE

```
  SSSSSSSS  TTTTTTTTTT  YY      YY  LL          EEEEEEEEEE
  SSSSSSSS  TTTTTTTTTT  YY      YY  LL          EEEEEEEEEE
SS              TT      YY      YY  LL          EE
SS              TT      YY      YY  LL          EE
SS              TT        YY  YY    LL          EE
SS              TT        YY  YY    LL          EE
  SSSSSS        TT          YY      LL          EEEEEEEE
  SSSSSS        TT          YY      LL          EEEEEEEE
        SS      TT          YY      LL          EE
        SS      TT          YY      LL          EE
        SS      TT          YY      LL          EE              ....
        SS      TT          YY      LL          EE              ....
SSSSSSSS        TT          YY      LLLLLLLLLL  EEEEEEEEEE      ....
SSSSSSSS        TT          YY      LLLLLLLLLL  EEEEEEEEEE      ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1          0001  0 %TITLE 'Verify and collect attributes for .STYLE <entity> commands'
;     2          0002  0 MODULE STYLE (  IDENT = 'V04-000'
;     3        P 0003  0                 %BLISS32[, ADDRESSING_MODE( EXTERNAL    = LONG_RELATIVE
;     4          0004  0                                           ,NONEXTERNAL = LONG_RELATIVE)]
;     5          0005  0                 ) =
;     6          0006  1 BEGIN
;     7          0007  1 !
;     8          0008  1 !*****************************************************************************
;     9          0009  1 !*                                                                           *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                      *
;    13          0013  1 !*                                                                           *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    19          0019  1 !*  TRANSFERRED.                                                              *
;    20          0020  1 !*                                                                           *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    23          0023  1 !*  CORPORATION.                                                              *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*****************************************************************************
;    30          0030  1 !
;    31          0031  1 !++
;    32          0032  1 ! FACILITY:      DSR (Digital Standard RUNOFF) / DSRPLUS
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT: Processes the attributes of a .S.YLE <entity> command.
;    35          0035  1 !
;    36          0036  1 ! ENVIRONMENT: Transportable
;    37          0037  1 !
;    38          0038  1 ! AUTHOR: Keith Dawson  CREATION DATE: April 1982
;    39          0039  1 !
```

```
;    41          0040  1 %SBTTL 'Revision History'
;    42          0041  1 !
;    43          0042  1 ! MODIFIED BY:
;    44          0043  1 !
;    45          0044  1 !       011     REM0011         Ray Marshall    21-Jul-1983
;    46          0045  1 !               The pass/return linkages for non-BLISS32 were incorrect.  They
;    47          0046  1 !                 were made to match the way they are defined in ATTRIB.BLI.
;    48          0047  1 !               There was also a missing comma in that same code (above) in the
;    49          0048  1 !                 original that caused a compilation error on TOPS-20.  But,
;    50          0049  1 !                 seeing as that code was replaced, it no longer matters.
;    51          0050  1 !
;    52          0051  1 !       010     KFA00010        Ken Alden       17-May-1983
;    53          0052  1 !               Fixed testpage bug.
;    54          0053  1 !
;    55          0054  1 !       009     REM00009        Ray Marshall    07-Mar-1983
;    56          0055  1 !               Global edit of all modules. Updated module names, idents,
;    57          0056  1 !               copyright dates. Changed require files to BLISS library.
;    58          0057  1 !
;    59          0058  1 !--
```

```
;    61          0059  1 %SBTTL 'Module Level Declarations'
;    62          0060  1 !
;    63          0061  1 ! TABLE OF CONTENTS:
;    64          0062  1 !
;    65          0063  1 LINKAGE
;    66          0064  1 %IF (%BLISS(BLISS32) OR %BLISS(BLISS16)) %THEN
;    67          0065  1     pass_1_return_1 =
;    68          0066  1         CALL (REGISTER=1; REGISTER=2),
;    69          0067  1     pass_4_return_1 =
;    70          0068  1         CALL (REGISTER=1, REGISTER=2, REGISTER=3, REGISTER=4; REGISTER=5),
;    71          0069  1     pass_2_return_2 =
;    72          0070  1         CALL (REGISTER=1, REGISTER=2; REGISTER=3, REGISTER=4);
;    73        U 0071  1 %ELSE
;    74        U 0072  1     pass_1_return_1 =
;    75        U 0073  1         PUSHJ (; REGISTER=2) : LINKAGE_REGS (15,13,1),
;    76        U 0074  1     pass_4_return_1 =
;    77        U 0075  1         PUSHJ (; REGISTER=2) : LINKAGE_REGS (15,13,1),
;    78        U 0076  1     pass_2_return_2 =
;    79        U 0077  1         PUSHJ (; REGISTER=2, REGISTER=3) : LINKAGE_REGS (15,13,1);
;    80          0078  1 %FI
;    81          0079  1
;    82          0080  1 ! INCLUDE FILES:
;    83          0081  1 !
;    84          0082  1
;    85          0083  1 LIBRARY 'NXPORT:XPORT';                 ! XPORT Library
;    86          0084  1 REQUIRE 'REQ:RNODEF';                   ! RUNOFF variant definitions
;    87          0215  1
;    88        U 0216  1 %IF DSRPLUS %THEN
;    89        U 0217  1 LIBRARY 'REQ:DPLLIB';                 ! DSRPLUS BLISS Library
;    90          0218  1 %ELSE
;    91          0219  1 LIBRARY 'REQ:DSRLIB';                 ! DSR BLISS Library
;    92          0220  1 %FI
;    93          0221  1
;    94          0222  1 FORWARD ROUTINE
;    95          0223  1     STYLE,                          !Processes .STYLE { EXAMPLE ! FIGURE ! TABLE ! HEADER } commands.
;    96          0224  1     SET_STHL        : NOVALUE       !Sets up ECC from .STHL command.
;    97        U 0225  1 %IF DSRPLUS %THEN
;    98        U 0226  1                   ,
;    99        U 0227  1     SET_ONE_STYLE : NOVALUE,      !For one attribute, stores KWS information in the proper ECC.
;   100        U 0228  1     CHECK_HL_VALUES             , !Range-checks supplied HL-level values.
;   101        U 0229  1     INIT_STYLE_KW : NOVALUE;      !Initializes the $KWS_BLOCKVECTOR for the STYLE <entity> commands.
;   102          0230  1 %ELSE
;   103          0231  1                                 ;
;   104          0232  1 %FI
;   105          0233  1
;   106          0234  1 !
;   107          0235  1 ! MACROS:
;   108          0236  1 !
;   109          0237  1 ! OWN STORAGE:
;   110          0238  1 !
;   111          0239  1
;   112        U 0240  1 %IF DSRPLUS %THEN
;   113        U 0241  1 OWN
;   114        U 0242  1     FIRST_TIME : INITIAL (1),     !Have we ever set up keywords?
;   115        U 0243  1     HLS_CHECKED,                  !Have we checked HL-range validity during this call?
;   116        U 0244  1     KEYWORDS : $KWS_BLOCKVECTOR;
;   117          0245  1 %FI
```

```
;  118          0246  1
;  119          0247  1 !
;  120          0248  1 ! EXTERNAL REFERENCES:
;  121          0249  1 !
;  122          0250  1
;  123          0251  1 EXTERNAL
;  124          0252  1     ECC : $ECC_BLOCKVECTOR,
;  125          0253  1     FSO1 : FIXED_STRING,
;  126          0254  1     HLC : HLC_DEFINITION,
;  127          0255  1     HLLIST : COUNTED_LIST,
;  128          0256  1     IRA : FIXED_STRING,
;  129          0257  1     KHAR;
;  130          0258  1
;  131          0259  1 EXTERNAL LITERAL                                    !Error messages
;  132        U 0260  1 %IF DSRPLUS %THEN
;  133        U 0261  1     RNFTMA,               ! too many attributes
;  134          0262  1 %FI
;  135          0263  1     RNFILE;
;  136          0264  1
;  137          0265  1 EXTERNAL ROUTINE
;  138        U 0266  1 %IF DSRPLUS %THEN
;  139        U 0267  1     ATTSYN : pass_1_return_1,   !Verifies syntax of attribute list.
;  140        U 0268  1     DOATTR : pass_4_return_1,   !Collects one attribute and sets its value.
;  141        U 0269  1     ENDCMT,
;  142        U 0270  1     SKPSEP,
;  143          0271  1 %FI
;  144          0272  1     ECCINI,
;  145          0273  1     ERMS,
;  146          0274  1     STHL;
```

```
;  148     0275  1 %sbttl 'STYLE -- process all attributes for .STYLE <entity> command'
;  149     0276  1 GLOBAL ROUTINE STYLE (HANDLER) =
;  150     0277  1
;  151     0278  1 !++
;  152     0279  1 ! FUNCTIONAL DESCRIPTION:
;  153     0280  1 !
;  154     0281  1 !       STYLE verifies the .STYLE <entity> attributes given by the user.
;  155     0282  1 !       It writes descriptions of the verified attributes into the ECC
;  156     0283  1 !       global area.
;  157     0284  1 !
;  158     0285  1 !       AFTER[=n]       leave n blank lines after each caption.
;  159     0286  1 !       ALLCAP          make caption all-caps.
;  160     0287  1 !       BEFORE[=n]      leave n blank lines before each caption.
;  161     0288  1 !       BETWEEN[=n]     leave n blank lines between 'CHAPTER' or 'APPENDIX' & caption.
;  162     0289  1 !       BOLD            make entity caption bold.
;  163     0290  1 !       BREAK           start a new line between the counter and caption.
;  164     0291  1 !       CENTER          center entity caption (and don't run in).
;  165     0292  1 !       DEFAULTS        return all values to initial defaults.
;  166     0293  1 !       FIRSTCAP        make entity caption initial-caps.
;  167     0294  1 !       LEAVECASE       do not change case of entity caption.
;  168     0295  1 !       LEFT            make the counter and caption left-justified.
;  169     0296  1 !       LEVEL=n         (.STHL only) command applies to HL n.
;  170     0297  1 !       NORUNIN         do not run in caption with text.
;  171     0298  1 !       NOBOLD          do not make entity caption bold.
;  172     0299  1 !       NOBREAK         do not start a new line after the counter.
;  173     0300  1 !       NOPAGE          do not start a new page before a CHAPTER, etc.
;  174     0301  1 !       NOTOCPAGE       do not insert a page number in the TOC.
;  175     0302  1 !       NOSTARTODD      do not force the first page of a chapter on an odd page.
;  176     0303  1 !       NOUNDERLINE     do not underline this entity caption.
;  177     0304  1 !       NUMBERED        display counter for this entity.
;  178     0305  1 !       PAGE            start a new page before CHAPTER or APPENDIX.
;  179     0306  1 !       RUNIN           run in caption with text (and don't center).
;  180     0307  1 !       SPACES[=n]      leave n spaces between counter and caption.
;  181     0308  1 !       STARTODD        force the first page of a chapter to start on an odd page.
;  182     0309  1 !       TESTPAGE[=n]    issue implicit test-page n.
;  183     0310  1 !       TOCPAGE insert a page number in the TOC.
;  184     0311  1 !       UNDERLINE       make entity caption underlined.
;  185     0312  1 !       UNNUMBERED      do not display counter for this entity.
;  186     0313  1 !
;  187     0314  1 ! FORMAL PARAMETERS:
;  188     0315  1 !
;  189     0316  1 !       HANDLER         specifies which command is being processed:
;  190     0317  1 !                       .STYLE EXAMPLE, .STYLE FIGURE, .STYLE TABLE,
;  191     0318  1 !                       .STYLE APPENDIX, .STYLE CHAPTER or .STYLE HEADERS.
;  192     0319  1 !
;  193     0320  1 ! IMPLICIT INPUTS:      None
;  194     0321  1 !
;  195     0322  1 ! IMPLICIT OUTPUTS:
;  196     0323  1 !
;  197     0324  1 !       The routine updates the ECC block corresponding to the handler code.
;  198     0325  1 !
;  199     0326  1 ! ROUTINE VALUE:
;  200     0327  1 ! COMPLETION CODES:
;  201     0328  1 !
;  202     0329  1 !       The routine returns 0 for normal completion. It returns -1 if
;  203     0330  1 !       the user-supplied attributes cannot be deciphered (mismatched
;  204     0331  1 !       delimiters).
```

```
;  205          0332  1 !
;  206          0333  1 ! SIDE EFFECTS:         None
;  207          0334  1 !--
;  208          0335  1
;  209          0336  2     BEGIN
;  210          0337  2
;  211        U 0338  2 %IF DSRPLUS %THEN
;  212        U 0339  2     STACKLOCAL
;  213        U 0340  2         first_keyword,          !First legal keyword in the tables for this command.
;  214        U 0341  2         last_keyword,           !Last legal keyword for this command.
;  215        U 0342  2         number_of_keywords,     !How many keywords are legal for this command.
;  216        U 0343  2         kw_offset,              !Offset into the KEYWORDS tables, pointing to the first
;  217        U 0344  2                                 ! legal keyword for this command.
;  218        U 0345  2         end_delimiter,          !The matching ) ] > }.
;  219        U 0346  2         match,                  !Index of which keyword was found, from DOATTR.
;  220        U 0347  2         ecc_offset,             !Offset into the ECC block, pointing to data for this entity.
;  221        U 0348  2         old_syntax_sthl,        !True if command is an .STHL in the old, 9-number syntax.
;  222        U 0349  2         result,                 !Completion code from ATTSYN and DOATTR.
;  223        U 0350  2         start_where;            !Assures that giving no attributes re-establishes defaults.
;  224        U 0351  2
;  225        U 0352  2 !+
;  226        U 0353  2 ! Initialize
;  227        U 0354  2 !-
;  228        U 0355  2     init_style_kw (.handler);   !Set up KWS structure for this STYLE command.
;  229        U 0356  2
;  230        U 0357  2     hls_checked = 0;            !Indicate that we have not checked a HL-range yet.
;  231        U 0358  2
;  232        U 0359  2     end_delimiter = %C')';      !Preset the terminator character.
;  233        U 0360  2
;  234        U 0361  2     SELECT .handler OF
;  235        U 0362  2         SET
;  236        U 0363  2
;  237        U 0364  2         [h_style_appendi, h_style_chapter] :
;  238        U 0365  2                 BEGIN
;  239        U 0366  2                 first_keyword = kws$k_first_chapp;
;  240        U 0367  2                 last_keyword = kws$k_last_chapp;
;  241        U 0368  2                 END;
;  242        U 0369  2
;  243        U 0370  2         [h_style_appendi] :     ecc_offset = append_offset;
;  244        U 0371  2
;  245        U 0372  2         [h_style_chapter] :     ecc_offset = chap_offset;
;  246        U 0373  2
;  247        U 0374  2         [h_style_example, h_style_figure, h_style_table] :
;  248        U 0375  2                 BEGIN
;  249        U 0376  2                 first_keyword = kws$k_first_entity;
;  250        U 0377  2                 last_keyword = kws$k_last_entity;
;  251        U 0378  2                 END;
;  252        U 0379  2
;  253        U 0380  2         [h_style_example] :     ecc_offset = examp_offset;
;  254        U 0381  2
;  255        U 0382  2         [h_style_figure] :      ecc_offset = figur_offset;
;  256        U 0383  2
;  257        U 0384  2         [h_style_table] :       ecc_offset = table_offset;
;  258        U 0385  2
;  259        U 0386  2         [h_style_headers] :
;  260        U 0387  2                 !
;  261        U 0388  2                 BEGIN
```

```
;  262     U 0389  2                   ecc_offset = hcoll_offset;
;  263     U 0390  2                   first_keyword = kws$k_first_header;
;  264     U 0391  2                   last_keyword = kws$k_last_header;
;  265     U 0392  2                   !
;  266       0393  2 %FI
;  267       0394  2                   !Initialize the ECC collecting area for .STYLE HEADERS commands.
;  268       0395  2                   eccini (hcoll_offset, hcoll_offset, false);
;  269       0396  2
;  270     U 0397  2 %IF DSRPLUS %THEN
;  271     U 0398  2                   END;
;  272     U 0399  2           TES;
;  273       0400  2 %FI
;  274       0401  2
;  275       0402  2 %IF NOT DSRPLUS %THEN
;  276       0403  2 !+
;  277       0404  2 ! This command must be .STYLE HEADERS (the only legal .STYLE command
;  278       0405  2 ! handled by this module for DSR). It is an old-syntax .STHL, with up
;  279       0406  2 ! to 9 numeric arguments.
;  280       0407  2 !-
;  281       0408  2     IF .handler EQL h_style_headers     !Defensive check.
;  282       0409  2     THEN
;  283       0410  3         BEGIN
;  284       0411  3         sthl (.handler);                !Process the .STHL in the old way.
;  285       0412  3         set_sthl (ecc, 1, 6, true);
;  286       0413  3         RETURN 0;
;  287       0414  3         END
;  288       0415  2     ELSE
;  289       0416  2         erms (rnfile, CH$PTR(UPLIT('STYLE1')), 6);
;  290       0417  2 %FI
;  291       0418  2
;  292     U 0419  2 %IF DSRPLUS %THEN
;  293     U 0420  2     !
;  294     U 0421  2     ! Set the number of keywords legal for this command.
;  295     U 0422  2     number_of_keywords = .last_keyword - .first_keyword + 1;
;  296     U 0423  2     !
;  297     U 0424  2     ! Assume that any .STHL command will be in the new, keyworded format.
;  298     U 0425  2     old_syntax_sthl = false;
;  299     U 0426  2     !
;  300     U 0427  2     ! Preset starting point for attribute search.
;  301     U 0428  2     start_where = .first_keyword;
;  302     U 0429  2     !
;  303     U 0430  2     ! Preset the offset into the keyword tables that applies for this command.
;  304     U 0431  2     ! This allows DOATTR not to have to know the structure of the keyword tables.
;  305     U 0432  2     kw_offset = .first_keyword * (kws$k_length * %UPVAL);
;  306     U 0433  2
;  307     U 0434  2 !+
;  308     U 0435  2 ! Verify the syntax of the attribute-list.
;  309     U 0436  2 !-
;  310     U 0437  2     result = attsyn (.handler; end_delimiter);
;  311     U 0438  2
;  312     U 0439  2     CASE .result FROM min_attcc TO max_attcc OF
;  313     U 0440  2         SET
;  314     U 0441  2
;  315     U 0442  2         [attcc_ok] : 0;
;  316     U 0443  2
;  317     U 0444  2         [attcc_dcf_error] :
;  318     U 0445  2             !
```

```
;  319        U 0446  2            BEGIN
;  320        U 0447  2            endcmt ();                  !Skip rest of text in the input.
;  321        U 0448  2            RETURN -1;
;  322        U 0449  2            END;
;  323        U 0450  2
;  324        U 0451  2        [attcc_mmd_error] :
;  325        U 0452  2            !
;  326        U 0453  2            BEGIN
;  327        U 0454  2            endcmt ();                  !Skip rest of text in the input.
;  328        U 0455  2            RETURN -1;
;  329        U 0456  2            END;
;  330        U 0457  2
;  331        U 0458  2        [attcc_leading_number] :        !Old-syntax .STHL was given.
;  332        U 0459  2            !
;  333        U 0460  2            IF .handler EQL h_style_headers     !Defensive check -- ATTRIB is supposed
;  334        U 0461  2            THEN                                !to return this code only for .STHL commands.
;  335        U 0462  2                BEGIN
;  336        U 0463  2                sthl (.handler);                !Process the .STHL in the old way.
;  337        U 0464  2                old_syntax_sthl = true;
;  338        U 0465  2                start_where = max_keywords + 1;
;  339        U 0466  2                END
;  340        U 0467  2            ELSE
;  341        U 0468  2                erms (rnfile, CH$PTR(UPLIT('STYLE1')), 6);
;  342        U 0469  2
;  343        U 0470  2        [attcc_none] :                  !No attributes were supplied.
;  344        U 0471  2            !
;  345        U 0472  2            !Do not try to get any keywords; act as if user had given DEFAULTS keyword.
;  346        U 0473  2            start_where = max_keywords + 1;
;  347        U 0474  2
;  348        U 0475  2        [INRANGE, OUTRANGE] : erms (rnfile, CH$PTR(UPLIT('STYLE2')), 6);
;  349        U 0476  2
;  350        U 0477  2        TES;
;  351        U 0478  2    !+
;  352        U 0479  2    ! Collect one or more attributes.
;  353        U 0480  2    !-
;  354        U 0481  2        INCR K FROM .START_WHERE TO .LAST_KEYWORD DO
;  355        U 0482  2            BEGIN
;  356        U 0483  2            MATCH = -1;                 !Assume we will not find the keyword.
;  357        U 0484  2            !
;  358        U 0485  2            RESULT = DOATTR (.HANDLER, .END_DELIMITER, (KEYWORDS + .KW_OFFSET),
;  359        U 0486  2                            .NUMBER_OF_KEYWORDS; MATCH);
;  360        U 0487  2            !
;  361        U 0488  2            CASE .RESULT FROM MIN_ATTCC TO MAX_ATTCC OF
;  362        U 0489  2                SET
;  363        U 0490  2
;  364        U 0491  2                !We take no action for the following errors; they were already
;  365        U 0492  2                !reported by DOATTR.
;  366        U 0493  2                !
;  367        U 0494  2                [ATTCC_AMA_ERROR, ATTCC_IKW_ERROR] : 0;
;  368        U 0495  2                [ATTCC_NNA_ERROR, ATTCC_MQS_ERROR] : 0;
;  369        U 0496  2
;  370        U 0497  2                [ATTCC_OK, ATTCC_NVS_ERROR] :
;  371        U 0498  2                    !
;  372        U 0499  2                    !It's ok not to specify a value; the effect is to invoke the default value.
;  373        U 0500  2                    SET_ONE_STYLE (.ECC_OFFSET, ECC [.ECC_OFFSET, 0,0,0,0], (.MATCH + .FIRST_KEYWORD) );
;  374        U 0501  2
;  375        U 0502  2                [ATTCC_NONE] : EXITLOOP;    !End of attribute list detected.
```

```
;  376        U 0503  2
;  377        U 0504  2              [INRANGE, OUTRANGE] : ERMS (RNFILE, CH$PTR(UPLIT('STYLE2')), 6);
;  378        U 0505  2
;  379        U 0506  2              TES;
;  380        U 0507  2          !
;  381        U 0508  2          END;                                ! End of processing for 1 attribute.
;  382        U 0509  2  !+
;  383        U 0510  2  ! Process defaulted arguments of extra keywords (unless the user issued
;  384        U 0511  2  ! an old-syntax .STHL command).
;  385        U 0512  2  !-
;  386        U 0513  2      IF NOT .old_syntax_sthl  THEN
;  387        U 0514  2          BEGIN
;  388        U 0515  2          !
;  389        U 0516  2          ! If the user specified no attributes, act as if DEFAULT has been specified.
;  390        U 0517  2          !
;  391        U 0518  2          IF .START_WHERE NEQ .FIRST_KEYWORD  THEN
;  392        U 0519  2              !User supplied no attributes. Act as if he gave the DEFAULTS keyword.
;  393        U 0520  2              SET_ONE_STYLE (.ECC_OFFSET, ECC [.ECC_OFFSET, 0,0,0,0], KWS$K_DEFAULTS)
;  394        U 0521  2          ELSE
;  395        U 0522  2              !User supplied some attributes; we should have processed all of them
;  396        U 0523  2              !by now. If any remain, skip over them and note the error.
;  397        U 0524  2              BEGIN
;  398        U 0525  2              IF .KHAR NEQ .END_DELIMITER  THEN
;  399        U 0526  2                  BEGIN
;  400        U 0527  2                  FS_INIT (FS01);
;  401        U 0528  2                  WHILE .KHAR NEQ .END_DELIMITER DO
;  402        U 0529  2                      BEGIN
;  403        U 0530  2                      KCNS ();
;  404        U 0531  2                      FS_WCHAR (FS01, .KHAR);
;  405        U 0532  2                      END;
;  406        U 0533  2                  !
;  407        U 0534  2                  !'Too many attributes; "..." ignored'. The "-1" assures that the
;  408        U 0535  2                  !end-delimiter is not included in the error message.
;  409        U 0536  2                  ERMS (RNFTMA, .FS_START (FS01), .FS_LENGTH (FS01) - 1);
;  410        U 0537  2                  END
;  411        U 0538  2              ELSE
;  412        U 0539  2                  !End of string; get the terminator and position past the
;  413        U 0540  2                  !attributes string.
;  414        U 0541  2                  BEGIN
;  415        U 0542  2                  KCNS ();
;  416        U 0543  2                  SKPSEP (IRA);
;  417        U 0544  2                  END;
;  418        U 0545  2              END;
;  419        U 0546  2          END;
;  420        U 0547  2  !+
;  421        U 0548  2  ! An .STHL command gets processed through its own action routine.
;  422        U 0549  2  !-
;  423        U 0550  2      !
;  424        U 0551  2      IF .handler EQL h_style_headers  THEN
;  425        U 0552  2          BEGIN
;  426        U 0553  2          LOCAL
;  427        U 0554  2              first, last;
;  428        U 0555  2          !
;  429        U 0556  2          ! The ARG_LENGTH value is used to store the second number of a range.
;  430        U 0557  2          ! If no range was given, this value is zero.
;  431        U 0558  2          !
;  432        U 0559  2          IF .keywords [kws$k_level,kws$h_arg_length] EQL 0  THEN
```

```
;  433          U 0560  2              first = (last = .keywords [kws$k_level,kws$h_arg_value])
;  434          U 0561  2          ELSE
;  435          U 0562  2              ! User did supply a range.
;  436          U 0563  2              BEGIN
;  437          U 0564  2              first = .keywords [kws$k_level,kws$h_arg_value];
;  438          U 0565  2              last = .keywords [kws$k_level,kws$h_arg_length];
;  439          U 0566  2              END;
;  440          U 0567  2      !+
;  441          U 0568  2      ! Check the order of FIRST and LAST, flipping if necessary; range-check
;  442          U 0569  2      ! the numbers. The routine returns FALSE if either number is out of
;  443          U 0570  2      ! range; in this case no STHL values are set.
;  444          U 0571  2      !-
;  445          U 0572  2          IF check_hl_values (first, last)  THEN
;  446          U 0573  2              set_sthl (ecc, .first, .last, .old_syntax_sthl);
;  447          U 0574  2      !+
;  448          U 0575  2      ! The value(s) given by the user are OK. Perform the style change.
;  449          U 0576  2      !-
;  450          U 0577  2          END;
;  451            0578  2 %FI
;  452            0579  2      RETURN 0;
;  453            0580  2
;  454            0581  1      END;                                        !End of STYLE


                                                                           .TITLE  STYLE Verify and collect attributes for .STYLE
;                                                                                  <entit
                                                                           .IDENT  \V04-000\

                                                                           .PSECT  $PLIT$,NOWRT,NOEXE,2

                    00  00  31  45  4C  59  54  53  00000 P.AAA:           .ASCII  \STYLE1\<0><0>                                  ;

                                                                           .EXTRN  ECC, FS01, HLC, HLLIST
                                                                           .EXTRN  IRA, KHAR, RNFILE
                                                                           .EXTRN  ECCINI, ERMS, STHL

                                                                           .PSECT  $CODE$,NOWRT,2

                                                    0000 00000             .ENTRY  STYLE, Save nothing                             ; 0276
                            7E                      03   7D 00002         MOVQ    #3, -(SP)                                       ; 0395
                                                    03   DD 00005         PUSHL   #3
                00000000G   EF                      03   FB 00007         CALLS   #3, ECCINI                                      ;
                000000C4    8F          04          AC   D1 0000E         CMPL    HANDLER, #196                                   ; 0408
                                                    1F   12 00016         BNEQ    1$
                                        04          AC   DD 00018         PUSHL   HANDLER                                         ; 0411
                00000000G   EF                      01   FB 0001B         CALLS   #1, STHL                                        ;
                                                    01   DD 00022         PUSHL   #1                                              ; 0412
                                                    06   DD 00024         PUSHL   #6
                                                    01   DD 00026         PUSHL   #1
                                    00000000G       EF   9F 00028         PUSHAB  ECC
                00000000V   EF                      04   FB 0002E         CALLS   #4, SET_STHL                                    ;
                                                    15   11 00035         BRB     2$                                              ; 0413
                                                    06   DD 00037 1$:     PUSHL   #6                                              ; 0416
                                    00000000'       EF   9F 00039         PUSHAB  P.AAA                                           ;
                                    00000000G       8F   DD 0003F         PUSHL   #RNFILE                                         ;
                00000000G   EF                      03   FB 00045         CALLS   #3, ERMS                                        ;
                                                    50   D4 0004C 2$:     CLRL    R0                                              ; 0581
```

```
                                                D 11
                                        04 0004E        RET                                                     ;

; Routine Size:  79 bytes,    Routine Base:  $CODE$ + 0000

;  455        U 0582  1 %IF DSRPLUS %THEN
```

```
;  457          U 0583  1 %SBTTL 'SET_ONE_STYLE -- set ECC information for one valid attribute'
;  458          U 0584  1 ROUTINE SET_ONE_STYLE (OFFSET, STYLE_BLOCK, MATCH) : NOVALUE =
;  459          U 0585  1 !++
;  460          U 0586  1 !
;  461          U 0587  1 ! FUNCTIONAL DESCRIPTION:
;  462          U 0588  1 !
;  463          U 0589  1 !       SET_ONE_STYLE   is the action routine for attributes. It is called,
;  464          U 0590  1 !                       when a legal attribute is processed, to set the proper
;  465          U 0591  1 !                       ECC values.
;  466          U 0592  1 !
;  467          U 0593  1 ! FORMAL PARAMETERS:
;  468          U 0594  1 !
;  469          U 0595  1 !       OFFSET          indicates which command is being processed: it is the
;  470          U 0596  1 !                       offset into the ECC blockvector pointing to the command.
;  471          U 0597  1 !
;  472          U 0598  1 !       STYLE_BLOCK     is the address of the $ECC_BLOCK to update.
;  473          U 0599  1 !
;  474          U 0600  1 !       MATCH           is the index into the $KWS_BLOCK that corresponds
;  475          U 0601  1 !                       to the attribute we are processing.
;  476          U 0602  1 !
;  477          U 0603  1 ! IMPLICIT INPUTS:      None
;  478          U 0604  1 !
;  479          U 0605  1 ! IMPLICIT OUTPUTS:
;  480          U 0606  1 !
;  481          U 0607  1 !       One piece of ECC information is updated.
;  482          U 0608  1 !
;  483          U 0609  1 ! ROUTINE VALUE:
;  484          U 0610  1 ! COMPLETION CODES:     None
;  485          U 0611  1 !
;  486          U 0612  1 ! SIDE EFFECTS:         None
;  487          U 0613  1 !--
;  488          U 0614  1     BEGIN
;  489          U 0615  1
;  490          U 0616  1     BIND KW = KEYWORDS [.MATCH, 0,0,0,0] : $KWS_BLOCK;
;  491          U 0617  1
;  492          U 0618  1     BIND SB = .STYLE_BLOCK : $ECC_BLOCK;
;  493          U 0619  1
;  494          U 0620  1     LOCAL
;  495          U 0621  1         NUMBER,
;  496          U 0622  1         SIGN;
;  497          U 0623  1
;  498          U 0624  1     NUMBER = .KW [KWS$H_ARG_VALUE];
;  499          U 0625  1     SIGN = .KW [KWS$H_ARG_SIGN];
;  500          U 0626  1
;  501          U 0627  1 !+
;  502          U 0628  1 ! We have a legal attribute. Set information in the ECC structure.
;  503          U 0629  1 !-
;  504          U 0630  1     SELECT .MATCH OF
;  505          U 0631  1         SET
;  506          U 0632  1
;  507          U 0633  1         [KWS$K_AFTER] :
;  508          U 0634  1                 !
;  509          U 0635  1                 BEGIN
;  510          U 0636  1                 LOCAL
;  511          U 0637  1                     res_number;
;  512          U 0638  1
;  513          U 0639  1                 res_number =
```

```
;  514          U 0640  1                    (IF   .sign EQL 0  THEN   .number
;  515          U 0641  1                     ELSE   .sb [ecc$h_after] + .number );
;  516          U 0642  1
;  517          U 0643  1                IF .RES_NUMBER LSS 0  THEN
;  518          U 0644  1                    RES_NUMBER = 0;
;  519          U 0645  1
;  520          U 0646  1                SB [ECC$H_AFTER] = .RES_NUMBER;
;  521          U 0647  1                END;
;  522          U 0648  1
;  523          U 0649  1           [KWS$K_ALLCAP] : SB [ECC$H_CASE] = ECC$K_ALLCAP;
;  524          U 0650  1
;  525          U 0651  1           [KWS$K_BEFORE] :
;  526          U 0652  1                !
;  527          U 0653  1                BEGIN
;  528          U 0654  1                LOCAL
;  529          U 0655  1                    RES_NUMBER;
;  530          U 0656  1
;  531          U 0657  1                RES_NUMBER =
;  532          U 0658  1                    (IF .SIGN EQL 0 THEN .NUMBER
;  533          U 0659  1                     ELSE .SB [ECC$H_BEFORE] + .NUMBER);
;  534          U 0660  1
;  535          U 0661  1                IF .RES_NUMBER LSS 0  THEN
;  536          U 0662  1                    RES_NUMBER = 0;
;  537          U 0663  1
;  538          U 0664  1                SB [ECC$H_BEFORE] = .RES_NUMBER;
;  539          U 0665  1                END;
;  540          U 0666  1
;  541          U 0667  1           [KWS$K_BETWEEN] :
;  542          U 0668  1                !
;  543          U 0669  1                BEGIN
;  544          U 0670  1                LOCAL
;  545          U 0671  1                    RES_NUMBER;
;  546          U 0672  1
;  547          U 0673  1                RES_NUMBER =
;  548          U 0674  1                    (IF .SIGN EQL 0 THEN .NUMBER
;  549          U 0675  1                     ELSE .SB [ECC$H_BETWEEN] + .NUMBER);
;  550          U 0676  1
;  551          U 0677  1                IF   .RES_NUMBER LSS 0  THEN
;  552          U 0678  1                    RES_NUMBER = 0;
;  553          U 0679  1
;  554          U 0680  1                SB [ECC$H_BETWEEN] = .RES_NUMBER;
;  555          U 0681  1                IF   .RES_NUMBER GTR 0  THEN
;  556          U 0682  1                    SB [ECC$V_BREAK] = TRUE;
;  557          U 0683  1                END;
;  558          U 0684  1
;  559          U 0685  1           [KWS$K_BOLD] : SB [ECC$V_BOLD] = TRUE;
;  560          U 0686  1
;  561          U 0687  1           [KWS$K_BREAK] :
;  562          U 0688  1                !
;  563          U 0689  1                BEGIN
;  564          U 0690  1                SB [ECC$V_BREAK] = TRUE;
;  565          U 0691  1                SB [ECC$H_RUNIN] = 0;
;  566          U 0692  1                END;
;  567          U 0693  1
;  568          U 0694  1           [KWS$K_CENTER] :
;  569          U 0695  1                !
;  570          U 0696  1                BEGIN
```

```
;  571          U 0697  1                   SB [ECC$H_POSITION] = ECC$K_CENTER;
;  572          U 0698  1                   SB [ECC$H_RUNIN] = 0;
;  573          U 0699  1                   END;
;  574          U 0700  1
;  575          U 0701  1           [KWS$K_DEFAULTS] :
;  576          U 0702  1                   !
;  577          U 0703  1                   IF  .OFFSET NEQ HCOLL_OFFSET  THEN      ! Initialize style info.
;  578          U 0704  1                       ECCINI (.OFFSET, .OFFSET, FALSE)    ! for one <entity>.
;  579          U 0705  1                   ELSE
;  580          U 0706  1                       BEGIN                               !Initialize style information for
;  581          U 0707  1                       LOCAL                               ! one or a range of header-levels.
;  582          U 0708  1                           FIRST, LAST;
;  583          U 0709  1                       !
;  584          U 0710  1                       ! The ARG_LENGTH value is used to store the second number of a range.
;  585          U 0711  1                       ! If no range was given, this value is zero.
;  586          U 0712  1                       !
;  587          U 0713  1                       IF .KEYWORDS [KWS$K_LEVEL,KWS$H_ARG_LENGTH] EQL 0  THEN
;  588          U 0714  1                           FIRST = (LAST = .KEYWORDS [KWS$R_LEVEL,KWS$H_ARG_VALUE])
;  589          U 0715  1                       ELSE
;  590          U 0716  1                           ! User did supply a range.
;  591          U 0717  1                           BEGIN
;  592          U 0718  1                           FIRST = .KEYWORDS [KWS$K_LEVEL, KWS$H_ARG_VALUE];
;  593          U 0719  1                           LAST = .KEYWORDS [KWS$K_LEVEL, KWS$H_ARG_LENGTH];
;  594          U 0720  1                           END;
;  595          U 0721  1                       !
;  596          U 0722  1                       ! Range-check the HL values.
;  597          U 0723  1                       IF  CHECK_HL_VALUES (FIRST, LAST)  THEN
;  598          U 0724  1                           BEGIN
;  599          U 0725  1                           FIRST = .FIRST + HCOLL_OFFSET;
;  600          U 0726  1                           LAST = .LAST + HCOLL_OFFSET;
;  601          U 0727  1                           ECCINI (.FIRST, .LAST, FALSE);
;  602          U 0728  1                           END;
;  603          U 0729  1                       END;
;  604          U 0730  1
;  605          U 0731  1           [KWS$K_FIRSTCAP] : SB [ECC$H_CASE] = ECC$K_FIRSTCAP;
;  606          U 0732  1
;  607          U 0733  1           [KWS$K_LEAVECASE] : SB [ECC$H_CASE] = ECC$K_LEAVECASE;
;  608          U 0734  1
;  609          U 0735  1           [KWS$K_LEFT]    : SB [ECC$H_POSITION] = ECC$K_LEFT;
;  610          U 0736  1
;  611          U 0737  1           [KWS$K_NOBOLD]  : SB [ECC$V_BOLD] = FALSE;
;  612          U 0738  1
;  613          U 0739  1           [KWS$K_NOBREAK] : SB [ECC$V_BREAK] = FALSE;
;  614          U 0740  1
;  615          U 0741  1           [KWS$K_NOPAGE]  :
;  616          U 0742  1                   !
;  617          U 0743  1                   BEGIN
;  618          U 0744  1                   SB [ECC$V_PAGE] = FALSE;
;  619          U 0745  1                   SB [ECC$V_STARTODD] = FALSE;
;  620          U 0746  1                   END;
;  621          U 0747  1
;  622          U 0748  1           [KWS$K_NOTOCPAGE] : SB [ECC$V_TOCPAGE] = FALSE;
;  623          U 0749  1
;  624          U 0750  1           [KWS$K_NOUNDERLINE] : SB [ECC$V_UNDERLINE] = FALSE;
;  625          U 0751  1
;  626          U 0752  1           [KWS$K_NORUNIN] : SB [ECC$H_RUNIN] = 0;
;  627          U 0753  1
```

```
;  628        U 0754  1          [KWS$K_NOSTARTODD] : SB [ECC$V_STARTODD] = FALSE;
;  629        U 0755  1
;  630        U 0756  1          [KWS$K_NUMBERED] : SB [ECC$V_UNNUMBERED] = FALSE;
;  631        U 0757  1
;  632        U 0758  1          [KWS$K_PAGE]     : SB [ECC$V_PAGE] = TRUE;
;  633        U 0759  1
;  634        U 0760  1          [KWS$K_RIGHT] :
;  635        U 0761  1                  !
;  636        U 0762  1                  BEGIN
;  637        U 0763  1                  SB [ECC$H_POSITION] = ECC$K_RIGHT;
;  638        U 0764  1                  SB [ECC$H_RUNIN] = 0;
;  639        U 0765  1                  END;
;  640        U 0766  1
;  641        U 0767  1          [KWS$K_RUNIN] :
;  642        U 0768  1                  !
;  643        U 0769  1                  BEGIN
;  644        U 0770  1                  SB [ECC$H_RUNIN] = 1;
;  645        U 0771  1                  SB [ECC$H_POSITION] = ECC$K_LEFT;
;  646        U 0772  1                  END;
;  647        U 0773  1
;  648        U 0774  1          [KWS$K_SPACES] :
;  649        U 0775  1                  !
;  650        U 0776  1                  BEGIN
;  651        U 0777  1                  LOCAL
;  652        U 0778  1                      RES_NUMBER;
;  653        U 0779  1
;  654        U 0780  1                  RES_NUMBER =
;  655        U 0781  1                      (IF .SIGN EQL 0 THEN .NUMBER
;  656        U 0782  1                       ELSE .SB [ECC$H_SPACES] + .NUMBER);
;  657        U 0783  1
;  658        U 0784  1                  IF  .RES_NUMBER LSS 0  THEN
;  659        U 0785  1                      RES_NUMBER = 0;
;  660        U 0786  1
;  661        U 0787  1                  SB [ECC$H_SPACES] = .RES_NUMBER;
;  662        U 0788  1                  END;
;  663        U 0789  1
;  664        U 0790  1          [KWS$K_STARTODD] :
;  665        U 0791  1                  !
;  666        U 0792  1                  BEGIN
;  667        U 0793  1                  SB [ECC$V_STARTODD] = TRUE;
;  668        U 0794  1                  SB [ECC$V_PAGE] = TRUE;
;  669        U 0795  1                  END;
;  670        U 0796  1
;  671        U 0797  1          [KWS$K_TESTPAGE] :
;  672        U 0798  1                  !
;  673        U 0799  1                  BEGIN
;  674        U 0800  1                  LOCAL
;  675        U 0801  1                      RES_NUMBER;
;  676        U 0802  1                  !If this is for a chapter-like header, NOPAGE must be in effect.
;  677        U 0803  1                  IF (.offset NEQ chap_offset)    OR
;  678        U 0804  1                     (.offset NEQ append_offset)  OR
;  679        U 0805  1                     (.OFFSET GEQ CHAP_OFFSET AND  .SB [ECC$V_PAGE] = FALSE)
;  680        U 0806  1                  THEN
;  681        U 0807  1                     BEGIN
;  682        U 0808  1                     RES_NUMBER =
;  683        U 0809  1                         (IF .SIGN EQL 0 THEN .NUMBER
;  684        U 0810  1                          ELSE .SB [ECC$H_TESTPAGE] + .NUMBER);
```

```
;  685          U 0811  1
;  686          U 0812  1                    IF  .RES_NUMBER LSS 0  THEN
;  687          U 0813  1                        RES_NUMBER = 0;
;  688          U 0814  1
;  689          U 0815  1                    SB [ECC$H_TESTPAGE] = .RES_NUMBER;
;  690          U 0816  1                    END;
;  691          U 0817  1                END;
;  692          U 0818  1
;  693          U 0819  1            [KWS$K_TOCPAGE] : SB [ECC$V_TOCPAGE] = TRUE;
;  694          U 0820  1
;  695          U 0821  1            [KWS$K_UNDERLINE] : SB [ECC$V_UNDERLINE] = TRUE;
;  696          U 0822  1
;  697          U 0823  1            [KWS$K_UNNUMBERED] : SB [ECC$V_UNNUMBERED] = TRUE;
;  698          U 0824  1
;  699          U 0825  1            [KWS$K_LEVEL] : 0;      !Nothing to do; the value already resides
;  700          U 0826  1                                    !in the KWS, from which it will be picked
;  701          U 0827  1                                    !up by SET_STHL.
;  702          U 0828  1
;  703          U 0829  1            [OTHERWISE] : ERMS (RNFILE, CH$PTR(UPLIT('SET_ONE_STYLE')), 13);
;  704          U 0830  1
;  705          U 0831  1            TES;
;  706          U 0832  1
;  707          U 0833  1        END;                                    ! End of SET_ONE_STYLE
;  708            0834  1 %FI
```

```
;  710          0835  1 %SBTTL 'SET_STHL -- Set up ECC from one .STYLE HEADERS command'
;  711          0836  1 ROUTINE SET_STHL (STYLE_BLOCK, FIRST_HL, LAST_HL, OLD_SYNTAX) : NOVALUE =
;  712          0837  1 !++
;  713          0838  1 !
;  714          0839  1 ! FUNCTIONAL DESCRIPTION:
;  715          0840  1 !
;  716          0841  1 !       SET_STHL        moves the collected .STYLE HEADER information from its
;  717          0842  1 !                       collecting place to the appropriate level of the ECC.
;  718          0843  1 !                       This is done in one of two ways:
;  719          0844  1 !
;  720          0845  1 !                       1) For a new-syntax, keyworded .STHL, information is moved
;  721          0846  1 !                          from ECC [HCOLL_OFFSET] to the proper level in the ECC.
;  722          0847  1 !
;  723          0848  1 !                       2) For an old-syntax, numbered .STHL, information is picked
;  724          0849  1 !                          up from the HLC structure and moved into the ECC.
;  725          0850  1 !
;  726          0851  1 ! FORMAL PARAMETERS:
;  727          0852  1 !
;  728          0853  1 !       STYLE_BLOCK     is the address of the $ECC_BLOCK to update.
;  729          0854  1 !
;  730          0855  1 !       FIRST_HL        is the number of the first level of header to update (from 1 to 6).
;  731          0856  1 !
;  732          0857  1 !       LAST_HL         is the number of the last level to update (from .FIRST_HL to 6).
;  733          0858  1 !
;  734          0859  1 !       OLD_SYNTAX      is TRUE if the .STHL command was in the old, numbered syntax.
;  735          0860  1 !
;  736          0861  1 ! IMPLICIT INPUTS:      None
;  737          0862  1 !
;  738          0863  1 ! IMPLICIT OUTPUTS:
;  739          0864  1 !
;  740          0865  1 !       One or more ECC fields are updated (new-syntax .STHL), or the entire
;  741          0866  1 !       header portion of the ECC_BLOCKVECTOR -- 6 $ECC_BLOCKs -- is replaced.
;  742          0867  1 !
;  743          0868  1 ! ROUTINE VALUE:
;  744          0869  1 ! COMPLETION CODES:     None
;  745          0870  1 !
;  746          0871  1 ! SIDE EFFECTS:        None
;  747          0872  1 !--
;  748          0873  1
;  749          0874  2     BEGIN
;  750          0875  2
;  751          0876  2     MAP
;  752          0877  2         STYLE_BLOCK : REF $ECC_BLOCKVECTOR;
;  753          0878  2
;  754          0879  2     BIND
;  755          0880  2         COL_BL  = STYLE_BLOCK [HCOLL_OFFSET, 0,0,0,0]
;  756          0881  2                 : $ECC_BLOCK,           !keyword information has been collected here.
;  757          0882  2
;  758          0883  2         OLD_BV  = STYLE_BLOCK [HCOLL_OFFSET, 0,0,0,0]
;  759          0884  2                 : $ECC_BLOCKVECTOR;     !Overwrite here, if an old-syntax .STHL.
;  760          0885  2
;  761          0886  2     MACRO
;  762        M 0887  2         move_bit (offset) =
;  763          0888  2             NEW_BL [offset] = (.NEW_BL [offset] OR .COL_BL [offset]) %,
;  764          0889  2
;  765        M 0890  2         move_numeric (offset) =
;  766        M 0891  2             IF .COL_BL [offset] NEQ -1  THEN
```

```
;  767     0892  2                 NEW_BL [offset] = .COL_BL [offset] %;
;  768     0893  2
;  769     0894  2
;  770     0895  2     IF NOT .OLD_SYNTAX  THEN
;  771     0896  2 !+
;  772     0897  2 ! The user issued a new-syntax, keyworded .STHL command. We have collected
;  773     0898  2 ! information from his keywords in the HCOLL_OFFSET of the ECC. Now move the
;  774     0899  2 ! collected information into the target ECC area.
;  775     0900  2 !-
;  776     0901  2         INCR I FROM .FIRST_HL TO .LAST_HL DO
;  777     0902  2             !
;  778     0903  3             BEGIN
;  779     0904  3             !Trarsfer collected keyword information to here.
;  780     0905  3             BIND NEW_BL =
;  781     0906  3                 STYLE_BLOCK [HCOLL_OFFSET + .I, 0,0,0,0] : $ECC_BLOCK;
;  782     0907  3             !+
;  783     0908  3             ! First move the bit information.
;  784     0909  3             !-
;  785     0910  3                 MOVE_BIT (ECC$V_BOLD);
;  786     0911  3                 MOVE_BIT (ECC$V_UNDERLINE);
;  787     0912  3                 MOVE_BIT (ECC$V_UNNUMBERED);
;  788     0913  3                 MOVE_BIT (ECC$V_BREAK);
;  789     0914  3                 MOVE_BIT (ECC$V_PAGE);
;  790     0915  3                 MOVE_BIT (ECC$V_TOCPAGE);
;  791     0916  3                 MOVE_BIT (ECC$V_STARTODD);
;  792     0917  3             !+
;  793     0918  3             ! Now move the numeric-value information.
;  794     0919  3             !-
;  795     0920  3                 MOVE_NUMERIC (ECC$H_RUNIN);
;  796     0921  3                 MOVE_NUMERIC (ECC$H_CASE);
;  797     0922  3                 MOVE_NUMERIC (ECC$H_POSITION);
;  798     0923  3                 MOVE_NUMERIC (ECC$H_AFTER);
;  799     0924  3                 MOVE_NUMERIC (ECC$H_BEFORE);
;  800     0925  3                 MOVE_NUMERIC (ECC$H_BETWEEN);
;  801     0926  3                 MOVE_NUMERIC (ECC$H_SPACES);
;  802     0927  3                 MOVE_NUMERIC (ECC$H_TESTPAGE);
;  803     0928  3                 END
;  804     0929  2     ELSE
;  805     0930  2 !+
;  806     0931  2 ! The user gave an old-syntax, numbered .STHL command. Overwrite the entire
;  807     0932  2 ! ECC header area -- all 6 levels -- with the results of that old .STHL.
;  808     0933  2 ! Note that this means the user loses the benefit of some of the advanced-
;  809     0934  2 ! syntax, keyword features: bolding and underlining, for example, are undone.
;  810     0935  2 !-
;  811     0936  2         INCR I FROM 1 TO 6 DO
;  812     0937  3             BEGIN
;  813     0938  3
;  814     0939  3             OLD_BV [.I, ECC$V_BOLD] = FALSE;
;  815     0940  3             OLD_BV [.I, ECC$V_UNDERLINE] = FALSE;
;  816     0941  3             OLD_BV [.I, ECC$V_UNNUMBERED] = (.I GEQ .HLC_NO_NUMBER);
;  817     0942  3
;  818     0943  3             OLD_BV [.I, ECC$H_RUNIN] = (.I GEQ .HLC_RUNON);
;  819     0944  3
;  820     0945  3             OLD_BV [.I, ECC$H_CASE] =
;  821     0946  4                 (IF  .I LEQ .HLC_UPPER  THEN
;  822     0947  4                     ECC$K_ALLCAP
;  823     0948  4                 ELSE
```

```
;  824     0949  4                    IF  .I LEQ .HLC_CAP  THEN
;  825     0950  4                        ECC$K_FIRSTCAP
;  826     0951  4                    ELSE
;  827     0952  3                        ECC$K_LEAVECASE);
;  828     0953  3
;  829     0954  3                OLD_BV [.I, ECC$H_POSITION] =
;  830     0955  4                    (IF .I GEQ .HLC_CENTER  THEN
;  831     0956  4                        ECC$K_CENTER
;  832     0957  4                     ELSE
;  833     0958  3                        ECC$K_LEFT);
;  834     0959  3
;  835     0960  3                OLD_BV [.I, ECC$H_AFTER] = .HLC_HEADLC;
;  836     0961  3                OLD_BV [.I, ECC$H_BEFORE] = .HLC_HEADLB;
;  837     0962  3                OLD_BV [.I, ECC$H_SPACES] = .HLC_HEADSP;
;  838     0963  3                OLD_BV [.I, ECC$H_TESTPAGE] = .HLC_HEADLT;
;  839     0964  3
;  840     0965  2                END;
;  841     0966  2
;  842     0967  1     END;                                    ! End of SET_STHL
```

```
                                003C 00000 SET_STHL:
                                           .WORD   Save R2,R3,R4,R5                    ; 0836
                55 00000000G   EF  9E 00002        MOVAB   HLC+28, R5                  ;
          52    04   AC 0000006C  8F  C1 00009     ADDL3   #108, STYLE_BLOCK, R2       ; 0880
                     03    10   AC  E9 00012     BLBC    OLD_SYNTAX, 1$              ; 0895
                             010F  31 00016     BRW     11$                         ;
          50    08   AC        01  C3 00019 1$: SUBL3   #1, FIRST_HL, I             ; 0901
                             00FF  31 0001E     BRW     10$                         ;
          51         50        24  C5 00021 2$: MULL3   #36, I, R1                  ; 0906
                     51    04   AC  C0 00025     ADDL2   STYLE_BLOCK, R1             ;
                     51    6C   A1  9E 00029     MOVAB   108(R1), R1                 ;
    53    61         01        00  EF 0002D     EXTZV   #0, #1, (R1), R3            ; 0910
    54    62         01        00  EF 00032     EXTZV   #0, #1, (R2), R4            ;
                     53        54  88 00037     BISB2   R4, R3                      ;
    61    01         00        53  F0 0003A     INSV    R3, #0, #1, (R1)            ;
    53    61         01        01  EF 0003F     EXTZV   #1, #1, (R1), R3            ; 0911
    54    62         01        01  EF 00044     EXTZV   #1, #1, (R2), R4            ;
                     53        54  88 00049     BISB2   R4, R3                      ;
    61    01         01        53  F0 0004C     INSV    R3, #1, #1, (R1)            ;
    53    61         01        02  EF 00051     EXTZV   #2, #1, (R1), R3            ; 0912
    54    62         01        02  EF 00056     EXTZV   #2, #1, (R2), R4            ;
                     53        54  88 0005B     BISB2   R4, R3                      ;
    61    01         02        53  F0 0005E     INSV    R3, #2, #1, (R1)            ;
    53    61         01        03  EF 00063     EXTZV   #3, #1, (R1), R3            ; 0913
    54    62         01        03  EF 00068     EXTZV   #3, #1, (R2), R4            ;
                     53        54  88 0006D     BISB2   R4, R3                      ;
    61    01         03        53  F0 00070     INSV    R3, #3, #1, (R1)            ;
    53    61         01        05  EF 00075     EXTZV   #5, #1, (R1), R3            ; 0914
    54    62         01        05  EF 0007A     EXTZV   #5, #1, (R2), R4            ;
                     53        54  88 0007F     BISB2   R4, R3                      ;
    61    01         05        53  F0 00082     INSV    R3, #5, #1, (R1)            ;
    53    61         01        04  EF 00087     EXTZV   #4, #1, (R1), R3            ; 0915
    54    62         01        04  EF 0008C     EXTZV   #4, #1, (R2), R4            ;
```

```
                                                53          54  88 00091          BISB2   R4, R3
        61              01                      04          53  F0 00094          INSV    R3, #4, #1, (R1)
        53              61                      01          06  EF 00099          EXTZV   #6, #1, (R1), R3                    ; 0916
        54              62                      01          06  EF 0009E          EXTZV   #6, #1, (R2), R4
                                                53          54  88 000A3          BISB2   R4, R3
        61              01                      06          53  F0 000A6          INSV    R3, #6, #1, (R1)
                                  FF    8F      02          A2  91 000AB          CMPB    2(R2), #-1                          ; 0920
                                                            05  13 000B0          BEQL    3$
                                  02    A1      02          A2  90 000B2          MOVB    2(R2), 2(R1)
                                  FF    8F      03          A2  91 000B7 3$:      CMPB    3(R2), #-1                          ; 0921
                                                            05  13 000BC          BEQL    4$
                                  03    A1      03          A2  90 000BE          MOVB    3(R2), 3(R1)
                                  FF    8F      04          A2  91 000C3 4$:      CMPB    4(R2), #-1                          ; 0922
                                                            05  13 000C8          BEQL    5$
                                  04    A1      04          A2  90 000CA          MOVB    4(R2), 4(R1)
                                FFFF    8F      05          A2  B1 000CF 5$:      CMPW    5(R2), #-1                          ; 0923
                                                            05  13 000D5          BEQL    6$
                                  05    A1      05          A2  B0 000D7          MOVW    5(R2), 5(R1)
        53      04      A2                      10          18  EE 000DC 6$:      EXTV    #24, #16, 4(R2), R3                 ; 0924
                        FFFFFFFF    8F                      53  D1 000E2          CMPL    R3, #-1
                                                            06  13 000E9          BEQL    7$
    04  A1              10                      18          53  F0 000EB          INSV    R3, #24, #16, 4(R1)
                                FFFF    8F      09          A2  B1 000F1 7$:      CMPW    9(R2), #-1                          ; 0925
                                                            05  13 000F7          BEQL    8$
                                  09    A1      09          A2  B0 000F9          MOVW    9(R2), 9(R1)
        53      08      A2                      10          18  EE 000FE 8$:      EXTV    #24, #16, 8(R2), R3                 ; 0926
                        FFFFFFFF    8F                      53  D1 00104          CMPL    R3, #-1
                                                            06  13 0010B          BEQL    9$
    08  A1              10                      18          53  F0 0010D          INSV    R3, #24, #16, 8(R1)
                                FFFF    8F      0D          A2  B1 00113 9$:      CMPW    13(R2), #-1                         ; 0927
                                                            05  13 00119          BEQL    10$
                                  0D    A1      0D          A2  B0 0011B          MOVW    13(R2), 13(R1)
  FEFA          50                      01      0C          AC  F1 00120 10$:     ACBL    LAST_HL, #1, I, 2$                  ; 0901
                                                            04  00127          RET
                                                51          01  D0 00128 11$:     MOVL    #1, I                               ; 0963
                50                              51          24  C5 0012B 12$:     MULL3   #36, I, R0                          ; 0939
                54                              52          50  C1 0012F          ADDL3   R0, R2, R4
                                                64          03  8A 00133          BICB2   #3, (R4)                            ; 0940
                                                53          D4  00136          CLRL    R3                                  ; 0941
                                                65          51  D1 00138          CMPL    I, HLC+28
                                                            02  19 0013B          BLSS    13$
                                                            53  D6 0013D          INCL    R3
    64          01                              02          53  F0 0013F 13$:     INSV    R3, #2, #1, (R4)
                                                            53  D4 00144          CLRL    R3                                  ; 0943
                                  EC    A5                  51  D1 00146          CMPL    I, HLC+8
                                                            02  19 0014A          BLSS    14$
                                                            53  D6 0014C          INCL    R3
                                  02    A4                  53  90 0014E 14$:     MOVB    R3, 2(R4)
                                  E4    A5                  51  D1 00152          CMPL    I, HLC                              ; 0946
                                                            05  14 00156          BGTR    15$
                                                53          02  D0 00158          MOVL    #2, R3
                                                            0D  11 0015B          BRB     17$
                                  E8    A5                  51  D1 0015D 15$:     CMPL    I, HLC+4                            ; 0949
                                                            05  14 00161          BGTR    16$
                                                53          01  D0 00163          MOVL    #1, R3
                                                            02  11 00166          BRB     17$
                                                            53  D4 00168 16$:     CLRL    R3
```

```
                                03   A4            53  90 0016A 17$:    MOVB    R3, 3(R4)                                  ; 0946
                                     54      04 A042  9E 0016E          MOVAB   4(R0)[R2], R4                              ; 0954
                                08   A5            51  D1 00173          CMPL    I, HLC+36                                  ; 0955
                                                   05  19 00177          BLSS    18$                                        ;
                                     53            02  D0 00179          MOVL    #2, R3                                     ;
                                                   02  11 0017C          BRB     19$                                        ;
                                                   53  D4 0017E 18$:     CLRL    R3                                         ;
                                     64            53  90 00180 19$:     MOVB    R3, (R4)                                   ;
                                01   A4       F4   A5  B0 00183          MOVW    HLC+16, 1(R4)                              ; 0960
                64                   18       FC   A5  F0 00188          INSV    HLC+24, #24, #16, (R4)                     ; 0961
        08 A042                 10   18       04   A5  F0 0018E          INSV    HLC+32, #24, #16, 8(R0)[R2]                ; 0962
                                                OD A042  9F 00196          PUSHAB  13(R0)[R2]                                 ; 0963
                                     9E       F8   A5  B0 0019A          MOVW    HLC+20, @(SP)+                             ;
                                89   51            06  F3 0019E          AOBLEQ  #6, I, 12$                                 ; 0936
                                                   04 001A2              RET                                                ; 0967

; Routine Size:  419 bytes,    Routine Base:  $CODE$ + 004F

;  843        U 0968  1 %IF DSRPLUS %THEN
```

```
;  845          U 0969  1 %SBTTL 'CHECK_HL_VALUES -- verify order and range of HL-level values'
;  846          U 0970  1 ROUTINE CHECK_HL_VALUES (FIRST_HL, LAST_HL) =
;  847          U 0971  1 !++
;  848          U 0972  1 !
;  849          U 0973  1 ! FUNCTIONAL DESCRIPTION:
;  850          U 0974  1 !
;  851          U 0975  1 !       CHECK_HL_VALUES Makes sure two supplied HL-level numbers are in
;  852          U 0976  1 !       ascending order, and range-checks them to make sure they are in the
;  853          U 0977  1 !       range from 1 to 6.
;  854          U 0978  1 !
;  855          U 0979  1 ! FORMAL PARAMETERS:
;  856          U 0980  1 !
;  857          U 0981  1 !       FIRST_HL, LAST_HL are the addresses of the two values to check.
;  858          U 0982  1 !
;  859          U 0983  1 ! IMPLICIT INPUTS:      None
;  860          U 0984  1 !
;  861          U 0985  1 ! IMPLICIT OUTPUTS:
;  862          U 0986  1 !
;  863          U 0987  1 !       If the two values were not in ascending order, they are reversed
;  864          U 0988  1 !       and the new, properly ordered values are "pushed back" through the
;  865          U 0989  1 !       routine formals.
;  866          U 0990  1 !
;  867          U 0991  1 ! ROUTINE VALUE:
;  868          U 0992  1 ! COMPLETION CODES:
;  869          U 0993  1 !
;  870          U 0994  1 !       The routine returns TRUE if the two values were in range, and FALSE
;  871          U 0995  1 !       otherwise.
;  872          U 0996  1 !
;  873          U 0997  1 ! SIDE EFFECTS:         None
;  874          U 0998  1 !--
;  875          U 0999  1     BEGIN
;  876          U 1000  1
;  877          U 1001  1     EXTERNAL ROUTINE
;  878          U 1002  1         CONVBB;
;  879          U 1003  1
;  880          U 1004  1     EXTERNAL LITERAL                    !Error messages
;  881          U 1005  1         RNFINM,
;  882          U 1006  1         RNFNNA;
;  883          U 1007  1
;  884          U 1008  1     LOCAL
;  885          U 1009  1         SWITCH,
;  886          U 1010  1         DIGITS : VECTOR [10],
;  887          U 1011  1         LEN;
;  888          U 1012  1
;  889          U 1013  1     OWN
;  890          U 1014  1         FS_ALLOCATE (STRING, 5);
;  891          U 1015  1
;  892          U 1016  1     BIND
;  893          U 1017  1         FIR = .FIRST_HL,
;  894          U 1018  1         LAS = .LAST_HL;
;  895          U 1019  1
;  896          U 1020  1     !+
;  897          U 1021  1     ! If this subroutine has been called yet during this invocation of STYLE,
;  898          U 1022  1     ! then there is nothing more to check.
;  899          U 1023  1     !-
;  900          U 1024  1     IF .HLS_CHECKED EQL -1  THEN        ! Have we checked already, and failed?
;  901          U 1025  1         RETURN FALSE
```

```
;  902          U 1026  1      ELSE
;  903          U 1027  1          IF .HLS_CHECKED EQL 1  THEN     !If not, have we checked and all was ok?
;  904          U 1028  1              RETURN TRUE
;  905          U 1029  1          ELSE
;  906          U 1030  1              HLS_CHECKED = 1;    !Haven't yet checked. Indicate that now we have.
;  907          U 1031  1
;  908          U 1032  1      FS_INIT (STRING);
;  909          U 1033  1
;  910          U 1034  1      !+
;  911          U 1035  1      ! Switch the two numbers if they are not monotone increasing.
;  912          U 1036  1      !-
;  913          U 1037  1      IF .FIR GTR .LAS  THEN
;  914          U 1038  1              BEGIN                   ! User gave a range like "LEVEL=3:1" -- which
;  915          U 1039  1              SWITCH = .LAS;          ! is legal -- so rationalize it.
;  916          U 1040  1              LAS = .FIR;
;  917          U 1041  1              FIR = .SWITCH;
;  918          U 1042  1              END;
;  919          U 1043  1      !+
;  920          U 1044  1      ! Range-check the supplied level-number(s).
;  921          U 1045  1      !-
;  922          U 1046  1      SELECT TRUE OF
;  923          U 1047  1          SET
;  924          U 1048  1
;  925          U 1049  1          [.FIR EQL 0] :
;  926          U 1050  1              !
;  927          U 1051  1              ERMS (RNFINM, CH$PTR (UPLIT('0')), 1);      !'Illegal number value'
;  928          U 1052  1
;  929          U 1053  1          [.LAS EQL 0] :
;  930          U 1054  1              !
;  931          U 1055  1              ERMS (RNFINM, CH$PTR (UPLIT('0')), 1);      !'Illegal number value'
;  932          U 1056  1
;  933          U 1057  1          [.FIR LSS 0] :
;  934          U 1058  1              !
;  935          U 1059  1              BEGIN
;  936          U 1060  1              CONVBB (.FIR, DIGITS, LEN, 10);
;  937          U 1061  1              FS_WCHAR (STRING, %C'-');
;  938          U 1062  1              DECR I FROM (MIN (5, .LEN)) TO 1 DO
;  939          U 1063  1                  FS_WCHAR (STRING, CH$PTR(.DIGITS[.I-1]));
;  940          U 1064  1              ERMS (RNFNNA, .FS_START (STRING), .FS_LENGTH (STRING));     !'Negative number not allowed'
;  941          U 1065  1              FS_INIT (STRING);
;  942          U 1066  1              END;
;  943          U 1067  1
;  944          U 1068  1          [.LAS LSS 0] :
;  945          U 1069  1              !
;  946          U 1070  1              BEGIN
;  947          U 1071  1              CONVBB (.LAS, DIGITS, LEN, 10);
;  948          U 1072  1              FS_WCHAR (STRING, %C'-');
;  949          U 1073  1              DECR I FROM (MIN (5, .LEN)) TO 1 DO
;  950          U 1074  1                  FS_WCHAR (STRING, CH$PTR(.DIGITS[.I-1]));
;  951          U 1075  1              ERMS (RNFNNA, .FS_START (STRING), .FS_LENGTH (STRING));     !'Negative number not allowed'
;  952          U 1076  1              FS_INIT (STRING);
;  953          U 1077  1              END;
;  954          U 1078  1
;  955          U 1079  1          [.FIR GTR 6] :
;  956          U 1080  1              !
;  957          U 1081  1              BEGIN
;  958          U 1082  1              CONVBB (.FIR, DIGITS, LEN, 10);
```

```
;  959          U 1083  1              DECR I FROM (MIN (5, .LEN)) TO 1 DO
;  960          U 1084  1                  FS_WCHAR (STRING, CH$PTR(.DIGITS[.I-1]));
;  961          U 1085  1              ERMS (RNFINM, .FS_START (STRING), .FS_LENGTH (STRING));     !'Illegal number value'
;  962          U 1086  1              FS_INIT (STRING);
;  963          U 1087  1              END;
;  964          U 1088  1
;  965          U 1089  1          [.LAS GTR 6] :
;  966          U 1090  1              !
;  967          U 1091  1              BEGIN
;  968          U 1092  1              CONVBB (.LAS, DIGITS, LEN, 10);
;  969          U 1093  1              DECR I FROM (MIN (5, .LEN)) TO 1 DO
;  970          U 1094  1                  FS_WCHAR (STRING, CH$PTR(.DIGITS[.I-1]));
;  971          U 1095  1              ERMS (RNFINM, .FS_START (STRING), .FS_LENGTH (STRING));     !'Illegal number value'
;  972          U 1096  1              FS_INIT (STRING);
;  973          U 1097  1              END;
;  974          U 1098  1
;  975          U 1099  1          [(.FIR LEQ 0) OR (.LAS GTR 6)] :
;  976          U 1100  1              !
;  977          U 1101  1              BEGIN
;  978          U 1102  1              HLS_CHECKED = -1;
;  979          U 1103  1              RETURN FALSE;
;  980          U 1104  1              END;
;  981          U 1105  1
;  982          U 1106  1          [OTHERWISE] :
;  983          U 1107  1              !
;  984          U 1108  1              RETURN TRUE;
;  985          U 1109  1
;  986          U 1110  1          TES;
;  987          U 1111  1
;  988          U 1112  1      TRUE
;  989          U 1113  1      END;                                        !End of CHECK_HL_VALUES
```

```
;  991          U 1114  1 %SBTTL 'INIT_STYLE_KW -- Initialize $KWS_BLOCKVECTOR for STYLE commands'
;  992          U 1115  1 ROUTINE INIT_STYLE_KW (HANDLER) : NOVALUE =
;  993          U 1116  1 !++
;  994          U 1117  1 !
;  995          U 1118  1 ! FUNCTIONAL DESCRIPTION:
;  996          U 1119  1 !
;  997          U 1120  1 !       INIT_STYLE_KW   initializes the $KWS_BLOCKVECTOR for STYLE commands.
;  998          U 1121  1 !
;  999          U 1122  1 ! FORMAL PARAMETERS:
; 1000          U 1123  1 !
; 1001          U 1124  1 !       HANDLER         tells which command is being processed.
; 1002          U 1125  1 !
; 1003          U 1126  1 ! IMPLICIT INPUTS:
; 1004          U 1127  1 !
; 1005          U 1128  1 !       The module-level OWN variable FIRST_TIME controls whether the keyword
; 1006          U 1129  1 !       data is initialized.
; 1007          U 1130  1 !
; 1008          U 1131  1 ! IMPLICIT OUTPUTS:
; 1009          U 1132  1 !
; 1010          U 1133  1 !       The module-level OWN variable KEYWORDS is initialized to represent
; 1011          U 1134  1 !       the legal keywords for this .STYLE command.
; 1012          U 1135  1 !
; 1013          U 1136  1 ! ROUTINE VALUE:
; 1014          U 1137  1 ! COMPLETION CODES:     None
; 1015          U 1138  1 !
; 1016          U 1139  1 ! SIDE EFFECTS:         None
; 1017          U 1140  1 !--
; 1018          U 1141  1     BEGIN
; 1019          U 1142  1
; 1020          U 1143  1     INCR I FROM 0 TO COUNT_STYL_KEYWORDS - 1 DO
; 1021          U 1144  1         BEGIN
; 1022          U 1145  1         KEYWORDS [.I, KWS$H_ARG_VALUE] = 0;
; 1023          U 1146  1         KEYWORDS [.I, KWS$H_ARG_SIGN] = 0;
; 1024          U 1147  1         KEYWORDS [.I, KWS$A_ARG_POINTER] = 0;
; 1025          U 1148  1         KEYWORDS [.I, KWS$H_ARG_LENGTH] = 0;
; 1026          U 1149  1         END;
; 1027          U 1150  1
; 1028          U 1151  1 !+
; 1029          U 1152  1 ! Set up default values for those keywords that take numeric parameters.
; 1030          U 1153  1 ! This lets us reestablish the default if the user does not give a value.
; 1031          U 1154  1 !-
; 1032          U 1155  1     KEYWORDS [KWS$K_AFTER, KWS$H_ARG_VALUE] =
; 1033          U 1156  1         (IF  .HANDLER EQL (H_STYLE_CHAPTER OR H_STYLE_APPENDI)
; 1034          U 1157  1              THEN 3
; 1035          U 1158  1              ELSE 1 );
; 1036          U 1159  1
; 1037          U 1160  1     KEYWORDS [KWS$K_BEFORE, KWS$H_ARG_VALUE] =
; 1038          U 1161  1         (IF .HANDLER EQL H_STYLE_HEADERS
; 1039          U 1162  1              THEN 2
; 1040          U 1163  1              ELSE
; 1041          U 1164  1                 IF .HANDLER EQL (H_STYLE_CHAPTER OR H_STYLE_APPENDI)
; 1042          U 1165  1                      THEN 12
; 1043          U 1166  1                      ELSE 1);
; 1044          U 1167  1
; 1045          U 1168  1     KEYWORDS [KWS$K_SPACES, KWS$H_ARG_VALUE] = 2;
; 1046          U 1169  1
; 1047          U 1170  1     KEYWORDS [KWS$K_BETWEEN, KWS$H_ARG_VALUE] = 1;
```

```
;  1048          U 1171  1
;  1049          U 1172  1      KEYWORDS [KWS$K_LEVEL, KWS$H_ARG_VALUE] = .HLLIST [CL_INDEX];
;  1050          U 1173  1
;  1051          U 1174  1      !
;  1052          U 1175  1      ! The module-level OWN variable FIRST_TIME is initialized to 1 in its
;  1053          U 1176  1      ! declaration; therefore, the first time this routine is called, the
;  1054          U 1177  1      ! keyword pointers and lengths are set up. On all subsequent calls,
;  1055          U 1178  1      ! only the keyword-value fields are initialized.
;  1056          U 1179  1      !
;  1057          U 1180  1      IF  .FIRST_TIME EQL 1  THEN
;  1058          U 1181  1          FIRST_TIME = -1
;  1059          U 1182  1      ELSE
;  1060          U 1183  1          RETURN;
;  1061          U 1184  1
;  1062          U 1185  1  !+
;  1063          U 1186  1  ! One-time-only initialization follows.
;  1064          U 1187  1  !-
;  1065          U 1188  1      ! AFTER keyword
;  1066          U 1189  1      KEYWORDS [KWS$K_AFTER, KWS$A_KW_POINTER] = CH$PTR(UPLIT('AFTER'));
;  1067          U 1190  1      KEYWORDS [KWS$K_AFTER, KWS$H_KW_LENGTH] = 5;
;  1068          U 1191  1      KEYWORDS [KWS$K_AFTER, KWS$H_ARG_TYPE] = KWS$K_NUMBER_OPTIONAL;
;  1069          U 1192  1      !
;  1070          U 1193  1      ! ALLCAP keyword
;  1071          U 1194  1      KEYWORDS [KWS$K_ALLCAP, KWS$A_KW_POINTER] = CH$PTR(UPLIT('ALLCAP'));
;  1072          U 1195  1      KEYWORDS [KWS$K_ALLCAP, KWS$H_KW_LENGTH] = 6;
;  1073          U 1196  1      KEYWORDS [KWS$K_ALLCAP, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1074          U 1197  1      !
;  1075          U 1198  1      ! BEFORE keyword
;  1076          U 1199  1      KEYWORDS [KWS$K_BEFORE, KWS$A_KW_POINTER] = CH$PTR(UPLIT('BEFORE'));
;  1077          U 1200  1      KEYWORDS [KWS$K_BEFORE, KWS$H_KW_LENGTH] = 6;
;  1078          U 1201  1      KEYWORDS [KWS$K_BEFORE, KWS$H_ARG_TYPE] = KWS$K_NUMBER_OPTIONAL;
;  1079          U 1202  1      !
;  1080          U 1203  1      ! BETWEEN keyword
;  1081          U 1204  1      KEYWORDS [KWS$K_BETWEEN, KWS$A_KW_POINTER] = CH$PTR(UPLIT('BETWEEN'));
;  1082          U 1205  1      KEYWORDS [KWS$K_BETWEEN, KWS$H_KW_LENGTH] = 7;
;  1083          U 1206  1      KEYWORDS [KWS$K_BETWEEN, KWS$H_ARG_TYPE] = KWS$K_NUMBER_OPTIONAL;
;  1084          U 1207  1      !
;  1085          U 1208  1      ! BOLD keyword
;  1086          U 1209  1      KEYWORDS [KWS$K_BOLD, KWS$A_KW_POINTER] = CH$PTR(UPLIT('BOLD'));
;  1087          U 1210  1      KEYWORDS [KWS$K_BOLD, KWS$H_KW_LENGTH] = 4;
;  1088          U 1211  1      KEYWORDS [KWS$K_BOLD, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1089          U 1212  1      !
;  1090          U 1213  1      ! BREAK keyword
;  1091          U 1214  1      KEYWORDS [KWS$K_BREAK, KWS$A_KW_POINTER] = CH$PTR(UPLIT('BREAK'));
;  1092          U 1215  1      KEYWORDS [KWS$K_BREAK, KWS$H_KW_LENGTH] = 5;
;  1093          U 1216  1      KEYWORDS [KWS$K_BREAK, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1094          U 1217  1      !
;  1095          U 1218  1      ! CENTER keyword
;  1096          U 1219  1      KEYWORDS [KWS$K_CENTER, KWS$A_KW_POINTER] = CH$PTR(UPLIT('CENTER'));
;  1097          U 1220  1      KEYWORDS [KWS$K_CENTER, KWS$H_KW_LENGTH] = 6;
;  1098          U 1221  1      KEYWORDS [KWS$K_CENTER, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1099          U 1222  1      !
;  1100          U 1223  1      ! DEFAULTS keyword
;  1101          U 1224  1      KEYWORDS [KWS$K_DEFAULTS, KWS$A_KW_POINTER] = CH$PTR(UPLIT('DEFAULTS'));
;  1102          U 1225  1      KEYWORDS [KWS$K_DEFAULTS, KWS$H_KW_LENGTH] = 8;
;  1103          U 1226  1      KEYWORDS [KWS$K_DEFAULTS, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1104          U 1227  1      !
```

```
;  1105        U 1228  1      ! FIRSTCAP keyword
;  1106        U 1229  1      KEYWORDS [KWS$K_FIRSTCAP, KWS$A_KW_POINTER] = CH$PTR(UPLIT('FIRSTCAP'));
;  1107        U 1230  1      KEYWORDS [KWS$K_FIRSTCAP, KWS$H_KW_LENGTH] = 8;
;  1108        U 1231  1      KEYWORDS [KWS$K_FIRSTCAP, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1109        U 1232  1      !
;  1110        U 1233  1      ! LEAVECASE keyword
;  1111        U 1234  1      KEYWORDS [KWS$K_LEAVECASE, KWS$A_KW_POINTER] = CH$PTR(UPLIT('LEAVECASE'));
;  1112        U 1235  1      KEYWORDS [KWS$K_LEAVECASE, KWS$H_KW_LENGTH] = 9;
;  1113        U 1236  1      KEYWORDS [KWS$K_LEAVECASE, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1114        U 1237  1      !
;  1115        U 1238  1      ! LEFT keyword
;  1116        U 1239  1      KEYWORDS [KWS$K_LEFT, KWS$A_KW_POINTER] = CH$PTR(UPLIT('LEFT'));
;  1117        U 1240  1      KEYWORDS [KWS$K_LEFT, KWS$H_KW_LENGTH] = 4;
;  1118        U 1241  1      KEYWORDS [KWS$K_LEFT, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1119        U 1242  1      !
;  1120        U 1243  1      ! LEVEL keyword
;  1121        U 1244  1      KEYWORDS [KWS$K_LEVEL, KWS$A_KW_POINTER] = CH$PTR(UPLIT('LEVEL'));
;  1122        U 1245  1      KEYWORDS [KWS$K_LEVEL, KWS$H_KW_LENGTH] = 5;
;  1123        U 1246  1      KEYWORDS [KWS$K_LEVEL, KWS$H_ARG_TYPE] = KWS$K_RANGE_OF_NUMBERS;
;  1124        U 1247  1      !
;  1125        U 1248  1      ! NOBOLD keyword
;  1126        U 1249  1      KEYWORDS [KWS$K_NOBOLD, KWS$A_KW_POINTER] = CH$PTR(UPLIT('NOBOLD'));
;  1127        U 1250  1      KEYWORDS [KWS$K_NOBOLD, KWS$H_KW_LENGTH] = 6;
;  1128        U 1251  1      KEYWORDS [KWS$K_NOBOLD, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1129        U 1252  1      !
;  1130        U 1253  1      ! NOBREAK keyword
;  1131        U 1254  1      KEYWORDS [KWS$K_NOBREAK, KWS$A_KW_POINTER] = CH$PTR(UPLIT('NOBREAK'));
;  1132        U 1255  1      KEYWORDS [KWS$K_NOBREAK, KWS$H_KW_LENGTH] = 7;
;  1133        U 1256  1      KEYWORDS [KWS$K_NOBREAK, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1134        U 1257  1      !
;  1135        U 1258  1      ! NOPAGE keyword
;  1136        U 1259  1      KEYWORDS [KWS$K_NOPAGE, KWS$A_KW_POINTER] = CH$PTR(UPLIT('NOPAGE'));
;  1137        U 1260  1      KEYWORDS [KWS$K_NOPAGE, KWS$H_KW_LENGTH] = 6;
;  1138        U 1261  1      KEYWORDS [KWS$K_NOPAGE, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1139        U 1262  1      !
;  1140        U 1263  1      ! NORUNIN keyword
;  1141        U 1264  1      KEYWORDS [KWS$K_NORUNIN, KWS$A_KW_POINTER] = CH$PTR(UPLIT('NORUNIN'));
;  1142        U 1265  1      KEYWORDS [KWS$K_NORUNIN, KWS$H_KW_LENGTH] = 7;
;  1143        U 1266  1      KEYWORDS [KWS$K_NORUNIN, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1144        U 1267  1      !
;  1145        U 1268  1      ! NOSTARTODD keyword
;  1146        U 1269  1      KEYWORDS [KWS$K_NOSTARTODD, KWS$A_KW_POINTER] = CH$PTR(UPLIT('NOSTARTO.D'));
;  1147        U 1270  1      KEYWORDS [KWS$K_NOSTARTODD, KWS$H_KW_LENGTH] = 11;
;  1148        U 1271  1      KEYWORDS [KWS$K_NOSTARTODD, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1149        U 1272  1      !
;  1150        U 1273  1      ! NOTOCPAGE keyword
;  1151        U 1274  1      KEYWORDS [KWS$K_NOTOCPAGE, KWS$A_KW_POINTER] = CH$PTR(UPLIT('NOTOCPAGE'));
;  1152        U 1275  1      KEYWORDS [KWS$K_NOTOCPAGE, KWS$H_KW_LENGTH] = 10;
;  1153        U 1276  1      KEYWORDS [KWS$K_NOTOCPAGE, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1154        U 1277  1      !
;  1155        U 1278  1      ! NOUNDERLINE keyword
;  1156        U 1279  1      KEYWORDS [KWS$K_NOUNDERLINE, KWS$A_KW_POINTER] = CH$PTR(UPLIT('NOUNDERLINE'));
;  1157        U 1280  1      KEYWORDS [KWS$K_NOUNDERLINE, KWS$H_KW_LENGTH] = 11;
;  1158        U 1281  1      KEYWORDS [KWS$K_NOUNDERLINE, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1159        U 1282  1      !
;  1160        U 1283  1      ! NUMBERED keyword
;  1161        U 1284  1      KEYWORDS [KWS$K_NUMBERED, KWS$A_KW_POINTER] = CH$PTR(UPLIT('NUMBERED'));
```

```
;  1162          U 1285  1      KEYWORDS [KWS$K_NUMBERED, KWS$H_KW_LENGTH] = 8;
;  1163          U 1286  1      KEYWORDS [KWS$K_NUMBERED, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1164          U 1287  1      !
;  1165          U 1288  1      ! PAGE keyword
;  1166          U 1289  1      KEYWORDS [KWS$K_PAGE, KWS$A_KW_POINTER] = CH$PTR(UPLIT('PAGE'));
;  1167          U 1290  1      KEYWORDS [KWS$K_PAGE, KWS$H_KW_LENGTH] = 4;
;  1168          U 1291  1      KEYWORDS [KWS$K_PAGE, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1169          U 1292  1      !
;  1170          U 1293  1      ! RIGHT keyword
;  1171          U 1294  1      KEYWORDS [KWS$K_RIGHT, KWS$A_KW_POINTER] = CH$PTR(UPLIT('RIGHT'));
;  1172          U 1295  1      KEYWORDS [KWS$K_RIGHT, KWS$H_KW_LENGTH] = 5;
;  1173          U 1296  1      KEYWORDS [KWS$K_RIGHT, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1174          U 1297  1      !
;  1175          U 1298  1      ! RUNIN keyword
;  1176          U 1299  1      KEYWORDS [KWS$K_RUNIN, KWS$A_KW_POINTER] = CH$PTR(UPLIT('RUNIN'));
;  1177          U 1300  1      KEYWORDS [KWS$K_RUNIN, KWS$H_KW_LENGTH] = 5;
;  1178          U 1301  1      KEYWORDS [KWS$K_RUNIN, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1179          U 1302  1      !
;  1180          U 1303  1      ! SPACES keyword
;  1181          U 1304  1      KEYWORDS [KWS$K_SPACES, KWS$A_KW_POINTER] = CH$PTR(UPLIT('SPACES'));
;  1182          U 1305  1      KEYWORDS [KWS$K_SPACES, KWS$H_KW_LENGTH] = 6;
;  1183          U 1306  1      KEYWORDS [KWS$K_SPACES, KWS$H_ARG_TYPE] = KWS$K_NUMBER_OPTIONAL;
;  1184          U 1307  1      !
;  1185          U 1308  1      ! STARTODD keyword
;  1186          U 1309  1      KEYWORDS [KWS$K_STARTODD, KWS$A_KW_POINTER] = CH$PTR(UPLIT('STARTODD'));
;  1187          U 1310  1      KEYWORDS [KWS$K_STARTODD, KWS$H_KW_LENGTH] = 9;
;  1188          U 1311  1      KEYWORDS [KWS$K_STARTODD, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1189          U 1312  1      !
;  1190          U 1313  1      ! TESTPAGE keyword
;  1191          U 1314  1      KEYWORDS [KWS$K_TESTPAGE, KWS$A_KW_POINTER] = CH$PTR(UPLIT('TESTPAGE'));
;  1192          U 1315  1      KEYWORDS [KWS$K_TESTPAGE, KWS$H_KW_LENGTH] = 8;
;  1193          U 1316  1      KEYWORDS [KWS$K_TESTPAGE, KWS$H_ARG_TYPE] = KWS$K_NUMBER_OPTIONAL;
;  1194          U 1317  1      !
;  1195          U 1318  1      ! TOCPAGE keyword
;  1196          U 1319  1      KEYWORDS [KWS$K_TOCPAGE, KWS$A_KW_POINTER] = CH$PTR(UPLIT('TOCPAGE'));
;  1197          U 1320  1      KEYWORDS [KWS$K_TOCPAGE, KWS$H_KW_LENGTH] = 8;
;  1198          U 1321  1      KEYWORDS [KWS$K_TOCPAGE, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1199          U 1322  1      !
;  1200          U 1323  1      ! UNDERLINE keyword
;  1201          U 1324  1      KEYWORDS [KWS$K_UNDERLINE, KWS$A_KW_POINTER] = CH$PTR(UPLIT('UNDERLINE'));
;  1202          U 1325  1      KEYWORDS [KWS$K_UNDERLINE, KWS$H_KW_LENGTH] = 9;
;  1203          U 1326  1      KEYWORDS [KWS$K_UNDERLINE, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1204          U 1327  1      !
;  1205          U 1328  1      ! UNNUMBERED keyword
;  1206          U 1329  1      KEYWORDS [KWS$K_UNNUMBERED, KWS$A_KW_POINTER] = CH$PTR(UPLIT('UNNUMBERED'));
;  1207          U 1330  1      KEYWORDS [KWS$K_UNNUMBERED, KWS$H_KW_LENGTH] = 10;
;  1208          U 1331  1      KEYWORDS [KWS$K_UNNUMBERED, KWS$H_ARG_TYPE] = KWS$K_NO_ARGUMENTS;
;  1209          U 1332  1      !
;  1210          U 1333  1
;  1211          U 1334  1      END;                                            ! End of INIT_STYLE_KW
;  1212            1335  1  %FI
;  1213            1336  1  END                                                 !End of module
;  1214            1337  0  ELUDOM
```

```
;                                   PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
;  $PLIT$                                  8  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                                498  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;
;
;                                 Library Statistics
;
;                                          -------- Symbols --------       Pages       Processing
;       File                               Total    Loaded   Percent       Mapped       Time
;
;  _$255$DUA28:[SYSLIB]XPORT.L32;1           590         0         0          252         00:00.2
;  _$255$DUA28:[RUNOFF.SRC]DSRLIB.L32;1     1248        49         3           86         00:00.3
;
;
;                                   COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS LIS$:STYLE/OBJ=OBJ$:STYLE MSRC$:STYLE/UPDATE=(ENH$:STYLE)
;
; Size:          498 code + 8 data bytes
; Run Time:          00:15.2
; Elapsed Time:      00:42.6
; Lines/CPU Min:      5274
; Lexemes/CPU-Min: 25518
; Memory Used:  130 pages
; Compilation Complete
```