```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBBBBBBBBBBB      AAA         AAA     SSSSSSSSS  RRRRRRRRRRR             TTT        LLL
BBBBBBBBBBBB      AAA         AAA     SSSSSSSSS  RRRRRRRRRRR             TTT        LLL
BBBBBBBBBBBB      AAA         AAA     SSSSSSSSS  RRRRRRRRRRR             TTT        LLL
BBB         BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR               TTT        LLL
BBB         BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR               TTT        LLL
BBB         BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR               TTT        LLL
BBB         BBB   AAA         AAA           SSS  RRR      RRR            TTT        LLL
BBB         BBB   AAA         AAA           SSS  RRR   RRR               TTT        LLL
BBB         BBB   AAA         AAA           SSS  RRR      RRR            TTT        LLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
```

```
BBBBBBBB      AAAAAA      SSSSSSSS  UU      UU  DDDDDDDD    FFFFFFFFFF  WW      WW  FFFFFFFFFF
BBBBBBBB      AAAAAA      SSSSSSSS  UU      UU  DDDDDDDD    FFFFFFFFFF  WW      WW  FFFFFFFFFF
BB      BB  AA      AA  SS        UU      UU  DD      DD  FF          WW      WW  FF
BB      BB  AA      AA  SS        UU      UU  DD      DD  FF          WW      WW  FF
BB      BB  AA      AA  SS        UU      UU  DD      DD  FF          WW      WW  FF
BB      BB  AA      AA  SS        U       UU  DD      DD  FF          WW      WW  FF
BBBBBBBB    AA      AA    SSSSSS  UU      UU  DD      DD  FFFFFFFF    WW      WW  FFFFFFFF
BBBBBBBB    AA      AA    SSSSSS  UU      UU  DD      DD  FFFFFFFF    WW      WW  FFFFFFFF
BB      BB  AAAAAAAAAA          SS  UU      UU  DD      DD  FF          WW  WW  WW  FF
BB      BB  AAAAAAAAAA          SS  UU      UU  DD      DD  FF          WW  WW  WW  FF
BB      BB  AA      AA          SS  UU      UU  DD      DD  FF          WWWW  WWWW  FF          ....
BB      BB  AA      AA          SS  UU      UU  DD      DD  FF          WWWW  WWWW  FF          ....
BBBBBBBB    AA      AA  SSSSSSSS    UUUUUUUUUU  DDDDDDDD    FF          WW      WW  FF          ....
BBBBBBBB    AA      AA  SSSSSSSS    UUUUUUUUUU  DDDDDDDD    FF          WW      WW  FF          ....

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

```
;     1          0001  0 MODULE BAS$$UDF_WF (                          ! BASIC Write Formatted UDF
;     2          0002  0                       IDENT = '1-013'          ! File: BASUDFWF.B32 Edit:PLL1013
;     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 ! FACILITY:  BASIC Support Library - not user callable
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !
;    34          0034  1 !       Perform the User data formatting required for Basic Print Using.
;    35          0035  1 !
;    36          0036  1 ! ENVIRONMENT:  User access mode; reentrant AST level or not.
;    37          0037  1 !
;    38          0038  1 ! AUTHOR:       Donald G. Petersen;     CREATION DATE: 14-May-79
;    39          0039  1 !
;    40          0040  1 ! MODIFIED BY:
;    41          0041  1 !
;    42          0042  1 ! 0-001 - original.  DGP 14-May-79
;    43          0043  1 ! 1-002 - Change linkage of BAS$$UDF_WF0.  DGP 22-May-79
;    44          0044  1 ! 1-003 - Pass a null return string to the format interpreter.  DGP 31-May-79
;    45          0045  1 ! 1-004 - Format string reversion must be handled here.  DGP  01-Jun-79
;    46          0046  1 ! 1-005 - Change the output routine which takes care of the string returned
;    47          0047  1 !         by the format interpreter.  DGP 04-Jun-79
;    48          0048  1 ! 1-006 - Remove the unused reference to STR$COPY.  JBS 16-JUL-1979
;    49          0049  1 ! 1-007 - Make format reversions always start a new record.  DGP 01-Aug-79
;    50          0050  1 ! 1-008 - Update cursor position.  DGP 02-Aug-79
;    51          0051  1 ! 1-009 - Set temp string pointers to 0 initially.  DGP 18-Sep-79
;    52          0052  1 ! 1-010 - Pick up the scale factor from the ISB and pass it to the format
;    53          0053  1 !         interpreter.  DGP 25-Nov-79
;    54          0054  1 ! 1-011 - Make PRINT USING look at the right margin.  DGP 25-Jan-80
;    55          0055  1 ! 1-012 - When the print line exceeds the buffer, a CRLF should not be inserted
;    56          0056  1 !         as it currently is.  DGP 03-Feb-1981
;    57          0057  1 ! 1-013 - The format string descriptor must specify the class and dtype to
```

```
;   58          0058  1 !       satisfy the enhanced STR$ routines.  PLL 28-Sep-81
;   59          0059  1 !--
;   60          0060  1
;   61          0061  1 !<BLF/PAGE>
```

```
;    63          0062  1 !
;    64          0063  1 ! SWITCHES:
;    65          0064  1 !
;    66          0065  1
;    67          0066  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    68          0067  1
;    69          0068  1 !
;    70          0069  1 ! LINKAGES
;    71          0070  1 !
;    72          0071  1
;    73          0072  1 REQUIRE 'RTLIN:OTSLNK';                         ! define all linkages
;    74          0501  1
;    75          0502  1 !
;    76          0503  1 ! TABLE OF CONTENTS:
;    77          0504  1 !
;    78          0505  1
;    79          0506  1 FORWARD ROUTINE
;    80          0507  1     BAS$$UDF_WF0 : JSB_UDF0 NOVALUE,            ! initialization
;    81          0508  1     BAS$$UDF_WF1 : CALL_CCB NOVALUE,            ! format one user I/O list element
;    82          0509  1     BAS$$UDF_WF9 : JSB_UDF9 NOVALUE;            ! end of user I/O list - finish
;    83          0510  1
;    84          0511  1 BUILTIN
;    85          0512  1     CVTLD,                                      ! Escape to  CVTLD instruction
;    86          0513  1     MOVTUC;                                     ! Escape to  MOVTUC instruction
;    87          0514  1 !
;    88          0515  1 ! INCLUDE FILES:
;    89          0516  1 !
;    90          0517  1
;    91          0518  1 REQUIRE 'RTLML:BASPAR';                         ! Intermodule BASIC parameters and constants
;    92          0540  1
;    93          0541  1 REQUIRE 'RTLML:OTSISB';                         ! I/O statement block (ISB) offsets
;    94          0709  1
;    95          0710  1 REQUIRE 'RTLML:OTSLUB';                         ! Only needed to get LUB length!
;    96          0850  1
;    97          0851  1 REQUIRE 'RTLIN:OTSMAC';                         ! Macros
;    98          1045  1
;    99          1046  1 REQUIRE 'RTLIN:RTLPSECT';                       ! Define DECLARE_PSECTS macro
;   100          1141  1
;   101          1142  1 LIBRARY 'RTLSTARLE';                            ! STARLET library for macros and symbols
;   102          1143  1
;   103          1144  1 !
;   104          1145  1 ! EQUATED SYMBOLS:
;   105          1146  1 !
;   106          1147  1 !
;   107          1148  1 !       NONE
;   108          1149  1                                                 ! output stream
;   109          1150  1 !
;   110          1151  1 ! MACROS:
;   111          1152  1 !
;   112          1153  1 !       NONE
;   113          1154  1 !
;   114          1155  1 ! PSECT DECLARATIONS:
;   115          1156  1 !
;   116          1157  1 DECLARE_PSECTS (BAS);                           ! declare PSECTs for BAS$ facility
;   117          1158  1 !
;   118          1159  1 ! OWN STORAGE:
;   119          1160  1 !
```

```
;  120          1161  1 !       NONE
;  121          1162  1 !
;  122          1163  1 ! EXTERNAL REFERENCES:
;  123          1164  1 !
;  124          1165  1
;  125          1166  1 EXTERNAL ROUTINE
;  126          1167  1     BAS$$FORMAT_INT : NOVALUE,                ! Basic format interpreter
;  127          1168  1     STR$FREE1_DX,                             ! Deallocate a dynamic string
;  128          1169  1     BAS$$DO_WRITE : JSB_DO_WRITE NOVALUE,     ! Output routine
;  129          1170  1     BAS$$REC_WF0 : JSB_REC0 NOVALUE,          ! initialize formatted output
;  130          1171  1     BAS$$REC_WF1 : JSB_REC1 NOVALUE,          ! write formatted
;  131          1172  1     BAS$$REC_WF9 : JSB_REC9 NOVALUE;          ! end formatted output
;  132          1173  1
```

```
;  134          1174  1 GLOBAL ROUTINE BAS$$UDF_WF0                   ! Write formatted UDF initialization
;  135          1175  1     : JSB_UDF0 NOVALUE =
;  136          1176  1
;  137          1177  1 !++
;  138          1178  1 ! FUNCTIONAL DESCRIPTION:
;  139          1179  1 !
;  140          1180  1 !       Initialize PRINT USING User data formatter (UDF)
;  141          1181  1 !
;  142          1182  1 ! FORMAL PARAMETERS:
;  143          1183  1 !
;  144          1184  1 !       NONE
;  145          1185  1 !
;  146          1186  1 ! IMPLICIT INPUTS:
;  147          1187  1 !
;  148          1188  1 !       LUB$V_AST_GUARD         Guard bit for AST reentrancy
;  149          1189  1 !       LUB$A_BUF_PTR           Pointer to next byte in user buffer
;  150          1190  1 !
;  151          1191  1 ! IMPLICIT OUTPUTS:
;  152          1192  1 !
;  153          1193  1 !       LUB$V_AST_GUARD         Guard bit for AST reentrancy
;  154          1194  1 !       LUB$A_BUF_BEG           Pointer to first byte of user buffer
;  155          1195  1 !       LUB$A_BUF_PTR           Adr of next byte of output
;  156          1196  1 !                               data buffer
;  157          1197  1 !       LUB$A_BUF_END           Adr of end of data buffer
;  158          1198  1 !       LUB$V_FORM_CHAR         indicates that last element transmitter ended in
;  159          1199  1 !                               a comma or semicolon format character
;  160          1200  1 !
;  161          1201  1 ! ROUTINE VALUE:
;  162          1202  1 ! COMPLETION CODES:
;  163          1203  1 !
;  164          1204  1 !       NONE
;  165          1205  1 !
;  166          1206  1 ! SIDE EFFECTS:
;  167          1207  1 !
;  168          1208  1 !       NONE
;  169          1209  1 !
;  170          1210  1 !--
;  171          1211  1
;  172          1212  2     BEGIN
;  173          1213  2
;  174          1214  2     EXTERNAL REGISTER
;  175          1215  2         CCB : REF BLOCK [, BYTE];
;  176          1216  2
;  177          1217  2     !+
;  178          1218  2     ! A guard bit is used to ensure AST reentrancy.  The bit is set to 1
;  179          1219  2     ! at the top of the routine, tested for 1 at the bottom of the routine,
;  180          1220  2     ! and set to 0 upon exitting.  If the test for 1 fails at the bottom of
;  181          1221  2     ! the routine, then an AST has gone off and used this routine possibly
;  182          1222  2     ! changing the buffer pointers. Therefore this routine will loop back and
;  183          1223  2     ! run itself again in its entirety.
;  184          1224  2     !-
;  185          1225  2
;  186          1226  2     DO
;  187          1227  3         BEGIN
;  188          1228  3
;  189          1229  3         !+
;  190          1230  3         ! Set the guard bit
```

```
;  191          1231  3          !-
;  192          1232
;  193          1233  3          CCB [LUB$V_AST_GUARD] = 1;
;  194          1234
;  195          1235  3          !+
;  196          1236  3          ! Call record level to get buffer pointers.
;  197          1237  3          !-
;  198          1238
;  199          1239  3          BAS$$REC_WF0 ();
;  200          1240
;  201          1241  3          !+
;  202          1242  3          ! set the beginning of the buffer if there is no format character pending
;  203          1243  3          !-
;  204          1244
;  205          1245  3          IF NOT .CCB [LUB$V_FORM_CHAR] THEN CCB [LUB$A_BUF_BEG] = .CCB [LUB$A_BUF_PTR];
;  206          1246
;  207          1247  3          !+
;  208          1248  3          ! Check the guard bit.  If it is now 0, then an AST has gone thru this routine
;  209          1249  3          ! Since the data base may have been altered in an unpredictable manner, it
;  210          1250  3          ! is necessary to redo the entire routine.  Note: in worst case processing,
;  211          1251  3          ! the run-time for this routine is essentially unbounded.
;  212          1252  3          !-
;  213          1253  3
;  214          1254  3          END
;  215          1255  2      UNTIL .CCB [LUB$V_AST_GUARD];              ! End of AST guard loop
;  216          1256  2
;  217          1257  2      CCB [LUB$V_AST_GUARD] = 0;
;  218          1258  1      END;


                                                                .TITLE  BAS$$UDF_WF
                                                                .IDENT  \1-013\

                                                                .EXTRN  BAS$$FORMAT_INT
                                                                .EXTRN  STR$FREE1_DX, BAS$$DO_WRITE
                                                                .EXTRN  BAS$$REC_WF0, BAS$$REC_WF1
                                                                .EXTRN  BAS$$REC_WF9

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

                                        52  DD 00000 BAS$$UDF_WF0::
                                                    PUSHL   R2                                  ;  1174
                                52        A0    AB  9E 00002      MOVAB   -96(R11), R2          ;  1233
                                62              20  88 00006 1$:   BISB2   #32, (R2)             ;
                                   00000000G    00  16 00009      JSB     BAS$$REC_WF0         ;  1239
                    05      FE  AB              02  E0 0000F      BBS     #2, -2(CCB), 2$      ;  1245
                            BC  AB      B0      AB  D0 00014      MOVL    -80(CCB), -68(CCB)   ;
                                52      A0      AB  9E 00019 2$:  MOVAB   -96(R11), R2         ;  1255
                    E5          62              05  E1 0001D      BBC     #5, (R2), 1$         ;
                                62              20  8A 00021      BICB2   #32, (R2)            ;  1257
                                                04  BA 00024      POPR    #^M<R2>              ;  1258
                                                    05 00026      RSB                          ;

; Routine Size:  39 bytes,    Routine Base:  _BAS$CODE + 0000

;  219          1259  1
```

```
;  221          1260  1 GLOBAL ROUTINE BAS$$UDF_WF1 (ELEM_TYPE, ELEM_SIZE, ELEM_ADR, FORMAT_CHAR        ! format character
;  222          1261  1     ) : CALL_CCB NOVALUE =
;  223          1262  1
;  224          1263  1 !++
;  225          1264  1 ! FUNCTIONAL DESCRIPTION:
;  226          1265  1 !
;  227          1266  1 !       Write formatted User Data Formatter.
;  228          1267  1 !       Accept an I/O element, format it, and put it in the record buffer.
;  229          1268  1 !       Calls record level processors to perform the actual I/O if the buffer
;  230          1269  1 !       is full or if non-forcible and end-of-record (no format character).
;  231          1270  1 !
;  232          1271  1 ! FORMAL PARAMETERS:
;  233          1272  1 !
;  234          1273  1 !       ELEM_TYPE.rlu.v         data type of the element
;  235          1274  1 !       ELEM_SIZE.rlu.v         size of the data element
;  236          1275  1 !       ELEM_ADR.rlu.r          adr of the data element to be written
;  237          1276  1 !                               Points to a descriptor for strings
;  238          1277  1 !       FORMAT_CHAR.rlu.v               type of format character which followed the data element
;  239          1278  1 !
;  240          1279  1 ! IMPLICIT INPUTS:
;  241          1280  1 !
;  242          1281  1 !       LUB$V_AST_GUARD         guard bit for AST reentrancy
;  243          1282  1 !       LUB$L_PRINT_POS         current cursor position
;  244          1283  1 !       LUB$V_OUTBUF_DR         indicates valid data in the output buffer.
;  245          1284  1 !       LUB$W_R_MARGIN          size of buffer specified in OPEN statement.
;  246          1285  1 !       LUB$V_FORM_CHAR         flag that a format character (',' or ';') was
;  247          1286  1 !                               seen on the last element.
;  248          1287  1 !       LUB$A_BUF_BEG           pointer to beginning of user buffer
;  249          1288  1 !       LUB$A_BUF_PTR           pointer to current position in the buffer.
;  250          1289  1 !       LUB$A_BUF_END           pointer to last byte of buffer + 1.
;  251          1290  1 !       ISB$B_SCALE_FAC         the 0 - -6 factor.
;  252          1291  1 !
;  253          1292  1 ! IMPLICIT OUTPUTS:
;  254          1293  1 !
;  255          1294  1 !       LUB$V_AST_GUARD         guard bit for AST reentrancy
;  256          1295  1 !       LUB$V_OUTBUF_DR         indicates valid data in output buffer
;  257          1296  1 !       LUB$V_FORM_CHAR         flag to indicate a format character was seen
;  258          1297  1 !       LUB$L_PRINT_POS         internal cursor position.
;  259          1298  1 !       LUB$A_BUF_PTR           next byte in the user buffer
;  260          1299  1 !
;  261          1300  1 ! ROUTINE VALUE:
;  262          1301  1 ! COMPLETION CODES:
;  263          1302  1 !
;  264          1303  1 !       NONE
;  265          1304  1 !
;  266          1305  1 ! SIDE EFFECTS:
;  267          1306  1 !
;  268          1307  1 !       If an AST goes off while we are in this routine and calls this routine,
;  269          1308  1 !       then this routine will be repeated upon return to the outer level.
;  270          1309  1 !       It will continue to be repeated until there are no more ASTs using this routine.
;  271          1310  1 !
;  272          1311  1 !--
;  273          1312  1
;  274          1313  2     BEGIN
;  275          1314  2
;  276          1315  2     EXTERNAL REGISTER
;  277          1316  2         CCB : REF BLOCK [, BYTE];
```

```
;  278          1317  2
;  279          1318  2      MAP
;  280          1319  2          ELEM_ADR : REF VECTOR;                  ! element is call-by-reference
;  281          1320  2
;  282          1321  2      LOCAL
;  283          1322  2          BUF_LENGTH,                            ! length of the output buffer
;  284          1323  2          BUF_END,                               ! the end of the print buffer
;  285          1324  2          RET_FORMAT_ADDR,                       ! points to next byte in format
;  286          1325  2                                                 ! string.  Returned by the format
;  287          1326  2                                                 ! interpreter.
;  288          1327  2          RET_STR_ADDR,                          ! Working storage in case the string
;  289          1328  2                                                 ! which is returned does not fit
;  290          1329  2                                                 ! in the buffer
;  291          1330  2          RET_STR_LENGTH,                        ! Working store in case the return
;  292          1331  2                                                 ! string is too long for the buffer.
;  293          1332  2          FORMAT_DSC : BLOCK [8, BYTE],          ! desc. for passing format string
;  294          1333  2                                                 ! to the format interpreter
;  295          1334  2          DSC : BLOCK [8, BYTE];                 ! dynamic string descriptor for
;  296          1335  2
;  297          1336  2                                                 ! output from the format interpreter
;  298          1337  2
;  299          1338  2      !+
;  300          1339  2      ! This loop is to ensure AST reentrancy.
;  301          1340  2      !-
;  302          1341  2
;  303          1342  2      DO
;  304          1343  3          BEGIN
;  305          1344  3          CCB [LUB$V_AST_GUARD] = 1;
;  306          1345  3
;  307          1346  3          !+
;  308          1347  3          ! Allocate a null dynamic string for the format interpreter to return its hand-
;  309          1348  3          ! icraft in.  The string is allocated here so that this will be AST
;  310          1349  3          ! reentrant and it is dynamic so that the return string will always fit.
;  311          1350  3          !-
;  312          1351  3
;  313          1352  3          DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  314          1353  3          DSC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  315          1354  3          DSC [DSC$W_LENGTH] = 0;
;  316          1355  3          DSC [DSC$A_POINTER] = 0;
;  317          1356  3
;  318          1357  3          !+
;  319          1358  3          ! Toggle the format character flag appropriately so that IO_END will know
;  320          1359  3          ! whether or not to do a PUT.
;  321          1360  3          !-
;  322          1361  3
;  323          1362  3          CASE .FORMAT_CHAR FROM BAS$K_SEMI_FORM TO BAS$K_NO_FORM OF
;  324          1363  3              SET
;  325          1364  3
;  326          1365  3              [BAS$K_SEMI_FORM] :
;  327          1366  3                  CCB [LUB$V_FORM_CHAR] = 1;
;  328          1367  3
;  329          1368  3              [BAS$K_COMMA_FOR] :
;  330          1369  3                  CCB [LUB$V_FORM_CHAR] = 1;
;  331          1370  3
;  332          1371  3              [BAS$K_NO_FORM] :
;  333          1372  3                  CCB [[UB$V_FORM_CHAR] = 0;
;  334          1373  3              TES;
```

```
;  335          1374  3
;  336          1375  3          !+
;  337          1376  3          ! Call the format interpreter.  It will scan the format string, check its
;  338          1377  3          ! validity, and format this element according to the string.
;  339          1378  3          ! Check for a format string length of 0 and reset to the front of
;  340          1379  3          ! the string if necessary.  Format reversion always starts a new record
;  341          1380  3          ! by definition.  So, put the current record.
;  342          1381  3          !-
;  343          1382  3
;  344          1383  3          IF .CCB [ISB$W_LEN_REM] EQL 0
;  345          1384  3          THEN
;  346          1385  4              BEGIN
;  347          1386  4              BAS$$DO_WRITE();
;  348          1387  4              CCB [ISB$A_FMT_PTR] = .CCB [ISB$A_FMT_BEG];
;  349          1388  4              CCB [ISB$W_LEN_REM] = .CCB [ISB$W_FMT_LEN];
;  350          1389  3              END;
;  351          1390  3
;  352          1391  3          FORMAT_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  353          1392  3          FORMAT_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  354          1393  3          FORMAT_DSC [DSC$W_LENGTH] = .CCB [ISB$W_LEN_REM];
;  355          1394  3          FORMAT_DSC [DSC$A_POINTER] = .CCB [ISB$A_FMT_PTR];
;  356          1395  3          BAS$$FORMAT_INT (.ELEM_ADR, FORMAT_DSC, .ELEM_TYPE, DSC, RET_FORMAT_ADDR, .CCB [ISB$B_SCALE_FAC]);
;  357          1396  3
;  358          1397  3          !+
;  359          1398  3          ! Update the format pointer so that it now points to the next format
;  360          1399  3          ! field.
;  361          1400  3          !-
;  362          1401  3
;  363          1402  3          CCB [ISB$W_LEN_REM] = .CCB [ISB$W_LEN_REM] - (.RET_FORMAT_ADDR - .CCB [ISB$A_FMT_PTR]);
;  364          1403  3          CCB [ISB$A_FMT_PTR] = .RET_FORMAT_ADDR;
;  365          1404  3
;  366          1405  3          !+
;  367          1406  3          ! Now that the format interpreter has been called, the length of the for-
;  368          1407  3          ! matted item is known exactly.  It is time to determine if the item will
;  369          1408  3          ! fit into the output buffer.  If the item is too big, then it is put out
;  370          1409  3          ! in sections.  No check is made to see whether the buffer is already
;  371          1410  3          ! 'dirty'.  The assumption is made that since this is formatted output,
;  372          1411  3          ! it will be put out exactly as specified.
;  373          1412  3          !-
;  374          1413  3
;  375          1414  3          RET_STR_ADDR = .DSC [DSC$A_POINTER];
;  376          1415  3          RET_STR_LENGTH = .DSC [DSC$W_LENGTH];
;  377          1416  4          BUF_END = (IF .CCB [LUB$W_R_MARGIN] GTR 0
;  378          1417  4            THEN MIN(.CCB [LUB$A_BUF_END], .CCB [LUB$A_BUF_BEG] + .CCB [LUB$W_R_MARGIN])
;  379          1418  3            ELSE .CCB [LUB$A_BUF_END]);
;  380          1419  3          BUF_LENGTH = .BUF_END - .CCB [LUB$A_BUF_PTR];
;  381          1420  3
;  382          1421  3          UNTIL .CCB [LUB$A_BUF_PTR] + .RET_STR_LENGTH LEQ .BUF_END DO
;  383          1422  4              BEGIN
;  384          1423  4              CH$MOVE (.BUF_LENGTH, .RET_STR_ADDR, .CCB [LUB$A_BUF_PTR]);
;  385          1424  4
;  386          1425  4              !+
;  387          1426  4              ! Dump the contents of the buffer and update the length and
;  388          1427  4              ! the pointer into the returned formatted string.
;  389          1428  4              !-
;  390          1429  4
;  391          1430  4              CCB [LUB$A_BUF_PTR] = .BUF_END;
```

```
;  392          1431  4              BAS$$DO_WRITE (BAS$K_BUF_EXC);
;  393          1432  4              RET_STR_LENGTH = .RET_STR_LENGTH - .BUF_LENGTH;
;  394          1433  4              RET_STR_ADDR = .RET_STR_ADDR + .BUF_LENGTH;
;  395          1434  4              BUF_LENGTH = .BUF_END - .CCB [LUB$A_BUF_PTR];
;  396          1435  3              END;
;  397          1436  3
;  398          1437  3          CH$MOVE (.RET_STR_LENGTH, .RET_STR_ADDR, .CCB [LUB$A_BUF_PTR]);
;  399          1438  3          CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] + .RET_STR_LENGTH;
;  400          1439  3      !+
;  401          1440  3      ! Update the current cursor position.
;  402          1441  3      !-
;  403          1442  3          CCB [LUB$L_PRINT_POS] = .CCB [LUB$L_PRINT_POS] + .RET_STR_LENGTH;
;  404          1443  3          CCB [LUB$V_OUTBUF_DR] = 1;
;  405          1444  3          END
;  406          1445  2      UNTIL .CCB [LUB$V_AST_GUARD];                ! End of AST guard loop
;  407          1446  2
;  408          1447  2      !+
;  409          1448  2      ! Free the heap storage allocated.
;  410          1449  2      !-
;  411          1450  2
;  412          1451  2      STR$FREE1_DX (DSC);
;  413          1452  2      CCB [LUB$V_AST_GUARD] = 1;
;  414          1453  2      RETURN;
;  415          1454  1      END;                                         ! END of BAS$$UDF_WF1


                                        07FC 00000          .ENTRY  BAS$$UDF_WF1, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 1260
                                                                    R10
                                    5E          1C  C2 00002        SUBL2   #28, SP
                                    5A      A0  AB  9E 00005        MOVAB   -96(R11), R10                          ; 1344
                                    57      FE  AB  9E 00009        MOVAB   -2(R11), R7                            ; 1366
                                    6A          20  88 0000D 1$:    BISB2   #32, (R10)                             ; 1344
                                0C  AE 020E0000 8F  D0 00010        MOVL    #34471936, DSC                         ; 1354
                                       10   AE  D4 00018            CLRL    DSC+4                                  ; 1355
                                02      01  10  AC  CF 0001B        CASEL   FORMAT_CHAR, #1, #2                    ; 1362
                              000B    0006      0006     00020 2$:  .WORD   3$-2$,-
                                                                    3$-2$,-
                                                                    4$-2$
                                    67          04  88 00026 3$:    BISB2   #4, (R7)                               ; 1369
                                                03  11 00029        BRB     5$
                                    67          04  8A 0002B 4$:    BICB2   #4, (R7)                               ; 1372
                                        8D      AB  B5 0002E 5$:    TSTW    -115(CCB)                              ; 1383
                                                12  12 00031        BNEQ    6$
                                    000000000G  00  16 00033        JSB     BAS$$DO_WRITE                          ; 1386
                                80  AB  FF7C    CB  D0 00039        MOVL    -132(CCB), -128(CCB)                   ; 1387
                                8D  AB  FF72    CB  B0 0003F        MOVW    -142(CCB), -115(CCB)                   ; 1388
                                16  AE  010E    8F  B0 00045 6$:    MOVW    #270, FORMAT_DSC+2                     ; 1391
                                14  AE    8D    AB  B0 0004B        MOVW    -115(CCB), FORMAT_DSC                  ; 1393
                                18  AE    80    AB  D0 00050        MOVL    -128(CCB), FORMAT_DSC+4                ; 1394
                                    7E  FF70    CB  98 00055        CVTBL   -144(CCB), -(SP)                       ; 1395
                                          0C    AE  9F 0005A        PUSHAB  RET_FORMAT_ADDR
                                          14    AE  9F 0005D        PUSHAB  DSC
                                          04    AC  DD 00060        PUSHL   ELEM_TYPE
                                          24    AE  9F 00063        PUSHAB  FORMAT_DSC
```

```
                                    0C   AC  DD 00066          PUSHL   ELEM_ADR                                                    ;
                 00000000G          00       06  FB 00069     CALLS   #6, BAS$$FORMAT_INT                                        ;
                50       80         AB   08  AE  C3 00070     SUBL3   RET_FORMAT_ADDR, -128(CCB), R0                          ; 1402
                         8D         AB       50  A0 00076     ADDW2   R0, -115(CCB)                                           ;
                         80         AB   08  AE  D0 0007A     MOVL    RET_FORMAT_ADDR, -128(CCB)                              ; 1403
                         6E         10   AE  D0 0007F         MOVL    DSC+4, RET_STR_ADDR                                     ; 1414
                         58         0C   AE  3C 00083         MOVZWL  DSC, RET_STR_LENGTH                                     ; 1415
                                    D4   AB  B5 00087         TSTW    -44(CCB)                                                ; 1416
                                         19  13 0008A         BEQL    8$                                                      ;
                         51         D4   AB  3C 0008C         MOVZWL  -44(CCB), R1                                            ; 1417
                         51         BC   AB  C0 00090         ADDL2   -68(CCB), R1                                            ;
                         50         B4   AB  D0 00094         MOVL    -76(CCB), R0                                            ;
                         51              50  D1 00098         CMPL    R0, R1                                                  ;
                                         03  15 0009B         BLEQ    7$                                                      ;
                         50              51  D0 0009D         MOVL    R1, R0                                                  ;
                         59              50  D0 000A0 7$:     MOVL    R0, BUF_END                                             ;
                                         04  11 000A3         BRB     9$                                                      ;
                         59         B4   AB  D0 000A5 8$:     MOVL    -76(CCB), BUF_END                                       ; 1418
            04   AE       59         B0   AB  C3 000A9 9$:     SUBL3   -80(CCB), BUF_END, BUF_LENGTH                           ; 1419
                         56         B0   AB  9E 000AF         MOVAB   -80(R11), R6                                            ; 1421
                 50      66              58  C1 000B3 10$:     ADDL3   RET_STR_LENGTH, (R6), R0                                ;
                         59              50  D1 000B7         CMPL    R0, BUF_END                                             ;
                                         26  15 000BA         BLEQ    11$                                                     ;
00   B6              00   BE         04   AE  28 000BC         MOVC3   BUF_LENGTH, @RET_STR_ADDR, @0(R6)                       ; 1423
                         66              59  D0 000C3         MOVL    BUF_END, (R6)                                           ; 1430
                         50              08  D0 000C6         MOVL    #8, R0                                                  ; 1431
                                 00000000G   00  16 000C9     JSB     BAS$$DO_WRITE                                           ;
                         58         04   AE  C2 000CF         SUBL2   BUF_LENGTH, RET_STR_LENGTH                              ; 1432
                         6E         04   AE  C0 000D3         ADDL2   BUF_LENGTH, RET_STR_ADDR                                ; 1433
                         56         B0   AB  9E 000D7         MOVAB   -80(R11), R6                                            ; 1434
            04   AE       59              66  C3 000DB         SUBL3   (R6), BUF_END, BUF_LENGTH                               ;
                                         D1  11 000E0         BRB     10$                                                     ; 1421
            B0   BB         00   BE         58  28 000E2 11$:     MOVC3   RET_STR_LENGTH, @RET_STR_ADDR, @-80(CCB)            ; 1437
                         B0   AB         58  C0 000E8         ADDL2   RET_STR_LENGTH, -80(CCB)                                ; 1438
                         C8   AB         58  C0 000EC         ADDL2   RET_STR_LENGTH, -56(CCB)                                ; 1442
                              57    FE   AB  9E 000F0         MOVAB   -2(R11), R7                                             ; 1443
                              67         08  88 000F4         BISB2   #8, (R7)                                                ;
                              5A    A0   AB  9E 000F7         MOVAB   -96(R11), R10                                           ; 1445
                    03        6A             05  E0 000FB         BBS     #5, (R10), 12$                                          ;
                                        FF0B  31 000FF         BRW     1$                                                      ;
                                    0C   AE  9F 00102 12$:     PUSHAB  DSC                                                     ; 1451
                      00000000G         00       01  FB 00105 CALLS   #1, STR$FREE1_DX                                        ;
                              6A             20  88 0010C         BISB2   #32, (R10)                                              ; 1452
                                             04  0010F         RET                                                             ; 1454

; Routine Size:  272 bytes,    Routine Base:  _BAS$CODE + 0027

;  416          1455  1
```

```
;  418          1456  1 GLOBAL ROUTINE BAS$$UDF_WF9                   ! I/O end for UDF level of Write Formatted.
;  419          1457  1     : JSB_UDF9 NOVALUE =
;  420          1458  1
;  421          1459  1 !++
;  422          1460  1 ! FUNCTIONAL DESCRIPTION:
;  423          1461  1 !
;  424          1462  1 !       Call the record level I/O end of list routine.  Reset the cursor position
;  425          1463  1 !       if a PUT was done
;  426          1464  1 !
;  427          1465  1 ! FORMAL PARAMETERS:
;  428          1466  1 !
;  429          1467  1 !       NONE
;  430          1468  1 !
;  431          1469  1 ! IMPLICIT INPUTS:
;  432          1470  1 !
;  433          1471  1 !       LUB$V_AST_GUARD         Guard for AST reentrancy
;  434          1472  1 !       LUB$V_FORM_CHAR         last element transmitter ended with a format char
;  435          1473  1 !
;  436          1474  1 ! IMPLICIT OUTPUTS:
;  437          1475  1 !
;  438          1476  1 !       LUB$V_AST_GUARD         guard for AST reentrancy
;  439          1477  1 !       LUB$L_PRINT_POS         current cursor position
;  440          1478  1 !
;  441          1479  1 ! ROUTINE VALUE:
;  442          1480  1 ! COMPLETION CODES:
;  443          1481  1 !
;  444          1482  1 !       NONE
;  445          1483  1 !
;  446          1484  1 ! SIDE EFFECTS:
;  447          1485  1 !
;  448          1486  1 !       This routine will loop back and reexecute if it detects that it was
;  449          1487  1 !       called by an AST while it was executing.
;  450          1488  1 !
;  451          1489  1 !--
;  452          1490  1
;  453          1491  2     BEGIN
;  454          1492  2
;  455          1493  2     EXTERNAL REGISTER
;  456          1494  2         CCB : REF BLOCK [, BYTE];
;  457          1495  2
;  458          1496  2     !+
;  459          1497  2     ! This outer loop is to detect an AST calling this routine while it is
;  460          1498  2     ! executing.
;  461          1499  2     !-
;  462          1500  2
;  463          1501  2     DO
;  464          1502  3         BEGIN
;  465          1503  3         CCB [LUB$V_AST_GUARD] = 1;              ! Initialize the guard bit
;  466          1504  3         BAS$$REC_WF9 ();
;  467          1505  3         ! Time to reset the cursor position to zero perhaps
;  468          1506  3
;  469          1507  3         IF NOT .CCB [LUB$V_FORM_CHAR] THEN CCB [LUB$L_PRINT_POS] = 0;
;  470          1508  3
;  471          1509  3         END
;  472          1510  2     UNTIL .CCB [LUB$V_AST_GUARD];               ! End of AST guard loop
;  473          1511  2
;  474          1512  2     CCB [LUB$V_AST_GUARD] = 0;
```

```
;  475          1513  1     END;


                                                52  DD 00000 BAS$$UDF_WF9::
                                                                 PUSHL   R2                                      ; 1456
                                          52        A0    AB  9E 00002                MOVAB   -96(R11), R2      ; 1503
                                          62              20  88 00006 1$:            BISB2   #32, (R2)         ;
                                                000000000G 00  16 00009               JSB     BAS$$REC_WF9      ; 1504
                                    03        FE    AB        02  E0 0000F            BBS     #2, -2(CCB), 2$   ; 1507
                                                      AB  D4 00014                    CLRL    -56(CCB)          ;
                                          52        A0    AB  9E 00017 2$:          MOVAB   -96(R11), R2      ; 1510
                                    E7            62        05  E1 0001B            BBC     #5, (R2), 1$      ;
                                                  62        20  8A 0001F            BICB2   #32, (R2)         ; 1512
                                                            04  BA 00022            POPR    #^M<R2>           ; 1513
                                                                05 00024            RSB                       ;

; Routine Size:  37 bytes,    Routine Base:  _BAS$CODE + 0137

;  476          1514  1
;  477          1515  1 END
;  478          1516  1
;  479          1517  0 ELUDOM


;                                    PSECT SUMMARY

;      Name                        Bytes                          Attributes

;  _BAS$CODE                         348  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          7         0        581        00:01.2


;                                    COMMAND QUALIFIERS

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

```
;  480          1518  0
; Size:         348 code + 0 data bytes
; Run Time:       00:15.5
; Elapsed Time:   00:34.1
; Lines/CPU Min:   5868
; Lexemes/CPU-Min: 36347
; Memory Used:  196 pages
; Compilation Complete
```

. End of module - BASUDFWF

BASUDFWF
LIS

BASSTRING
LIS

BASTERMIO
LIS

BASTRM
LIS

BASTAB
LIS

BASUDFRM
LIS

BASSYS
LIS

BASSTR
LIS

BASUDFRL
LIS

BASUDFWL
LIS

BASUNLOCK
LIS

BASVECTOR
LIS

BASVIRTUA
LIS

BASVAL
LIS

BASVIRTIO
LIS

BASUNWIND
LIS

BASUPDATE
LIS

BASVECTR2
LIS