```
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
RRRRRRRRRRRR     UUU          UUU  NNN  NNN     NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN     NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN     NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
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
STR
STR
```

FILEID**GETNUM

N 2

```
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  NN      NN  UU      UU  MM      MM
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  NN      NN  UU      UU  MM      MM
GG          EE              TT      NN      NN  UU      UU  MMMM  MMMM
GG          EE              TT      NN      NN  UU      UU  MMMM  MMMM
GG          EE              TT      NNNN    NN  UU      UU  MM  MM  MM
GG          EE              TT      NNNN    NN  UU      UU  MM  MM  MM
GG          EEEEEEEE        TT      NN  NN  NN  UU      UU  MM      MM
GG          EEEEEEEE        TT      NN  NN  NN  UU      UU  MM      MM
GG  GGGGGG  EE              TT      NN    NNNN  UU      UU  MM      MM
GG  GGGGGG  EE              TT      NN    NNNN  UU      UU  MM      MM
GG      GG  EE              TT      NN      NN  UU      UU  MM      MM      ....
GG      GG  EE              TT      NN      NN  UU      UU  MM      MM      ....
  GGGGGG    EEEEEEEEEE      TT      NN      NN  UUUUUUUUUU  MM      MM      ....
  GGGGGG    EEEEEEEEEE      TT      NN      NN  UUUUUUUUUU  MM      MM      ....

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
```

GET
V04

```
;  1          0001  0 %TITLE 'Get a number and convert it to binary.'
;  2          0002  0 MODULE getnum ( 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 !++
; 33          0033  1 ! FACILITY:     DSR (Digital Standard RUNOFF) / DSRPLUS
; 34          0034  1 !
; 35          0035  1 ! ABSTRACT:     Get a number and convert it to binary.
; 36          0036  1 !
; 37          0037  1 ! ENVIRONMENT:  Transportable
; 38          0038  1 !
; 39          0039  1 ! AUTHOR:       R.W. Friday      CREATION DATE: May, 1978/Revised July, 1983
```

```
;    41          0040  1 ! MODIFIED BY:
;    42          0041  1 !
;    43          0042  1 !       011     KFA00011        Ken Alden       23-Sep-1983
;    44          0043  1 !               Made call in GETLET to gslu now call gname.
;    45          0044  1 !                 This will allow user to mix letters and numbers in number
;    46          0045  1 !                 strings.
;    47          0046  1 !
;    48          0047  1 !       010     REM00010        Ray Marshall    22-Jul-1983
;    49          0048  1 !               Changed TEMP_MRA_SIZE to TEMP_SIZE_MRA so that it will be
;    50          0049  1 !                 unique within the first 6 characters for the TOPS-20 linker.
;    51          0050  1 !
;    52          0051  1 !       009     KFA00009        Ken Alden       06-Jul-1983
;    53          0052  1 !               Fixed the newsub problems in dealing with alphanumeric strings
;    54          0053  1 !               Added two new routines to handles $$entities & strings.
;    55          0054  1 !
;    56          0055  1 !       008     KFA00008        Ken Alden       23-Jun-1983
;    57          0056  1 !               Added call to ENDWRD to flush out the last character
;    58          0057  1 !                 that NEWSUB put into the mra.(DSRPLUS)
;    59          0058  1 !
;    60          0059  1 !       007     KFA00007        Ken Alden       15-Jun-1983
;    61          0060  1 !               For DSRPLUS:  Conditionalized call to kcns if this
;    62          0061  1 !                 routine had called NEWSUB for the number.
;    63          0062  1 !
;    64          0063  1 !       006     RER00006        Ron Randall     9-Jun-1983
;    65          0064  1 !               For DSRPLUS:
;    66          0065  1 !               Improved resolution of entities by setting/resetting a
;    67          0066  1 !               flag causing ENDCHR not to be called by NEWSUB while
;    68          0067  1 !               entities are being resolved.
;    69          0068  1 !
;    70          0069  1 !       005     RER00005        Ron Randall     16-May-1983
;    71          0070  1 !               Fixed a bug related to leading zeros.
;    72          0071  1 !
;    73          0072  1 !       004     RER00004        Ron Randall     11-May-1983
;    74          0073  1 !               For DSRPLUS:
;    75          0074  1 !               Allow GETNUM to resolve numerical $ and $$ entities.
;    76          0075  1 !
;    77          0076  1 !       003     RER00003        Ron Randall     7-Mar-1983
;    78          0077  1 !               Global edit of all modules. Updated module names, idents,
;    79          0078  1 !               copyright dates. Changed require files to BLISS library.
;    80          0079  1 !--
```

```
;    82          0080  1 !
;    83          0081  1 ! INCLUDE FILES:
;    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     GETNUM,                         ! Get a number
;    96        U 0224  1 %IF DSRPLUS %THEN
;    97        U 0225  1     GETSUB,                         ! Get a number from a $$entity
;    98          0226  1 %FI
;    99          0227  1     GETLET : NOVALUE;               ! Get a number from a string of letters
;   100          0228  1
;   101          0229  1 !
;   102          0230  1 ! EXTERNAL REFERENCES:
;   103          0231  1 !
;   104          0232  1 EXTERNAL
;   105          0233  1         FS01    : FIXED_STRING,
;   106          0234  1         rnoiob : REF $XPO_IOB (),       ! Output file (document being built)
;   107          0235  1         khar;
;   108          0236  1
;   109          0237  1 EXTERNAL LITERAL
;   110          0238  1         rnfmfn;
;   111          0239  1
;   112          0240  1 EXTERNAL ROUTINE
;   113          0241  1         convlb,
;   114          0242  1         gname,
;   115          0243  1         erma,
;   116          0244  1         rskips;
;   117          0245  1
;   118        U 0246  1 %IF DSRPLUS %THEN
;   119        U 0247  1 EXTERNAL LITERAL
;   120        U 0248  1         rintes  : UNSIGNED (8),
;   121        U 0249  1         temp_size_mra;                        ! Allocated size for temporary mra.
;   122        U 0250  1
;   123        U 0251  1 EXTERNAL
;   124        U 0252  1         entity_in_footnote,                   ! Flag used by FCIMRA.
;   125        U 0253  1         processing_entity,                    ! Flag used by NEWSUB.
;   126        U 0254  1         flgt    : flgt_definition,      ! Flag characters.
;   127        U 0255  1         fnct    : fnct_definition,      ! Footnote control table.
;   128        U 0256  1         hold_mra,                             ! Holds mra address.
;   129        U 0257  1         hold_tsf,                             ! Holds tsf address.
;   130        U 0258  1         mra             : REF fixed_string,
;   131        U 0259  1         sca             : sca_definition,     ! Flag for justification.
;   132        U 0260  1         temp_mra        : fixed_string,       ! Substitute mra.
;   133        U 0261  1         temp_tsf        : VECTOR [tsf_size],  ! Substitute tsf.
;   134        U 0262  1         tsf             : tsf_definition;
;   135        U 0263  1
;   136        U 0264  1 EXTERNAL ROUTINE
;   137        U 0265  1
;   138        U 0266  1         endwrd,                               ! Flushes last character into mra.
```

```
;  139        U 0267  1          gtoken,                         ! Handles tokens.
;  140        U 0268  1          newsub;                         ! Resolves entity value.
;  141          0269  1          %FI
;  142          0270  1
;  143          0271  1 OWN
;  144          0272  1          sign_convert;          ! Either +1 or -1; used in multiplying
;  145          0273  1                                 ! to get the right sign.
;  146          0274  1
;  147          0275  1
;  148          0276  1
```

```
;   150     0277  1 GLOBAL ROUTINE getnum (input_string, number_value, number_sign, number_length) =
;   151     0278  1
;   152     0279  1 !++
;   153     0280  1 ! FUNCTIONAL DESCRIPTION:
;   154     0281  1 !
;   155     0282  1 !       Beginning with khar, getnum skips spaces and tabs until it
;   156     0283  1 !       finds something that is neither.  From that point on, it processes
;   157     0284  1 !       characters until it encounters what cannot be part of a number;
;   158     0285  1 !       though a + or - sign will be accepted.
;   159     0286  1 !
;   160     0287  1 !       If GETNUM encounters a "number" that is incorrect (too long or not
;   161     0288  1 !       numeric), it returns FALSE; otherwise, it returns TRUE.
;   162     0289  1 !
;   163     0290  1 ! FORMAL PARAMETERES:
;   164     0291  1 !
;   165     0292  1 !       input_string    - The string containing the value sought.
;   166     0293  1 !       number_value    - Contains the binary equivalent of the original input.
;   167     0294  1 !       number_sign     - Contains the value 1 if a + sign was encountered,
;   168     0295  1 !                         -1 if a minus sign was encountered,
;   169     0296  1 !                         or 0 if no sign was encountered.
;   170     0297  1 !       number_length   - Contains the number of digits that were processed.
;   171     0298  1 !                         If number_length is zero on return, no number was
;   172     0299  1 !                         found.  In such a case, all other parameters will
;   173     0300  1 !                         also have a zero value.
;   174     0301  1 !
;   175     0302  1 ! IMPLICIT INPUTS:      None
;   176     0303  1 !
;   177     0304  1 ! IMPLICIT OUTPUTS:     None
;   178     0305  1 !
;   179     0306  1 ! ROUTINE VALUE:
;   180     0307  1 ! COMPLETION CODES:
;   181     0308  1 !
;   182     0309  1 !       Returns TRUE if a valid number was found, else FALSE.
;   183     0310  1 !
;   184     0311  1 ! SIDE EFFECTS:
;   185     0312  1 !
;   186     0313  1 !       Issues an error message in the case of an invalid number.
;   187     0314  1 !--
;   188     0315  1
;   189     0316  2 BEGIN
;   190     0317  2 BIND
;   191     0318  2     ira = input_string : REF FIXED_STRING,
;   192     0319  2     nv  = .number_value,
;   193     0320  2     ns  = .number_sign,
;   194     0321  2     nl  = .number_length;
;   195     0322  2
;   196     0323  2 LOCAL
;   197     0324  2         leading_zeros;                  ! TRUE if leading zeros were found.
;   198     0325  2
;   199     0326  2     nv = 0;
;   200     0327  2     ns = 0;
;   201     0328  2     nl = 0;
;   202     0329  2 !
;   203     0330  2 ! Ignore leading spaces and tabs.
;   204     0331  2 !
;   205     0332  2 rskips (.input_string);
;   206     0333  2 !
```

```
;   207          0334  2 ! Check for leading sign.
;   208          0335  2 !
;   209          0336  3 IF  (.khar EQL %C'+') OR (.khar EQL %C'-')
;   210          0337  2 THEN
;   211          0338  3         BEGIN
;   212          0339  3         ns = (IF .khar EQL %C'+' THEN 1 ELSE -1);
;   213          0340  3         kcns ();
;   214          0341  2         END;
;   215          0342  2
;   216          0343  2 sign_convert  = (IF .ns GEQ 0 THEN 1 ELSE -1);
;   217          0344  2 leading_zeros = .khar EQL %C'0';
;   218          0345  2 !
;   219          0346  2 ! Skip leading zeros.
;   220          0347  2 !
;   221          0348  2 WHILE .khar EQL %C'0' DO
;   222          0349  2         kcns ();
;   223          0350  2
;   224          0351  2 !
;   225          0352  2 ! Detect special cases.
;   226          0353  2 !
;   227          0354  3 IF  NOT digit (.khar)
;   228          0355  2 THEN
;   229          0356  3     BEGIN
;   230          0357  3     !
;   231          0358  3     ! Sign alone, or + or - zero.
;   232          0359  3     !
;   233          0360  3     IF  .ns NEQ 0
;   234          0361  3     THEN
;   235          0362  4         BEGIN
;   236        U 0363  4 %IF DSRPLUS %THEN
;   237        U 0364  4         IF (.flgt [sub_flag, flag_character] EQL .khar) AND
;   238        U 0365  4             .flgt [sub_flag, flag_enabled]
;   239        U 0366  4         THEN
;   240        U 0367  4             RETURN (getsub(ira, nv, nl, false))
;   241        U 0368  4         ELSE
;   242          0369  4 %FI
;   243          0370  5             BEGIN
;   244          0371  5             erma (rnfmfn, false);
;   245          0372  5             RETURN false;
;   246          0373  4             END;
;   247          0374  4         END
;   248          0375  3     ELSE
;   249          0376  4         BEGIN
;   250          0377  4         !
;   251          0378  4         ! Set number length to 1 (arbitrarily) for any string of zeros.
;   252          0379  4         !
;   253          0380  4         IF  .leading_zeros
;   254          0381  4         THEN
;   255          0382  4             nl = 1;
;   256          0383  4
;   257        U 0384  4 %IF DSRPLUS %THEN
;   258        U 0385  4         IF (.flgt [sub_flag, flag_character] EQL .khar) AND
;   259        U 0386  4             .flgt [sub_flag, flag_enabled]
;   260        U 0387  4         THEN
;   261        U 0388  4             BEGIN
;   262        U 0389  4             LOCAL
;   263        U 0390  4                 status;
```

```
;  264        U 0391  4
;  265        U 0392  4                status = getsub(ira, nv, nl, false);
;  266        U 0393  4                IF .status
;  267        U 0394  4                THEN
;  268        U 0395  4                    kcns ();
;  269        U 0396  4                RETURN true;
;  270        U 0397  4                END;
;  271           0398  4 %ELSE
;  272           0399  4                RETURN true;
;  273           0400  4 %FI
;  274           0401  3                END;
;  275           0402  3
;  276           0403  2            END;
;  277           0404  2
;  278           0405  2 WHILE digit (.khar) DO
;  279           0406  3            BEGIN
;  280           0407  3            LOCAL
;  281           0408  3                converted_digit;
;  282           0409  3
;  283           0410  4            IF  .nl EQL most_digits_9
;  284           0411  3            THEN
;  285           0412  4                BEGIN
;  286           0413  4                !
;  287           0414  4                ! Skip overflow.
;  288           0415  4                !
;  289           0416  4                WHILE digit (.khar) DO
;  290           0417  4                    kcns ();
;  291           0418  4
;  292           0419  4                erma (rnfmfn, false);
;  293           0420  4                RETURN false;
;  294           0421  3                END;
;  295           0422  3            !
;  296           0423  3            ! Convert characters to decimal number.
;  297           0424  3            !
;  298           0425  3            converted_digit = .khar - %C'0';
;  299           0426  3            nv = (.nv * 10) + (.converted_digit * .sign_convert);
;  300           0427  3            nl = .nl + 1;
;  301           0428  3            !
;  302           0429  3            ! Get next character.
;  303           0430  3            !
;  304           0431  3            kcns ();
;  305           0432  2            END;
;  306           0433  2
;  307           0434  2 RETURN true;
;  308           0435  1 END;                                    ! End of GETNUM.


                                                .TITLE  GETNUM Get a number and convert it to binary.
                                                .IDENT  \V04-000\

                                                .PSECT  $OWN$,NOEXE,2

                                        00000 SIGN_CONVERT:
                                                .BLKB   4

                                                .EXTRN  FS01, RNOIOB, KHAR
                                                .EXTRN  RNFMFN, CONVLB, GNAME
```

```
                                                .EXTRN  ERMA, RSKIPS, RINTES

                                                .PSECT  $CODE$,NOWRT,2

                                 007C 00000     .ENTRY  GETNUM, Save R2,R3,R4,R5,R6                     ; 0277
                    56 00000000'  EF  9E 00002  MOVAB   SIGN_CONVERT, R6                                ;
                    55 00000000G  EF  9E 00009  MOVAB   KHAR, R5                                        ;
                    54       10   AC  D0 00010  MOVL    NUMBER_LENGTH, R4                               ; 0321
                             08   BC  D4 00014  CLRL    @NUMBER_VALUE                                   ; 0326
                             0C   BC  D4 00017  CLRL    @NUMBER_SIGN                                    ; 0327
                             64   D4  0001A     CLRL    (R4)                                            ; 0328
                    52       04   AC  D0 0001C  MOVL    INPUT_STRING, R2                                ; 0332
                             52   DD  00020     PUSHL   R2                                              ;
          00000000G  EF           01  FB 00022  CALLS   #1, RSKIPS                                      ;
                             50   D4  00029     CLRL    R0                                              ; 0336
                    2B           65  D1 0002B  CMPL    KHAR, #43                                       ;
                             04   12  0002E     BNEQ    1$                                              ;
                             50   D6  00030     INCL    R0                                              ;
                             05   11  00032     BRB     2$                                              ;
                    2D           65  D1 00034  1$:     CMPL    KHAR, #45                               ;
                             28   12  00037     BNEQ    6$                                              ;
                    05           50  E9 00039  2$:     BLBC    R0, 3$                                  ; 0339
                    50           01  D0 0003C  MOVL    #1, R0                                          ;
                             03   11  0003F     BRB     4$                                              ;
                    50           01  CE 00041  3$:     MNEGL   #1, R0                                  ;
                 0C BC           50  D0 00044  4$:     MOVL    R0, @NUMBER_SIGN                        ;
                          0C     A2  D5 00048  TSTL    12(R2)                                          ; 0340
                             0A   14  0004B     BGTR    5$                                              ;
                    65       00G 8F  9A 0004D  MOVZBL  #RINTES, KHAR                                   ;
                 0C A2           01  CE 00051  MNEGL   #1, 12(R2)                                      ;
                             0A   11  00055     BRB     6$                                              ;
                    65       04  B2  9A 00057  5$:     MOVZBL  @4(R2), KHAR                            ;
                             04  A2  D6 0005B  INCL    4(R2)                                           ;
                             0C  A2  D7 0005E  DECL    12(R2)                                          ;
                             0C  BC  D5 00061  6$:     TSTL    @NUMBER_SIGN                            ; 0343
                             05  19  00064     BLSS    7$                                              ;
                    50           01  D0 00066  MOVL    #1, R0                                          ;
                             03  11  00069     BRB     8$                                              ;
                    50           01  CE 0006B  7$:     MNEGL   #1, R0                                  ;
                    66           50  D0 0006E  8$:     MOVL    R0, SIGN_CONVERT                        ;
                             50  D4  00071     CLRL    R0                                              ; 0344
                    30           65  D1 00073  CMPL    KHAR, #48                                       ;
                             02  12  00076     BNEQ    9$                                              ;
                             50  D6  00078     INCL    R0                                              ;
                    53       0C  A2  9E 0007A  9$:     MOVAB   12(R2), R3                              ; 0349
                    30           65  D1 0007E  10$:    CMPL    KHAR, #48                               ; 0348
                             18  12  00081     BNEQ    12$                                             ;
                             63  D5  00083     TSTL    (R3)                                            ; 0349
                             09  14  00085     BGTR    11$                                             ;
                    65       00G 8F  9A 00087  MOVZBL  #RINTES, KHAR                                   ;
                    63           01  CE 0008B  MNEGL   #1, (R3)                                        ;
                             EE  11  0008E     BRB     10$                                             ;
                    65       04  B2  9A 00090  11$:    MOVZBL  @4(R2), KHAR                            ;
                             04  A2  D6 00094  INCL    4(R2)                                           ;
                             63  D7  00097     DECL    (R3)                                            ;
                             E3  11  00099     BRB     10$                                             ;
                                 05  19 0009B  12$:    BLSS    13$                                     ; 0354
```

```
                        39              65  D1 0009D            CMPL    KHAR, #57                                       ;
                                        1E  15 000A0            BLEQ    16$                                             ;
                                    0C  BC  D5 000A2 13$:       TSTL    @NUMBER_SIGN                                    ; 0360
                                        11  13 000A5            BEQL    15$                                             ;
                                        7E  D4 000A7 14$:       CLRL    -(SP)                                           ; 0371
                                00000000G   8F  DD 000A9        PUSHL   #RNFMFN
                            00000000G   EF      02  FB 000AF    CALLS   #2, ERMA
                                        68  11 000B6            BRB     22$                                             ; 0376
                        61                  50  E9 000B8 15$:   BLBC    LEADING_ZEROS, 21$                              ; 0380
                        64                  01  D0 000BB        MOVL    #1, (R4)                                        ; 0382
                                            5C  11 000BE        BRB     21$                                             ; 0399
                        30                  65  D1 000C0 16$:   CMPL    KHAR, #48                                       ; 0405
                                            57  19 000C3        BLSS    21$
                        39                  65  D1 000C5        CMPL    KHAR, #57
                                            52  14 000C8        BGTR    21$
                        09                  64  D1 000CA        CMPL    (R4), #9                                        ; 0410
                                            22  12 000CD        BNEQ    19$
                        30                  65  D1 000CF 17$:   CMPL    KHAR, #48                                       ; 0416
                                            D3  19 000D2        BLSS    14$
                        39                  65  D1 000D4        CMPL    KHAR, #57
                                            CE  14 000D7        BGTR    14$
                                            63  D5 000D9        TSTL    (R3)                                            ; 0417
                                            09  14 000DB        BGTR    18$
                        65              00G 8F  9A 000DD        MOVZBL  #RINTES, KHAR
                        63                  01  CE 000E1        MNEGL   #1, (R3)
                                            E9  11 000E4        BRB     17$
                        65              04  B2  9A 000E6 18$:   MOVZBL  @4(R2), KHAR
                                        04  A2  D6 000EA        INCL    4(R2)
                                            63  D7 000ED        DECL    (R3)
                                            DE  11 000EF        BRB     17$
                    50                  65          30  C3 000F1 19$:   SUBL3   #48, KHAR, CONVERTED_DIGIT              ; 0425
                    51              08  BC          0A  C5 000F5        MULL3   #10, @NUMBER_VALUE, R1                  ; 0426
                                        50          66  C4 000FA        MULL2   SIGN_CONVERT, R0
                08  BC                  51          50  C1 000FD        ADDL3   R0, R1, @NUMBER_VALUE
                                                    64  D6 00102        INCL    (R4)                                    ; 0427
                                                    63  D5 00104        TSTL    (R3)                                    ; 0431
                                                    09  14 00106        BGTR    20$
                                        65      00G 8F  9A 00108        MOVZBL  #RINTES, KHAR
                                        63          01  CE 0010C        MNEGL   #1, (R3)
                                                    AF  11 0010F        BRB     16$
                                        65      04  B2  9A 00111 20$:   MOVZBL  @4(R2), KHAR
                                                04  A2  D6 00115        INCL    4(R2)
                                                    63  D7 00118        DECL    (R3)
                                                    A4  11 0011A        BRB     16$                                     ; 0405
                                        50          01  D0 0011C 21$:   MOVL    #1, R0                                  ; 0434
                                                    04  00 0011F        RET
                                                    50  D4 00120 22$:   CLRL    R0                                      ; 0435
                                                    04  00122        RET

; Routine Size:  291 bytes,    Routine Base:  $CODE$ + 0000

;  309          0436  1
```

```
;   311        U 0437  1 %IF DSRPLUS %THEN
;   312        U 0438  1 GLOBAL ROUTINE getsub (input_string, number_value, number_length, expand_string) =
;   313        U 0439  1
;   314        U 0440  1 !++
;   315        U 0441  1 ! FUNCTIONAL DESCRIPTION:
;   316        U 0442  1 !
;   317        U 0443  1 !       Khar is assumed to be the substitute flag and must be to
;   318        U 0444  1 !       to get processed through this routine.  GETSUB will call NEWSUB
;   319        U 0445  1 !       to pick up the $entity and if expand_string is true, then GETLET
;   320        U 0446  1 !       will be called.
;   321        U 0447  1 !
;   322        U 0448  1 ! FORMAL PARAMETERS:
;   323        U 0449  1 !
;   324        U 0450  1 !       input_string    - The string containing the value sought.
;   325        U 0451  1 !       number_value    - Contains the binary equivalent of the original input.
;   326        U 0452  1 !       number_length   - Contains the number of digits that were processed.
;   327        U 0453  1 !                         If number_length is zero on return, no number was
;   328        U 0454  1 !                         found.  In such a case, all other parameters will
;   329        U 0455  1 !                         also have a zero value.
;   330        U 0456  1 !       expand_string   - If true, then any string 'number' will be expanded
;   331        U 0457  1 !                         and converted into a 'number'.
;   332        U 0458  1 !
;   333        U 0459  1 !
;   334        U 0460  1 ! IMPLICIT INPUTS:      None
;   335        U 0461  1 !
;   336        U 0462  1 ! IMPLICIT OUTPUTS:     None
;   337        U 0463  1 !
;   338        U 0464  1 ! ROUTINE VALUE:
;   339        U 0465  1 ! COMPLETION CODES:
;   340        U 0466  1 !
;   341        U 0467  1 !       Returns TRUE if a valid number was found, else FALSE.
;   342        U 0468  1 !
;   343        U 0469  1 ! SIDE EFFECTS:
;   344        U 0470  1 !
;   345        U 0471  1 !       Issues an error message in the case of an invalid number.
;   346        U 0472  1 !--
;   347        U 0473  1
;   348        U 0474  1 BEGIN
;   349        U 0475  1 OWN
;   350        U 0476  1     value,
;   351        U 0477  1     length,
;   352        U 0478  1     token_desc : $STR_DESCRIPTOR (CLASS = DYNAMIC),
;   353        U 0479  1     status     : $gtoken_status,  ! Status returned by GTOKEN.
;   354        U 0480  1     entity_value;                 ! Value returned by GTOKEN.
;   355        U 0481  1
;   356        U 0482  1 BIND
;   357        U 0483  1     ira = .input_string : REF FIXED_STRING,
;   358        U 0484  1     nv  = .number_value,
;   359        U 0485  1     nl  = .number_length;
;   360        U 0486  1
;   361        U 0487  1 !
;   362        U 0488  1 ! Check to make sure khar is the substitute flag.
;   363        U 0489  1 !
;   364        U 0490  1 IF .khar NEQ .flgt [sub_flag, flag_character] AND
;   365        U 0491  1           .flgt [sub_flag, flag_enabled]
;   366        U 0492  1 THEN
;   367        U 0493  1           RETURN false;
```

```
;  368          U 0494  1
;  369          U 0495  1 !
;  370          U 0496  1 ! Knowing that khar is the subs flag, we can hand the input string off
;  371          U 0497  1 ! to NEWSUB.
;  372          U 0498  1 !
;  373          U 0499  1     BEGIN
;  374          U 0500  1     LOCAL
;  375          U 0501  1         hold_temp_mra_length,
;  376          U 0502  1         hold_khar,
;  377          U 0503  1         hold_ira_next,
;  378          U 0504  1         hold_ira_length;
;  379          U 0505  1
;  380          U 0506  1     length = 0;
;  381          U 0507  1     value = 0;
;  382          U 0508  1
;  383          U 0509  1     !
;  384          U 0510  1     ! Remember location of current tsf and ira.
;  385          U 0511  1     !
;  386          U 0512  1     hold_khar = .khar;
;  387          U 0513  1     hold_ira_next = .fs_next(ira);
;  388          U 0514  1     hold_ira_length     = .fs_length(ira);
;  389          U 0515  1     hold_tsf = .tsf;
;  390          U 0516  1     !
;  391          U 0517  1     ! Set up new (temporary) tsf.
;  392          U 0518  1     !
;  393          U 0519  1     tsf = temp_tsf;
;  394          U 0520  1
;  395          U 0521  1     INCR I FROM 0 TO (tsf_size - 1) DO
;  396          U 0522  1     tsf [.I] = 0;
;  397          U 0523  1
;  398          U 0524  1     !
;  399          U 0525  1     ! Remember location of current mra.
;  400          U 0526  1     !
;  401          U 0527  1     hold_mra = .mra;
;  402          U 0528  1     !
;  403          U 0529  1     ! Set up new (temporary) mra.
;  404          U 0530  1     !
;  405          U 0531  1     fs_maxsize (temp_mra) = temp_size_mra;
;  406          U 0532  1     fs_init    (temp_mra);
;  407          U 0533  1     mra = temp_mra;
;  408          U 0534  1     !
;  409          U 0535  1     ! Set a flag to indicate that we are within a footnote.
;  410          U 0536  1     ! The flag is used by FCIMRA to avoid writing into the new mra.
;  411          U 0537  1     !
;  412          U 0538  1     IF  .fnct_collecting
;  413          U 0539  1     THEN
;  414          U 0540  1         entity_in_footnote = true;
;  415          U 0541  1     !
;  416          U 0542  1     ! Set a flag to indicate that we are processing an entity.
;  417          U 0543  1     ! The flag is used by NEWSUB to avoid calling ENDCHR.
;  418          U 0544  1     ! It also tells GETNUM that NEWSUB has already called kcns.
;  419          U 0545  1     !
;  420          U 0546  1     processing_entity = true;
;  421          U 0547  1     !
;  422          U 0548  1     ! Get the string (expected to be digits) referred to by the entity.
;  423          U 0549  1     ! Place it in the current mra.
;  424          U 0550  1     !
```

```
;  425     U 0551  1      newsub ();
;  426     U 0552  1      processing_entity = false;
;  427     U 0553  1      !
;  428     U 0554  1      ! Flush out last character into the mra.
;  429     U 0555  1      !
;  430     U 0556  1      endwrd (false, false, false);
;  431     U 0557  1      !
;  432     U 0558  1      ! Reset one flag; job is done.
;  433     U 0559  1      !
;  434     U 0560  1      entity_in_footnote = false;
;  435     U 0561  1      !
;  436     U 0562  1      ! Reset next character pointer to the start of the mra.
;  437     U 0563  1      !
;  438     U 0564  1      hold_temp_mra_length = .fs_length (mra);
;  439     U 0565  1      fs_next (mra) = .fs_start (mra);
;  440     U 0566  1      !
;  441     U 0567  1      ! Convert substitute string in the temp. mra to a value
;  442     U 0568  1      ! and get statistics about it.
;  443     U 0569  1      !
;  444     U 0570  1      status = gtoken (.mra, 0, token_desc, entity_value, 0);
;  445     U 0571  1      !
;  446     U 0572  1      IF .status [gt$v_numeric]
;  447     U 0573  1      THEN
;  448     U 0574  1          BEGIN
;  449     U 0575  1          !
;  450     U 0576  1          ! User supplied a valid numeric number.
;  451     U 0577  1          ! Apply sign to returned value.
;  452     U 0578  1          !
;  453     U 0579  1          nv = (.entity_value * .sign_convert);
;  454     U 0580  1          !
;  455     U 0581  1          ! Get length of returned value.
;  456     U 0582  1          !
;  457     U 0583  1          nl = .token_desc [str$h_length];
;  458     U 0584  1          !
;  459     U 0585  1          ! Restore original mra.
;  460     U 0586  1          !
;  461     U 0587  1          mra = .hold_mra;
;  462     U 0588  1          !
;  463     U 0589  1          ! Restore original tsf.
;  464     U 0590  1          !
;  465     U 0591  1          tsf = .hold_tsf;
;  466     U 0592  1          RETURN true;
;  467     U 0593  1          END;
;  468     U 0594  1      !
;  469     U 0595  1      ! If we picked up something but it wasn't a number then check to see
;  470     U 0596  1      ! if we are allowed to go further and use the string as a 'number'.
;  471     U 0597  1      !
;  472     U 0598  1      IF  .status [gt$v_token] AND NOT .status [gt$v_numeric]
;  473     U 0599  1      THEN
;  474     U 0600  1          BEGIN
;  475     U 0601  1          IF .expand_string AND (.token_desc [str$h_length] GTR 0)
;  476     U 0602  1          THEN
;  477     U 0603  1              BEGIN
;  478     U 0604  1              !
;  479     U 0605  1              ! Take a picture of the mra and ira for restoring when finished.
;  480     U 0606  1              !
;  481     U 0607  1              hold_ira_next       = .fs_next(ira);
```

```
;  482          U 0608  1              hold_ira_length     = .fs_length(ira);
;  483          U 0609  1              fs_next (mra)       = .fs_start (mra);
;  484          U 0610  1              fs_length (mra)     = .hold_temp_mra_length;
;  485          U 0611  1              !
;  486          U 0612  1              ! Reset next character pointer
;  487          U 0613  1              !
;  488          U 0614  1              fs_rchar (mra, khar);
;  489          U 0615  1              !
;  490          U 0616  1              ! Go convert the string into a 'number'
;  491          U 0617  1              !
;  492          U 0618  1              getlet(.mra, value, length);
;  493          U 0619  1              !
;  494          U 0620  1              nv = .value;
;  495          U 0621  1              nl = .length;
;  496          U 0622  1              !
;  497          U 0623  1              ! Restore original mra and ira.
;  498          U 0624  1              !
;  499          U 0625  1              mra = .hold_mra;
;  500          U 0626  1              fs_next(ira) = .hold_ira_next;
;  501          U 0627  1              fs_length(ira) = .hold_ira_length;
;  502          U 0628  1              !
;  503          U 0629  1              ! Restore original tsf.
;  504          U 0630  1              !
;  505          U 0631  1              tsf = .hold_tsf;
;  506          U 0632  1              RETURN true;
;  507          U 0633  1              END
;  508          U 0634  1          ELSE
;  509          U 0635  1              BEGIN
;  510          U 0636  1              !
;  511          U 0637  1              ! Restore original mra.
;  512          U 0638  1              !
;  513          U 0639  1              mra = .hold_mra;
;  514          U 0640  1              !
;  515          U 0641  1              ! Restore original tsf and put the IRA back into shape.
;  516          U 0642  1              !
;  517          U 0643  1              tsf = .hold_tsf;
;  518          U 0644  1              !
;  519          U 0645  1              khar = .hold_khar;
;  520          U 0646  1              fs_next(ira) = .hold_ira_next;
;  521          U 0647  1              fs_length(ira) = .hold_ira_length;
;  522          U 0648  1              RETURN false;
;  523          U 0649  1              END;
;  524          U 0650  1      END;
;  525          U 0651  1      END;
;  526          U 0652  1      RETURN true;                    ! to make BLISS happy.
;  527          U 0653  1      END;                            ! End of GETSUB.
;  528          U 0654  1
;  529            0655  1      %FI
```

```
;  531     0656  1 GLOBAL ROUTINE getlet (input_string, number_value, number_length) : NOVALUE =
;  532     0657  1
;  533     0658  1 !++
;  534     0659  1 ! FUNCTIONAL DESCRIPTION:
;  535     0660  1 !
;  536     0661  1 !       GETLET is called from GETSUB or NM in order to resolve a string that
;  537     0662  1 !       will be turned into a 'number'.
;  538     0663  1 !
;  539     0664  1 ! FORMAL PARAMETERS:
;  540     0665  1 !
;  541     0666  1 !       input_string    - The string containing the value sought.
;  542     0667  1 !       number_value    - Contains the binary equivalent of the original input.
;  543     0668  1 !       number_length   - Contains the number of digits that were processed.
;  544     0669  1 !                         If number_length is zero on return, no number was
;  545     0670  1 !                         found.  In such a case, all other parameters will
;  546     0671  1 !                         also have a zero value.
;  547     0672  1 !
;  548     0673  1 ! IMPLICIT INPUTS:      None
;  549     0674  1 !
;  550     0675  1 ! IMPLICIT OUTPUTS:     None
;  551     0676  1 !
;  552     0677  1 ! ROUTINE VALUE:
;  553     0678  1 ! COMPLETION CODES:
;  554     0679  1 !
;  555     0680  1 !       Returns TRUE if a valid number was found, else FALSE.
;  556     0681  1 !
;  557     0682  1 ! SIDE EFFECTS:
;  558     0683  1 !
;  559     0684  1 !       none
;  560     0685  1 !--
;  561     0686  1
;  562     0687  2 BEGIN
;  563     0688  2 BIND
;  564     0689  2     ira = .input_string :  REF FIXED_STRING,
;  565     0690  2     nv  = .number_value,
;  566     0691  2     nl  = .number_length;
;  567     0692  2
;  568     0693  2 LOCAL
;  569     0694  2     gname_result;
;  570     0695  2
;  571     0696  2 ! Initialize the temporary fixed string where the result is returned.
;  572     0697  2 fs_init (fs01);
;  573     0698  2 !
;  574     0699  2 ! The routine GNAME uses khar in doing its processing.  Therefore we must
;  575     0700  2 ! update khar so everthing works correctly.
;  576     0701  2 !
;  577     0702  2 !       $XPO_PUT ( IOB    = .rnoiob
;  578     0703  2 !                 ,STRING = $STR_CONCAT( ' *****  Value of khar: '
;  579     0704  2 !                                       ,$STR_ASCII (.khar)
;  580     0705  2 !                                       ));
;  581     0706  2 !       $XPO_PUT ( IOB    = .rnoiob
;  582     0707  2 !                 ,STRING = $STR_CONCAT( ' length of ''ira'': '
;  583     0708  2 !                                       ,$STR_ASCII (.fs_length(ira))
;  584     0709  2 !                                       ));
;  585     0710  2 !
;  586     0711  2 ! Now try to get an appendix name specified as a string of letters.
;  587     0712  2 gname_result = gname (ira, fs01);
```

```
;  588          0713  2
;  589          0714  2 ! If the user specified a string of letters convert to binary representation.
;  590          0715  2 IF .gname_result NEQ gname_no_name
;  591          0716  2 THEN
;  592          0717  3         BEGIN
;  593          0718  3         nl = .fs_length (fs01);
;  594          0719  3         nv = convlb (.fs_start (fs01), .fs_length (fs01));
;  595          0720  2         END;
;  596          0721  1 END;                ! End of GETLET.



                                        0004 00000         .ENTRY  GETLET, Save R2                 ; 0656
                              52 00000000G  EF   9E 00002  MOVAB   FS01, R2                        ;
                                        0C  A2   D4 00009  CLRL    FS01+12                       ; 0697
                              62        10  A2   9E 0000C  MOVAB   FS01+16, FS01                  ;
                           04   A2           62   D0 00010  MOVL    FS01, FS01+4                  ;
                                             52   DD 00014  PUSHL   R2                            ; 0712
                                        04   AC   DD 00016  PUSHL   INPUT_STRING                   ;
                        00000000G   EF           02   FB 00019  CALLS   #2, GNAME                 ;
                                    02           50   D1 00020  CMPL    GNAME_RESULT, #2          ; 0715
                                                 15   13 00023  BEQL    1$                        ;
                           0C   BC       0C      A2   D0 00025  MOVL    FS01+12, @NUMBER_LENGTH   ; 0718
                                        0C  A2   DD 0002A  PUSHL   FS01+12                       ; 0719
                                             62   DD 0002D  PUSHL   FS01                         ;
                        00000000G   EF           02   FB 0002F  CALLS   #2, CONVLB               ;
                              08    BC           50   D0 00036  MOVL    R0, @NUMBER_VALUE         ;
                                             04   0003A 1$:     RET                               ; 0721

; Routine Size:  59 bytes,    Routine Base:  $CODE$ + 0123

;  597          0722  1
;  598          0723  1 END
;  599          0724  0 ELUDOM



;                                  PSECT SUMMARY

;       Name                    Bytes                           Attributes

; $OWN$                            4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
; $CODE$                         350  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        74        12       252          00:00.1
;  _$255$DUA28:[RUNOFF.SRC]DSRLIB.L32;1    1248        16         1        86          00:00.3




;                                       COMMAND QUALIFIERS

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

; Size:          350 code + 4 data bytes
; Run Time:          00:10.0
; Elapsed Time:      00:28.4
; Lines/CPU Min:      4348
; Lexemes/CPU-Min: 17195
; Memory Used:  114 pages
; Compilation Complete
```