K 6

```
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFF              000         000  RRR         RRR  RRR         RRR        TTT        LLL
FFFFFFFFFFFF     000         000  RRRRRRRRRRRR     RRRRRRRRRRRR           TTT        LLL
FFFFFFFFFFFF     000         000  RRRRRRRRRRRR     RRRRRRRRRRRR           TTT        LLL
FFFFFFFFFFFF     000         000  RRRRRRRRRRRR     RRRRRRRRRRRR           TTT        LLL
FFF              000         000  RRR   RRR        RRR   RRR              TTT        LLL
FFF              000         000  RRR   RRR        RRR   RRR              TTT        LLL
FFF              000         000  RRR   RRR        RRR   RRR              TTT        LLL
FFF              000         000  RRR      RRR     RRR      RRR           TTT        LLL
FFF              000         000  RRR      RRR     RRR      RRR           TTT        LLL
FFF              000         000  RRR      RRR     RRR      RRR           TTT        LLL
FFF                 000000000     RRR         RRR  RRR         RRR        TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR  RRR         RRR        TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR  RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

FILEID**FORNMLTAB

```
FFFFFFFFFF    000000    RRRRRRRR    NN      NN  MM      MM  LL          TTTTTTTTTT    AAAAAA    BBBBBBBB
FFFFFFFFFF    000000    RRRRRRRR    NN      NN  MM      MM  LL          TTTTTTTTTT    AAAAAA    BBBBBBBB
FF          00      00  RR      RR  NN      NN  MMMM  MMMM  LL              TT      AA      AA  BB      BB
FF          00      00  RR      RR  NN      NN  MMMM  MMMM  LL              TT      AA      AA  BB      BB
FF          00      00  RR      RR  NNNN    NN  MM  MM  MM  LL              TT      AA      AA  BB      BB
FF          00      00  RR      RR  NNNN    NN  MM  MM  MM  LL              TT      AA      AA  BB      BB
FFFFFFFF    00      00  RRRRRRRR    NN  NN  NN  MM      MM  LL              TT      AA      AA  BBBBBBBB
FFFFFFFF    00      00  RRRRRRRR    NN  NN  NN  MM      MM  LL              TT      AA      AA  BBBBBBBB
FF          00      00  RR  RR      NN    NNNN  MM      MM  LL              TT      AAAAAAAAAA  BB      BB
FF          00      00  RR  RR      NN    NNNN  MM      MM  LL              TT      AAAAAAAAAA  BB      BB
FF          00      00  RR    RR    NN      NN  MM      MM  LL              TT      AA      AA  BB      BB
FF          00      00  RR    RR    NN      NN  MM      MM  LL              TT      AA      AA  BB      BB      ....
FF            000000    RR      RR  NN      NN  MM      MM  LLLLLLLLLL      TT      AA      AA  BBBBBBBB        ....
FF            000000    RR      RR  NN      NN  MM      MM  LLLLLLLLLL      TT      AA      AA  BBBBBBBB        ....

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
```

```
0 MODULE FOR$NML_TABLES (%TITLE 'FOR$NML_TABLES - TPARSE state tables for NAMELIST input'
0                 IDENT = '1-012'         ! File: FORNMLTAB.B32 Edit: SBL1012
0                 ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIAB.  TY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:     FORTRAN Language Support
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains the LIB$TPARSE state tables used in
1 !       implementing FORTRAN NAMELIST input.  It also contains the
1 !       action routines associated with the state tables.
1 !
1 ! ENVIRONMENT:  User mode - AST reentrant
1 !
1 ! AUTHOR: Steven B. Lionel, CREATION DATE: 10-July-1980
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  SBL 10-July-1980
1 ! 1-002 - Disallow superfluous commas.  SBL 18-Nov-1980
1 ! 1-003 - Reflect change in group block spec so that number-of-variables is
1 !         a word; second word is reserved.  SBL 5-Dec-1980
1 ! 1-004 - Don't require a delimiter before ending $ or &.  These characters can
1 !         no longer be a part of a logical constant.  SBL 17-Dec-1980
1 ! 1-005 - Allow repeated nulls of the form "r*".  Don't consider repeated values
1 !         as candidates for being identifiers.  Add comments.  SBL 2-Mar-1981
1 ! 1-006 - Add text describing the NAMELIST descriptor block.  Disallow an array
1 !         substring without a subscript.  SBL 15-April-1981
1 ! 1-007 - Change to use new OTS$CVT_T_F routine.  SBL 15-April-1981
1 ! 1-008 - Also use OTS$CVT_T_F in STORE_COMPLEX.  SBL 5-June-1981
1 ! 1-009 - Use new ONE_OF macro where necessary.  SBL 18-Dec-1981
```

```
1 ! **** Start post-V3.0 enhancements. ****
1 ! 1-010 - Enhancements and minor bug fixes:  SBL 17-Dec-1982
1 !             1. Allow "!" to begin an end-of-line comment.  It is allowed
1 !                wherever "end-of-line" is allowed, except in character values,
1 !                and is equivalent to end-of-line.
1 !             2. Disallow signed integer as syntactically correct for
1 !                repeat count.
1 !             3. Improve error reporting by chaining FOR$_INVTEXREC for
1 !                input conversion errors.
1 !             4. Use prologue file.
1 ! 1-011 - Turn off NML$V_IMAG in STCRE_VALUE so that subsequent complex values
1 !         get stored correctly.  (Same as BUG 1-009A)  SPR 11-nnnnn SBL 7-Mar-1983
1 ! 1-012 - Add inquiry feature.  SBL 24-May-1983
1 !--
1
```

```
;    74         0073  1 %SBTTL 'Declarations'
;    75         0074  1 !
;    76         0075  1 ! PROLOGUE FILE:
;    77         0076  1 !
;    78         0077  1
;    79         0078  1 REQUIRE 'RTLIN:FORPROLOG';                          ! FORTRAN-specific declarations
;    80         0144  1
;    81         0145  1 !
;    82         0146  1 ! LINKAGES:
;    83         0147  1 !
;    84         0148  1
;    85         0149  1 LINKAGE
;    86         0150  1     JSB_COMPARE_UPCASE = JSB (REGISTER=4, REGISTER=5) :
;    87         0151  1                          NOPRESERVE (0,1,2,3,4) NOTUSED (6,7,8,9,10,11);
;    88         0152  1
;    89         0153  1 !
;    90         0154  1 ! TABLE OF CONTENTS:
;    91         0155  1 !
;    92         0156  1
;    93         0157  1 FORWARD ROUTINE
;    94         0158  1     NEXT_RECORD,                            ! Read another record
;    95         0159  1     LOOKUP_IDENTIFIER,                      ! Lookup identifier
;    96         0160  1     SUBSTRING_COLON,                        ! Process colon in substring
;    97         0161  1     INIT_SUBS,                              ! Start a subscript/substring
;    98         0162  1     STORE_SUBS,                             ! Store a subscript/substring
;    99         0163  1     END_SUBSCRIPT,                          ! End a subscript
;   100         0164  1     END_SUBSTRING,                          ! End a substring
;   101         0165  1     CONVERT_INTEGER,                        ! Convert a decimal integer
;   102         0166  1     STORE_LOGICAL,                          ! Store a logical into CONSBLOCK
;   103         0167  1     STORE_REAL,                             ! Store a real value into CONSBLOCK
;   104         0168  1     STORE_COMPLEX,                          ! Store a complex value into CONSBLOCK
;   105         0169  1     STORE_REPEAT,                           ! Store repeat count
;   106         0170  1     END_REPEAT,                             ! End a repeated value
;   107         0171  1     STORE_CHARACTER,                        ! Store a character string character
;   108         0172  1     END_CHARACTER,                          ! End a character string
;   109         0173  1     STRING_OK,                              ! Is a string value ok?
;   110         0174  1     STORE_VALUE,                            ! Store a value
;   111         0175  1     NULL_VALUE,                             ! Skip an element
;   112         0176  1     SET_VALUE_IDENT,                        ! Indicate last value was an identifier
;   113         0177  1     WAS_VALUE_IDENT,                        ! Lookup last value token as an identifier
;   114         0178  1     SYNTAX_ERROR,                           ! Signal a syntax error
;   115         0179  1     INVREFVAR_ERROR,                        ! Signal invalid ref to variable error
;   116         0180  1     INPCONERR_ERROR,                        ! Signal input conversion error
;   117         0181  1     BLANKS_OFF,                             ! Turn explicit blanks off
;   118         0182  1     BLANKS_ON,                              ! Turn explicit blanks on
;   119         0183  1     COMPUTE_INDEX,                          ! Compute the subscript index
;   120         0184  1     COMPARE_UPCASE: JSB_COMPARE_UPCASE,     ! Compare strings upcased
;   121         0185  1     DUMP_NAMES,                             ! Respond to '?' inquiry
;   122         0186  1     DUMP_VALUES;                            ! Respond to '=?' inquiry
;   123         0187  1
;   124         0188  1 !
;   125         0189  1 ! REQUIRE FILES:
;   126         0190  1 !
;   127         0191  1
;   128         0192  1 LIBRARY 'RTLTPAMAC';                                ! TPARSE library of macros
;   129         0193  1
;   130         0194  1 !
;  131          0195  1 ! EQUATED SYMBOLS:
;  132          0196  1 !
;  133          0197  1
;  134          0198  1 LITERAL
;  135          0199  1     SINGLE_QUOTE = 39,                        ! ASCII value for "'"
;  136          0200  1     K_NULL = 0,                               ! Constant type for null value
;  137          0201  1     K_LOGICAL = 1,                            ! Constant type for logical
;  138          0202  1     K_INTEGER = 2,                            ! Constant type for integer
;  139          0203  1     K_REAL = 3,                               ! Constant type for real
;  140          0204  1     K_COMPLEX = 4,                            ! Constant type for complex
;  141          0205  1     K_CHARACTER = 5;                          ! Constant type for character
;  142          0206  1
;  143          0207  1 !
;  144          0208  1 ! FIELDS:
;  145          0209  1 !
;  146          0210  1 !       NONE
;  147          0211  1 !
;  148          0212  1 ! OWN STORAGE:
;  149          0213  1 !
;  150          0214  1 !       NONE
;  151          0215  1 !
;  152          0216  1 ! BUILTIN DECLARATIONS:
;  153          0217  1 !
;  154          0218  1 BUILTIN
;  155          0219  1     CALLG,
;  156          0220  1     INDEX;
;  157          0221  1
;  158          0222  1 !
;  159          0223  1 ! EXTERNAL REFERENCES:
;  160          0224  1 !
;  161          0225  1
;  162          0226  1 EXTERNAL ROUTINE
;  163          0227  1     FOR$$CVT_TYPE,                    ! Convert a value to destination type
;  164          0228  1     FOR$$DO_NML_OUTPUT: CALL_CCB,     ! Do Namelist output
;  165          0229  1     FOR$$REC_RSNO: JSB_RECO,          ! Read a record
;  166          0230  1     FOR$$REC_WSNO: JSB_RECO,          ! Start a write
;  167          0231  1     FOR$$SIGNAL: NOVALUE,             ! Signal continuable error
;  168          0232  1     FOR$$SIGNAL_STO: NOVALUE,         ! Signal fatal error
;  169          0233  1     OTS$CVT_TI_L,                     ! Convert decimal to longword
;  170          0234  1     OTS$CVT_TL_L,                     ! Convert logical to longword
;  171          0235  1     OTS$CVT_T_F,                      ! Convert text to F_floating
;  172          0236  1     OTS$CVT_T_D,                      ! Convert text to D_floating
;  173          0237  1     OTS$CVT_T_G,                      ! Convert text to G_floating
;  174          0238  1     OTS$CVT_T_H,                      ! Convert text to H_floating
;  175          0239  1     LIB$SIG_TO_RET;                   ! Convert signal to return value
;  176          0240  1
;  177          0241  1 !<BLF/PAGE>
```

```
1 !++
1 !   Each NAMELIST descriptor block has the following form:
1 !
1 !        3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 !        1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
1 !
1 !       +---------------------------------------------------------------+
1 !     0 !             Address of ASCIC name of NAMELIST group           !
1 !       +---------------------------------------------------------------+
1 !     1 !          Reserved           !   Count of NAMELIST variables   !
1 !       +---------------------------------------------------------------+
1 !     2 !               Address of ASCIC name of variable 1             !
1 !       +---------------------------------------------------------------+
1 !     3 !         Address of standard VAX descriptor for variable 1     !
1 !       +---------------------------------------------------------------+
1 !     4 !                              ...                              !
1 !       +---------------------------------------------------------------+
1 !     5 !               Address of ASCIC name of variable n             !
1 !       +---------------------------------------------------------------+
1 !     6 !         Address of standard VAX descriptor for variable n     !
1 !       +---------------------------------------------------------------+
1 !
1 !
1 !   The NAMELIST group name and the variable names which are pointed to in
1 !   the  NAMELIST  descriptor  block  are  upper  case  only.  The FORTRAN
1 !   compiler or other calling program is responsible for  case  conversion
1 !   of the name strings.  In NAMELIST input data, case is significant only
1 !   in character literals.  The run-time library is responsible  for  case
1 !   conversion of NAMELIST input data.
1 !
1 !   The allowable data types in variable descriptors are  BU  (BYTE),  WU,
1 !   LU,  W,  L,  F,  D, G, H, T, FC, DC, and GC.  The allowable descriptor
1 !   classes are scalar and array.  For the  array  class  descriptor,  the
1 !   descriptor  flags  COLUMN,  COEFF,  and BOUNDS must be set, indicating
1 !   column-major order and the presence of coefficient and bounds  blocks.
1 !   The number of dimensions must not exceed 7.
1 !--
1
1 !<BLF/PAGE>
```

```
  1 %SBTTL 'FOR$$NML_TABLES - TPARSE tables for NAMELIST input'
  1
  1 !++
  1 ! FUNCTIONAL DESCRIPTION:
  1 !
  1 !       The following are the state tables used to perform FORTRAN
  1 !       NAMELIST input.
  1 !
  1 !--
  1
  1 $INIT_STATE (FOR$$A_NMLSTATE, FOR$$A_NMLKEYWD);
  1
  1 !+
  1 ! Main scanning loop.  Look for assignments.
  1 ! If a $ or & is found, terminate the statement.
  1 !-
P 1 $STATE (BEGIN_SCAN,
P 1     ((END_OF_LINE), BEGIN_SCAN, NEXT_RECORD),
P 1     ('$', TPA$_EXIT),
P 1     ('&', TPA$_EXIT),
P 1     ((ASSIGNMENT), BEGIN_SCAN, BLANKS_OFF),
P 1     (TPA$_LAMBDA, ERROR_STATE)
  1     );
  1
  1 !+
  1 ! This state matches the equivalent of an end-of-line; either the
  1 ! actual end-of-line or a comment beginning with "!", but it does
  1 ! not consume the "!".
  1 !-
P 1 $STATE (END_OF_LINE,
P 1     (TPA$_EOS, TPA$_EXIT),
P 1     ((NO_COMMENT), TPA$_FAIL),
P 1     (TPA$_LAMBDA, TPA$_EXIT)
  1     );
  1
  1 !+
  1 ! An assignment consists of a variable, an equals sign, and a list of values.
  1 !-
P 1 $STATE (ASSIGNMENT,
P 1     ((VARIABLE), ASSN_EQL ,BLANKS_OFF),
P 1     ('?', FLUSH_RECORD, DUMP_NAMES),            ! Dump names
P 1     ((EQUALS_QUESTION), FLUSH_RECORD, DUMP_VALUES), ! Dump values and retry
P 1     (TPA$_LAMBDA, ERROR_STATE)
  1     );
  1
P 1 $STATE (FLUSH_RECORD,
P 1     (TPA$_EOS, TPA$_EXIT),
P 1     (TPA$_ANY, FLUSH_RECORD)
  1     );
  1
P 1 $STATE (ASSN_EQL,
P 1     ((END_OF_LINE), ASSN_EQL, NEXT_RECORD),
P 1     ('=', VALUE_LIST),
P 1     (TPA$_LAMBDA, ERROR_STATE)
  1     );
  1
  1 !+
;  276          0338  1 ! A value list consists of simple values and repeated values, possibly separated
;  277          0339  1 ! by commas.  A comma instead of a value indicates an omitted value, where that
;  278          0340  1 ! element of the variable should remain unchanged.
;  279          0341  1 !-
;  280        P 0342  1 $STATE (VALUE_LIST,
;  281        P 0343  1     ((END_OF_LINE), VALUE_LIST, NEXT_RECORD),
;  282        P 0344  1     (',', VALUE_LIST, NULL_VALUE),
;  283        P 0345  1     ((REPEATED_VALUE), VALUE_LIST1, BLANKS_ON),
;  284        P 0346  1     ((VALUE), VALUE_LIST1, BLANKS_ON),
;  285        P 0347  1     (TPA$_LAMBDA, TPA$_EXIT)
;  286          0348  1     );
;  287          0349  1
;  288          0350  1 !+
;  289          0351  1 ! A value has been found.  The next delimiter tells us if that token was really
;  290          0352  1 ! a value or was an identifier that looked like a value.
;  291          0353  1 !-
;  292        P 0354  1 $STATE (VALUE_LIST1,
;  293        P 0355  1     ((END_OF_LINE), VALUE_LIST2, BLANKS_OFF),
;  294        P 0356  1     (TPA$_BLANK, VALUE_LIST2, BLANKS_OFF),
;  295        P 0357  1     ((NO_LPAREN), VALUE_LIST2, BLANKS_OFF),      ! Succeeds if "(" NOT found
;  296        P 0358  1     (TPA$_LAMBDA, TPA$_EXIT, SET_VALUE_IDENT)    ! Last token was an identifier
;  297          0359  1     );
;  298          0360  1
;  299          0361  1 !+
;  300          0362  1 ! At this point, the last token was an identifier only if the next significant
;  301          0363  1 ! character is an "=".  The other case, a "(", was taken care of in the
;  302          0364  1 ! previous state.
;  303          0365  1 !-
;  304        P 0366  1 $STATE (VALUE_LIST2,
;  305        P 0367  1     ((END_OF_LINE), VALUE_LIST2, NEXT_RECORD),
;  306        P 0368  1     (TPA$_BLANK, VALUE_LIST2),                   ! Even though explicit blank
;  307        P 0369  1                                                  ! processing is off, use up
;  308        P 0370  1                                                  ! blanks in the record to aid
;  309        P 0371  1                                                  ! error reporting.
;  310        P 0372  1     (',', VALUE_LIST, STORE_VALUE),
;  311        P 0373  1     ((NO_EQUALS), VALUE_LIST, STORE_VALUE),      ! Succeeds if "=" NOT found
;  312        P 0374  1     (TPA$_LAMBDA, TPA$_EXIT, SET_VALUE_IDENT)    ! Last token was an identifier
;  313          0375  1     );
;  314          0376  1
;  315          0377  1 !+
;  316          0378  1 ! This type of state determines if the next character is "(", without consuming
;  317          0379  1 ! the character.  In this case, failure indicates that the desired character
;  318          0380  1 ! was found.  This scheme is used in the next, and in other states.
;  319          0381  1 !-
;  320        P 0382  1 $STATE (NO_LPAREN,
;  321        P 0383  1     ('(', TPA$_FAIL),
;  322        P 0384  1     (TPA$_LAMBDA, TPA$_EXIT)
;  323          0385  1     );
;  324          0386  1
;  325        P 0387  1 $STATE (NO_EQUALS,
;  326        P 0388  1     ((NO_EQUALS_QUESTION), NO_EQUALS2),
;  327        P 0389  1     (TPA$_LAMBDA, TPA$_EXIT)
;  328          0390  1     );
;  329          0391  1
;  330        P 0392  1 $STATE (NO_EQUALS2,
;  331        P 0393  1     ('=', TPA$_FAIL),
;  332        P 0394  1     (TPA$_LAMBDA, TPA$_EXIT)
```

```
   1        );
   1
P  1  $STATE (NO_EQUALS_QUESTION,
P  1        ((EQUALS_QUESTION), TPA$_FAIL),
P  1        (TPA$_LAMBDA, TPA$_EXIT)
   1        );
   1
   1  !+
   1  ! Look for '=?'
   1  !-
P  1  $STATE (EQUALS_QUESTION,
P  1        ('=', , BLANKS_ON)                      ! Does it start with '='?
   1        );
   1
P  1  $STATE (,
P  1        ('?', TPA$_EXIT, BLANKS_OFF),          ! '=?' found
P  1        (TPA$_LAMBDA, TPA$_FAIL, BLANKS_OFF)
   1        );
   1
P  1  $STATE (NO_COMMENT,
P  1        ('!', TPA$_FAIL),
P  1        (TPA$_LAMBDA, TPA$_EXIT)
   1        );
   1
   1  !+
   1  ! A repeated value is of the form n*value, where n is an unsigned integer and
   1  ! no delimiters appear on either side of the "*".  A repeated null is of the
   1  ! form "n*" where a delimiter follows the "*".
   1  !-
   1
P  1  $STATE (REPEATED_VALUE,
P  1        (TPA$_DECIMAL, REPEAT2, BLANKS_ON)  ! Value stored in TPA$L_NUMBER
   1        );
   1
P  1  $STATE (REPEAT2,
P  1        ('*', REPEAT3, STORE_REPEAT),
P  1        (TPA$_LAMBDA, TPA$_FAIL, BLANKS_OFF)
   1        );
   1
P  1  $STATE (REPEAT3,
P  1        ((VALUE), TPA$_EXIT, END_REPEAT),   ! n*c
P  1        ((NOT_DELIM), ERROR_STATE),         ! Not n*
P  1        (TPA$_LAMBDA, TPA$_EXIT, BLANKS_OFF)! Is "n*", skipping will be done by STORE_VALUE
   1        );
   1
   1  !+
   1  ! A value can be one of four types.  Integers look like reals, for our purposes.
   1  ! This state can fail if the current string isn't matched by any of these patterns.
   1  !-
P  1  $STATE (VALUE,
P  1        ((LOGICAL), TPA$_EXIT, STORE_LOGICAL),
P  1        ((REAL), TPA$_EXIT, STORE_REAL),
P  1        ((COMPLEX), TPA$_EXIT),         ! Stores are done for each part
P  1        ((CHARACTER), TPA$_EXIT, END_CHARACTER)
   1        );
   1
   1
```

```
;  390          0452  1 !+
;  391          0453  1 ! A variable consists of an identifier, followed by an optional subscript,
;  392          0454  1 ! followed by an optional substring.  If, while parsing values for the previous
;  393          0455  1 ! assignment, it was determined that the last "value" was really an identifier,
;  394          0456  1 ! WAS_VALUE_IDENT will retrieve the token from NML$T_TOKEN and call LOOKUP_IDENTIFIER
;  395          0457  1 ! itself.  Otherwise, we look for an identifier here.
;  396          0458  1 !-
;  397          0459  1
;  398        P 0460  1 $STATE (VARIABLE,
;  399        P 0461  1     (TPA$_LAMBDA, VARIABLE2, WAS_VALUE_IDENT),  ! Fails if last token was not
;  400        P 0462  1                                                 ! an identifier.  If it succeeds,
;  401        P 0463  1                                                 ! lookup is done.
;  402        P 0464  1     ((IDENTIFIER), VARIABLE2, LOOKUP_IDENTIFIER),
;  403          0465  1     );
;  404          0466  1
;  405        P 0467  1 $STATE (VARIABLE2,
;  406        P 0468  1     (TPA$_LAMBDA, , BLANKS_ON)
;  407          0469  1     );
;  408          0470  1
;  409          0471  1 !+
;  410          0472  1 ! Look for subscript or substring.
;  411          0473  1 !-
;  412        P 0474  1 $STATE (SUBSCRIPT_START,
;  413        P 0475  1     ('(', SUB_LOOP1, INIT_SUBS),        ! Signals error if subscript/substring not ok
;  414        P 0476  1     (TPA$_LAMBDA, TPA$_EXIT)
;  415          0477  1     );
;  416          0478  1
;  417          0479  1 !+
;  418          0480  1 ! Get first subscript or first substring.  We can't tell which is which until
;  419          0481  1 ! we see the ":".
;  420          0482  1 !-
;  421        P 0483  1 $STATE (SUB_LOOP1,
;  422        P 0484  1     ((END_OF_LINE), SUB_LOOP1, NEXT_RECORD),
;  423        P 0485  1     (TPA$_BLANK, SUB_LOOP1),
;  424        P 0486  1     ((DECIMAL_INTEGER), , STORE_SUBS),
;  425        P 0487  1     (':', RIGHT_SUBSTRING, SUBSTRING_COLON),    ! Succeeds if substring ok
;  426        P 0488  1                                                 ! otherwise signals FOR$_INVREFVAR
;  427        P 0489  1     (TPA$_LAMBDA, INVREFVAR_STATE)              ! Signal FOR$_INVREFVAR
;  428          0490  1     );
;  429          0491  1
;  430          0492  1 !+
;  431          0493  1 ! This state and the next one consist of the loop looking for subscripts.
;  432          0494  1 ! if a colon is found, control transfers to the substring processor.
;  433          0495  1 !-
;  434        P 0496  1 $STATE (SUB_LOOP2,
;  435        P 0497  1     ((END_OF_LINE), SUB_LOOP2, NEXT_RECORD),
;  436        P 0498  1     (TPA$_BLANK, SUB_LOOP2),
;  437        P 0499  1     (',', SUB_LOOP3),
;  438        P 0500  1     (':', RIGHT_SUBSTRING, SUBSTRING_COLON),    ! Succeeds if substring ok
;  439        P 0501  1                                                 ! otherwise signals FOR$_INVREFVAR
;  440        P 0502  1     (')', START_SUBSTRING, END_SUBSCRIPT),
;  441        P 0503  1     (TPA$_LAMBDA, ERROR_STATE)
;  442          0504  1     );
;  443          0505  1
;  444        P 0506  1 $STATE (SUB_LOOP3,
;  445        P 0507  1     ((END_OF_LINE), SUB_LOOP3, NEXT_RECORD),
;  446        P 0508  1     (TPA$_BLANK, SUB_LOOP3),
;  447        P 0509  1       ((DECIMAL_INTEGER), SUB_LOOP2, STORE_SUBS),
;  448        P 0510  1       (TPA$_LAMBDA, INVREFVAR_STATE)                ! Signal FOR$_INVREFVAR
;  449          0511  1       );
;  450          0512  1
;  451          0513  1 !+
;  452          0514  1 ! This state is reached if we have already processed a subscript.  At this point,
;  453          0515  1 ! only a substring is allowed.
;  454          0516  1 !-
;  455        P 0517  1 $STATE (START_SUBSTRING,
;  456        P 0518  1       ('(', , INIT_SUBS),
;  457        P 0519  1       (TPA$_LAMBDA, TPA$_EXIT)
;  458          0520  1       );
;  459          0521  1
;  460        P 0522  1 $STATE (LEFT_SUBSTRING,
;  461        P 0523  1       ((END_OF_LINE), LEFT_SUBSTRING, NEXT_RECORD),
;  462        P 0524  1       (TPA$_BLANK, LEFT_SUBSTRING),
;  463        P 0525  1       ((DECIMAL_INTEGER), SUBSTRING2, STORE_SUBS),
;  464        P 0526  1       (':', RIGHT_SUBSTRING, SUBSTRING_COLON),
;  465        P 0527  1       (TPA$_LAMBDA, INVREFVAR_STATE)                ! Signal FOR$_INVREFVAR
;  466          0528  1       );
;  467          0529  1
;  468        P 0530  1 $STATE (SUBSTRING2,
;  469        P 0531  1       ((END_OF_LINE), SUBSTRING2, NEXT_RECORD),
;  470        P 0532  1       (TPA$_BLANK, SUBSTRING2),
;  471        P 0533  1       (':', RIGHT_SUBSTRING, SUBSTRING_COLON),
;  472        P 0534  1       (TPA$_LAMBDA, ERROR_STATE)
;  473          0535  1       );
;  474          0536  1
;  475        P 0537  1 $STATE (RIGHT_SUBSTRING,
;  476        P 0538  1       ((END_OF_LINE), RIGHT_SUBSTRING, NEXT_RECORD),
;  477        P 0539  1       (TPA$_BLANK, RIGHT_SUBSTRING),
;  478        P 0540  1       ((DECIMAL_INTEGER), SUBSTRING3, STORE_SUBS),
;  479        P 0541  1       (')', TPA$_EXIT, END_SUBSTRING),
;  480        P 0542  1       (TPA$_LAMBDA, INVREFVAR_STATE)                ! Signal FOR$_INVREFVAR
;  481          0543  1       );
;  482          0544  1
;  483        P 0545  1 $STATE (SUBSTRING3,
;  484        P 0546  1       ((END_OF_LINE), SUBSTRING3, NEXT_RECORD),
;  485        P 0547  1       (TPA$_BLANK, SUBSTRING3),
;  486        P 0548  1       (')', TPA$_EXIT, END_SUBSTRING),
;  487        P 0549  1       (TPA$_LAMBDA, ERROR_STATE)
;  488          0550  1       );
;  489          0551  1
;  490          0552  1 !+
;  491          0553  1 ! An identifier is a letter followed by 0 or more letters, digits, "$" or "_".
;  492          0554  1 !-
;  493        P 0555  1 $STATE (IDENTIFIER,
;  494        P 0556  1       (TPA$_ALPHA, , BLANKS_ON)
;  495          0557  1       );
;  496          0558  1
;  497        P 0559  1 $STATE (,
;  498        P 0560  1       (TPA$_SYMBOL, TPA$_EXIT, BLANKS_OFF),         ! Matches any string whose characters
;  499        P 0561  1                                                     ! consist of letters, digits,
;  500        P 0562  1                                                     ! "$" and "_".
;  501        P 0563  1       (TPA$_LAMBDA, TPA$_EXIT, BLANKS_OFF),
;  502          0564  1       );
;  503          0565  1
;  504        P 0566  1 $STATE (DECIMAL_INTEGER,
;  505        P 0567  1     ((INTEGER), TPA$_EXIT, CONVERT_INTEGER)
;  506          0568  1     );
;  507          0569  1
;  508        P 0570  1 $STATE (INTEGER,
;  509        P 0571  1     ('+', , BLANKS_ON),
;  510        P 0572  1     ('-', , BLANKS_ON),
;  511        P 0573  1     (TPA$_LAMBDA, , BLANKS_ON)
;  512          0574  1     );
;  513          0575  1
;  514        P 0576  1 $STATE (,
;  515        P 0577  1     (TPA$_DECIMAL, TPA$_EXIT, BLANKS_OFF),
;  516        P 0578  1     (TPA$_LAMBDA, TPA$_FAIL, BLANKS_OFF)
;  517          0579  1     );
;  518          0580  1
;  519          0581  1 !+
;  520          0582  1 ! Pattern for a REAL value.
;  521          0583  1 !-
;  522        P 0584  1 $STATE (REAL,
;  523        P 0585  1     ('+', , BLANKS_ON),
;  524        P 0586  1     ('-', , BLANKS_ON),
;  525        P 0587  1     (TPA$_LAMBDA, , BLANKS_ON)
;  526          0588  1     );
;  527          0589  1
;  528        P 0590  1 $STATE (REAL1,
;  529        P 0591  1     (TPA$_DIGIT, REAL1),
;  530        P 0592  1     ('.'),
;  531        P 0593  1     (TPA$_LAMBDA)
;  532          0594  1     );
;  533          0595  1
;  534        P 0596  1 $STATE (REAL2,
;  535        P 0597  1     (TPA$_DIGIT, REAL2),
;  536        P 0598  1     (TPA$_LAMBDA)
;  537          0599  1     );
;  538          0600  1
;  539        P 0601  1 $STATE (EXPONENT,
;  540        P 0602  1     ('E'),
;  541        P 0603  1     ('e'),
;  542        P 0604  1     ('D'),
;  543        P 0605  1     ('d'),
;  544        P 0606  1     ('Q'),
;  545        P 0607  1     ('q'),
;  546        P 0608  1     (TPA$_LAMBDA)
;  547          0609  1     );
;  548          0610  1
;  549        P 0611  1 $STATE (,
;  550        P 0612  1     ('+'),
;  551        P 0613  1     ('-'),
;  552        P 0614  1     (TPA$_LAMBDA)
;  553          0615  1     );
;  554          0616  1
;  555        P 0617  1 $STATE (EXPONENT2,
;  556        P 0618  1     (TPA$_DIGIT, EXPONENT2),
;  557        P 0619  1     (TPA$_LAMBDA)
;  558          0620  1     );
;  559          0621  1
;  560        P 0622  1 $STATE (,                          ! Fail if next character is not a delimiter
    ((NOT_DELIM), TPA$_FAIL),     . but don't consume the character.
    (TPA$_LAMBDA, TPA$_EXIT)
    );

$STATE (NOT_DELIM,                ! Fails if next character is a delimiter
    ((END_OF_LINE), TPA$_FAIL),
    (TPA$_BLANK, TPA$_FAIL),
    (',', TPA$_FAIL),
    ('$', TPA$_FAIL),
    ('&', TPA$_FAIL),
    (')', TPA$_FAIL),             ! Can show in complex values
    (TPA$_LAMBDA, TPA$_EXIT)
    );

!+
! Pattern for a logical value.  It is complex because any string can follow
! after the initial T, F, .T or .F up to the next "delimiter".
!-
$STATE (LOGICAL,
    ('.', , BLANKS_ON),
    (TPA$_LAMBDA, , BLANKS_ON)
    );

$STATE (,
    ('T'),
    ('t'),
    ('F'),
    ('f')
    );

!+
! Consume characters up to but not including the next delimiter.
!-
$STATE (LOGICAL1,
    ((LOGICAL2), LOGICAL1),
    (TPA$_LAMBDA, TPA$_EXIT, BLANKS_OFF)
    );

!+
! Indicates by failing if any of the selected characters are found.
!-
$STATE (LOGICAL2,
    ((END_OF_LINE), TPA$_FAIL),
    (TPA$_BLANK, TPA$_FAIL),
    (',', TPA$_FAIL),
    ('(', TPA$_FAIL),
    ('=', TPA$_FAIL),
    ('$', TPA$_FAIL),
    ('&', TPA$_FAIL),
    (TPA$_ANY, TPA$_EXIT)
    );

!+
! Parse and store the representation of a complex value.  This is safe because
! a complex value can not possibly be an identifier.
$STATE (COMPLEX,
    ('(', COMPLEX2)
;  618           0680  1      );
;  619           0681  1
;  620         P 0682  1  $STATE (COMPLEX2,
;  621         P 0683  1      ((END_OF_LINE), COMPLEX2, NEXT_RECORD),
;  622         P 0684  1      (TPA$_BLANK, COMPLEX2),
;  623         P 0685  1      ((REAL), COMPLEX3, STORE_COMPLEX),  ! Store real part
;  624         P 0686  1      (TPA$_LAMBDA, ERROR_STATE)
;  625           0687  1      );
;  626           0688  1
;  627         P 0689  1  $STATE (COMPLEX3,
;  628         P 0690  1      ((END_OF_LINE), COMPLEX3, NEXT_RECORD),
;  629         P 0691  1      (TPA$_BLANK, COMPLEX3),
;  630         P 0692  1      (',', COMPLEX4),
;  631         P 0693  1      (TPA$_LAMBDA, ERROR_STATE)
;  632           0694  1      );
;  633           0695  1
;  634         P 0696  1  $STATE (COMPLEX4,
;  635         P 0697  1      ((END_OF_LINE), COMPLEX4, NEXT_RECORD),
;  636         P 0698  1      (TPA$_BLANK, COMPLEX4),
;  637         P 0699  1      ((REAL), COMPLEX5, STORE_COMPLEX),  ! Store imaginary part
;  638         P 0700  1      (TPA$_LAMBDA, ERROR_STATE)
;  639           0701  1      );
;  640           0702  1
;  641         P 0703  1  $STATE (COMPLEX5,
;  642         P 0704  1      ((END_OF_LINE), COMPLEX5, NEXT_RECORD),
;  643         P 0705  1      (TPA$_BLANK, COMPLEX5),
;  644         P 0706  1      (')', TPA$_EXIT),
;  645         P 0707  1      (TPA$_LAMBDA, ERROR_STATE)
;  646           0708  1      );
;  647           0709  1
;  648           0710  1  !+
;  649           0711  1  ! Pattern for a character string.  Inside the string, two consecutive quotes
;  650           0712  1  ! are counted as one.  This value is stored in the user variable as it goes,
;  651           0713  1  ! since this can not possibly be an identifier.
;  652           0714  1  !-
;  653         P 0715  1  $STATE (CHARACTER,
;  654         P 0716  1      (SINGLE_QUOTE, CHARACTER1, STRING_OK)       ! Signals error if not type CHARACTER
;  655           0717  1      );                                          ! Also turns on TPA$V_BLANKS
;  656           0718  1
;  657         P 0719  1  $STATE (CHARACTER1,
;  658         P 0720  1      (TPA$_EOS, CHARACTER1, NEXT_RECORD),        ! Don't use END_OF_LINE because
;  659         P 0721  1      (SINGLE_QUOTE, NEXT_QUOTE),                 ! a "!" is a valid character.
;  660         P 0722  1      (TPA$_ANY, CHARACTER1, STORE_CHARACTER)
;  661           0723  1      );
;  662           0724  1
;  663         P 0725  1  $STATE (NEXT_QUOTE,
;  664         P 0726  1      (TPA$_EOS, NEXT_QUOTE, NEXT_RECORD),        ! Don't use END_OF_LINE.
;  665         P 0727  1      (SINGLE_QUOTE, CHARACTER1, STORE_CHARACTER),
;  666         P 0728  1      (TPA$_LAMBDA, TPA$_EXIT)
;  667           0729  1      );
;  668           0730  1
;  669           0731  1  !+
;  670           0732  1  ! This state is transferred to if a syntax error is detected in the parsing. It
;  671           0733  1  ! calls SYNTAX_ERROR with a token which is at or near where the error was.
;  672           0734  1  ! SYNTAX_ERROR signals FOR$_SYNERRNAM.
;  673           0735  1  !-
;  674         P 0736  1  $STATE (ERROR_STATE,
;  675        P 0737  1      (TPA$_ANY, TPA$_FAIL, SYNTAX_ERROR),
;  676        P 0738  1      (TPA$_LAMBDA, TPA$_FAIL, SYNTAX_ERROR)
;  677          0739  1      );
;  678          0740  1
;  679          0741  1 !+
;  680          0742  1 ! This state is transferred to when there is some invalid reference on a
;  681          0743  1 ! variable, i.e. subscripting a scalar, substringing a non-character or using
;  682          0744  1 ! non-integers in subscripts/substrings.  It calls INVREFVAR_ERROR which
;  683          0745  1 ! signals FOR$_INVREFVAR.
;  684          0746  1 !-
;  685        P 0747  1 $STATE (INVREFVAR_STATE,
;  686        P 0748  1      (TPA$_LAMBDA, TPA$_FAIL, INVREFVAR_ERROR)
;  687          0749  1      );
;  688          0750  1 !<BLF/PAGE>
```

```
1 %SBTTL 'NEXT_RECORD - Get next record'
1 ROUTINE NEXT_RECORD =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Reads a new record from the current unit and updates the STRING pointers
1 !       in PARAM_BLOCK.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = NEXT_RECORD ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       PARAM_BLOCK [TPA$L_STRINGPTR] is address of new record
1 !       PARAM_BLOCK [TPA$L_STRINGCNT] is record length
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success; all errors are signalled.
1 !
1 ! SIDE EFFECTS:
1 !
1 !
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     GLOBAL REGISTER
2         CCB = 11: REF $FOR$CCB_DECL;
2
2     CCB = .AP [NML$A_CCB];      ! Fetch CCB address
2     DO
3         BEGIN
3         FOR$$REC_RSNO ();                 ! Read the next record
3         CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_PTR] + 1; ! Start with second byte
3         AP [TPA$L_STRINGPTR] = .CCB [LUB$A_BUF_PTR];
3         AP [TPA$L_STRINGCNT] = .CCB [LUB$A_BUF_END] - .CCB [LUB$A_BUF_PTR];
3         END
2     UNTIL .AP [TPA$L_STRINGCNT] GTR 0;
2     RETURN 1;
```

```
;  747          0808  2
;  748          0809  1       END;

                                                             .TITLE  FOR$$NML_TABLES FOR$$NML_TABLES - TPARSE state
;                                                                                    tables for NAMEL
                                                             .IDENT  \1-012\

                                                             .PSECT  _LIB$STATE$,NOWRT,  SHR,  PIC,1

                                               00000 FOR$$A_NMLSTATE::
                                                                     .BLKB   0
                                               00000 BEGIN_SCAN:
                                                                     .BLKB   0
                                         99F8  00000 ;TPA$TYPE
                                                     U.2:    .WORD   -26120                                          ;
                                        0000*  00002 ;TPA$SUBEXP
                                                     U.4:    .WORD   <<U.3-U.4>-2>                                   ;
                                    00000000*  00004 ;TPA$ACTION
                                                     U.5:    .LONG   <<NEXT_RECORD-U.5>-4>                           ;
                                        0000*  00008 ;TPA$TARGET
                                                     U.6:    .WORD   <<BEGIN_SCAN-U.6>-2>                            ;
                                         1024  0000A ;TPA$TYPE
                                                     U.7:    .WORD   4132                                            ;
                                         FFFF  0000C ;TPA$TARGET
                                                     U.8:    .WORD   -1                                              ;
                                         1026  0000E ;TPA$TYPE
                                                     U.9:    .WORD   4134                                            ;
                                         FFFF  00010 ;TPA$TARGET
                                                     U.10:   .WORD   -1                                              ;
                                         99F8  00012 ;TPA$TYPE
                                                     U.11:   .WORD   -26120                                          ;
                                        0000*  00014 ;TPA$SUBEXP
                                                     U.13:   .WORD   <<U.12-U.13>-2>                                 ;
                                    00000000V  00016 ;TPA$ACTION
                                                     U.14:   .LONG   <<BLANKS_OFF-U.14>-4>                           ;
                                        0000*  0001A ;TPA$TARGET
                                                     U.15:   .WORD   <<BEGIN_SCAN-U.15>-2>                           ;
                                         15F6  0001C ;TPA$TYPE
                                                     U.16:   .WORD   5622                                            ;
                                        0000*  0001E ;TPA$TARGET
                                                     U.18:   .WORD   <<U.17-U.18>-2>                                 ;
                                               00020 ;END_OF_LINE
                                                     U.3:    .BLKB   0
                                         11F7  00020 ;TPA$TYPE
                                                     U.19:   .WORD   4599                                            ;
                                         FFFF  00022 ;TPA$TARGET
                                                     U.20:   .WORD   -1                                              ;
                                         19F8  00024 ;TPA$TYPE
                                                     U.21:   .WORD   6648                                            ;
                                        0000*  00026 ;TPA$SUBEXP
                                                     U.23:   .WORD   <<U.22-U.23>-2>                                 ;
                                         FFFE  00028 ;TPA$TARGET
                                                     U.24:   .WORD   -2                                              ;
                                         15F6  0002A ;TPA$TYPE
                                                     U.25:   .WORD   5622                                            ;
                                         FFFF  0002C ;TPA$TARGET
```

```
                                   U.26:    .WORD   -1                                            ;
                           0002E ;ASSIGNMENT
                                   U.12:    .BLKB   0
                    99F8   0002E ;TPA$TYPE
                                   U.27:    .WORD   -26120                                        ;
                   0000*   00030 ;TPA$SUBEXP
                                   U.29:    .WORD   <<U.28-U.29>-2>                               ;
               00000000V   00032 ;TPA$ACTION
                                   U.30:    .LONG   <<BLANKS_OFF-U.30>-4>                         ;
                   0000*   00036 ;TPA$TARGET
                                   U.32:    .WORD   <<U.31-U.32>-2>                               ;
                    903F   00038 ;TPA$TYPE
                                   U.33:    .WORD   -28609                                        ;
               00000000V   0003A ;TPA$ACTION
                                   U.34:    .LONG   <<DUMP_NAMES-U.34>-4>                         ;
                   0000*   0003E ;TPA$TARGET
                                   U.36:    .WORD   <<U.35-U.36>-2>                               ;
                    99F8   00040 ;TPA$TYPE
                                   U.37:    .WORD   -26120                                        ;
                   0000*   00042 ;TPA$SUBEXP
                                   U.39:    .WORD   <<U.38-U.39>-2>                               ;
               00000000V   00044 ;TPA$ACTION
                                   U.40:    .LONG   <<DUMP_VALUES-U.40>-4>                        ;
                   0000*   00048 ;TPA$TARGET
                                   U.41:    .WORD   <<U.35-U.41>-2>                               ;
                    15F6   0004A ;TPA$TYPE
                                   U.42:    .WORD   5622                                          ;
                   0000*   0004C ;TPA$TARGET
                                   U.43:    .WORD   <<U.17-U.43>-2>                               ;
                           0004E ;FLUSH_RECORD
                                   U.35:    .BLKB   0
                    11F7   0004E ;TPA$TYPE
                                   U.44:    .WORD   4599                                          ;
                    FFFF   00050 ;TPA$TARGET
                                   U.45:    .WORD   -1                                            ;
                    15ED   00052 ;TPA$TYPE
                                   U.46:    .WORD   5613                                          ;
                   0000*   00054 ;TPA$TARGET
                                   U.47:    .WORD   <<U.35-U.47>-2>                               ;
                           00056 ;ASSN_EQL
                                   U.31:    .BLKB   0
                    99F8   00056 ;TPA$TYPE
                                   U.48:    .WORD   -26120                                        ;
                   0000*   00058 ;TPA$SUBEXP
                                   U.49:    .WORD   <<U.3-U.49>-2>                                ;
               00000000*   0005A ;TPA$ACTION
                                   U.50:    .LONG   <<NEXT_RECORD-U.50>-4>                        ;
                   0000*   0005E ;TPA$TARGET
                                   U.51:    .WORD   <<U.31-U.51>-2>                               ;
                    103D   00060 ;TPA$TYPE
                                   U.52:    .WORD   4157                                          ;
                   0000*   00062 ;TPA$TARGET
                                   U.54:    .WORD   <<U.53-U.54>-2>                               ;
                    15F6   00064 ;TPA$TYPE
                                   U.55:    .WORD   5622                                          ;
                   0000*   00066 ;TPA$TARGET
                                   U.56:    .WORD   <<U.17-U.56>-2>                               ;
                    00068 ;VALUE_LIST
                          U.53:     .BLKB   0
              99F8  00068 ;TPA$TYPE
                          U.57:     .WORD   -26120                                    ;
             0000*  0006A ;TPA$SUBEXP
                          U.58:     .WORD   <<U.3-U.58>-2>                            ;
         00000000*  0006C ;TPA$ACTION
                          U.59:     .LONG   <<NEXT_RECORD-U.59>-4>                    ;
             0000*  00070 ;TPA$TARGET
                          U.60:     .WORD   <<U.53-U.60>-2>                           ;
              902C  00072 ;TPA$TYPE
                          U.61:     .WORD   -28628                                    ;
         00000000V  00074 ;TPA$ACTION
                          U.62:     .LONG   <<NULL_VALUE-U.62>-4>                     ;
             0000*  00078 ;TPA$TARGET
                          U.63:     .WORD   <<U.53-U.63>-2>                           ;
              99F8  0007A ;TPA$TYPE
                          U.64:     .WORD   -26120                                    ;
             0000*  0007C ;TPA$SUBEXP
                          U.66:     .WORD   <<U.65-U.66>-2>                           ;
         00000000V  0007E ;TPA$ACTION
                          U.67:     .LONG   <<BLANKS_ON-U.67>-4>                      ;
             0000*  00082 ;TPA$TARGET
                          U.69:     .WORD   <<U.68-U.69>-2>                           ;
              99F8  00084 ;TPA$TYPE
                          U.70:     .WORD   -26120                                    ;
             0000*  00086 ;TPA$SUBEXP
                          U.72:     .WORD   <<U.71-U.72>-2>                           ;
         00000000V  00088 ;TPA$ACTION
                          U.73:     .LONG   <<BLANKS_ON-U.73>-4>                      ;
             0000*  0008C ;TPA$TARGET
                          U.74:     .WORD   <<U.68-U.74>-2>                           ;
              15F6  0008E ;TPA$TYPE
                          U.75:     .WORD   5622                                      ;
              FFFF  00090 ;TPA$TARGET
                          U.76:     .WORD   -1                                        ;
                    00092 ;VALUE_LIST1
                          U.68:     .BLKB   0
              99F8  00092 ;TPA$TYPE
                          U.77:     .WORD   -26120                                    ;
             0000*  00094 ;TPA$SUBEXP
                          U.78:     .WORD   <<U.3-U.78>-2>                            ;
         00000000V  00096 ;TPA$ACTION
                          U.79:     .LONG   <<BLANKS_OFF-U.79>-4>                     ;
             0000*  0009A ;TPA$TARGET
                          U.81:     .WORD   <<U.80-U.81>-2>                           ;
              91F2  0009C ;TPA$TYPE
                          U.82:     .WORD   -28174                                    ;
         00000000V  0009E ;TPA$ACTION
                          U.83:     .LONG   <<BLANKS_OFF-U.83>-4>                     ;
             0000*  000A2 ;TPA$TARGET
                          U.84:     .WORD   <<U.80-U.84>-2>                           ;
              99F8  000A4 ;TPA$TYPE
                          U.85:     .WORD   -26120                                    ;
             0000*  000A6 ;TPA$SUBEXP
                          U.87:     .WORD   <<U.86-U.87>-2>                           ;
         00000000V  000A8 ;TPA$ACTION
```

```
                              U.88:     .LONG    <<BLANKS_OFF-U.88>-4>                             ;
            0000*  000AC ;TPA$TARGET
                              U.89:     .WORD    <<U.80-U.89>-2>                                   ;
             95F6  000AE ;TPA$TYPE
                              U.90:     .WORD    -27146                                            ;
        00000000V  000B0 ;TPA$ACTION
                              U.91:     .LONG    <<SET_VALUE_IDENT-U.91>-4>                        ;
             FFFF  000B4 ;TPA$TARGET
                              U.92:     .WORD    -1                                                ;
                   000B6 ;VALUE_LIST2
                              U.80:     .BLKB    0
             99F8  000B6 ;TPA$TYPE
                              U.93:     .WORD    -26120                                            ;
            0000*  000B8 ;TPA$SUBEXP
                              U.94:     .WORD    <<U.3-U.94>-2>                                    ;
        00000000*  000BA ;TPA$ACTION
                              U.95:     .LONG    <<NEXT_RECORD-U.95>-4>                            ;
            0000*  000BE ;TPA$TARGET
                              U.96:     .WORD    <<U.80-U.96>-2>                                   ;
             11F2  000C0 ;TPA$TYPE
                              U.97:     .WORD    4594                                              ;
            0000*  000C2 ;TPA$TARGET
                              U.98:     .WORD    <<U.80-U.98>-2>                                   ;
             902C  000C4 ;TPA$TYPE
                              U.99:     .WORD    -28628                                            ;
        00000000V  000C6 ;TPA$ACTION
                              U.100:    .LONG    <<STORE_VALUE-U.100>-4>                           ;
            0000*  000CA ;TPA$TARGET
                              U.101:    .WORD    <<U.53-U.101>-2>                                  ;
             99F8  000CC ;TPA$TYPE
                              U.102:    .WORD    -26120                                            ;
            0000*  000CE ;TPA$SUBEXP
                              U.104:    .WORD    <<U.103-U.104>-2>                                 ;
        00000000V  000D0 ;TPA$ACTION
                              U.105:    .LONG    <<STORE_VALUE-U.105>-4>                           ;
            0000*  000D4 ;TPA$TARGET
                              U.106:    .WORD    <<U.53-U.106>-2>                                  ;
             95F6  000D6 ;TPA$TYPE
                              U.107:    .WORD    -27146                                            ;
        00000000V  000D8 ;TPA$ACTION
                              U.108:    .LONG    <<SET_VALUE_IDENT-U.108>-4>                       ;
             FFFF  000DC ;TPA$TARGET
                              U.109:    .WORD    -1                                                ;
                   000DE ;NO_LPAREN
                              U.86:     .BLKB    0
             1028  000DE ;TPA$TYPE
                              U.110:    .WORD    4136                                              ;
             FFFE  000E0 ;TPA$TARGET
                              U.111:    .WORD    -2                                                ;
             15F6  000E2 ;TPA$TYPE
                              U.112:    .WORD    5622                                              ;
             FFFF  000E4 ;TPA$TARGET
                              U.113:    .WORD    -1                                                ;
                   000E6 ;NO_EQUALS
                              U.103:    .BLKB    0
             19F8  000E6 ;TPA$TYPE
                              U.114:    .WORD    6648                                              ;
```

```
     0000* 000E8 ;TPA$SUBEXP
                 U.116:  .WORD   <<U.115-U.116>-2>                          ;
     0000* 000EA ;TPA$TARGET
                 U.118:  .WORD   <<U.117-U.118>-2>                          ;
     15F6  000EC ;TPA$TYPE
                 U.119:  .WORD   5622                                       ;
     FFFF  000EE ;TPA$TARGET
                 U.120:  .WORD   -1                                         ;
           000F0 ;NO_EQUALS2
                 U.117:  .BLKB   0
     103D  000F0 ;TPA$TYPE
                 U.121:  .WORD   4157                                       ;
     FFFE  000F2 ;TPA$TARGET
                 U.122:  .WORD   -2                                         ;
     15F6  000F4 ;TPA$TYPE
                 U.123:  .WORD   5622                                       ;
     FFFF  000F6 ;TPA$TARGET
                 U.124:  .WORD   -1                                         ;
           000F8 ;NO_EQUALS_QUESTION
                 U.115:  .BLKB   0
     19F8  000F8 ;TPA$TYPE
                 U.125:  .WORD   6648                                       ;
     0000* 000FA ;TPA$SUBEXP
                 U.126:  .WORD   <<U.38-U.126>-2>                           ;
     FFFE  000FC ;TPA$TARGET
                 U.127:  .WORD   -2                                         ;
     15F6  000FE ;TPA$TYPE
                 U.128:  .WORD   5622                                       ;
     FFFF  00100 ;TPA$TARGET
                 U.129:  .WORD   -1                                         ;
           00102 ;EQUALS_QUESTION
                 U.38:   .BLKB   0
     843D  00102 ;TPA$TYPE
                 U.130:  .WORD   -31683                                     ;
 00000000V 00104 ;TPA$ACTION
                 U.131:  .LONG   <<BLANKS_ON-U.131>-4>                      ;
     903F  00108 ;TPA$TYPE
                 U.132:  .WORD   -28609                                     ;
 00000000V 0010A ;TPA$ACTION
                 U.133:  .LONG   <<BLANKS_OFF-U.133>-4>                     ;
     FFFF  0010E ;TPA$TARGET
                 U.134:  .WORD   -1                                         ;
     95F6  00110 ;TPA$TYPE
                 U.135:  .WORD   -27146                                     ;
 00000000V 00112 ;TPA$ACTION
                 U.136:  .LONG   <<BLANKS_OFF-U.136>-4>                     ;
     FFFE  00116 ;TPA$TARGET
                 U.137:  .WORD   -2                                         ;
           00118 ;NO_COMMENT
                 U.22:   .BLKB   0
     1021  00118 ;TPA$TYPE
                 U.138:  .WORD   4129                                       ;
     FFFE  0011A ;TPA$TARGET
                 U.139:  .WORD   -2                                         .
     15F6  0011C ;TPA$TYPE
                 U.140:  .WORD   5622                                       .
     FFFF  0011E ;TPA$TARGET
```

```
                           U.141:  .WORD   -1                                    ;
                    00120 ;REPEATED_VALUE
                           U.65:    .BLKB   0
              95F3  00120 ;TPA$TYPE
                           U.142:  .WORD   -27149                                ;
         00000000V  00122 ;TPA$ACTION
                           U.143:  .LONG   <<BLANKS_ON-U.143>-4>                 ;
             0000*  00126 ;TPA$TARGET
                           U.145:  .WORD   <<U.144-U.145>-2>                     ;
                    00128 ;REPEAT2
                           U.144:  .BLKB   0
              902A  00128 ;TPA$TYPE
                           U.146:  .WORD   -28630                                ;
         00000000V  0012A ;TPA$ACTION
                           U.147:  .LONG   <<STORE_REPEAT-U.147>-4>              ;
             0000*  0012E ;TPA$TARGET
                           U.149:  .WORD   <<U.148-U.149>-2>                     ;
              95F6  00130 ;TPA$TYPE
                           U.150:  .WORD   -27146                                ;
         00000000V  00132 ;TPA$ACTION
                           U.151:  .LONG   <<BLANKS_OFF-U.151>-4>                ;
              FFFE  00136 ;TPA$TARGET
                           U.152:  .WORD   -2                                    ;
                    00138 ;REPEAT3
                           U.148:  .BLKB   0
              99F8  00138 ;TPA$TYPE
                           U.153:  .WORD   -26120                                ;
             0000*  0013A ;TPA$SUBEXP
                           U.154:  .WORD   <<U.71-U.154>-2>                      ;
         00000000V  0013C ;TPA$ACTION
                           U.155:  .LONG   <<END_REPEAT-U.155>-4>                ;
              FFFF  00140 ;TPA$TARGET
                           U.156:  .WORD   -1                                    ;
              19F8  00142 ;TPA$TYPE
                           U.157:  .WORD   6648                                  ;
             0000*  00144 ;TPA$SUBEXP
                           U.159:  .WORD   <<U.158-U.159>-2>                     ;
             0000*  00146 ;TPA$TARGET
                           U.160:  .WORD   <<U.17-U.160>-2>                      ;
              95F6  00148 ;TPA$TYPE
                           U.161:  .WORD   -27146                                ;
         00000000V  0014A ;TPA$ACTION
                           U.162:  .LONG   <<BLANKS_OFF-U.162>-4>                ;
              FFFF  0014E ;TPA$TARGET
                           U.163:  .WORD   -1                                    ;
                    00150 ;VALUE
                           U.71:    .BLKB   0
              99F8  00150 ;TPA$TYPE
                           U.164:  .WORD   -26120                                ;
             0000*  00152 ;TPA$SUBEXP
                           U.166:  .WORD   <<U.165-U.166>-2>                     ;
         00000000V  00154 ;TPA$ACTION
                           U.167:  .LONG   <<STORE_LOGICAL-U.167>-4>             ;
              FFFF  00158 ;TPA$TARGET
                           U.168:  .WORD   -1                                    ;
              99F8  0015A ;TPA$TYPE
                           U.169:  .WORD   -26120                                ;
```

```
     0000* 0015C ;TPA$SUBEXP
                 U.171:   .WORD    <<U.170-U.171>-2>                          ;
 00000000V 0015E ;TPA$ACTION
                 U.172:   .LONG    <<STORE_REAL-U.172>-4>                     ;
      FFFF 00162 ;TPA$TARGET
                 U.173:   .WORD    -1                                         ;
      19F8 00164 ;TPA$TYPE
                 U.174:   .WORD    6648                                       ;
     0000* 00166 ;TPA$SUBEXP
                 U.176:   .WORD    <<U.175-U.176>-2>                          ;
      FFFF 00168 ;TPA$TARGET
                 U.177:   .WORD    -1                                         ;
      9DF8 0016A ;TPA$TYPE
                 U.178:   .WORD    -25096                                     ;
     0000* 0016C ;TPA$SUBEXP
                 U.180:   .WORD    <<U.179-U.180>-2>                          ;
 00000000V 0016E ;TPA$ACTION
                 U.181:   .LONG    <<END_CHARACTER-U.181>-4>                  ;
      FFFF 00172 ;TPA$TARGET
                 U.182:   .WORD    -1                                         ;
           00174 ;VARIABLE
                 U.28:    .BLKB    0
      91F6 00174 ;TPA$TYPE
                 U.183:   .WORD    -28170                                     ;
 00000000V 00176 ;TPA$ACTION
                 U.184:   .LONG    <<WAS_VALUE_IDENT-U.184>-4>                ;
     0000* 0017A ;TPA$TARGET
                 U.186:   .WORD    <<U.185-U.186>-2>                          ;
      9DF8 0017C ;TPA$TYPE
                 U.187:   .WORD    -25096                                     ;
     0000* 0017E ;TPA$SUBEXP
                 U.189:   .WORD    <<U.188-U.189>-2>                          ;
 00000000V 00180 ;TPA$ACTION
                 U.190:   .LONG    <<LOOKUP_IDENTIFIER-U.190>-4>              ;
     0000* 00184 ;TPA$TARGET
                 U.191:   .WORD    <<U.185-U.191>-2>                          ;
           00186 ;VARIABLE2
                 U.185:   .BLKB    0
      85F6 00186 ;TPA$TYPE
                 U.192:   .WORD    -31242                                     ;
 00000000V 00188 ;TPA$ACTION
                 U.193:   .LONG    <<BLANKS_ON-U.193>-4>                      ;
           0018C SUBSCRIPT_START:
                          .BLKB    0
      9028 0018C ;TPA$TYPE
                 U.194:   .WORD    -28632                                     ;
 00000000V 0018E ;TPA$ACTION
                 U.195:   .LONG    <<INIT_SUBS-U.195>-4>                      ;
     0000* 00192 ;TPA$TARGET
                 U.197:   .WORD    <<U.196-U.197>-2>                          ;
      15F6 00194 ;TPA$TYPE
                 U.198:   .WORD    5622                                       ;
      FFFF 00196 ;TPA$TARGET
                 U.199:   .WORD    -1                                         ;
           00198 ;SUB_LOOP1
                 U.196:   .BLKB    0
      99F8 00198 ;TPA$TYPE
```

```
                                 U.200:  .WORD   -26120                           ;
                  0000*  0019A  ;TPA$SUBEXP
                                 U.201:  .WORD   <<U.3-U.201>-2>                  ;
              00000000*  0019C  ;TPA$ACTION
                                 U.202:  .LONG   <<NEXT_RECORD-U.202>-4>          ;
                  0000*  001A0  ;TPA$TARGET
                                 U.203:  .WORD   <<U.196-U.203>-2>                ;
                  11F2   001A2  ;TPA$TYPE
                                 U.204:  .WORD   4594                             ;
                  0000*  001A4  ;TPA$TARGET
                                 U.205:  .WORD   <<U.196-U.205>-2>                ;
                  89F8   001A6  ;TPA$TYPE
                                 U.206:  .WORD   -30216                           ,
                  0000*  001A8  ;TPA$SUBEXP
                                 U.208:  .WORD   <<U.207-U.208>-2>                ;
              00000000V  001AA  ;TPA$ACTION
                                 U.209:  .LONG   <<STORE_SUBS-U.209>-4>           ;
                  903A   001AE  ;TPA$TYPE
                                 U.210:  .WORD   -28614                           ;
              00000000V  001B0  ;TPA$ACTION
                                 U.211:  .LONG   <<SUBSTRING_COLON-U.211>-4>      ;
                  0000*  001B4  ;TPA$TARGET
                                 U.213:  .WORD   <<U.212-U.213>-2>                ;
                  15F6   001B6  ;TPA$TYPE
                                 U.214:  .WORD   5622                             ;
                  0000*  001B8  ;TPA$TARGET
                                 U.216:  .WORD   <<U.215-U.216>-2>                ;
                         001BA  SUB_LOOP2:
                                         .BLKB   0
                  99F8   001BA  ;TPA$TYPE
                                 U.217:  .WORD   -26120                           ;
                  0000*  001BC  ;TPA$SUBEXP
                                 U.218:  .WORD   <<U.3-U.218>-2>                  ;
              00000000*  001BE  ;TPA$ACTION
                                 U.219:  .LONG   <<NEXT_RECORD-U.219>-4>          ;
                  0000*  001C2  ;TPA$TARGET
                                 U.220:  .WORD   <<SUB_LOOP2-U.220>-2>            ;
                  11F2   001C4  ;TPA$TYPE
                                 U.221:  .WORD   4594                             ;
                  0000*  001C6  ;TPA$TARGET
                                 U.222:  .WORD   <<SUB_LOOP2-U.222>-2>            ;
                  102C   001C8  ;TPA$TYPE
                                 U.223:  .WORD   4140                             ;
                  0000*  001CA  ;TPA$TARGET
                                 U.225:  .WORD   <<U.224-U.225>-2>                ;
                  903A   001CC  ;TPA$TYPE
                                 U.226:  .WORD   -28614                           ;
              00000000V  001CE  ;TPA$ACTION
                                 U.227:  .LONG   <<SUBSTRING_COLON-U.227>-4>      ;
                  0000*  001D2  ;TPA$TARGET
                                 U.228:  .WORD   <<U.212-U.228>-2>                ;
                  9029   001D4  ;TPA$TYPE
                                 U.229:  .WORD   -28631                           ;
              00000000V  001D6  ;TPA$ACTION
                                 U.230:  .LONG   <<END_SUBSCRIPT-U.230>-4>        ;
                  0000*  001DA  ;TPA$TARGET
                                 U.232:  .WORD   <<U.231-U.232>-2>                ;
```

```
          15F6  001DC ;TPA$TYPE
                      U.233:    .WORD   5622                                            ;
         0000*  001DE ;TPA$TARGET
                      U.234:    .WORD   <<U.17-U.234>-2>                                ;
                001E0 ;SUB_LOOP3
                      U.224:    .BLKB   0
          99F8  001E0 ;TPA$TYPE
                      U.235:    .WORD   -26120                                          ;
         0000*  001E2 ;TPA$SUBEXP
                      U.236:    .WORD   <<U.3-U.236>-2>                                 ;
     00000000*  001E4 ;TPA$ACTION
                      U.237:    .LONG   <<NEXT_RECORD-U.237>-4>                         ;
         0000*  001E8 ;TPA$TARGET
                      U.238:    .WORD   <<U.224-U.238>-2>                               ;
          11F2  001EA ;TPA$TYPE
                      U.239:    .WORD   4594                                            ;
         0000*  001EC ;TPA$TARGET
                      U.240:    .WORD   <<U.?24-U.240>-2>                               ;
          99F8  001EE ;TPA$TYPE
                      U.241:    .WORD   -26120                                          ;
         0000*  001F0 ;TPA$SUBEXP
                      U.242:    .WORD   <<U.207-U.242>-2>                               ;
     00000000V  001F2 ;TPA$ACTION
                      U.243:    .LONG   <<STORE_SUBS-U.243>-4>                          ;
         0000*  001F6 ;TPA$TARGET
                      U.244:    .WORD   <<SUB_LOOP2-U.244>-2>                           ;
          15F6  001F8 ;TPA$TYPE
                      U.245:    .WORD   5622                                            ;
         0000*  001FA ;TPA$TARGET
                      U.246:    .WORD   <<U.215-U.246>-2>                               ;
                001FC ;START_SUBSTRING
                      U.231:    .BLKB   0
          8028  001FC ;TPA$TYPE
                      U.247:    .WORD   -32728                                          ;
     00000000V  001FE ;TPA$ACTION
                      U.248:    .LONG   <<INIT_SUBS-U.248>-4>                           ;
          15F6  00202 ;TPA$TYPE
                      U.249:    .WORD   5622                                            ;
          FFFF  00204 ;TPA$TARGET
                      U.250:    .WORD   -1                                              ;
                00206 LEFT_SUBSTRING:
                                .BLKB   0
          99F8  00206 ;TPA$TYPE
                      U.251:    .WORD   -26120                                          ;
         0000*  00208 ;TPA$SUBEXP
                      U.252:    .WORD   <<U.3-U.252>-2>                                 ;
     00000000*  0020A ;TPA$ACTION
                      U.253:    .LONG   <<NEXT_RECORD-U.253>-4>                         ;
         0000*  0020E ;TPA$TARGET
                      U.254:    .WORD   <<LEFT_SUBSTRING-U.254>-2>                      ;
          11F2  00210 ;TPA$TYPE
                      U.255:    .WORD   4594                                            ;
         0000*  00212 ;TPA$TARGET
                      U.256:    .WORD   <<LEFT_SUBSTRING-U.256>-2>                      ;
          99F8  00214 ;TPA$TYPE
                      U.257:    .WORD   -26120                                          ;
         0000*  00216 ;TPA$SUBEXP
```

```
                    U.258:   .WORD    <<U.207-U.258>-2>                          ;
00000000V 00218 ;TPA$ACTION
                    U.259:   .LONG    <<STORE_SUBS-U.259>-4>                     ;
    0000* 0021C ;TPA$TARGET
                    U.261:   .WORD    <<U.260-U.261>-2>                          ;
    903A  0021E ;TPA$TYPE
                    U.262:   .WORD    -28614                                     ;
00000000V 00220 ;TPA$ACTION
                    U.263:   .LONG    <<SUBSTRING_COLON-U.263>-4>                ;
    0000* 00224 ;TPA$TARGET
                    U.264:   .WORD    <<U.212-U.264>-2>                          ;
    15F6  00226 ;TPA$TYPE
                    U.265:   .WORD    5622                                       ;
    0000* 00228 ;TPA$TARGET
                    U.266:   .WORD    <<U.215-U.266>-2>                          ;
          0022A ;SUBSTRING2
                    U.260:   .BLKB    0
    99F8  0022A ;TPA$TYPE
                    U.267:   .WORD    -26120                                     ;
    0000* 0022C ;TPA$SUBEXP
                    U.268:   .WORD    <<U.3-U.268>-2>                            ;
00000000* 0022E ;TPA$ACTION
                    U.269:   .LONG    <<NEXT_RECORD-U.269>-4>                    ;
    0000* 00232 ;TPA$TARGET
                    U.270:   .WORD    <<U.260-U.270>-2>                          ;
    11F2  00234 ;TPA$TYPE
                    U.271:   .WORD    4594                                       ;
    0000* 00236 ;TPA$TARGET
                    U.272:   .WORD    <<U.260-U.272>-2>                          ;
    903A  00238 ;TPA$TYPE
                    U.273:   .WORD    -28614                                     ;
00000000V 0023A ;TPA$ACTION
                    U.274:   .LONG    <<SUBSTRING_COLON-U.274>-4>                ;
    0000* 0023E ;TPA$TARGET
                    U.275:   .WORD    <<U.212-U.275>-2>                          ;
    15F6  00240 ;TPA$TYPE
                    U.276:   .WORD    5622                                       ;
    0000* 00242 ;TPA$TARGET
                    U.277:   .WORD    <<U.17-U.277>-2>                           ;
          00244 ;RIGHT_SUBSTRING
                    U.212:   .BLKB    0
    99F8  00244 ;TPA$TYPE
                    U.278:   .WORD    -26120                                     ;
    0000* 00246 ;TPA$SUBEXP
                    U.279:   .WORD    <<U.3-U.279>-2>                            ;
00000000* 00248 ;TPA$ACTION
                    U.280:   .LONG    <<NEXT_RECORD-U.280>-4>                    ;
    0000* 0024C ;TPA$TARGET
                    U.281:   .WORD    <<U.212-U.281>-2>                          ;
    11F2  0024E ;TPA$TYPE
                    U.282:   .WORD    4594                                       ;
    0000* 00250 ;TPA$TARGET
                    U.283:   .WORD    <<U.212-U.283>-2>                          ;
    99F8  00252 ;TPA$TYPE
                    U.284:   .WORD    -26120                                     ;
    0000* 00254 ;TPA$SUBEXP
                    U.285:   .WORD    <<U.207-U.285>-2>                          ;
```

```
00000000V 00256 ;TPA$ACTION
                U.286:  .LONG   <<STORE_SUBS-U.286>-4>                                        ;
    0000* 0025A ;TPA$TARGET
                U.288:  .WORD   <<U.287-U.288>-2>                                             ;
     9029 0025C ;TPA$TYPE
                U.289:  .WORD   -28631                                                        ;
00000000V 0025E ;TPA$ACTION
                U.290:  .LONG   <<END_SUBSTRING-U.290>-4>                                     ;
     FFFF 00262 ;TPA$TARGET
                U.291:  .WORD   -1                                                            ;
     15F6 00264 ;TPA$TYPE
                U.292:  .WORD   5622                                                          ;
    0000* 00266 ;TPA$TARGET
                U.293:  .WORD   <<U.215-U.293>-2>                                             ;
          00268 ;SUBSTRING3
                U.287:  .BLKB   0
     99F8 00268 ;TPA$TYPE
                U.294:  .WORD   -26120                                                        ;
    0000* 0026A ;TPA$SUBEXP
                U.295:  .WORD   <<U.3-U.295>-2>                                               ;
00000000* 0026C ;TPA$ACTION
                U.296:  .LONG   <<NEXT_RECORD-U.296>-4>                                       ;
    0000* 00270 ;TPA$TARGET
                U.297:  .WORD   <<U.287-U.297>-2>                                             ;
     11F2 00272 ;TPA$TYPE
                U.298:  .WORD   4594                                                          ;
    0000* 00274 ;TPA$TARGET
                U.299:  .WORD   <<U.287-U.299>-2>                                             ;
     9029 00276 ;TPA$TYPE
                U.300:  .WORD   -28631                                                        ;
00000000V 00278 ;TPA$ACTION
                U.301:  .LONG   <<END_SUBSTRING-U.301>-4>                                     ;
     FFFF 0027C ;TPA$TARGET
                U.302:  .WORD   -1                                                            ;
     15F6 0027E ;TPA$TYPE
                U.303:  .WORD   5622                                                          ;
    0000* 00280 ;TPA$TARGET
                U.304:  .WORD   <<U.17-U.304>-2>                                              ;
          00282 ;IDENTIFIER
                U.188:  .BLKB   0
     85EE 00282 ;TPA$TYPE
                U.305:  .WORD   -31250                                                        ;
00000000V 00284 ;TPA$ACTION
                U.306:  .LONG   <<BLANKS_ON-U.306>-4>                                         ;
     91F1 00288 ;TPA$TYPE
                U.307:  .WORD   -28175                                                        ;
00000000V 0028A ;TPA$ACTION
                U.308:  .LONG   <<BLANKS_OFF-U.308>-4>                                        ;
     FFFF 0028E ;TPA$TARGET
                U.309:  .WORD   -1                                                            ;
     95F6 00290 ;TPA$TYPE
                U.310:  .WORD   -27146                                                        ;
00000000V 00292 ;TPA$ACTION
                U.311:  .LONG   <<BLANKS_OFF-U.311>-4>                                        ;
     FFFF 00296 ;TPA$TARGET
                U.312:  .WORD   -1                                                            ;
          00298 ;DECIMAL_INTEGER
```

```
                                U.207:  .BLKB   0
           9DF8  00298 ;TPA$TYPE
                                U.313:  .WORD   -25096                                      ;
          0000*  0029A ;TPA$SUBEXP
                                U.315:  .WORD   <<U.314-U.315>-2>                           ;
      00000000V  0029C ;TPA$ACTION
                                U.316:  .LONG   <<CONVERT_INTEGER-U.316>-4>                 ;
           FFFF  002A0 ;TPA$TARGET
                                U.317:  .WORD   -1                                          ;
                 002A2 ;INTEGER
                                U.314:  .BLKB   0
           802B  002A2 ;TPA$TYPE
                                U.318:  .WORD   -32725                                      ;
      00000000V  002A4 ;TPA$ACTION
                                U.319:  .LONG   <<BLANKS_ON-U.319>-4>                       ;
           802D  002A8 ;TPA$TYPE
                                U.320:  .WORD   -32723                                      ;
      00000000V  002AA ;TPA$ACTION
                                U.321:  .LONG   <<BLANKS_ON-U.321>-4>                       ;
           85F6  002AE ;TPA$TYPE
                                U.322:  .WORD   -31242                                      ;
      00000000V  002B0 ;TPA$ACTION
                                U.323:  .LONG   <<BLANKS_ON-U.323>-4>                       ;
           91F3  002B4 ;TPA$TYPE
                                U.324:  .WORD   -28173                                      ;
      00000000V  002B6 ;TPA$ACTION
                                U.325:  .LONG   <<BLANKS_OFF-U.325>-4>                      ;
           FFFF  002BA ;TPA$TARGET
                                U.326:  .WORD   -1                                          ;
           95F6  002BC ;TPA$TYPE
                                U.327:  .WORD   -27146                                      ;
      00000000V  002BE ;TPA$ACTION
                                U.328:  .LONG   <<BLANKS_OFF-U.328>-4>                      ;
           FFFE  002C2 ;TPA$TARGET
                                U.329:  .WORD   -2                                          ;
                 002C4 ;REAL
                                U.170:  .BLKB   0
           802B  002C4 ;TPA$TYPE
                                U.330:  .WORD   -32725                                      ;
      00000000V  002C6 ;TPA$ACTION
                                U.331:  .LONG   <<BLANKS_ON-U.331>-4>                       ;
           802D  002CA ;TPA$TYPE
                                U.332:  .WORD   -32723                                      ;
      00000000V  002CC ;TPA$ACTION
                                U.333:  .LONG   <<BLANKS_ON-U.333>-4>                       ;
           85F6  002D0 ;TPA$TYPE
                                U.334:  .WORD   -31242                                      ;
      00000000V  002D2 ;TPA$ACTION
                                U.335:  .LONG   <<BLANKS_ON-U.335>-4>                       ;
                 002D6 REAL1:   .BLKB   0
           11EF  002D6 ;TPA$TYPE
                                U.336:  .WORD   4591                                        ;
          0000*  002D8 ;TPA$TARGET
                                U.337:  .WORD   <<REAL1-U.337>-2>                           ;
           002E  002DA ;TPA$TYPE
                                U.338:  .WORD   46                                          ;
           05F6  002DC ;TPA$TYPE
```

```
                 U.339:  .WORD    1526                      ;
          002DE REAL2:   .BLKB    0
   11EF   002DE ;TPA$TYPE
                 U.340:  .WORD    4591                      ;
   0000*  002E0 ;TPA$TARGET
                 U.341:  .WORD    <<REAL2-U.341>-2>         ;
   05F6   002E2 ;TPA$TYPE
                 U.342:  .WORD    1526                      ;
          002E4 EXPONENT:
                         .BLKB    0
   0045   002E4 ;TPA$TYPE
                 U.343:  .WORD    69                        ;
   0065   002E6 ;TPA$TYPE
                 U.344:  .WORD    101                       ;
   0044   002E8 ;TPA$TYPE
                 U.345:  .WORD    68                        ;
   0064   002EA ;TPA$TYPE
                 U.346:  .WORD    100                       ;
   0051   002EC ;TPA$TYPE
                 U.347:  .WORD    81                        ;
   0071   002EE ;TPA$TYPE
                 U.348:  .WORD    113                       ;
   05F6   002F0 ;TPA$TYPE
                 U.349:  .WORD    1526                      ;
   002B   002F2 ;TPA$TYPE
                 U.350:  .WORD    43                        ;
   002D   002F4 ;TPA$TYPE
                 U.351:  .WORD    45                        ;
   05F6   002F6 ;TPA$TYPE
                 U.352:  .WORD    1526                      ;
          002F8 EXPONENT2:
                         .BLKB    0
   11EF   002F8 ;TPA$TYPE
                 U.353:  .WORD    4591                      ;
   0000*  002FA ;TPA$TARGET
                 U.354:  .WORD    <<EXPONENT2-U.354>-2>     ;
   05F6   002FC ;TPA$TYPE
                 U.355:  .WORD    1526                      ;
   19F8   002FE ;TPA$TYPE
                 U.356:  .WORD    6648                      ;
   0000*  00300 ;TPA$SUBEXP
                 U.357:  .WORD    <<U.158-U.357>-2>         ;
   FFFE   00302 ;TPA$TARGET
                 U.358:  .WORD    -2                        ;
   15F6   00304 ;TPA$TYPE
                 U.359:  .WORD    5622                      ;
   FFFF   00306 ;TPA$TARGET
                 U.360:  .WORD    -1                        ;
          00308 ;NOT_DELIM
                 U.158:  .BLKB    0
   19F8   00308 ;TPA$TYPE
                 U.361:  .WORD    6648                      ;
   0000*  0030A ;TPA$SUBEXP
                 U.362:  .WORD    <<U.3-U.362>-2>           ;
   FFFE   0030C ;TPA$TARGET
                 U.363:  .WORD    -2                        ;
   11F2   0030E ;TPA$TYPE
```

```
                                   U.364:   .WORD    4594                                           ;
                      FFFE  00310 ;TPA$TARGET
                                   U.365:   .WORD    -2                                             ;
                      102C  00312 ;TPA$TYPE
                                   U.366:   .WORD    4140                                           ;
                      FFFE  00314 ;TPA$TARGET
                                   U.367:   .WORD    -2                                             ;
                      1024  00316 ;TPA$TYPE
                                   U.368:   .WORD    4132                                           ;
                      FFFE  00318 ;TPA$TARGET
                                   U.369:   .WORD    -2                                             ;
                      1026  0031A ;TPA$TYPE
                                   U.370:   .WORD    4134                                           ;
                      FFFE  0031C ;TPA$TARGET
                                   U.371:   .WORD    -2                                             ;
                      1029  0031E ;TPA$TYPE
                                   U.372:   .WORD    4137                                           ;
                      FFFE  00320 ;TPA$TARGET
                                   U.373:   .WOPD    -2                                             ;
                      15F6  00322 ;TPA$TYPE
                                   U.374:   .WORD    5622                                           ;
                      FFFF  00324 ;TPA$TARGET
                                   U.375:   .WORD    -1                                             ;
                            00326 ;LOGICAL
                                   U.165:   .BLKB    0
                      802E  00326 ;TPA$TYPE
                                   U.376:   .WORD    -32722                                         ;
                 00000000V  00328 ;TPA$ACTION
                                   U.377:   .LONG    <<BLANKS_ON-U.377>-4>                          ;
                      85F6  0032C ;TPA$TYPE
                                   U.378:   .WORD    -31242                                         ;
                 00000000V  0032E ;TPA$ACTION
                                   U.379:   .LONG    <<BLANKS_ON-U.379>-4>                          ;
                      0054  00332 ;TPA$TYPE
                                   U.380:   .WORD    84                                             ;
                      0074  00334 ;TPA$TYPE
                                   U.381:   .WORD    116                                            ;
                      0046  00336 ;TPA$TYPE
                                   U.382:   .WORD    70                                             ;
                      0466  00338 ;TPA$TYPE
                                   U.383:   .WORD    1126                                           ;
                            0033A LOGICAL1:
                                            .BLKB    0
                      19F8  0033A ;TPA$TYPE
                                   U.384:   .WORD    6648                                           ;
                      0000* 0033C ;TPA$SUBEXP
                                   U.386:   .WORD    <<U.385-U.386>-2>                              ;
                      0000* 0033E ;TPA$TARGET
                                   U.387:   .WORD    <<LOGICAL1-U.387>-2>                           ;
                      95F6  00340 ;TPA$TYPE
                                   U.388:   .WORD    -27146                                         ;
                 00000000V  00342 ;TPA$ACTION
                                   U.389:   .LONG    <<BLANKS_OFF-U.389>-4>                         ;
                      FFFF  00346 ;TPA$TARGET
                                   U.390:   .WORD    -1                                             ;
                            00348 ;LOGICAL2
                                   U.385:   .BLKB    0
```

```
           19F8  00348 ;TPA$TYPE
                       U.391:  .WORD   6648                                       ;
          0000*  0034A ;TPA$SUBEXP
                       U.392:  .WORD   <<U.3-U.392>-2>                            ;
           FFFE  0034C ;TPA$TARGET
                       U.393:  .WORD   -2                                         ;
           11F2  0034E ;TPA$TYPE
                       U.394:  .WORD   4594                                       ;
           FFFE  00350 ;TPA$TARGET
                       U.395:  .WORD   -2                                         ;
           102C  00352 ;TPA$TYPE
                       U.396:  .WORD   4140                                       ;
           FFFE  00354 ;TPA$TARGET
                       U.397:  .WORD   -2                                         ;
           1028  00356 ;TPA$TYPE
                       U.398:  .WORD   4136                                       ;
           FFFE  00358 ;TPA$TARGET
                       U.399:  .WORD   -2                                         ;
           103D  0035A ;TPA$TYPE
                       U.400:  .WORD   4157                                       ;
           FFFE  0035C ;TPA$TARGET
                       U.401:  .WORD   -2                                         ;
           1024  0035E ;TPA$TYPE
                       U.402:  .WORD   4132                                       ;
           FFFE  00360 ;TPA$TARGET
                       U.403:  .WORD   -2                                         ;
           1026  00362 ;TPA$TYPE
                       U.404:  .WORD   4134                                       ;
           FFFE  00364 ;TPA$TARGET
                       U.405:  .WORD   -2                                         ;
           15ED  00366 ;TPA$TYPE
                       U.406:  .WORD   5613                                       ;
           FFFF  00368 ;TPA$TARGET
                       U.407:  .WORD   -1                                         ;
                 0036A ;COMPLEX
                       U.175:  .BLKB   0
           1428  0036A ;TPA$TYPE
                       U.408:  .WORD   5160                                       ;
          0000*  0036C ;TPA$TARGET
                       U.410:  .WORD   <<U.409-U.410>-2>                          ;
                 0036E ;COMPLEX2
                       U.409:  .BLKB   0
           99F8  0036E ;TPA$TYPE
                       U.411:  .WORD   -26120                                     ;
          0000*  00370 ;TPA$SUBEXP
                       U.412:  .WORD   <<U.3-U.412>-2>                            ;
      00000000*  00372 ;TPA$ACTION
                       U.413:  .LONG   <<NEXT_RECORD-U.413>-4>                    ;
          0000*  00376 ;TPA$TARGET
                       U.414:  .WORD   <<U.409-U.414>-2>                          ;
           11F2  00378 ;TPA$TYPE
                       U.415:  .WORD   4594                                       ;
          0000*  0037A ;TPA$TARGET
                       U.416:  .WORD   <<U.409-U.416>-2>                          ;
           99F8  0037C ;TPA$TYPE
                       U.417:  .WORD   -26120                                     ;
          0000*  0037E ;TPA$SUBEXP
```

```
                            U.418:   .WORD    <<U.170-U.418>-2>                     ;
        00000000V 00380 ;TPA$ACTION
                            U.419:   .LONG    <<STORE_COMPLEX-U.419>-4>             ;
            0000* 00384 ;TPA$TARGET
                            U.421:   .WORD    <<U.420-U.421>-2>                     ;
            15F6  00386 ;TPA$TYPE
                            U.422:   .WORD    5622                                  ;
            0000* 00388 ;TPA$TARGET
                            U.423:   .WORD    <<U.17-U.423>-2>                      ;
                  0038A ;COMPLEX3
                            U.420:   .BLKB    0
            99F8  0038A ;TPA$TYPE
                            U.424:   .WORD    -26120                                ;
            0000* 0038C ;TPA$SUBEXP
                            U.425:   .WORD    <<U.3-U.425>-2>                       ;
        00000000* 0038E ;TPA$ACTION
                            U.426:   .LONG    <<NEXT_RECORD-U.426>-4>               ;
            0000* 00392 ;TPA$TARGET
                            U.427:   .WORD    <<U.420-U.427>-2>                     ;
            11F2  00394 ;TPA$TYPE
                            U.428:   .WORD    4594                                  ;
            0000* 00396 ;TPA$TARGET
                            U.429:   .WORD    <<U.420-U.429>-2>                     ;
            102C  00398 ;TPA$TYPE
                            U.430:   .WORD    4140                                  ;
            0000* 0039A ;TPA$TARGET
                            U.432:   .WORD    <<U.431-U.432>-2>                     ;
            15F6  0039C ;TPA$TYPE
                            U.433:   .WORD    5622                                  ;
            0000* 0039E ;TPA$TARGET
                            U.434:   .WORD    <<U.17-U.434>-2>                      ;
                  003A0 ;COMPLEX4
                            U.431:   .BLKB    0
            99F8  003A0 ;TPA$TYPE
                            U.435:   .WORD    -26120                                ;
            0000* 003A2 ;TPA$SUBEXP
                            U.436:   .WORD    <<U.3-U.436>-2>                       ;
        00000000* 003A4 ;TPA$ACTION
                            U.437:   .LONG    <<NEXT_RECORD-U.437>-4>               ;
            0000* 003A8 ;TPA$TARGET
                            U.438:   .WORD    <<U.431-U.438>-2>                     ;
            11F2  003AA ;TPA$TYPE
                            U.439:   .WORD    4594                                  .
            0000* 003AC ;TPA$TARGET
                            U.440:   .WORD    <<U.431-U.440>-2>                     ;
            99F8  003AE ;TPA$TYPE
                            U.441:   .WORD    -26120                                ;
            0000* 003B0 ;TPA$SUBEXP
                            U.442:   .WORD    <<U.170-U.442>-2>                     ;
        00000000V 003B2 ;TPA$ACTION
                            U.443:   .LONG    <<STORE_COMPLEX-U.443>-4>             ;
            0000* 003B6 ;TPA$TARGET
                            U.445:   .WORD    <<U.444-U.445>-2>                     ;
            15F6  003B8 ;TPA$TYPE
                            U.446:   .WORD    5622                                  ;
            0000* 003BA ;TPA$TARGET
                            U.447:   .WORD    <<U.17-U.447>-2>                      ;
```

```
                     003BC ;COMPLEX5
                           U.444:   .BLKB    0
           99F8      003BC ;TPA$TYPE
                           U.448:   .WORD    -26120                                  ;
           0000*     003BE ;TPA$SUBEXP
                           U.449:   .WORD    <<U.3-U.449>-2>                         ;
       00000000*     003C0 ;TPA$ACTION
                           U.450:   .LONG    <<NEXT_RECORD-U.450>-4>                 ;
           0000*     003C4 ;TPA$TARGET
                           U.451:   .WORD    <<U.444-U.451>-2>                       ;
           11F2      003C6 ;TPA$TYPE
                           U.452:   .WORD    4594                                    ;
           0000*     003C8 ;TPA$TARGET
                           U.453:   .WORD    <<U.444-U.453>-2>                       ;
           1029      003CA ;TPA$TYPE
                           U.454:   .WORD    4137                                    ;
           FFFF      003CC ;TPA$TARGET
                           U.455:   .WORD    -1                                      ;
           15F6      003CE ;TPA$TYPE
                           U.456:   .WORD    5622                                    ;
           0000*     003D0 ;TPA$TARGET
                           U.457:   .WORD    <<U.17-U.457>-2>                        ;
                     003D2 ;CHARACTER
                           U.179:   .BLKB    0
           9427      003D2 ;TPA$TYPE
                           U.458:   .WORD    -27609                                  ;
       00000000V     003D4 ;TPA$ACTION
                           U.459:   .LONG    <<STRING_OK-U.459>-4>                   ;
           0000*     003D8 ;TPA$TARGET
                           U.461:   .WORD    <<U.460-U.461>-2>                       ;
                     003DA ;CHARACTER1
                           U.460:   .BLKB    0
           91F7      003DA ;TPA$TYPE
                           U.462:   .WORD    -28169                                  ;
       00000000*     003DC ;TPA$ACTION
                           U.463:   .LONG    <<NEXT_RECORD-U.463>-4>                 ;
           0000*     003E0 ;TPA$TARGET
                           U.464:   .WORD    <<U.460-U.464>-2>                       ;
           1027      003E2 ;TPA$TYPE
                           U.465:   .WORD    4135                                    ;
           0000*     003E4 ;TPA$TARGET
                           U.467:   .WORD    <<U.466-U.467>-2>                       ;
           95ED      003E6 ;TPA$TYPE
                           U.468:   .WORD    -27155                                  ;
       00000000V     003E8 ;TPA$ACTION
                           U.469:   .LONG    <<STORE_CHARACTER-U.469>-4>             ;
           0000*     003EC ;TPA$TARGET
                           U.470:   .WORD    <<U.460-U.470>-2>                       ;
                     003EE ;NEXT_QUOTE
                           U.466:   .BLKB    0
           91F7      003EE ;TPA$TYPE
                           U.471:   .WORD    -28169                                  ;
       00000000*     003F0 ;TPA$ACTION
                           U.472:   .LONG    <<NEXT_RECORD-U.472>-4>                 ;
           0000*     003F4 ;TPA$TARGET
                           U.473:   .WORD    <<U.466-U.473>-2>                       ;
           9027      003F6 ;TPA$TYPE
```

```
                                        U.474:  .WORD   -28633                                   ;
                    00000000V 003F8 ;TPA$ACTION
                                        U.475:  .LONG   <<STORE_CHARACTER-U.475>-4>              ;
                        0000* 003FC ;TPA$TARGET
                                        U.476:  .WORD   <<U.460-U.476>-2>                        ;
                        15F6  003FE ;TPA$TYPE
                                        U.477:  .WORD   5622                                     ;
                        FFFF  00400 ;TPA$TARGET
                                        U.478:  .WORD   -1                                       ;
                              00402 ;ERROR_STATE
                                        U.17:   .BLKB   0
                        91ED  00402 ;TPA$TYPE
                                        U.479:  .WORD   -28179                                   ;
                    00000000V 00404 ;TPA$ACTION
                                        U.480:  .LONG   <<SYNTAX_ERROR-U.480>-4>                 ;
                        FFFE  00408 ;TPA$TARGET
                                        U.481:  .WORD   -2                                       ;
                        95F6  0040A ;TPA$TYPE
                                        U.482:  .WORD   -27146                                   ;
                    00000000V 0040C ;TPA$ACTION
                                        U.483:  .LONG   <<SYNTAX_ERROR-U.483>-4>                 ;
                        FFFE  00410 ;TPA$TARGET
                                        U.484:  .WORD   -2                                       ;
                              00412 ;INVREFVAR_STATE
                                        U.215:  .BLKB   0
                        95F6  00412 ;TPA$TYPE
                                        U.485:  .WORD   -27146                                   ;
                    00000000V 00414 ;TPA$ACTION
                                        U.486:  .LONG   <<INVREFVAR_ERROR-U.486>-4>              ;
                        FFFE  00418 ;TPA$TARGET
                                        U.487:  .WORD   -2                                       ;

                                                .PSECT  _LIB$KEY0$,NOWRT,  SHR,  PIC,1

                              00000 FOR$$A_NMLKEYWD::
                                                .BLKB   0
                              00000 ;TPA$KEY0
                                        U.1:    .BLKB   0

                                                .EXTRN  FOR$$CVT_TYPE, FOR$$DO_NML_OUTPUT
                                                .EXTRN  FOR$$REC_RSNO, FOR$$REC_WSNO
                                                .EXTRN  FOR$$SIGNAL, FOR$$SIGNAL_STO
                                                .EXTRN  OTS$CVT_TI_L, OTS$CVT_TL_L
                                                .EXTRN  OTS$CVT_T_F, OTS$CVT_T_D
                                                .EXTRN  OTS$CVT_T_G, OTS$CVT_T_H
                                                .EXTRN  LIB$SIG_TO_RET

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

                         083C 00000 NEXT_RECORD:
                                                .WORD   Save R2,R3,R4,R5,R11                     ; 0752
              5B        40    AC   D0 00002     MOVL    64(AP), CCB                              ; 0798
                00000000G    00   16 00006 1$:  JSB     FOR$$REC_RSNO                            ; 0801
                        B0    AB   D6 0000C     INCL    -80(CCB)                                 ; 0802
             0C    AC   B0    AB   D0 0000F     MOVL    -80(CCB), 12(AP)                         ; 0803
  08   AC    B4    AB   B0    AB   C3 00014     SUBL3   -80(CCB), -76(CCB), 8(AP)                ; 0804
                        E9    15      0001B     BLEQ    1$                                       ; 0806
```

```
                                    50          01  D0 0001D          MOVL    #1, R0                                      ; 0807
                                                    04 00020          RET                                                 ; 0809

; Routine Size:  33 bytes,    Routine Base:  _FOR$CODE + 0000
```

```
1 %SBTTL 'INIT_SUBS - Start a subscript/substring '
1 ROUTINE INIT_SUBS =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which initiates the evaluation of a subscript
1 !       or substring.  If the current variable can not have a subscript or
1 !       a substring, an error routine is called.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = INIT_SUBS ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       PARAM_BLOCK [NML$L_CURIDX] = 0
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Can call INVREFVAR_ERROR
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     LOCAL
2         DESCR: REF BLOCK [, BYTE];
2
2     DESCR = .AP [NML$A_DESCR];  ! Get descriptor address
2
2     !+
2     ! If this variable is not an array or a string, signal FOR$_INVREFVAR
2     !-
2
3     IF ((.DESCR [DSC$B_CLASS] EQL DSC$K_CLASS_A) OR
3             (.DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_T))
2     THEN
2         AP [NML$L_CURIDX] = 0   ! Set up for start of subscript/substring
```

```
;  807          0867  2     ELSE
;  808          0868  2         CALLG (.AP, INVREFVAR_ERROR);
;  809          0869  2
;  810          0870  2     RETURN 1;
;  811          0871  2
;  812          0872  1     END;



                                                                  0000 00000 INIT_SUBS:
                                                                                        .WORD   Save nothing                                     ; 0811
                                               50        3C    AC  D0 00002             MOVL    60(AP), DESCR                                    ; 0857
                                               04        03    A0  91 00006             CMPB    3(DESCR), #4                                     ; 0863
                                                               06  13 0000A             BEQL    1$                                               ;
                                               0E        02    A0  91 0000C             CMPB    2(DESCR), #14                                    ; 0864
                                                               05  12 00010             BNEQ    2$                                               ;
                                                         48    AC  D4 00012 1$:         CLRL    72(AP)                                           ; 0866
                                                               05  11 00015             BRB     3$                                               ;
                                       0000V   CF              6C  FA 00017 2$:         CALLG   (AP), INVREFVAR_ERROR                            ; 0868
                                               50              01  D0 0001C 3$:         MOVL    #1, R0                                           ; 0870
                                                                   04 0001F             RET                                                      ; 0872

; Routine Size:  32 bytes,    Routine Base:  _FOR$CODE + 0021


;  813          0873  1 !<BLF/PAGE>
```

FO
1-

;

;

```
1 %SBTTL 'SUBSTRING_COLON - Mark presence of colon in substring'
1 ROUTINE SUBSTRING_COLON =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which is called when a colon is found in
1 !       a substring.  If no left part has been found, it sets the left part
1 !       to 1 indicating that the low column was omitted.  If the current
1 !       variable is not of type CHARACTER, then an error routine is called.
1 !       If the variable is an array, a subscript must have been previously
1 !       processed, otherwise an error is given.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = SUBSTRING_COLON ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       If NML$L_CURIDX = 0 then NML$L_CURIDX = 1 and NML$L_SUBSCR[0] = 1
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2    BEGIN
2
2    LOCAL
2       DESCR: REF BLOCK [, BYTE];      ! Address of variable descriptor
2
2    BUILTIN
2       AP;             ! Argument pointer points to parameter block
2
2    MAP
2       AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2    IF .AP [NML$B_DTYPE] NEQ DSC$K_DTYPE_T
2    THEN
2       CALLG (.AP, INVREFVAR_ERROR);   ! Substring not allowed with non-CHARACTER
2
2    !+
2    ! If this variable is an array, then a subscript must have been previously
2    ! seen for a substring to be valid.
```

```
;  872          0931  2      !-
;  873          0932  2
;  874          0933  2      DESCR = .AP [NML$A_DESCR];
;  875          0934  2      IF .DESCR [DSC$B_CLASS] EQL DSC$K_CLASS_A AND NOT .AP [NML$V_SUBSCRIPT]
;  876          0935  2      THEN
;  877          0936  2          CALLG (.AP, INVREFVAR_ERROR);   ! Substring not allowed with unsubscripted array
;  878          0937  2
;  879          0938  2      IF .AP [NML$L_CURIDX] EQL 0 ! Substring of the form (:n)?
;  880          0939  2      THEN
;  881          0940  3          BEGIN
;  882          0941  3          AP [NML$L_CURIDX] = 1;  ! Left bound is first character
;  883          0942  3          AP [NML$L_SUBSCR] = 1;
;  884          0943  2          END;
;  885          0944  2
;  886          0945  2      AP [NML$V_SUBSTRING] = 1;   ! Indicate substring
;  887          0946  2
;  868          0947  2      RETURN 1;
;  889          0948  2
;  890          0949  1      END;



                                             0000 00000 SUBSTRING_COLON:
                                                                .WORD   Save nothing                             ; 0875
                      0E        44    AC   91 00002             CMPB    68(AP), #14                             ; 0924
                                      05   13 00006             BEQL    1$                                       ;
                 0000V  CF            6C   FA 00008             CALLG   (AP), INVREFVAR_ERROR                    ; 0926
                      50        3C    AC   D0 0000D 1$:         MOVL    60(AP), DESCR                            ; 0933
                      04        03    A0   91 00011             CMPB    3(DESCR), #4                             ; 0934
                                      0A   12 00015             BNEQ    2$                                       ;
          05        45  AC            03   E0 00017             BBS     #3, 69(AP), 2$                           ;
                 0000V  CF            6C   FA 0001C             CALLG   (AP), INVREFVAR_ERROR                    ; 0936
                                48    AC   D5 00021 2$:         TSTL    72(AP)                                   ; 0938
                                      08   12 00024             BNEQ    3$                                       ;
                    48  AC            01   D0 00026             MOVL    #1, 72(AP)                               ; 0941
                    4C  AC            01   D0 0002A             MOVL    #1, 76(AP)                               ; 0942
                    45  AC            01   88 0002E 3$:         BISB2   #1, 69(AP)                               ; 0945
                        50            01   D0 00032             MOVL    #1, R0                                   ; 0947
                                           04 00035             RET                                              ; 0949

; Routine Size:  54 bytes,    Routine Base:  _FOR$CODE + 0041

;  891          0950  1 !<BLF/PAGE>
```

```
1 %SBTTL 'STORE_SUBS - Store a subscript or substring'
1 ROUTINE STORE_SUBS =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which stores the value of a subscript or
1 !       substring column.  It also checks to see if the allowed number of
1 !       subscripts have not been exceeded.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = STORE_SUBS ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       PARAM_BLOCK [NML$L_CURIDX] is incremented by 1
1 !       The value of the subscript is stored in the current subscript vector
1 !       location.
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May call SYNTAX_ERROR
1 !       May call INVREFVAR_ERROR
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;               ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     LOCAL
2         SUBSCRIPTS: REF VECTOR [, LONG],
2         DESCR: REF BLOCK [, BYTE];
2
2     SUBSCRIPTS = AP [NML$L_SUBSCR];     ! Address of subscript vector
2
2     IF .AP [NML$V_SUBSTRING]
2     THEN
3         BEGIN
3         IF .AP [NML$L_CURIDX] GTR 1     ! Only two substring values allowed!
```

```
;  950          1008  3             THEN
;  951          1009  3                 CALLG (.AP, SYNTAX_ERROR);
;  952          1010  3             IF .AP [TPA$L_NUMBER] LEQ 0      ! Substring column can't be LEQ 0
;  953          1011  3             THEN
;  954          1012  3                 CALLG (.AP, INVREFVAR_ERROR);
;  955          1013  3             END
;  956          1014  2         ELSE
;  957          1015  3             BEGIN
;  958          1016  3             DESCR = .AP [NML$A_DESCR];       ! Get descriptor address
;  959          1017  3             IF .DESCR [DSC$B_CLASS] EQL DSC$K_CLASS_A
;  960          1018  3             THEN
;  961          1019  3                 IF .AP [NML$L_CURIDX] GEQ .DESCR [DSC$B_DIMCT]
;  962          1020  3                 THEN
;  963          1021  3                     CALLG (.AP, INVREFVAR_ERROR);   ! Too many subscripts
;  964          1022  2             END;
;  965          1023  2
;  966          1024  2         SUBSCRIPTS [.AP [NML$L_CURIDX]] = .AP [TPA$L_NUMBER];        ! Store subscript
;  967          1025  2         AP [NML$L_CURIDX] = .AP [NML$L_CURIDX] + 1;
;  968          1026  2
;  969          1027  2         RETURN 1;
;  970          1028  2
;  971          1029  1         END;



                                                  0004 00000 STORE_SUBS:
                                                                     .WORD   Save R2                                        ; 0952
                                52       4C   AC   9E 00002          MOVAB   76(AP), SUBSCRIPTS                             ; 1002
                                10       45   AC   E9 00006          BLBC    69(AP), 2$                                     ; 1004
                                01       48   AC   D1 0000A          CMPL    72(AP), #1                                     ; 1007
                                               05  15 0000E          BLEQ    1$                                             ;
                         0000V  CF             6C  FA 00010          CALLG   (AP), SYNTAX_ERROR                             ; 1009
                                         1C   AC   D5 00015 1$:      TSTL    28(AP)                                         ; 1010
                                               11  11 00018          BRB     3$                                             ;
                                50       3C   AC   D0 0001A 2$:      MOVL    60(AP), DESCR                                  ; 1016
                                04       03   A0   91 0001E          CMPB    3(DESCR), #4                                   ; 1017
                                               0E  12 00022          BNEQ    4$                                             ;
      48   AC        0B   A0              08       00  ED 00024          CMPZV   #0, #8, 11(DESCR), 72(AP)                      ; 1019
                                               05  14 0002B 3$:      BGTR    4$                                             ;
                         0000V  CF             6C  FA 0002D          CALLG   (AP), INVREFVAR_ERROR                          ; 1021
                                50       48   AC   D0 00032 4$:      MOVL    72(AP), R0                                     ; 1024
                              6240       1C   AC   D0 00036          MOVL    28(AP), (SUBSCRIPTS)[R0]                       ;
                                         48   AC   D6 0003B          INCL    72(AP)                                         ; 1025
                                50            01   D0 0003E          MOVL    #1, R0                                         ; 1027
                                                   04 00041          RET                                                    ; 1029

; Routine Size:  66 bytes,    Routine Base:  _FOR$CODE + 0077


;  972          1030  1 !<BLF/PAGE>
```

```
1 %SBTTL 'END_SUBSCRIPT - End an array subscript'
1 ROUTINE END_SUBSCRIPT =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which is called at the end of an array subscript.
1 !       It calls COMPUTE_INDEX to calculate the starting position in the array.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = END_SUBSCRIPT ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       See COMPUTE_INDEX
1 !       NML$V_SUBSCRIPT = 1, to indicate subscript processed.
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals FOR$_INVREFVAR if a subscript is out of bounds.
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
3     IF NOT (CALLG (.AP, COMPUTE_INDEX))
2     THEN
2         CALLG (.AP, INVREFVAR_ERROR);
2
2     AP [NML$V_SUBSCRIPT] = 1;   ! Allows substring to follow for arrays
2
2     RETURN 1;
2
1     END;
```

```
                                        0000 00000 END_SUBSCRIPT:
                                                           .WORD   Save nothing                              ; 1032
                        0000V  CF            6C  FA 00002          CALLG   (AP), COMPUTE_INDEX                       ; 1075
                               05            50  E8 00007          BLBS    R0, 1$                                    ;
                        0000V  CF            6C  FA 0000A          CALLG   (AP), INVREFVAR_ERROR                     ; 1077
                          45   AC            08  88 0000F 1$:      BISB2   #8, 69(AP)                                ; 1079
                               50            01  D0 00013          MOVL    #1, R0                                    ; 1081
                                                 04 00016          RET                                               ; 1083

; Routine Size:  23 bytes,    Routine Base:  _FOR$CODE + 00B9


; 1027          1084  1 !<BLF/PAGE>
```

FO
1-

```
1 %SBTTL 'COMPUTE_INDEX - Compute the array index'
1 ROUTINE COMPUTE_INDEX =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Routine which computes the starting location for the current
1 !       variable based on the array subscripts seen.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = COMPUTE_INDEX ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       PARAM_BLOCK [NML$A_VARCUR] = Starting address
1 !       PARAM_BLOCK [NML$A_VARSTART] - Starting address
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !       SS$_SUBRNG for subscript out of range
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     LOCAL
2         DESCR: REF BLOCK [, BYTE],       ! Variable descriptor
2         MULTIPLIERS: REF VECTOR [,LONG],! Multiplier array
2         LAST_MULT,                       ! Previous bounds' multiplier
2         BOUNDS: REF VECTOR [,LONG],      ! Current bounds
2         SUBSCRIPT: REF VECTOR [,LONG],   ! Current subscript
2         DIMENSION,                       ! Current dimension
2         OFFSET;                          ! Offset into array
2
2     ENABLE
2         LIB$SIG_TO_RET; ! Return SS$_SUBRNG as a status
2
```

```
2      DESCR = .AP [NMLSA_DESCR];           ! Get descriptor address
2
2      !+
2      ! If the descriptor class is not ARRAY, then a subscript is illegal.
2      !-
2
2      IF .DESCR [DSC$B_CLASS] NEQ DSC$K_CLASS_A
2      THEN
2          RETURN 0;
2
2      !+
2      ! If the number of subscripts doesn't match the number of dimensions, then
2      ! it is an error.
2      !-
2
2      IF .DESCR [DSC$B_DIMCT] NEQ .AP [NML$L_CURIDX]
2      THEN
2          RETURN 0;       ! Number of subscripts don't match
2
2      DIMENSION = .AP [NML$L_CURIDX] - 1;
2      SUBSCRIPT = AP [NML$L_SUBSCR] + (4 * .DIMENSION);
2      MULTIPLIERS = DESCR [DSC$L_M1] + (4 * .DIMENSION) - 4;
2      LAST_MULT = .MULTIPLIERS [0];
2      BOUNDS = MULTIPLIERS [2] + (8 * .DIMENSION);
2      OFFSET = 0;
2
2      !+
2      ! For each dimension, from last to first, compute the offset into the
2      ! array.  If a subscript is out of bounds, the INDEX instruction will
2      ! signal an error.
2      !-
2
2      DECR DIM FROM .DIMENSION TO 0 DO
3          BEGIN
3          IF .DIM EQL 0
3          THEN
3              LAST_MULT = 1;
3          INDEX (SUBSCRIPT [0], BOUNDS [0], BOUNDS [1], LAST_MULT,
3                 OFFSET, OFFSET);
3          MULTIPLIERS = MULTIPLIERS [-1];
3          LAST_MULT = .MULTIPLIERS [0];
3          BOUNDS = BOUNDS [-2];
3          SUBSCRIPT = SUBSCRIPT [-1];
2          END;
2
2      AP [NML$A_VARCUR] = .DESCR [DSC$A_A0] + (.OFFSET * .AP [NML$W_STRIDE]);
2      AP [NML$A_VARSTART] = .AP [NML$A_VARCUR];
2
2      RETURN 1;
2
1      END;


                                        007C 00000 COMPUTE_INDEX:
```

```
                                                                 .WORD   Save R2,R3,R4,R5,R6                        ; 1086
                                  6D      0063   CF   DE 00002       MOVAL   5$, (FP)                                   ; 1122
                                  53        3C   AC   D0 00007       MOVL    60(AP), DESCR                              ; 1142
                                  04        03   A3   91 0000B       CMPB    3(DESCR), #4                               ; 1148
                                                 55   12 0000F       BNEQ    4$                                         ;
      48    AC        0B    A3          08       00   ED 00011       CMPZV   #0, #8, 11(DESCR), 72(AP)                  ; 1157
                                                 4C   12 00018       BNEQ    4$                                         ;
                            50        48  AC     01   C3 0001A       SUBL3   #1, 72(AP), DIMENSION                      ; 1161
                                          56  4C AC40 DE 0001F       MOVAL   76(AP)[DIMENSION], SUBSCRIPT               ; 1162
                                          54  10 A340 DE 00024       MOVAL   16(DESCR)[DIMENSION], MULTIPLIERS          ; 1163
                                          55         64 D0 00029     MOVL    (MULTIPLIERS), LAST_MULT                   ; 1164
                                          52  08 A440 7E 0002C       MOVAQ   8(MULTIPLIERS)[DIMENSION], BOUNDS          ; 1165
                                                 51   D4 00031       CLRL    OFFSET                                     ; 1166
                                                 50   D6 00033       INCL    DIM                                        ; 1174
                                                 16   11 00035       BRB     3$                                         ;
                                                 03   12 00037 1$:   BNEQ    2$                                         ; 1176
                                          55     01   D0 00039       MOVL    #1, LAST_MULT                              ; 1178
            55        04    A2            62     66   0A 0003C 2$:   INDEX   (SUBSCRIPT), (BOUNDS), 4(BOUNDS), -        ; 1179
                                          51     51      00042               LAST_MULT, OFFSET, OFFSET                  ;
                                          55     74   D0 00044       MOVL    -(MULTIPLIERS), LAST_MULT                  ; 1182
                                          52     08   C2 00047       SUBL2   #8, BOUNDS                                 ; 1183
                                          56     04   C2 0004A       SUBL2   #4, SUBSCRIPT                              ; 1184
                                          E7     50   F4 0004D 3$:   SOBGEQ  DIM, 1$                                    ; 1174
                                          50   3A  AC   3C 00050     MOVZWL  58(AP), R0                                 ; 1187
                                          51     50   C4 00054       MULL2   R0, R1                                     ;
                                  34      AC  10 B341 9E 00057       MOVAB   @16(DESCR)[R1], 52(AP)                     ;
                                  2C      AC   34  AC D0 0005D       MOVL    52(AP), 44(AP)                             ; 1188
                                          50     01   D0 00062       MOVL    #1, R0                                     ; 1190
                                                      04 00065       RET                                                ;
                                                 50   D4 00066 4$:   CLRL    R0                                         ; 1192
                                                      04 00068       RET                                                ;
                                                    0000 00069 5$:   .WORD   Save nothing                               ; 1122
                                                 7E   D4 0006B       CLRL    -(SP)                                      ;
                                                 5E   DD 0006D       PUSHL   SP                                         ;
                                          7E  04  AC  7D 0006F       MOVQ    4(AP), -(SP)                               ;
                           00000000G      00     03   FB 00073       CALLS   #3, LIB$SIG_TO_RET                         ;
                                                      04 0007A       RET                                                ;

; Routine Size:  123 bytes,    Routine Base:  _FOR$CODE + 00D0


; 1137          1193  1 !<BLF/PAGE>
```

```
1 %SBTTL 'END_SUBSTRING - End a substring'
1 ROUTINE END_SUBSTRING =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which evaluates a substring reference.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = END_SUBSTRING ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       PARAM_BLOCK [NML$A_VARCUR] - Set to starting point
1 !       PARAM_BLOCK [NML$W_VARSIZE] - Set to string size
1 !       PARAM_BLOCK [NML$A_VARSTART] - Set to starting point
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Can call INVREFVAR_ERROR if the substring is out-of-bounds.
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     IF .AP [NML$L_CURIDX] EQL 1
2     THEN
2         AP [NML$L_SUBSTRHI] = .AP [NML$W_VARSIZE]
2     ELSE IF .AP [NML$L_CURIDX] NEQ 2
2     THEN
2         CALLG (.AP, SYNTAX_ERROR);
2
2     IF .AP [NML$L_SUBSTRLO] LEQ 0 OR .AP [NML$L_SUBSTRHI] LSS .AP [NML$L_SUBSTRLO] OR
2         .AP [NML$L_SUBSTRHI] GTR .AP [NML$W_VARSIZE]
2     THEN
2         CALLG (.AP, INVREFVAR_ERROR);
2
2     AP [NML$A_VARCUR] = .AP [NML$A_VARCUR] + .AP [NML$L_SUBSTRLO] - 1;
```

```
; 1196          1251  2      AP [NML$A_VARSTART] = .AP [NML$A_VARCUR];
; 1197          1252  2      AP [NML$W_VARSIZE] = (.AP [NML$L_SUBSTRHI] - .AP [NML$L_SUBSTRLO]) + 1;
; 1198          1253  2
; 1199          1254  2      RETURN 1;
; 1200          1255  2
; 1201          1256  1      END;



                                                0000 00000 END_SUBSTRING:
                                                                   .WORD    Save nothing                            ; 1195
                                 01      48    AC  D1 00002        CMPL     72(AP), #1                              ; 1238
                                               07  12 00006        BNEQ     1$                                      ;
                            50   AC      38    AC  3C 00008        MOVZWL   56(AP), 80(AP)                          ; 1240
                                               0B  11 0000D        BRB      2$                                      ;
                                 02      48    AC  D1 0000F 1$:    CMPL     72(AP), #2                              ; 1241
                                               05  13 00013        BEQL     2$                                      ;
                         0000V   CF            6C  FA 00015        CALLG    (AP), SYNTAX_ERROR                      ; 1243
                                         4C    AC  D5 0001A 2$:    TSTL     76(AP)                                  ; 1245
                                               10  15 0001D        BLEQ     3$                                      ;
                            4C   AC      50    AC  D1 0001F        CMPL     80(AP), 76(AP)                          ;
                                               09  19 00024        BLSS     3$                                      ;
     50   AC     38   AC         10            00  ED 00026        CMPZV    #0, #16, 56(AP), 80(AP)                 ; 1246
                                               05  18 0002D        BGEQ     4$                                      ;
                         0000V   CF            6C  FA 0002F 3$:    CALLG    (AP), INVREFVAR_ERROR                   ; 1248
                      50    34   AC      4C    AC  C1 00034 4$:    ADDL3    76(AP), 52(AP), R0                      ; 1250
                            34   AC      FF    A0  9E 0003A        MOVAB    -1(R0), 52(AP)                          ;
                            2C   AC      34    AC  D0 0003F        MOVL     52(AP), 44(AP)                          ; 1251
                      50    50   AC      4C    AC  C3 00044        SUBL3    76(AP), 80(AP), R0                      ; 1252
                 38   AC         50            01  A1 0004A        ADDW3    #1, R0, 56(AP)                          ;
                                 50            01  D0 0004F        MOVL     #1, R0                                  ; 1254
                                                   04 00052        RET                                              ; 1256

; Routine Size:  83 bytes,    Routine Base:  _FOR$CODE + 014B


; 1202          1257  1 !<BLF/PAGE>
```

```
1 %SBTTL 'CONVERT_INTEGER - Convert a decimal integer'
1 ROUTINE CONVERT_INTEGER =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which converts the current token to a
1 !       longword integer which is stored in TPA$L_NUMBER.  If the conversion
1 !       fails, an error is signalled.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = CONVERT_INTEGER ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       TPA$L_NUMBER gets the binary value of the integer token
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL if success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May signal FOR$_INPCONERR, input conversion error
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     IF NOT OTS$CVT_TI_L (AP [TPA$L_TOKENCNT], AP [TPA$L_NUMBER])
2     THEN
2         CALLG (.AP, INPCONERR_ERROR);
2
2     RETURN 1;
2
1     END;
```

```
                                        0000 00000 CONVERT_INTEGER:
```

```
                                                                .WORD   Save nothing                                        ; 1259
                                     1C    AC  9F 00002        PUSHAB  28(AP)                                               ; 1302
                                     10    AC  9F 00005        PUSHAB  16(AP)                                               ;
                     00000000G  00         02  FB 00008        CALLS   #2, OTS$CVT_TI_L                                     ;
                                05         50  E8 0000F        BLBS    R0, 1$                                               ;
                         0000V  CF         6C  FA 00012        CALLG   (AP), INPCONERR_ERROR                                ; 1304
                                50         01  D0 00017 1$:    MOVL    #1, R0                                               ; 1306
                                               04 0001A        RET                                                          ; 1308

; Routine Size:  27 bytes,    Routine Base:  _FOR$CODE + 019E


; 1255          1309  1 !<BLF/PAGE>
```

```
%SBTTL 'STRING_OK - Is a string value ok?'
ROUTINE STRING_OK =

!++
! FUNCTIONAL DESCRIPTION:
!
!       LIB$TPARSE action routine which returns success if the current variable
!       datatype is CHARACTER.  It also sets TPA$V_BLANKS if successful.
!       If the datatype is not CHARACTER, INPCONERR is signalled.
!
! CALLING SEQUENCE:
!
!       status = STRING_OK ()
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       AP      Points to PARAM_BLOCK
!
! IMPLICIT OUTPUTS:
!
!       PARAM_BLOCK [TPA$V_BLANKS] = 1 if successful
!
! COMPLETION STATUS:
!
!       1 - success
!
! SIDE EFFECTS:
!
!       May signal FOR$_INPCONERR, input conversion error
!
!--

    BEGIN

    BUILTIN
        AP;             ! Argument pointer points to parameter block

    MAP
        AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);

    IF .AP [NML$B_DTYPE] NEQ DSC$K_DTYPE_T
    THEN
        CALLG (.AP, INPCONERR_ERROR);   ! Input conversion error

    IF .AP [NML$A_VARSTART] GEQA .AP [NML$A_VAREND]
    THEN
        FOR$$SIGNAL_STO (FOR$K_TOOMANYVAL, .AP [NML$A_VARNAME]);

    AP [TPA$V_BLANKS] = 1;
    AP [NML$B_CONSTYPE] = K_CHARACTER;
    AP [NML$L_CONSBLOCK] = .AP [NML$A_VARSTART];
    RETURN 1;

```

```
; 1314          1367  1     END;



                                          0000 00000 STRING_OK:
                                                               .WORD    Save nothing                                    ; 1311
                         0E        44    AC  91 00002          CMPB     68(AP), #14                                     ; 1354
                                         05  13 00006          BEQL     1$                                              ;
                  0000V  CF              6C  FA 00008          CALLG    (AP), INPCONERR_ERROR                           ; 1356
                     30  AC        2C    AC  D1 0000D 1$:      CMPL     44(AP), 48(AP)                                  ; 1358
                                         0C  1F 00012          BLSSU    2$                                              ;
                                   28    AC  DD 00014          PUSHL    40(AP)                                          ; 1360
                                         12  DD 00017          PUSHL    #18                                             ;
              00000000G  00              02  FB 00019          CALLS    #2, FOR$$SIGNAL_STO                             ;
                     04  AC              01  88 00020 2$:      BISB2    #1, 4(AP)                                       ; 1362
                     46  AC              05  90 00024          MOVB     #5, 70(AP)                                      ; 1363
                     68  AC        2C    AC  D0 00028          MOVL     44(AP), 104(AP)                                 ; 1364
                         50              01  D0 0002D          MOVL     #1, R0                                          ; 1365
                                             04 00030          RET                                                      ; 1367

; Routine Size:  49 bytes,    Routine Base:  _FOR$CODE + 01B9


; 1315          1368  1 !<BLF/PAGE>
```

```
1 %SBTTL 'STORE_CHARACTER - Store a character in a string'
1 ROUTINE STORE_CHARACTER =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which stores the character at TPA$B_CHAR
1 !       at the location referenced by NML$A_VARCUR.  NML$A_VARCUR is then
1 !       incremented by 1.  If the character would be stored past the end
1 !       of the string, the procedure returns success without storirg anything.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = STORE_CHARACTER ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE]  FIELD (NML$FIELDS);
2
2     IF .AP [NML$A_VARCUR] - .AP [NML$A_VARSTART] GEQA .AP [NML$W_VARSIZE]
2     THEN
2         RETURN 1;
2
2     CH$WCHAR_A (.AP [TPA$B_CHAR], AP [NML$A_VARCUR]);
2     RETURN 1;
2
1     END;
```

```
                                               0000 00000 STORE_CHARACTER:
                                                                  .WORD   Save nothing                             ; 1370
                    50      34   AC      2C   AC  C3 00002        SUBL3   44(AP), 52(AP), R0                       ; 1414
     50      38     AC           10           00  ED 00008        CMPZV   #0, #16, 56(AP), R0                      ;
                                              08  1B 0000E        BLEQU   1$                                       ;
                            34   BC      18   AC  90 00010        MOVB    24(AP), @52(AP)                          ; 1418
                                         34   AC  D6 00015        INCL    52(AP)                                   ;
                                 50           01  D0 00018 1$:    MOVL    #1, R0                                   ; 1419
                                                  04 0001B        RET                                              ; 1421

; Routine Size:  28 bytes,    Routine Base:  _FOR$CODE + 01EA


; 1370          1422  1 !<BLF/PAGE>
```

```
1 %SBTTL 'END_CHARACTER - End a character string'
1 ROUTINE END_CHARACTER =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which is called at the end of a character string value.
1 !       It blank fills the string if necessary and advances NML$A_VARSTART and
1 !       NML$A_VARCUR.  If the repeat count is greater than 1, multiple copies
1 !       are stored.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = END_CHARACTER ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NML$A_VARCUR = start of next string
1 !       NML$A_VARSTART = start of next string
1 !       User variable is modified.
1 !       NML$L_REPEATCT <= 1
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2    BEGIN
2
2    BUILTIN
2       AP;             ! Argument pointer points to parameter block
2
2    MAP
2       AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2    LOCAL
2       STRINGSIZE;     ! Size of string constant
2
2    STRINGSIZE = .AP [NML$A_VARCUR] - .AP [NML$A_VARSTART];
2    IF .STRINGSIZE LSSU .AP [NML$W_VARSIZE]
2    THEN
2       CH$FILL (%C' ', (.AP [NML$W_VARSIZE] - .STRINGSIZE), .AP [NML$A_VARCUR]);
2
2    !+
; 1429          1480  2        ! Update the current position in the variable.
; 1430          1481  2        !-
; 1431          1482  2
; 1432          1483  2        IF .AP [NML$W_STRIDE] NEQ 0
; 1433          1484  2        THEN
; 1434          1485  2            AP [NML$A_VARCUR] = .AP [NML$A_VARSTART] + .AP [NML$W_STRIDE]
; 1435          1486  2        ELSE
; 1436          1487  2            AP [NML$A_VARCUR] = .AP [NML$A_VAREND];
; 1437          1488  2
; 1438          1489  2        !+
; 1439          1490  2        ! While repeat count is greater than 1, store multiple copies.
; 1440          1491  2        !-
; 1441          1492  2
; 1442          1493  2        WHILE .AP [NML$L_REPEATCT] GTR 1 DO
; 1443          1494  3            BEGIN
; 1444          1495  3            IF .AP [NML$A_VARCUR] GEQA .AP [NML$A_VAREND]
; 1445          1496  3            THEN
; 1446          1497  3                FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
; 1447          1498  3            CH$MOVE (.AP [NML$W_VARSIZE], .AP [NML$A_VARSTART], .AP [NML$A_VARCUR]);
; 1448          1499  3            AP [NML$A_VARCUR] = .AP [NML$A_VARCUR] + .AP [NML$W_STRIDE];     ! Must be array!
; 1449          1500  3            AP [NML$L_REPEATCT] = .AP [NML$L_REPEATCT] - 1;
; 1450          1501  2            END;
; 1451          1502  2
; 1452          1503  2        AP [NML$A_VARSTART] = .AP [NML$A_VARCUR];
; 1453          1504  2        RETURN 1;
; 1454          1505  2
; 1455          1506  1        END;



                                                     007C 00000 END_CHARACTER:
                                                                        .WORD   Save R2,R3,R4,R5,R6                       ; 1424
                             56         34   AC   9E 00002              MOVAB   52(AP), R6                                ; 1474
                   50        66         2C   AC   C3 00006              SUBL3   44(AP), (R6), STRINGSIZE                  ;
     50     38     AC        10              00   ED 0000B              CMPZV   #0, #16, 56(AP), STRINGSIZE               ; 1475
                                             0F   1B 00011              BLEQU   1$                                        ;
                             51         38   AC   3C 00013              MOVZWL  56(AP), R1                                ; 1477
                   50        51              50   C3 00017              SUBL3   STRINGSIZE, R1, R0                        ;
     50            20        6E              00   2C 0001B              MOVC5   #0, (SP), #32, R0, @0(R6)                 ;
                                        00   B6      00020
                                        3A   AC   B5 00022 1$:          TSTW    58(AP)                                    ; 1483
                                             0B   13 00025              BEQL    2$                                        ;
                             50         3A   AC   3C 00027              MOVZWL  58(AP), R0                                ; 1485
                             66         2C BC40   9E 0002B              MOVAB   @44(AP)[R0], (R6)                         ;
                                             04   11 00030              BRB     3$                                        ;
                             66         30   AC   D0 00032 2$:          MOVL    48(AP), (R6)                              ; 1487
                             01         78   AC   D1 00036 3$:          CMPL    120(AP), #1                               ; 1493
                                             25   15 0003A              BLEQ    5$                                        ;
                        30   AC              66   D1 0003C              CMPL    (R6), 48(AP)                              ; 1495
                                             0C   1F 00040              BLSSU   4$                                        ;
                                        28   AC   DD 00042              PUSHL   40(AP)                                    ; 1497
                                             12   DD 00045              PUSHL   #18                                       ;
                 00000000G   00              02   FB 00047              CALLS   #2, FOR$$SIGNAL_STO                       ;
            00   B6     2C   BC         38   AC   28 0004E 4$:          MOVC3   56(AP), @44(AP), @0(R6)                   ; 1498
                             50         3A   AC   3C 00055              MOVZWL  58(AP), R0                                ; 1499
```

```
                          66           50  C0 00059          ADDL2   R0, (R6)                              ;
                                 78    AC  D7 0005C          DECL    120(AP)                               ; 1500
                                       D5  11 0005F          BRB     3$                                    ; 1493
                      2C  AC           66  D0 00061 5$:      MOVL    (R6), 44(AP)                          ; 1503
                          50           01  D0 00065          MOVL    #1, R0                                ; 1504
                                           04 00068          RET                                           ; 1506

; Routine Size:  105 bytes,    Routine Base:  _FOR$CODE + 0206


; 1456          1507  1 :<BLF/PAGE>
```

```
1 %SBTTL 'STORE_REAL - Store a real constant'
1 ROUTINE STORE_REAL =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which converts the real constant at
1 !       TPA$L_TOKENCNT and stores the value in NML$L_CONSBLOCK.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = STORE_REAL ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !       TPA$L_TOKENCNT - Descriptor of token
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NML$L_CONSBLOCK set to value of token
1 !       NML$B_CONSTYPE set to K_REAL
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !       0 if the token is of zero length.  This is because the pattern matches
1 !         the null string.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May call INPCONERR_ERROR
1 !       May signal FOR$_INVARGFOR
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;                 ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     !+
2     ! If token is of zero length, then return failure.
2     !-
2
2     IF .AP [TPA$L_TOKENCNT] EQL 0
2     THEN
2         RETURN 0;
2
2     !+
; 1515          1565  2       ! Since the pattern for a real matches a string such as "D123", which
; 1516          1566  2       ! might be an identifier, check for the first character being a letter.
; 1517          1567  2       ! If it is, then store the token, set the value to zero and return.
; 1518          1568  2       ! If we don't do this, an identifier like D99999999999 would get a
; 1519          1569  2       ! conversion error immediately.  No other "real" token can possibly
; 1520          1570  2       ! be an identifier.
; 1521          1571  2       !-
; 1522          1572  2
; 1523          1573  2       IF CH$RCHAR (.AP [TPA$L_TOKENPTR]) GEQU %C'A' AND
; 1524          1574  2          CH$RCHAR (.AP [TPA$L_TOKENPTR]) LEQU %C'z'
; 1525          1575  2       THEN
; 1526          1576  3           BEGIN
; 1527          1577  3           AP [NML$L_CONSBLOCK] = 0;        ! Set value to zero
; 1528          1578  3           AP [NML$B_CONSTYPE] = K_INTEGER;
; 1529          1579  3           IF .AP [TPA$L_TOKENCNT] LEQ 31
; 1530          1580  3           THEN
; 1531          1581  4               BEGIN
; 1532          1582  4               LOCAL
; 1533          1583  4                   TOKEN: REF VECTOR [, BYTE];
; 1534          1584  4               TOKEN = AP [NML$T_TOKEN];
; 1535          1585  4               TOKEN [0] = .AP [TPA$L_TOKENCNT];
; 1536          1586  4               CH$MOVE (.AP [TPA$L_TOKENCNT], .AP [TPA$L_TOKENPTR], TOKEN [1]);
; 1537          1587  4               END
; 1538          1588  3           ELSE
; 1539          1589  3               AP [NML$T_TOKEN] = 0;
; 1540          1590  3           RETURN 1;
; 1541          1591  3           END
; 1542          1592  2       ELSE
; 1543          1593  2           AP [NML$T_TOKEN] = 0;
; 1544          1594  2
; 1545          1595  2       !+
; 1546          1596  2       ! Depending on the destination type, convert the token appropriately.
; 1547          1597  2       !-
; 1548          1598  2
; 1549        P 1599  2       IF ONE_OF (.AP [NML$B_DTYPE],
; 1550        P 1600  2           DSC$K_DTYPE_L, DSC$K_DTYPE_W, DSC$K_DTYPE_B, DSC$K_DTYPE_WU,
; 1551          1601  3           DSC$K_DTYPE_LU, DSC$K_DTYPE_D, DSC$K_DTYPE_DC
; 1552          1602  2       THEN
; 1553          1603  3           BEGIN
; 1554          1604  3           IF NOT OTS$CVT_T_D (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK])
; 1555          1605  3           THEN
; 1556          1606  3               CALLG (.AP, INPCONERR_ERROR);
; 1557          1607  3           END
; 1558          1608  3
; 1559          1609  3       ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_F, DSC$K_DTYPE_FC)
; 1560          1610  2       THEN
; 1561          1611  3           BEGIN
; 1562          1612  3           IF NOT OTS$CVT_T_F (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK])
; 1563          1613  3           THEN
; 1564          1614  3               CALLG (.AP, INPCONERR_ERROR);
; 1565          1615  3           END
; 1566          1616  3
; 1567          1617  3       ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_G, DSC$K_DTYPE_GC)
; 1568          1618  2       THEN
; 1569          1619  3           BEGIN
; 1570          1620  3           IF NOT OTS$CVT_T_G (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK])
; 1571          1621  3           THEN
; 1572          1622  3                 CALLG (.AP, INPCONERR_ERROR);
; 1573          1623  3             END
; 1574          1624  3
; 1575          1625  3         ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_H)
; 1576          1626  2         THEN
; 1577          1627  3             BEGIN
; 1578          1628  3             IF NOT OTS$CVT_T_H (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK])
; 1579          1629  3             THEN
; 1580          1630  3                 CALLG (.AP, INPCONERR_ERROR);
; 1581          1631  3             END
; 1582          1632  3
; 1583          1633  3         ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_T)
; 1584          1634  2         THEN
; 1585          1635  3             BEGIN
; 1586          1636  3             AP [NML$L_CONSBLOCK] = 0;        ! Store zero result
; 1587          1637  3             CALLG (.AP, INPCONERR_ERROR);
; 1588          1638  3             END
; 1589          1639  3
; 1590          1640  2         ELSE          ! Invalid datatype
; 1591          1641  3             BEGIN
; 1592          1642  3             FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
; 1593          1643  2             END;
; 1594          1644  2
; 1595          1645  2
; 1596          1646  2         AP [NML$B_CONSTYPE] = K_REAL;
; 1597          1647  2
; 1598          1648  2         RETURN 1;
; 1599          1649  2
; 1600          1650  1         END;



                                            007C 00000 STORE_REAL:
                                                             .WORD   Save R2,R3,R4,R5,R6                          ; 1509
                          56       10   AC   9E 00002        MOVAB   16(AP), R6                                   ; 1560
                                        66   D5 00006        TSTL    (R6)                                         ;
                                        03   12 00008        BNEQ    1$                                           ;
                                      00B4   31 0000A        BRW     16$                                          ;
                       41   8F     14   BC   91 0000D 1$:    CMPB    @20(AP), #65                                 ; 1573
                                        28   1F 00012        BLSSU   4$                                           ;
                       7A   8F     14   BC   91 00014        CMPB    @20(AP), #122                                ; 1574
                                        21   1A 00019        BGTRU   4$                                           ;
                                   68   AC   D4 0001B        CLRL    104(AP)                                      ; 1577
                       46   AC          02   90 0001E        MOVB    #2, 70(AP)                                   ; 1578
                            1F          66   D1 00022        CMPL    (R6), #31                                    ; 1579
                                        0F   14 00025        BGTR    2$                                           ;
                          50       7C   AC   9E 00027        MOVAB   124(AP), TOKEN                               ; 1584
                          60            66   90 0002B        MOVB    (R6), (TOKEN)                                ; 1585
         01     A0      14   BC         66   28 0002E        MOVC3   (R6), @20(AP), 1(TOKEN)                      ; 1586
                                        03   11 00034        BRB     3$                                           ; 1579
                                   7C   AC   94 00036 2$:    CLRB    124(AP)                                      ; 1589
                                      0081   31 00039 3$:    BRW     15$                                          ; 1590
                                   7C   AC   94 0003C 4$:    CLRB    124(AP)                                      ; 1593
                          52       44   AC   9A 0003F        MOVZBL  68(AP), R2                                   ; 1601
            50 1B940000   8F            52   78 00043        ASHL    R2, #462684160, R0                           ;
```

```
                                    11  18 0004B          BGEQ    6$                                  ;
                              68    AC  9F 0004D          PUSHAB  104(AP)                             ; 1604
                                    56  DD 00050          PUSHL   R6                                  ;
               00000000G  00        02  FB 00052          CALLS   #2, OTS$CVT_T_D                     ;
                          5D        50  E8 00059 5$:      BLBS    R0, 14$                             ;
                                    4B  11 0005C          BRB     12$                                 ; 1606
                          0A        52  91 0005E 6$:      CMPB    R2, #10                             ; 1609
                                    05  13 00061          BEQL    7$                                  ;
                          0C        52  91 00063          CMPB    R2, #12                             ;
                                    0E  12 00066          BNEQ    8$                                  ;
                              68    AC  9F 00068 7$:      PUSHAB  104(AP)                             ; 1612
                                    56  DD 0006B          PUSHL   R6                                  ;
               00000000G  00        02  FB 0006D          CALLS   #2, OTS$CVT_T_F                     ;
                                    E3  11 00074          BRB     5$                                  ;
                          1B        52  91 00076 8$:      CMPB    R2, #27                             ; 1617
                                    05  13 00079          BEQL    9$                                  ;
                          1D        52  91 0007B          CMPB    R2, #29                             ;
                                    0E  12 0007E          BNEQ    10$                                 ;
                              68    AC  9F 00080 9$:      PUSHAB  104(AP)                             ; 1620
                                    56  DD 00083          PUSHL   R6                                  ;
               00000000G  00        02  FB 00085          CALLS   #2, OTS$CVT_T_G                     ;
                                    CB  11 0008C          BRB     5$                                  ;
                          1C        52  91 0008E 10$:     CMPB    R2, #28                             ; 1625
                                    0E  12 00091          BNEQ    11$                                 ;
                              68    AC  9F 00093          PUSHAB  104(AP)                             ; 1628
                                    56  DD 00096          PUSHL   R6                                  ;
               00000000G  00        02  FB 00098          CALLS   #2, OTS$CVT_T_H                     ;
                                    B8  11 0009F          BRB     5$                                  ;
                          0E        52  91 000A1 11$:     CMPB    R2, #14                             ; 1633
                                    0A  12 000A4          BNEQ    13$                                 ;
                              68    AC  D4 000A6          CLRL    104(AP)                             ; 1636
                   0000V  CF        6C  FA 000A9 12$:     CALLG   (AP), INPCONERR_ERROR               ; 1637
                                    09  11 000AE          BRB     14$                                 ; 1633
                                    30  DD 000B0 13$:     PUSHL   #48                                 ; 1642
               00000000G  00        01  FB 000B2          CALLS   #1, FOR$$SIGNAL_STO                 ;
                      46  AC        03  90 000B9 14$:     MOVB    #3, 70(AP)                          ; 1646
                          50        01  D0 000BD 15$:     MOVL    #1, R0                              ; 1648
                                        04 000C0          RET                                         ;
                                    50  D4 000C1 16$:     CLRL    R0                                  ; 1650
                                        04 000C3          RET                                         ;

; Routine Size:  196 bytes,    Routine Base:  _FOR$CODE + 026F

; 1601          1651  1 !<BLF/PAGE>
```

```
%SBTTL 'STORE_LOGICAL - Store a logical value'
ROUTINE STORE_LOGICAL =

!++
! FUNCTIONAL DESCRIPTION:
!
!       LIB$TPARSE action routine which converts the logical value at
!       TPA$L_TOKENCNT and stores the value at NML$L_CONSBLOCK.  If the
!       token is possibly an identifier, the token is saved at NML$T_TOKEN.
!
! CALLING SEQUENCE:
!
!       status = STORE_LOGICAL ()
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       AP      Points to PARAM_BLOCK
!       TPA$L_TOKENCNT is descriptor of token
!
! IMPLICIT OUTPUTS:
!
!       NML$L_CONSBLOCK gets converted value
!       NML$B_CONSTYPE gets K_LOGICAL
!       NML$T_TOKEN gets token if possibly an identifier
!
! COMPLETION STATUS:
!
!       1 for success
!
! SIDE EFFECTS:
!
!       NONE
!
!--

    BEGIN

    BUILTIN
        AP;                 ! Argument pointer points to parameter block

    MAP
        AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);

    IF CH$RCHAR (.AP [TPA$L_TOKENPTR]) NEQ %C'.' AND
        .AP [TPA$L_TOKENCNT] LEQ 31
    THEN
        BEGIN
        LOCAL
            TOKEN: REF VECTOR [, BYTE];
        TOKEN = AP [NML$T_TOKEN];
        TOKEN [0] = .AP [TPA$L_TOKENCNT];
        CH$MOVE (.AP [TPA$L_TOKENCNT], .AP [TPA$L_TOKENPTR], TOKEN [1]);
        END
```

```
; 1660          1709  2        ELSE
; 1661          1710  2            AP [NML$T_TOKEN] = 0;
; 1662          1711  2
; 1663          1712  2        OTS$CVT_TL_L (AP [TPA$L_TOKENCNT], AP [NML$L_CONSBLOCK]);
; 1664          1713  2        AP [NML$B_CONSTYPE] = K_LOGICAL;
; 1665          1714  2        RETURN 1;
; 1666          1715  2
; 1667          1716  1        END;



                                              003C 00000 STORE_LOGICAL:
                                                                .WORD   Save R2,R3,R4,R5                              ; 1653
                         2E       14    BC   91 00002           CMPB    @20(AP), #46                                  ; 1699
                                        17   13 00006           BEQL    1$                                            ;
                         1F       10    AC   D1 00008           CMPL    16(AP), #31                                   ; 1700
                                        11   14 0000C           BGTR    1$                                            ;
                         50       7C    AC   9E 0000E           MOVAB   124(AP), TOKEN                                ; 1705
                         60       10    AC   90 00012           MOVB    16(AP), (TOKEN)                               ; 1706
       01    A0      14  BC       10    AC   28 00016           MOVC3   16(AP), @20(AP), 1(TOKEN)                     ; 1707
                                        03   11 0001D           BRB     2$                                            ; 1699
                                  7C    AC   94 0001F 1$:       CLRB    124(AP)                                       ; 1710
                                  68    AC   9F 00022 2$:       PUSHAB  104(AP)                                       ; 1712
                                  10    AC   9F 00025           PUSHAB  16(AP)                                        ;
                 00000000G  00          02   FB 00028           CALLS   #2, OTS$CVT_TL_L                              ;
                        46  AC          01   90 0002F           MOVB    #1, 70(AP)                                    ; 1713
                         50             01   D0 00033           MOVL    #1, R0                                        ; 1714
                                             04 00036           RET                                                   ; 1716

; Routine Size:  55 bytes,    Routine Base:  _FOR$CODE + 0333

; 1668          1717  1 !<BLF/PAGE>
```

```
1 %SBTTL 'STORE_COMPLEX - Store a complex constant'
1 ROUTINE STORE_COMPLEX =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which converts the current token as a real
1 !       value and converts it to either the real part or the imaginary part
1 !       of a complex value.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = STORE_COMPLEX ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !       TPA$L_TOKENCNT - Descriptor of token
1 !       NML$V_IMAG     - Set if real part already seen
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NML$L_CONSBLOCK set to value of token
1 !       NML$B_CONSTYPE set to K_COMPLEX
1 !       NML$V_IMAG set to 1
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !       0 if the token is of zero length.  This is because the pattern matches
1 !         the null string.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May call INPCONERR_ERROR
1 !       May signal FOR$_INVARGFOR
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;                 ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     LOCAL
2         L_DTYPE,                          ! Local data type
2         L_CONSBLOCK: VECTOR [4, LONG];   ! Local constant block
2
2     !+
2     ! If token is of zero length, then return failure.
```

```
        !-

        IF .AP [TPA$L_TOKENCNT] EQL 0
        THEN
            RETURN 0;

        !+
        ! Depending on the destination type, convert the token appropriately.
        !-

        IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_F, DSC$K_DTYPE_FC)
        THEN
            BEGIN
            IF NOT OTS$CVT_T_F (AP [TPA$L_TOKENCNT], L_CONSBLOCK)
            THEN
                CALLG (.AP, INPCONERR_ERROR);
            END

        ELSE IF ONE_OF (.AP [NML$B_DTYPE],
            DSC$K_DTYPE_L, DSC$K_DTYPE_W, DSC$K_DTYPE_B, DSC$K_DTYPE_LU,
            DSC$K_DTYPE_WU, DSC$K_DTYPE_D, DSC$K_DTYPE_DC)
        THEN
            BEGIN
            IF NOT OTS$CVT_T_D (AP [TPA$L_TOKENCNT], L_CONSBLOCK)
            THEN
                CALLG (.AP, INPCONERR_ERROR);
            END

        ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_G, DSC$K_DTYPE_GC)
        THEN
            BEGIN
            IF NOT OTS$CVT_T_G (AP [TPA$L_TOKENCNT], L_CONSBLOCK)
            THEN
                CALLG (.AP, INPCONERR_ERROR);
            END

        ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_H)
        THEN
            BEGIN
            IF NOT OTS$CVT_T_H (AP [TPA$L_TOKENCNT], L_CONSBLOCK)
            THEN
                CALLG (.AP, INPCONERR_ERROR);
            END

        ELSE IF ONE_OF (.AP [NML$B_DTYPE], DSC$K_DTYPE_T)
        THEN
            BEGIN
            L_CONSBLOCK [0] = 0;     ! Store zero result
            CALLG (.AP, INPCONERR_ERROR);
            END

        ELSE
            BEGIN
            FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
            END;

```

```
2
2       AP [NML$B_CONSTYPE] = K_COMPLEX;
2
2       !+
2       ! Now convert the local constant to the proper complex type and store in
2       ! either the real or imaginary part of NML$L_CONSBLOCK.
2       !-
2
2       SELECTONE .AP [NML$B_DTYPE] OF
2           SET
2           [DSC$K_DTYPE_FC]:
2               L_DTYPE = DSC$K_DTYPE_F;
2           [DSC$K_DTYPE_DC]:
2               L_DTYPE = DSC$K_DTYPE_D;
2           [DSC$K_DTYPE_GC]:
2               L_DTYPE = DSC$K_DTYPE_G;
2           [OTHERWISE]:
2               L_DTYPE = .AP [NML$B_DTYPE];
2           TES;
2       IF NOT .AP [NML$V_IMAG]                 ! If real part
2       THEN
3           BEGIN
3           IF NOT FOR$$CVT_TYPE (K_REAL, L_CONSBLOCK,
3                                 .L_DTYPE, AP [NML$L_CONSBLOCK], 0)
3           THEN
4               BEGIN
4               AP [NML$L_CONSBLOCK] = 0;    ! Store zero result
4               CALLG (.AP, INPCONERR_ERROR);
3               END;
3           AP [NML$V_IMAG] = 1;
3           END
2       ELSE
3           BEGIN
3           IF .L_DTYPE EQL DSC$K_DTYPE_H
3           THEN
3               RETURN 1;
3           IF NOT FOR$$CVT_TYPE (K_REAL, L_CONSBLOCK,
3                                 .L_DTYPE,
4                                 (IF .L_DTYPE EQL DSC$K_DTYPE_F
4                                  THEN
4                                     AP [NML$L_CONSBLOCK] + 4
4                                  ELSE
3                                     AP [NML$L_CONSBLOCK] + 8),
3                                  0)
3           THEN
4               BEGIN
4               IF .L_DTYPE EQL DSC$K_DTYPE_F
4               THEN
4                   AP [NML$L_CONSBLOCK]+4 = 0      ! Store zero result
4               ELSE
4                   AP [NML$L_CONSBLOCK]+8 = 0;
4               CALLG (.AP, INPCONERR_ERROR);
3               END;
2           END;
2
2       RETURN 1;
2
```

```
; 1841          1889  1      END;



                                          003C 00000 STORE_COMPLEX:
                                                           .WORD   Save R2,R3,R4,R5                               ; 1719
                          55       0000V  CF    9E 00002          MOVAB   INPCONERR_ERROR, R5
                          54 00000000G    00    9E 00007          MOVAB   FOR$$CVT_TYPE, R4
                          5E              10    C2 0000E          SUBL2   #16, SP
                          53         10   AC    9E 00011          MOVAB   16(AP), R3                                     ; 1777
                                          63    D5 00015          TSTL    (R3)
                                          03    12 00017          BNEQ    1$
                                        00FA    31 00019          BRW     23$
                          52         44   AC    9A 0001C  1$:     MOVZBL  68(AP), R2                                     ; 1786
                          0A              52    91 00020          CMPB    R2, #10
                                          05    13 00023          BEQL    2$
                          0C              52    91 00025          CMPB    R2, #12
                                          10    12 00028          BNEQ    4$
                                   4008   8F    BB 0002A  2$:     PUSHR   #^M<R3,SP>                                     ; 1789
                 00000000G  00            02    FB 0002E          CALLS   #2, OTS$CVT_T_F
                            57            50    E8 00035  3$:     BLBS    R0, 11$
                                          47    11 00038          BRB     9$                                             ; 1791
            50 1B940000     8F            52    78 0003A  4$:     ASHL    R2, #462684160, R0                             ; 1796
                                          0D    18 00042          BGEQ    5$
                                   4008   8F    BB 00044          PUSHR   #^M<R3,SP>                                     ; 1799
                 00000000G  00            02    FB 00048          CALLS   #2, OTS$CVT_T_D
                                          E4    11 0004F          BRB     3$
                            1B            52    91 00051  5$:     CMPB    R2, #27                                        ; 1804
                                          05    13 00054          BEQL    6$
                            1D            52    91 00056          CMPB    R2, #29
                                          0D    12 00059          BNEQ    7$
                                   4008   8F    BB 0005B  6$:     PUSHR   #^M<R3,SP>                                     ; 1807
                 00000000G  00            02    FB 0005F          CALLS   #2, OTS$CVT_T_G
                                          CD    11 00066          BRB     3$
                            1C            52    91 00068  7$:     CMPB    R2, #28                                        ; 1812
                                          0D    12 0006B          BNEQ    8$
                                   4008   8F    BB 0006D          PUSHR   #^M<R3,SP>                                     ; 1815
                 00000000G  00            02    FB 00071          CALLS   #2, OTS$CVT_T_H
                                          BB    11 00078          BRB     3$
                            0E            52    91 0007A  8$:     CMPB    R2, #14                                        ; 1820
                                          07    12 0007D          BNEQ    10$
                                          6E    D4 0007F          CLRL    L_CONSBLOCK                                    ; 1823
                            65            6C    FA 00081  9$:     CALLG   (AP), INPCONERR_ERROR                          ; 1824
                                          09    11 00084          BRB     11$                                            ; 1820
                                          30    DD 00086  10$:    PUSHL   #48                                            ; 1829
                 00000000G  00            01    FB 00088          CALLS   #1, FOR$$SIGNAL_STO
                        46  AC            04    90 0008F  11$:    MOVB    #4, 70(AP)                                     ; 1833
                            50       44   AC    9A 00093          MOVZBL  68(AP), R0                                     ; 1840
                            0C            50    91 00097          CMPB    R0, #12                                        ; 1842
                                          05    12 0009A          BNEQ    12$
                            52            0A    D0 0009C          MOVL    #10, L_DTYPE                                   ; 1843
                                          17    11 0009F          BRB     15$
                            0D            50    91 000A1  12$:    CMPB    R0, #13                                        ; 1844
                                          05    12 000A4          BNEQ    13$
                            52            0B    D0 000A6          MOVL    #11, L_DTYPE                                   ; 1845
```

```
                                          0D  11 000A9          BRB     15$                                    ;
                            1D            50  91 000AB 13$:     CMPB    R0, #29                                ; 1846
                                          05  12 000AE          BNEQ    14$                                    ;
                            52            1B  D0 000B0          MOVL    #27, L_DTYPE                           ; 1847
                                          03  11 000B3          BRB     15$                                    ;
                            52            50  D0 000B5 14$:     MOVL    R0, L_DTYPE                            ; 1849
              1E       45   AC            01  E0 000B8 15$:     BBS     #1, 69(AP), 17$                        ; 1851
                                          7E  D4 000BD          CLRL    -(SP)                                  ; 1855
                                   68     AC  9F 000BF          PUSHAB  104(AP)                                ;
                                          52  DD 000C2          PUSHL   L_DTYPE                                ;
                                   0C     AE  9F 000C4          PUSHAB  L_CONSBLOCK                            ; 1854
                                          03  DD 000C7          PUSHL   #3                                     ; 1855
                            64            05  FB 000C9          CALLS   #5, FOR$$CVT_TYPE                      ;
                            06            50  E8 000CC          BLBS    R0, 16$                                ;
                                   68     AC  D4 000CF          CLRL    104(AP)                                ; 1858
                            65            6C  FA 000D2          CALLG   (AP), INPCONERR_ERROR                  ; 1859
                       45   AC            02  88 000D5 16$:     BISB2   #2, 69(AP)                             ; 1861
                                          37  11 000D9          BRB     22$                                    ; 1851
                            1C            52  D1 000DB 17$:     CMPL    L_DTYPE, #28                           ; 1865
                                          32  13 000DE          BEQL    22$                                    ;
                                          7E  D4 000E0          CLRL    -(SP)                                  ; 1868
                                          53  D4 000E2          CLRL    R3                                     ; 1870
                            0A            52  D1 000E4          CMPL    L_DTYPE, #10                           ;
                                          08  12 000E7          BNEQ    18$                                    ;
                                          53  D6 000E9          INCL    R3                                     ;
                            50     6C     AC  9E 000EB          MOVAB   108(AP), R0                            ; 1872
                                          04  11 000EF          BRB     19$                                    ;
                            50     70     AC  9E 000F1 18$:     MOVAB   112(AP), R0                            ; 1874
                                          50  DD 000F5 19$:     PUSHL   R0                                     ;
                                          52  DD 000F7          PUSHL   L_DTYPE                                ; 1869
                                   0C     AE  9F 000F9          PUSHAB  L_CONSBLOCK                            ; 1868
                                          03  DD 000FC          PUSHL   #3                                     ;
                            64            05  FB 000FE          CALLS   #5, FOR$$CVT_TYPE                      ;
                            0E            50  E8 00101          BLBS    R0, 22$                                ;
                            05            53  E9 00104          BLBC    R3, 20$                                ; 1878
                                   6C     AC  D4 00107          CLRL    108(AP)                                ; 1880
                                          03  11 0010A          BRB     21$                                    ;
                                   70     AC  D4 0010C 20$:     CLRL    112(AP)                                ; 1882
                            65            6C  FA 0010F 21$:     CALLG   (AP), INPCONERR_ERROR                  ; 1883
                            50            01  D0 00112 22$:     MOVL    #1, R0                                 ; 1887
                                              04 00115          RET                                            ;
                                          50  D4 00116 23$:     CLRL    R0                                     ; 1889
                                              04 00118          RET                                            ;

; Routine Size:  281 bytes,    Routine Base:  _FOR$CODE + 036A


; 1842          1890  1 !<BLF/PAGE>
```

```
%SBTTL 'STORE_REPEAT - Store a repeat count'
ROUTINE STORE_REPEAT =

!++
! FUNCTIONAL DESCRIPTION:
!
!       LIB$TPARSE action routine which stores the repeat count into the
!       parameter block.
!
! CALLING SEQUENCE:
!
!       status = STORE_REPEAT ()
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       AP      Points to PARAM_BLOCK
!
! IMPLICIT OUTPUTS:
!
!       NML$L_REPEATCT gets the repeat count
!       NML$B_CONSTYPE = K_NULL
!
! COMPLETION STATUS:
!
!       1 for success
!
! SIDE EFFECTS:
!
!       May signal FOR$_SYNERRNAM, syntax error in NAMELIST input
!
!--

    BEGIN

    BUILTIN
        AP;             ! Argument pointer points to parameter block

    MAP
        AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);

    IF .AP [TPA$L_NUMBER] LEQ 0
    THEN
        CALLG (.AP, SYNTAX_ERROR);

    AP [NML$L_REPEATCT] = .AP [TPA$L_NUMBER];
    AP [NML$B_CONSTYPE] = K_NULL;       ! Initially treat as null value

    RETURN 1;

    END;
```

```
                              0000 00000 STORE_REPEAT:
                                              .WORD    Save nothing                        ; 1892
                       1C   AC  D5 00002       TSTL     28(AP)                             ; 1935
                            05  14 00005       BGTR     1$                                 ;
          0000V  CF         6C  FA 00007       CALLG    (AP), SYNTAX_ERROR                 ; 1937
             78  AC    1C   AC  D0 0000C 1$:   MOVL     28(AP), 120(AP)                    ; 1939
                       46   AC  94 00011       CLRB     70(AP)                             ; 1940
                 50         01  D0 00014       MOVL     #1, R0                             ; 1942
                                04 00017       RET                                         ; 1944

; Routine Size:  24 bytes,    Routine Base:  _FOR$CODE + 0483

; 1898         1945  1 !<BLF/PAGE>
; 1900          1946  1 %SBTTL 'END_REPEAT - End a repeated value'
; 1901          1947  1 ROUTINE END_REPEAT =
; 1902          1948  1
; 1903          1949  1 !++
; 1904          1950  1 ! FUNCTIONAL DESCRIPTION:
; 1905          1951  1 !
; 1906          1952  1 !       LIB$TPARSE action routine which marks the end of a repeated value.
; 1907          1953  1 !
; 1908          1954  1 ! CALLING SEQUENCE:
; 1909          1955  1 !
; 1910          1956  1 !       status = END_REPEAT ()
; 1911          1957  1 !
; 1912          1958  1 ! FORMAL PARAMETERS:
; 1913          1959  1 !
; 1914          1960  1 !       NONE
; 1915          1961  1 !
; 1916          1962  1 ! IMPLICIT INPUTS:
; 1917          1963  1 !
; 1918          1964  1 !       AP      Points to PARAM_BLOCK
; 1919          1965  1 !
; 1920          1966  1 ! IMPLICIT OUTPUTS:
; 1921          1967  1 !
; 1922          1968  1 !       NML$T_TOKEN = 0, meaning that this value can't be an identifier
; 1923          1969  1 !       TPA$V_BLANKS = 0, disabling explicit blank processing
; 1924          1970  1 !
; 1925          1971  1 ! COMPLETION STATUS:
; 1926          1972  1 !
; 1927          1973  1 !       1 for success
; 1928          1974  1 !
; 1929          1975  1 ! SIDE EFFECTS:
; 1930          1976  1 !
; 1931          1977  1 !       NONE
; 1932          1978  1 !
; 1933          1979  1 !--
; 1934          1980  1
; 1935          1981  2   BEGIN
; 1936          1982  2
; 1937          1983  2   BUILTIN
; 1938          1984  2       AP;             ! Argument pointer points to parameter block
; 1939          1985  2
; 1940          1986  2   MAP
; 1941          1987  2       AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
; 1942          1988  2
; 1943          1989  2   AP [NML$T_TOKEN] = 0;       ! Inhibit use of this token as an identifier
; 1944          1990  2   AP [TPA$V_BLANKS] = 0;      ! Turn off explicit blank processing
; 1945          1991  2
; 1946          1992  2   RETURN 1;
; 1947          1993  2
; 1948          1994  1   END;



                                      0000 00000 END_REPEAT:
                                                           .WORD   Save nothing                           ; 1947
                                  7C    AC  94 00002       CLRB    124(AP)                                ; 1989
```

```
                               04    AC          01  8A 00005          BICB2   #1, 4(AP)                                 ; 1990
                                     50          01  D0 00009          MOVL    #1, R0                                    ; 1992
                                                     04 0000C          RET                                               ; 1994

; Routine Size:  13 bytes,    Routine Base:  _FOR$CODE + 049B


; 1949          1995  1 .<BLF/PAGE>
```

```
1 %SBTTL 'STORE_VALUE - Store a value in a variable'
1 ROUTINE STORE_VALUE=
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which stores the value just read in the
1 !       current variable.  If the repeat count is greater than 1, multiple
1 !       copies are moved.  However, if the value was of type CHARACTER,
1 !       all copies have been stored and this routine only returns success.
1 !       If the constant type is NULL, then "repeat-count" values are skipped.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = STORE_VALUE ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       The user variable is modified (if value not NULL)
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals FOR$_SYNERRNAM if an error occurs during conversion.
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2       AP;                ! Argument pointer points to parameter block
2
2     MAP
2       AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     !+
2     ! If this was a character string, all values have been stored.
2     !-
2
2     IF .AP [NML$B_CONSTYPE] EQL K_CHARACTER
2     THEN
2       RETURN 1;
2
2     !+
2     ! Check to see if we are past the end of the variable or array
2     !-
```

```
2
2       IF .AP [NML$A_VARSTART] GEQA .AP [NML$A_VAREND]
2       THEN
3           BEGIN
3           FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
3           RETURN 0;
2           END;
2
2       !+
2       ! If this was a repeated null (n*), then skip over values.
2       !-
2
2       IF .AP [NML$B_CONSTYPE] EQL K_NULL
2       THEN
2           WHILE .AP [NML$L_REPEATCT] GTR 0 DO
3               BEGIN
3               IF .AP [NML$A_VARCUR] GEQA .AP [NML$A_VAREND]
3               THEN
4                   BEGIN
4                   FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
4                   RETURN 0;
3                   END;
3               AP [NML$A_VARCUR] = .AP [NML$A_VARCUR] + .AP [NML$W_VARSIZE];
3               AP [NML$L_REPEATCT] = .AP [NML$L_REPEATCT] - 1;
3               END
2       ELSE
3           BEGIN
3
3           !+
3           ! Call routine to convert value to the appropriate destination type.
3           ! If conversion fails, signal an error.
3           !-
3
3           IF NOT FOR$$CVT_TYPE (.AP [NML$B_CONSTYPE], AP [NML$L_CONSBLOCK],
3                               .AP [NML$B_DTYPE], .AP [NML$A_VARSTART], 0)
3           THEN
3               CALLG (.AP, INPCONERR_ERROR);
3           AP [NML$A_VARCUR] = .AP [NML$A_VARSTART] + .AP [NML$W_VARSIZE];
3
3           !+
3           ! While repeat count is greater than 1, store copies of the value.
3           !-
3
3           WHILE .AP [NML$L_REPEATCT] GTR 1 DO
4               BEGIN
4               IF .AP [NML$A_VARCUR] GEQA .AP [NML$A_VAREND]
4               THEN
5                   BEGIN
5                   FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
5                   RETURN 0;
4                   END;
4               AP [NML$A_VARCUR] = CH$MOVE (.AP [NML$W_VARSIZE], .AP [NML$A_VARSTART],
4                                           .AP [NML$A_VARCUR]);
4               AP [NML$L_REPEATCT] = .AP [NML$L_REPEATCT] - 1;
3               END;
2           END;
2
```

```
; 2065          2110  2         !+
; 2066          2111  2         ! Turn off NML$V_IMAG if set.  This lets subsequent complex values get
; 2067          2112  2         ! stored correctly.
; 2068          2113  2         !-
; 2069          2114  2
; 2070          2115  2         AP [NML$V_IMAG] = 0;
; 2071          2116  2
; 2072          2117  2         !+
; 2073          2118  2         ! Update VARSTART with new position
; 2074          2119  2         !-
; 2075          2120  2
; 2076          2121  2         AP [NML$A_VARSTART] = .AP [NML$A_VARCUR];
; 2077          2122  2         RETURN 1;
; 2078          2123  1         END;



                                          00FC 00000 STORE_VALUE:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7                    ; 1997
                    05       46   AC   91 00002            CMPB    70(AP), #5                                ; 2046
                                  03   12 00006            BNEQ    1$                                        ;
                                0080   31 00008            BRW     9$                                        ;
              30   AC        2C   AC   D1 0000B 1$:        CMPL    44(AP), 48(AP)                            ; 2054
                                  55   1E 00010            BGEQU   6$                                        ;
                    57       78   AC   9E 00012            MOVAB   120(AP), R7                               ; 2067
                    56       34   AC   9E 00016            MOVAB   52(AP), R6                                ; 2069
                             46   AC   95 0001A            TSTB    70(AP)                                    ; 2065
                                  15   12 0001D            BNEQ    3$                                        ;
                                  67   D5 0001F 2$:        TSTL    (R7)                                      ; 2067
                                  60   15 00021            BLEQ    8$                                        ;
              30   AC             66   D1 00023            CMPL    (R6), 48(AP)                              ; 2069
                                  3E   1E 00027            BGEQU   6$                                        ;
                    50       38   AC   3C 00029            MOVZWL  56(AP), R0                                ; 2075
                    66            50   C0 0002D            ADDL2   R0, (R6)                                  ;
                                  67   D7 00030            DECL    (R7)                                      ; 2076
                                  EB   11 00032            BRB     2$                                        ; 2067
                                  7E   D4 00034 3$:        CLRL    -(SP)                                     ; 2086
                             2C   AC   DD 00036            PUSHL   44(AP)                                    ; 2087
                    7E       44   AC   9A 00039            MOVZBL  68(AP), -(SP)                             ;
                             68   AC   9F 0003D            PUSHAB  104(AP)                                   ; 2086
                    7E       46   AC   9A 00040            MOVZBL  70(AP), -(SP)                             ;
          00000000G 00            05   FB 00044            CALLS   #5, FOR$$CVT_TYPE                         ;
                    05            50   E8 0004B            BLBS    R0, 4$                                    ;
              0000V CF            6C   FA 0004E            CALLG   (AP), INPCONERR_ERROR                     ; 2089
                    50       38   AC   3C 00053 4$:        MOVZWL  56(AP), R0                                ; 2090
                    66       2C BC40   9E 00057            MOVAB   @44(AP)[R0], (R6)                         ;
                    01            67   D1 0005C 5$:        CMPL    (R7), #1                                  ; 2096
                                  22   15 0005F            BLEQ    8$                                        ;
              30   AC             66   D1 00061            CMPL    (R6), 48(AP)                              ; 2098
                                  0E   1F 00065            BLSSU   7$                                        ;
                             28   AC   DD 00067 6$:        PUSHL   40(AP)                                    ; 2101
                                  12   DD 0006A            PUSHL   #18                                       ;
          00000000G 00            02   FB 0006C            CALLS   #2, FOR$$SIGNAL_STO                       ;
                                  1A   11 00073            BRB     10$                                       ; 2102
     00   B6     2C   BC         38   AC   28 00075 7$:        MOVC3   56(AP), @44(AP), @0(R6)                   ; 2105
```

```
                         66          53  D0 0007C          MOVL    R3, (R6)                             ;
                                     67  D7 0007F          DECL    (R7)                                 ; 2106
                                     D9  11 00081          BRB     5$                                   ; 2096
                    45   AC          02  8A 00083 8$:      BICB2   #2, 69(AP)                           ; 2115
                    2C   AC          66  D0 00087          MOVL    (R6), 44(AP)                         ; 2121
                         50          01  D0 0008B 9$:      MOVL    #1, R0                               ; 2122
                                         04 0008E          RET                                          ;
                                     50  D4 0008F 10$:     CLRL    R0                                   ; 2123
                                         04 00091          RET                                          ;

; Routine Size:  146 bytes,    Routine Base:  _FOR$CODE + 04A8


; 2079          2124  1 !<BLF/PAGE>
```

```
1 %SBTTL 'NULL_VALUE - Skip an element'
1 ROUTINE NULL_VALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which is called when a comma is found in place
1 !       of a value.  The pointer to the current element is advanced one element
1 !       with no change being made to the current element.  Note that if the
1 !       current variable is not an array, an attempt to store a following value
1 !       will be an error.  If we have already passed the last element, give
1 !       an error.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = NULL_VALUE ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NML$A_VARSTART is advanced one element.
1 !       NML$A_VARCUR = NML$A_VARSTART
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !
1 !       FOR$_TOOMANVAL - if this comma implies a skip past the end of the
1 !       variable.
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     !+
2     ! If we are already past the end of the variable, give an error.
2     !-
2
2     IF .AP [NML$A_VARSTART] GEQA .AP [NML$A_VAREND]
2     THEN
2         FOR$$SIGNAL_STO (FOR$K_TOOMANVAL, .AP [NML$A_VARNAME]);
2
```

```
; 2138          2182  2      IF .AP [NML$W_STRIDE] NEQ 0
; 2139          2183  2      THEN
; 2140          2184  2          AP [NML$A_VARSTART] = .AP [NML$A_VARSTART] + .AP [NML$W_STRIDE]
; 2141          2185  2      ELSE
; 2142          2186  2          AP [NML$A_VARSTART] = .AP [NML$A_VAREND];
; 2143          2187  2
; 2144          2188  2      AP [NML$A_VARCUR] = .AP [NML$A_VARSTART];
; 2145          2189  2
; 2146          2190  2      RETURN 1;
; 2147          2191  2
; 2148          2192  1      END;


                                                0000 00000 NULL_VALUE:
                                                                   .WORD   Save nothing                          ; 2126
                30    AC        2C    AC  D1 00002                 CMPL    44(AP), 48(AP)                        ; 2178
                                      0C  1F 00007                 BLSSU   1$                                    ;
                                28    AC  DD 00009                 PUSHL   40(AP)                                ; 2180
                                      12  DD 0000C                 PUSHL   #18                                   ;
         00000000G  00                02  FB 0000E                 CALLS   #2, FOR$$SIGNAL_STO                   ;
                                3A    AC  B5 00015 1$:             TSTW    58(AP)                                ; 2182
                                      0A  13 00018                 BEQL    2$                                    ;
                      50        3A    AC  3C 0001A                 MOVZWL  58(AP), R0                            ; 2184
                2C    AC              50  C0 0001E                 ADDL2   R0, 44(AP)                            ;
                                      05  11 00022                 BRB     3$                                    ;
                2C    AC        30    AC  D0 00024 2$:             MOVL    48(AP), 44(AP)                        ; 2186
                34    AC        2C    AC  D0 00029 3$:             MOVL    44(AP), 52(AP)                        ; 2188
                      50              01  D0 0002E                 MOVL    #1, R0                                ; 2190
                                          04 00031                 RET                                           ; 2192

; Routine Size:  50 bytes,    Routine Base:  _FOR$CODE + 053A


; 2149          2193  1 !<BLF/PAGE>
```

```
1 %SBTTL 'SYNTAX_ERROR - Signal syntax error'
1 ROUTINE SYNTAX_ERROR =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which signals a syntax error.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = SYNTAX_ERROR ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals FOR$_SYNERRNAM - Syntax error in NAMELIST
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;                ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     IF .AP [TPA$L_TOKENCNT] LSS 6
2     THEN
3         BEGIN
3         LOCAL
3             EXTRA,
3             CCB: REF $FOR$CCB_DECL;
3         CCB = .AP [NML$A_CCB];
3
3         !+
3         ! Try to make the string reported include the part of the record
3         ! where the error was.
3         !-
3
3         IF .AP [TPA$L_TOKENPTR] GEQA .CCB [LUB$A_BUF_PTR]
3         THEN
```

```
; 2208          2251  4               BEGIN
; 2209          2252  4               EXTRA = MAX (0, (6 - .AP [TPA$L_TOKENCNT]));
; 2210          2253  4               IF .AP [TPA$L_TOKENPTR] - .EXTRA LSSA .CCB [LUB$A_BUF_PTR]
; 2211          2254  4               THEN
; 2212          2255  4                   EXTRA = .AP [TPA$L_TOKENPTR] - .CCB [LUB$A_BUF_PTR];
; 2213          2256  4               AP [TPA$L_TOKENCNT] = .AP [TPA$L_TOKENCNT] + .EXTRA;
; 2214          2257  4               AP [TPA$L_TOKENPTR] = .AP [TPA$L_TOKENPTR] - .EXTRA;
; 2215          2258  3               END;
; 2216          2259  2           END;
; 2217          2260  2
; 2218          2261  2       FOR$$SIGNAL_STO (FOR$K_SYNERRNAM, AP [TPA$L_TOKENCNT]);
; 2219          2262  2       RETURN 0;
; 2220          2263  2
; 2221          2264  1       END;



                                                    0004 00000 SYNTAX_ERROR:
                                                                              .WORD   Save R2                              ; 2195
                                 06         10   AC  D1 00002                 CMPL    16(AP), #6                           ; 2236
                                                 30  18 00006                 BGEQ    3$                                   ;
                                 51         40   AC  D0 00008                 MOVL    64(AP), CCB                          ; 2242
                         B0      A1         14   AC  D1 0000C                 CMPL    20(AP), -80(CCB)                     ; 2249
                                                 25  1F 00011                 BLSSU   3$                                   ;
                50               06         10   AC  C3 00013                 SUBL3   16(AP), #6, R0                       ; 2252
                                                 02  18 00018                 BGEQ    1$                                   ;
                                                 50  D4 0001A                 CLRL    R0                                   ;
                                 52              50  D0 0001C 1$:             MOVL    R0, EXTRA                            ;
                50       14      AC              52  C3 0001F                 SUBL3   EXTRA, 20(AP), R0                    ; 2253
                         B0      A1              50  D1 00024                 CMPL    R0, -80(CCB)                         ;
                                                 06  1E 00028                 BGEQU   2$                                   ;
                52       14      AC         B0   A1  C3 0002A                 SUBL3   -80(CCB), 20(AP), EXTRA              ; 2255
                         10      AC              52  C0 00030 2$:             ADDL2   EXTRA, 16(AP)                        ; 2256
                         14      AC              52  C2 00034                 SUBL2   EXTRA, 20(AP)                        ; 2257
                                            10   AC  9F 00038 3$:             PUSHAB  16(AP)                               ; 2261
                                                 11  DD 0003B                 PUSHL   #17                                  ;
                   00000000G     00              02  FB 0003D                 CALLS   #2, FOR$$SIGNAL_STO                  ;
                                                 50  D4 00044                 CLRL    R0                                   ; 2262
                                                     04 00046                 RET                                          ; 2264

; Routine Size:  71 bytes,    Routine Base:  _FOR$CODE + 056C


; 2222          2265  1 !<BLF/PAGE>
```

```
1 %SBTTL 'INVREFVAR_ERROR - Signal invalid variable reference error'
1 ROUTINE INVREFVAR_ERROR =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which signals an invalid variable
1 !       reference  error.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = INVREFVAR_ERROR ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals FOR$_INVREFVAR - Invalid reference to variable in NAMELIST
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;             ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     LOCAL
2         DESCR: VECTOR [2, LONG],
2         VARNAME: REF VECTOR [, BYTE];
2
2     VARNAME = .AP [NML$A_VARNAME];
2     DESCR [0] = .VARNAME [0];
2     DESCR [1] = VARNAME [1];
2     FOR$$SIGNAL_STO (FOR$K_INVREFVAR, DESCR);
2     RETURN 0;
2
1     END;
```

```
                                         0000 00000 INVREFVAR_ERROR:
                                                           .WORD    Save nothing                                ; 2267
                      5E            04  C2 00002           SUBL2    #4, SP                                      ;
                      50      28    AC  D0 00005           MOVL     40(AP), VARNAME                             ; 2313
                      7E            60  9A 00009           MOVZBL   (VARNAME), DESCR                            ; 2314
              04      AE      01    A0  9E 0000C           MOVAB    1(R0), DESCR+4                              ; 2315
                                    5E  DD 00011           PUSHL    SP                                          ; 2316
                                    13  DD 00013           PUSHL    #19                                         ;
         00000000G    00            02  FB 00015           CALLS    #2, FOR$$SIGNAL_STO                         ;
                                    50  D4 0001C           CLRL     R0                                          ; 2317
                                        04 0001E           RET                                                  ; 2319

; Routine Size:  31 bytes,    Routine Base:  _FOR$CODE + 05B3


; 2278          2320  1 !<BLF/PAGE>
```

```
1 %SBTTL 'INPCONERR_ERROR - Signal input conversion error'
1 ROUTINE INPCONERR_ERROR =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Routine which signals FOR$_INPCONERR, "input conversion error",
1 !       with a chained message giving the text and record number.  Although
1 !       called as if it were a LIB$TPARSE action routine, in fact it is
1 !       only called from other action routines.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = INPCONERR_ERROR ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals FOR$_INPCONERR, input conversion error
1 !
1 !--
1
2    BEGIN
2
2    BUILTIN
2       AP;             ! Argument pointer points to parameter block
2
2    MAP
2       AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2    LOCAL
2       CCB: REF $FOR$CCB_DECL;
2
2    CCB = .AP [NML$A_CCB];      ! Get CCB address
2
2    !+
2    ! If the file is indexed organization or is an internal file (unlikely,
2    ! since that's not allowed), then use the message that doesn't have
2    ! a record number.  Otherwise chain the message with both text and
2    ! record number.  Signal it as a continuable error.
2    !-
2
```

```
; 2337          2378  2       IF (.CCB [LUB$B_ORGAN] EQL LUB$K_ORG_INDEX) OR
; 2338          2379  3          (.CCB [LUB$W_LUN] EQL LUB$K_LUN_ENCD)
; 2339          2380  2       THEN
; 2340          2381  2          FOR$$SIGNAL (FOR$K_INPCONERR, FOR$_INVTEX, 1, AP [TPA$L_TOKENCNT])
; 2341          2382  2       ELSE
; 2342          2383  2          FOR$$SIGNAL (FOR$K_INPCONERR, FOR$_INVTEXREC, 2,
; 2343          2384  2             AP [TPA$L_TOKENCNT], .CCB [LUB$L_LOG_RECNO] - 1);
; 2344          2385  2       RETURN 0;
; 2345          2386  2
; 2346          2387  1       END;



                                             001C 00000 INPCONERR_ERROR:
                                                                   .WORD    Save R2,R3,R4                    ; 2322
                          54 00000000G  00  9E 00002               MOVAB    FOR$$SIGNAL, R4                  ;
                          52        40  AC  D0 00009               MOVL     64(AP), CCB                      ; 2369
                          53        10  AC  9E 0000D               MOVAB    16(AP), R3                       ; 2381
                          03        C4  A2  91 00011               CMPB     -60(CCB), #3                     ; 2378
                                        08  13 00015               BEQL     1$                               ;
                  FFFB    8F        C6  A2  B1 00017               CMPW     -58(CCB), #-5                    ; 2379
                                        13  12 0001D               BNEQ     2$                               ;
                                        53  DD 0001F 1$:           PUSHL    R3                               ; 2381
                                        01  DD 00021               PUSHL    #1                               ;
                             0018883C   8F  DD 00023               PUSHL    #1607740                         ;
                          7E        40  8F  9A 00029               MOVZBL   #64, -(SP)                       ;
                          64            04  FB 0002D               CALLS    #4, FOR$$SIGNAL                  ;
                                        16  11 00030               BRB      3$                               ;
           7E        E0   A2            01  C3 00032 2$:           SUBL3    #1, -32(CCB), -(SP)              ; 2384
                                        53  DD 00037               PUSHL    R3                               ;
                                        02  DD 00039               PUSHL    #2                               ;
                             00188834   8F  DD 0003B               PUSHL    #1607732                         ;
                          7E        40  8F  9A 00041               MOVZBL   #64, -(SP)                       ;
                          64            05  FB 00045               CALLS    #5, FOR$$SIGNAL                  ;
                                        50  D4 00048 3$:           CLRL     R0                               ; 2385
                                            04 0004A               RET                                       ; 2387

; Routine Size:  75 bytes,    Routine Base:  _FOR$CODE + 05D2


; 2347          2388  1 !<BLF/PAGE>
; 2349          2389  1 %SBTTL 'BLANKS_OFF - Turn off explicit blanks'
; 2350          2390  1 ROUTINE BLANKS_OFF =
; 2351          2391  1
; 2352          2392  1 !++
; 2353          2393  1 ! FUNCTIONAL DESCRIPTION:
; 2354          2394  1 !
; 2355          2395  1 !       Turns off explicit blank processing for LIB$TPARSE.  When off, blanks
; 2356          2396  1 !       are implicit separators.
; 2357          2397  1 !
; 2358          2398  1 ! CALLING SEQUENCE:
; 2359          2399  1 !
; 2360          2400  1 !       status = BLANKS_OFF ()
; 2361          2401  1 !
; 2362          2402  1 ! FORMAL PARAMETERS:
; 2363          2403  1 !
; 2364          2404  1 !       NONE
; 2365          2405  1 !
; 2366          2406  1 ! IMPLICIT INPUTS:
; 2367          2407  1 !
; 2368          2408  1 !       AP      Points to PARAM_BLOCK
; 2369          2409  1 !
; 2370          2410  1 ! IMPLICIT OUTPUTS:
; 2371          2411  1 !
; 2372          2412  1 !       PARAM_BLOCK [TPA$V_BLANKS] = 0
; 2373          2413  1 !
; 2374          2414  1 ! COMPLETION STATUS:
; 2375          2415  1 !
; 2376          2416  1 !       1 for success
; 2377          2417  1 !
; 2378          2418  1 ! SIDE EFFECTS:
; 2379          2419  1 !
; 2380          2420  1 !       NONE
; 2381          2421  1 !
; 2382          2422  1 !--
; 2383          2423  1
; 2384          2424  2    BEGIN
; 2385          2425  2
; 2386          2426  2    BUILTIN
; 2387          2427  2       AP;               ! Argument pointer points to parameter block
; 2388          2428  2
; 2389          2429  2    MAP
; 2390          2430  2       AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
; 2391          2431  2
; 2392          2432  2    AP [TPA$V_BLANKS] = 0;       ! Turn off blank processing
; 2393          2433  2    RETURN 1;
; 2394          2434  2
; 2395          2435  1    END;



                                       0000 00000 BLANKS_OFF:
                                                        .WORD   Save nothing                              ; 2390
                 04   AC          01  8A 00002          BICB2   #1, 4(AP)                                 ; 2432
                      50          01  D0 00006          MOVL    #1, R0                                    ; 2433
                                      04 00009          RET                                               ; 2435
```

```
; Routine Size:  10 bytes,    Routine Base:  _FOR$CODE + 061D

; 2396          2436  1 !<BLF/PAGE>
; 2398          2437  1 %SBTTL 'BLANKS_ON - Turn on explicit blanks'
; 2399          2438  1 ROUTINE BLANKS_ON =
; 2400          2439  1
; 2401          2440  1 !++
; 2402          2441  1 ! FUNCTIONAL DESCRIPTION:
; 2403          2442  1 !
; 2404          2443  1 !       Turns on explicit blank processing for LIB$TPARSE.  When on, blanks
; 2405          2444  1 !       are not implicit separators.
; 2406          2445  1 !
; 2407          2446  1 ! CALLING SEQUENCE:
; 2408          2447  1 !
; 2409          2448  1 !       status = BLANKS_ON ()
; 2410          2449  1 !
; 2411          2450  1 ! FORMAL PARAMETERS:
; 2412          2451  1 !
; 2413          2452  1 !       NONE
; 2414          2453  1 !
; 2415          2454  1 ! IMPLICIT INPUTS:
; 2416          2455  1 !
; 2417          2456  1 !       AP      Points to PARAM_BLOCK
; 2418          2457  1 !
; 2419          2458  1 ! IMPLICIT OUTPUTS:
; 2420          2459  1 !
; 2421          2460  1 !       PARAM_BLOCK [TPA$V_BLANKS] = 0
; 2422          2461  1 !
; 2423          2462  1 ! COMPLETION STATUS:
; 2424          2463  1 !
; 2425          2464  1 !       1 for success
; 2426          2465  1 !
; 2427          2466  1 ! SIDE EFFECTS:
; 2428          2467  1 !
; 2429          2468  1 !       NONE
; 2430          2469  1 !
; 2431          2470  1 !--
; 2432          2471  1
; 2433          2472  2     BEGIN
; 2434          2473  2
; 2435          2474  2     BUILTIN
; 2436          2475  2         AP;                 ! Argument pointer points to parameter block
; 2437          2476  2
; 2438          2477  2     MAP
; 2439          2478  2         AP: REF BLOCK [, BYTE] FIELD (NMLSFIELDS);
; 2440          2479  2
; 2441          2480  2     AP [TPA$V_BLANKS] = 1;          ! Turn on blank processing
; 2442          2481  2     RETURN 1;
; 2443          2482  2
; 2444          2483  1     END;


                                       0000 00000 BLANKS_ON:
                                                          .WORD   Save nothing                                        ; 2438
                04    AC             01  88 00002         BISB2   #1, 4(AP)                                           ; 2480
                      50             01  D0 00006         MOVL    #1, R0                                              ; 2481
                                         04 00009         RET                                                         ; 2483
```

```
; Routine Size:  10 bytes,    Routine Base:  _FOR$CODE + 0627

; 2445          2484  1 !<BLF/PAGE>
```

```
1 %SBTTL 'LOOKUP_IDENTIFIER - Lookup identifier in NAMELIST group'
1 ROUTINE LOOKUP_IDENTIFIER =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Searches the NAMELIST group for an identifier which matches the
1 !       current token.  If found, the descriptor information is entered into
1 !       the parameter block.  If not found, an error is signalled.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = LOOKUP_IDENTIFIER ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       PARAM_BLOCK [NML$A_VARNAME] = address of variable name counted string
1 !       PARAM_BLOCK [NML$A_VARSTART] = address of variable low byte
1 !       PARAM_BLOCK [NML$A_VAREND] = address of next byte past end of variable
1 !       PARAM_BLOCK [NML$A_VARCUR] = same as VARSTART
1 !       PARAM_BLOCK [NML$W_VARSIZE] = size of a variable element in bytes
1 !       PARAM_BLOCK [NML$W_STRIDE] = stride between elements if array, else 0
1 !       PARAM_BLOCK [NML$B_DTYPE] = descriptor datatype code of variable
1 !       PARAM_BLOCK [NML$B_CONSTYPE] = 0
1 !       PARAM_BLOCK [NML$L_REPEATCT] = 1
1 !       PARAM_BLOCK [NML$V_IMAG] = 0
1 !       PARAM_BLOCK [NML$V_VALUE_IDENT] = 0
1 !       PARAM_BLOCK [NML$V_SUBSTRING] = 0;
1 !       PARAM_BLOCK [NML$V_SUBSCRIPT] = 0;
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 for success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals FOR$_INVREFVAR - Invalid NAMELIST variable if identifier is not in
1 !       the current group.
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2       AP;                 ! Argument pointer points to parameter block
2
2     MAP
2       AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
```

```
  2     LOCAL
  2         NML_LIST: REF VECTOR [, LONG];          ! Pointer to list descriptor
  2
  2     NML_LIST = .AP [NML$A_LISTBLOCK];           ! Get list block address
  2
  2     !+
  2     ! Loop through identifier list looking for a matching identifier.  If none
  2     ! found, signal an error.  Loop value will be true if no match found.
  2     !-
  2
  3     IF (
  3         DECRU I FROM .(NML_LIST [1])<0,16,0> TO 1 DO    ! Count is first word of second longword
  4             BEGIN
  4             NML_LIST = NML_LIST [2];            ! Move to next identifier in list
  4             IF COMPARE_UPCASE (.NML_LIST[0], AP [TPA$L_TOKENCNT])
  4             THEN
  4                 EXITLOOP 0;     ! Loop value false if a match is found
  3             END)
  2     THEN
  3         BEGIN
  3         !+
  3         ! If we get here, there is no match.  Signal an error giving the variable
  3         ! name.
  3         !-
  3
  3         FOR$$SIGNAL_STO (FOR$K_INVREFVAR, AP [TPA$L_TOKENCNT]);
  3
  3         RETURN 0;       ! Execution should never return here
  3         END
  3
  2     ELSE
  2
  3         BEGIN
  3
  3         !+
  3         ! A match has been found.  Fill in the parameter block from the
  3         ! descriptor.
  3         !-
  3
  3         LOCAL
  3             DESC: REF BLOCK [, BYTE];   ! Variable descriptor
  3
  3         AP [NML$A_VARNAME] = .NML_LIST [0];     ! Address of name counted string
  3         DESC = .NML_LIST [1];           ! Descriptor address
  3
  3         !+
  3         ! Validate descriptor class and datatype
  3         ! \\ Note:  The use of the ONE_OF macro here assumes that
  3         !     neither a datatype code of 0 nor a class code
  3         !     of 0 is one of the valid ones.  If this
  3         !     is no longer true, the value must first be tested to
  3         !     ensure that it is not greater than 127 (unsigned). \\
  3         !-
  3
P 3         IF NOT ONE_OF (.DESC [DSC$B_DTYPE],
P 3             DSC$K_DTYPE_BU, DSC$K_DTYPE_B , DSC$K_DTYPE_WU,
P 3             DSC$K_DTYPE_W , DSC$K_DTYPE_LU, DSC$K_DTYPE_L ,
P  3                    DSC$K_DTYPE_T , DSC$K_DTYPE_F , DSC$K_DTYPE_D ,
P  3                    DSC$K_DTYPE_G , DSC$K_DTYPE_H , DSC$K_DTYPE_FC,
   3                    DSC$K_DTYPE_DC, DSC$K_DTYPE_GC) OR
P  3                NOT ONE_OF (.DESC [DSC$B_CLASS],
   4                    DSC$K_CLASS_S, DSC$K_CLASS_A)
   3            THEN
   4                BEGIN
   4                FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
   4                RETURN 0;
   3                END;
   3
   3            !+
   3            ! Fill in parameter block.
   3            !-
   3
   3            AP [NML$A_VARSTART] = .DESC [DSC$A_POINTER];
   3            AP [NML$A_VARCUR] = .DESC [DSC$A_POINTER];
   3            AP [NML$W_VARSIZE] = .DESC [DSC$W_LENGTH];
   3            AP [NML$B_DTYPE] = .DESC [DSC$B_DTYPE];
   3            AP [NML$A_DESCR] = .DESC;
   3
   3            IF .DESC [DSC$B_CLASS] EQL DSC$K_CLASS_A
   3            THEN
   4                BEGIN
   4                !+
   4                ! If the array descriptor doesn't have COLUMN order and
   4                ! coefficient and bounds blocks, or if it has
   4                ! more than 7 dimensions, then the descriptor is
   4                ! invalid for us.
   4                !-
   4
   5                IF NOT (.DESC [DSC$V_FL_COLUMN] AND
   5                        .DESC [DSC$V_FL_COEFF] AND
   5                        .DESC [DSC$V_FL_BOUNDS] AND
   5                        (.DESC [DSC$B_DIMCT] LEQU 7))
   4                THEN
   5                    BEGIN
   5                    FOR$$SIGNAL_STO (FOR$K_INVARGFOR);
   5                    RETURN 0;
   4                    END;
   4
   4                AP [NML$A_VAREND] = .AP [NML$A_VARSTART] + .DESC [DSC$L_ARSIZE];
   4                AP [NML$W_STRIDE] = .AP [NML$W_VARSIZE];
   4                END
   3            ELSE
   4                BEGIN
   4                AP [NML$A_VAREND] = .AP [NML$A_VARSTART] + .AP [NML$W_VARSIZE];
   4                AP [NML$W_STRIDE] = 0;
   3                END;
   3
   3            AP [NML$B_CONSTYPE] = 0;
   3            AP [NML$L_REPEATCT] = 1;
   3            AP [NML$V_IMAG] = 0;
   3            AP [NML$V_VALUE_IDENT] = 0;
   3            AP [NML$V_SUBSCRIPT] = 0;
   3            AP [NML$V_SUBSTRING] = 0;
   3
```

```
; 2618          2656  3           !+
; 2619          2657  3           ! Since FORTRAN insists on passing us datatype BU for a signed byte,
; 2620          2658  3           ! change it here.
; 2621          2659  3           !-
; 2622          2660  3
; 2623          2661  3           IF .AP [NML$B_DTYPE] EQL DSC$K_DTYPE_BU
; 2624          2662  3           THEN
; 2625          2663  3               AP [NML$B_DTYPE] = DSC$K_DTYPE_B;
; 2626          2664  2           END;
; 2627          2665  2
; 2628          2666  2       RETURN 1;    ! Success
; 2629          2667  2
; 2630          2668  1       END;



                                    01FC 00000 LOOKUP_IDENTIFIER:
                                                         .WORD    Save R2,R3,R4,R5,R6,R7,R8                   ; 2486
                   58 00000000G  00  9E 00002            MOVAB    FOR$$SIGNAL_STO, R8                         ;
                   56         24  AC  D0 00009            MOVL     36(AP), NML_LIST                            ; 2545
                   55         10  AC  9E 0000D            MOVAB    16(AP), R5                                  ; 2555
                   57         04  A6  3C 00011            MOVZWL   4(NML_LIST), I                              ;
                                  0E  11 00015            BRB      2$                                          ;
                   56             08  C0 00017 1$:        ADDL2    #8, NML_LIST                                ; 2555
                   54             66  D0 0001A            MOVL     (NML_LIST), R4                              ; 2556
                               0000V  30 0001D            BSBW     COMPARE_UPCASE                              ;
                   0D             50  E8 00020            BLBS     R0, 3$                                      ;
                                  57  D7 00023            DECL     I                                           ; 2553
                                  F0  12 00025 2$:        BNEQ     1$                                          ;
                                  55  DD 00027            PUSHL    R5                                          ; 2567
                                  13  DD 00029            PUSHL    #19                                         ;
                   68             02  FB 0002B            CALLS    #2, FOR$$SIGNAL_STO                         ;
                                  56  11 0002E            BRB      6$                                          ; 2569
              28   AC             66  D0 00030 3$:        MOVL     (NML_LIST), 40(AP)                          ; 2584
                   52         04  A6  D0 00034            MOVL     4(NML_LIST), DESC                           ; 2585
      50 3BBE001C  8F         02  A2  78 00038            ASHL     2(DESC), #1002307612, R0                    ; 2601
                                  3E  18 00041            BGEQ     5$                                          ;
                   01         03  A2  91 00043            CMPB     3(DESC), #1                                 ; 2603
                                  06  13 00047            BEQL     4$                                          ;
                   04         03  A2  91 00049            CMPB     3(DESC), #4                                 ;
                                  32  12 0004D            BNEQ     5$                                          ;
              2C   AC         04  A2  D0 0004F 4$:        MOVL     4(DESC), 44(AP)                             ; 2614
              34   AC         04  A2  D0 00054            MOVL     4(DESC), 52(AP)                             ; 2615
              38   AC             62  B0 00059            MOVW     (DESC), 56(AP)                              ; 2616
              44   AC         02  A2  90 0005D            MOVB     2(DESC), 68(AP)                             ; 2617
              3C   AC             52  D0 00062            MOVL     DESC, 60(AP)                                ; 2618
                   04         03  A2  91 00066            CMPB     3(DESC), #4                                 ; 2620
                                  2A  12 0006A            BNEQ     8$                                          ;
      10        0A  A2            05  E1 0006C            BBC      #5, 10(DESC), 5$                            ; 2630
      0B        0A  A2            06  E1 00071            BBC      #6, 10(DESC), 5$                            ; 2631
                              0A  A2  95 00076            TSTB     10(DESC)                                    ; 2632
                                  06  18 00079            BGEQ     5$                                          ;
                   07         0B  A2  91 0007B            CMPB     11(DESC), #7                                ; 2633
                                  07  1B 0007F            BLEQU    7$                                          ;
                                  30  DD 00081 5$:        PUSHL    #48                                         ; 2636
```

```
                           68          01  FB 00083          CALLS   #1, FOR$$SIGNAL_STO                       :
                                       34  11 00086 6$:      BRB     11$                                       : 2637
          30   AC      2C  AC     0C   A2  C1 00088 7$:      ADDL3   12(DESC), 44(AP), 48(AP)                  : 2640
                       3A  AC     38   AC  B0 0008F          MOVW    56(AP), 58(AP)                            : 2641
                                       0D  11 00094          BRB     9$                                        : 2620
                           50     38   AC  3C 00096 8$:      MOVZWL  56(AP), R0                                : 2645
                       30  AC     2C BC40  9E 0009A          MOVAB   @44(AP)[R0], 48(AP)                       :
                                  3A   AC  B4 000A0          CLRW    58(AP)                                    : 2646
                                  46   AC  94 000A3 9$:      CLRB    70(AP)                                    : 2649
                       78  AC          01  D0 000A6          MOVL    #1, 120(AP)                               : 2650
                       45  AC          0F  8A 000AA          BICB2   #15, 69(AP)                               : 2654
                           02     44   AC  91 000AE          CMPB    68(AP), #2                                : 2661
                                       04  12 000B2          BNEQ    10$                                       :
                       44  AC          06  90 000B4          MOVB    #6, 68(AP)                                : 2663
                           50          01  D0 000B8 10$:     MOVL    #1, R0                                    : 2666
                                       04 000BB              RET                                               :
                                       50  D4 000BC 11$:     CLRL    R0                                        : 2668
                                       04 000BE              RET                                               :

; Routine Size:  191 bytes,    Routine Base:  _FOR$CODE + 0631


; 2631          2669  1 !<BLF/PAGE>
; 2633          2670  1 %SBTTL 'SET_VALUE_IDENT - Mark that last token is supposed to be an identifier'
; 2634          2671  1 ROUTINE SET_VALUE_IDENT =
; 2635          2672  1
; 2636          2673  1 !++
; 2637          2674  1 ! FUNCTIONAL DESCRIPTION:
; 2638          2675  1 !
; 2639          2676  1 !       LIB$TPARSE action routine which is called when the character following
; 2640          2677  1 !       a value token indicates that the last token is supposed to be an
; 2641          2678  1 !       identifier.  It sets a flag in the parameter block which is checked
; 2642          2679  1 !       when the next identifier is needed.
; 2643          2680  1 !
; 2644          2681  1 ! CALLING SEQUENCE:
; 2645          2682  1 !
; 2646          2683  1 !       status = SET_VALUE_IDENT ()
; 2647          2684  1 !
; 2648          2685  1 ! FORMAL PARAMETERS:
; 2649          2686  1 !
; 2650          2687  1 !       NONE
; 2651          2688  1 !
; 2652          2689  1 ! IMPLICIT INPUTS:
; 2653          2690  1 !
; 2654          2691  1 !       AP      Points to PARAM_BLOCK
; 2655          2692  1 !
; 2656          2693  1 ! IMPLICIT OUTPUTS:
; 2657          2694  1 !
; 2658          2695  1 !       NML$V_VALUE_IDENT = 1
; 2659          2696  1 !
; 2660          2697  1 ! COMPLETION STATUS:
; 2661          2698  1 !
; 2662          2699  1 !       1
; 2663          2700  1 !
; 2664          2701  1 ! SIDE EFFECTS:
; 2665          2702  1 !
; 2666          2703  1 !       NONE
; 2667          2704  1 !
; 2668          2705  1 !--
; 2669          2706  1
; 2670          2707  2     BEGIN
; 2671          2708  2
; 2672          2709  2     BUILTIN
; 2673          2710  2         AP;             ! Argument pointer points to parameter block
; 2674          2711  2
; 2675          2712  2     MAP
; 2676          2713  2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
; 2677          2714  2
; 2678          2715  2     AP [NML$V_VALUE_IDENT] = 1;
; 2679          2716  2     RETURN 1;
; 2680          2717  2
; 2681          2718  1     END;


                                    0000 00000 SET_VALUE_IDENT:
                                                      .WORD   Save nothing                        ; 2671
                     45   AC          04  88 00002    BISB2   #4, 69(AP)                          ; 2715
```

```
                        50          01  D0 00006        MOVL    #1, R0                                    ; 2716
                                        04 00009        RET                                               ; 2718

; Routine Size:  10 bytes,    Routine Base:  _FOR$CODE + 06F0


; 2682          2719  1 !<BLF/PAGE>
```

```
1 %SBTTL 'WAS_VALUE_IDENT - Lookup last value as an identifier'
1 ROUTINE WAS_VALUE_IDENT =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which is called when an identifier is needed.
1 !       If NML$V_VALUE_IDENT is 1 then the last value token is supposed to be
1 !       an identifier. Otherwise, 0 is returned.  The last value token, if it
1 !       could possibly be an identifier, was stored in NML$T_TOKEN.  We call
1 !       LOOKUP_IDENTIFIER to see if it is.  If the last token wasn't of type
1 !       REAL or LOGICAL or if the token length is zero, we fail.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = WAS_VALUE_IDENT ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !       NML$V_VALUE_IDENT
1 !       NML$T_TOKEN
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       See LOOKUP_IDENTIFIER
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1 if LOOKUP_IDENTIFIER succeeds
1 !       0 if last token isn't an identifier
1 !
1 ! SIDE EFFECTS:
1 !
1 !       See LOOKUP_IDENTIFIER
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2       AP;             ! Argument pointer points to parameter block
2
2     MAP
2       AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     LOCAL
2       TOKEN: REF VECTOR [, BYTE];
2
2     !+
2     ! If NML$V_VALUE_IDENT is 0, then fail.
2     !-
2
```

```
; 2741          2777  2      IF NOT .AP [NML$V_VALUE_IDENT]
; 2742          2778  2      THEN
; 2743          2779  2          RETURN 0;
; 2744          2780  2
; 2745          2781  2      !+
; 2746          2782  2      ! If last constant type is not REAL or LOGICAL or INTEGER or if token length
; 2747          2783  2      ! is zero, then we have a syntax error.
; 2748          2784  2      !-
; 2749          2785  2
; 2750          2786  2      IF NOT ONE_OF (.AP [NML$B_CONSTYPE], K_REAL, K_LOGICAL, K_INTEGER) OR
; 2751          2787  2          .AP [NML$T_TOKEN] EQL 0
; 2752          2788  2      THEN
; 2753          2789  3          BEGIN
; 2754          2790  3          !+
; 2755          2791  3          ! We reached this state by matching TPA$_LAMBDA just at the delimiter
; 2756          2792  3          ! that caused us to think that the last value token was really an
; 2757          2793  3          ! identifier.  TOKENPTR points to that delimiter and TOKENCNT is 0.
; 2758          2794  3          ! Increment TOKENCNT so that the delimiter will be in the error
; 2759          2795  3          ! message.
; 2760          2796  3          !-
; 2761          2797  3
; 2762          2798  3          AP [TPA$L_TOKENCNT] = .AP [TPA$L_TOKENCNT] + 1;
; 2763          2799  3          CALLG (.AP, SYNTAX_ERROR);
; 2764          2800  2          END;
; 2765          2801  2
; 2766          2802  2
; 2767          2803  2      !+
; 2768          2804  2      ! Construct token from NML$T_TOKEN.
; 2769          2805  2      !-
; 2770          2806  2
; 2771          2807  2      TOKEN = AP [NML$T_TOKEN];
; 2772          2808  2      AP [TPA$L_TOKENCNT] = .TOKEN [0];
; 2773          2809  2      AP [TPA$L_TOKENPTR] = TOKEN [1];
; 2774          2810  2      RETURN CALLG (.AP, LOOKUP_IDENTIFIER);
; 2775          2811  2
; 2776          2812  1      END;



                                      0000 00000 WAS_VALUE_IDENT:
                                                           .WORD   Save nothing                              ; 2721
       2B       45    AC          02   E1 00002            BBC     #2, 69(AP), 3$                            ; 2777
       50 70000000    8F     46   AC   78 00007            ASHL    70(AP), #1879048192, R0                   ; 2786
                                  05   18 00010            BGEQ    1$                                        ;
                             7C   AC   95 00012            TSTB    124(AP)                                   ; 2787
                                  08   12 00015            BNEQ    2$                                        ;
                             10   AC   D6 00017 1$:        INCL    16(AP)                                    ; 2798
             FE53     CF          6C   FA 0001A            CALLG   (AP), SYNTAX_ERROR                        ; 2799
                      50     7C   AC   9E 0001F 2$:        MOVAB   124(AP), TOKEN                            ; 2807
               10     AC          60   9A 00023            MOVZBL  (TOKEN), 16(AP)                           ; 2808
               14     AC     01   A0   9E 00027            MOVAB   1(R0), 20(AP)                             ; 2809
             FF06     CF          6C   FA 0002C            CALLG   (AP), LOOKUP_IDENTIFIER                   ; 2810
                                       04 00031            RET                                               ;
                                  50   D4 00032 3$:        CLRL    R0                                        ; 2812
                                       04 00034            RET                                               ;
```

```
; Routine Size:  53 bytes,    Routine Base:  _FOR$CODE + 06FA

; 2777          2813  1 !<BLF/PAGE>
```

```
1 %SBTTL 'COMPARE_UPCASE - Compare strings upcased'
1 ROUTINE COMPARE_UPCASE (
1                           CSTRING_ADR,
1                           STRING2_DSC
1                           ) : JSB_COMPARE_UPCASE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Compare two strings: the counted string whose address is CSTRING_ADR
1 !       and the string described by the descriptor STRING2_DSC.  The
1 !       STRING2_DSC string is upcased for the comparison; the CSTRING_ADR
1 !       string is assumed to be already upcased.
1 !
1 !       Comparison continues until a non-matching character is found or until
1 !       one of the strings is empty.  No blank-filling is done.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       matches = COMPARE_UPCASE (CSTRING_ADR, STRING2_DSC)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       CSTRING_ADR       - The address of a counted string whose count is in the
1 !                           first byte.  Assumed to be uppercase.
1 !
1 !       STRING2_DSC       - The address of a string descriptor.  This string will
1 !                           be forced to upper case during the comparison.  The
1 !                           string itself is not modified.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! FUNCTION VALUE:
1 !
1 !       1 if the strings are equal
1 !       0 otherwise
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     MAP
2         CSTRING_ADR: REF VECTOR [, BYTE],
2         STRING2_DSC: REF BLOCK [, BYTE];
2
2     LOCAL
2         STRING2_ADR: REF VECTOR [, BYTE],
; 2836          2871  2         STRING1_LEN: WORD;
; 2837          2872  2
; 2838          2873  2     !+
; 2839          2874  2     ! Compare string lengths.  If they don't match, return failure.
; 2840          2875  2     !-
; 2841          2876  2
; 2842          2877  2     STRING1_LEN = .CSTRING_ADR [0];
; 2843          2878  2     IF .STRING1_LEN NEQU .STRING2_DSC [DSC$W_LENGTH]
; 2844          2879  2     THEN
; 2845          2880  2         RETURN 0;
; 2846          2881  2
; 2847          2882  2     !+
; 2848          2883  2     ! Compare strings for equality.  Lengths must match.
; 2849          2884  2     !-
; 2850          2885  2
; 2851          2886  2     STRING2_ADR = .STRING2_DSC [DSC$A_POINTER];
; 2852          2887  2     INCRU I FROM 1 TO .STRING1_LEN DO
; 2853          2888  3         BEGIN
; 2854          2889  3         IF .CSTRING_ADR [.I] NEQU
; 2855          2890  4             (
; 2856          2891  4             IF .STRING2_ADR [0] GEQU %C'a' AND .STRING2_ADR [0] LEQU %C'z'
; 2857          2892  4             THEN
; 2858          2893  5                 CH$RCHAR_A (STRING2_ADR) - (%C'a' - %C'A')
; 2859          2894  4             ELSE
; 2860          2895  4                 CH$RCHAR_A (STRING2_ADR)
; 2861          2896  4             )
; 2862          2897  3         THEN
; 2863          2898  3             RETURN 0;    ! Unequal character found
; 2864          2899  2         END;
; 2865          2900  2
; 2866          2901  2     !+
; 2867          2902  2     ! If we get here, then the match is successful.
; 2868          2903  2     !-
; 2869          2904  2
; 2870          2905  2     RETURN 1;
; 2871          2906  2
; 2872          2907  1     END;


                     50           64  9B 00000 COMPARE_UPCASE:
                                                        MOVZBW  (CSTRING_ADR), STRING1_LEN                  ; 2877
                     53           50  3C 00003          MOVZWL  STRING1_LEN, R3                             ; 2878
                     53           65  B1 00006          CMPW    (STRING2_DSC), R3                           ;
                                  33  12 00009          BNEQ    5$                                          ;
                     51      04   A5  D0 0000B          MOVL    4(STRING2_DSC), STRING2_ADR                 ; 2886
                     52           01  D0 0000F          MOVL    #1, I                                       ; 2887
                                  21  11 00012          BRB     4$                                          ;
                61   8F           61  91 00014 1$:      CMPB    (STRING2_ADR), #97                          ; 2891
                                  0E  1F 00018          BLSSU   2$                                          ;
                7A   8F           61  91 0001A          CMPB    (STRING2_ADR), #122                         ;
                                  08  1A 0001E          BGTRU   2$                                          ;
                     50           81  9A 00020          MOVZBL  (STRING2_ADR)+, R0                          ; 2893
                     50           20  C2 00023          SUBL2   #32, R0                                     ;
                                  03  11 00026          BRB     3$                                          ;
```

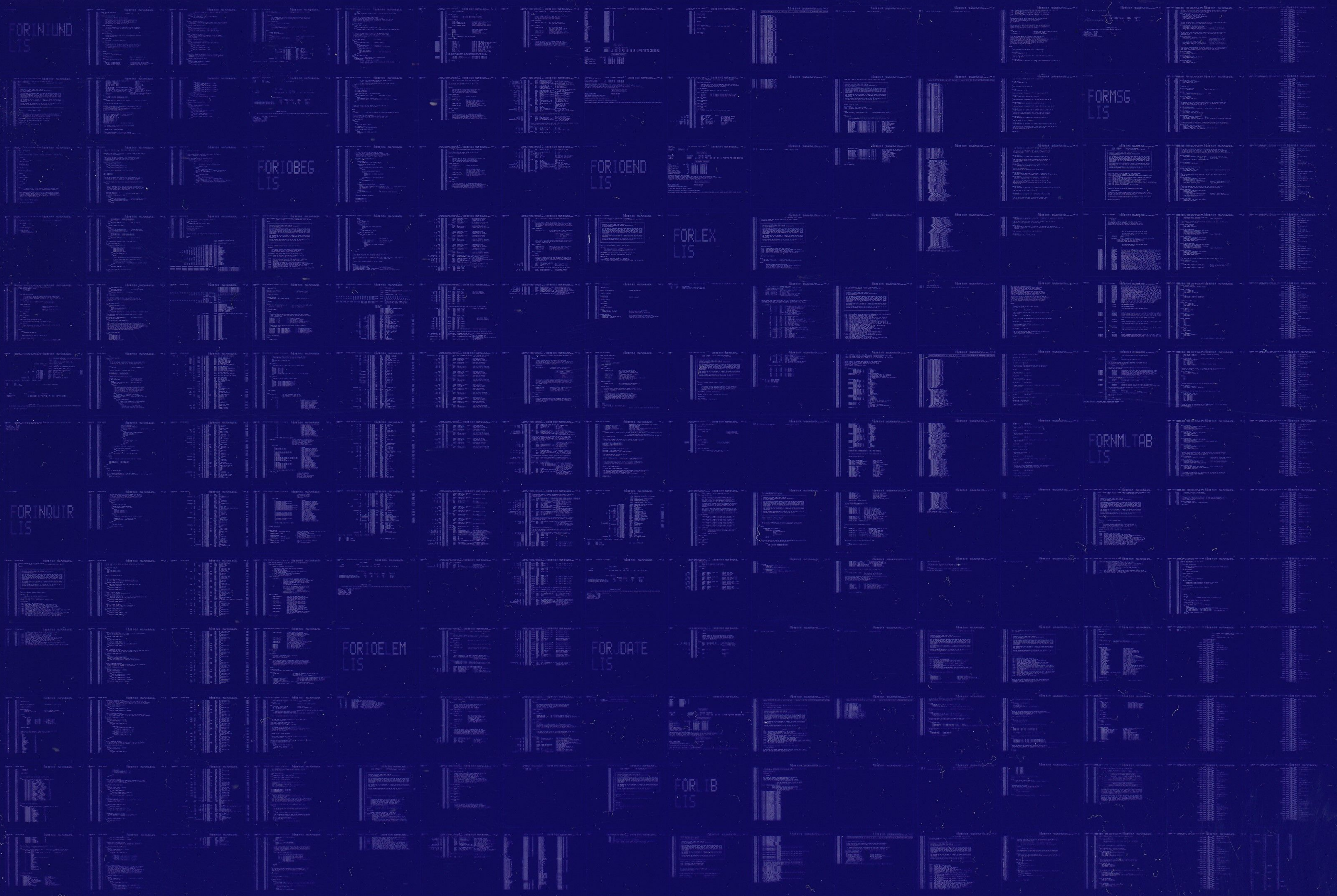
```
                                   50          81  9A 00028 2$:      MOVZBL  (STRING2_ADR)+, R0                    ; 2895
        50           6244          08          00  ED 0002B 3$:      CMPZV   #0, #8, (I)[CSTRING_ADR], R0          ; 2890
                                               0B  12 00031          BNEQ    5$                                    ;
                                               52  D6 00033          INCL    I                                     ; 2887
                                   53          52  D1 00035 4$:      CMPL    I, R3                                 ;
                                               DA  1B 00038          BLEQU   1$                                    ;
                                   50          01  D0 0003A          MOVL    #1, R0                                ; 2905
                                                   05 0003D          RSB                                           ;
                                               50  D4 0003E 5$:      CLRL    R0                                    ; 2907
                                                   05 00040          RSB                                           ;

; Routine Size:  65 bytes,    Routine Base:  _FOR$CODE + 072F
```

```
1 %SBTTL 'DUMP_NAMES - Respond to "?" inquiry'
1 ROUTINE DUMP_NAMES =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which is called when '?' is seen
1 !       in place of a variable.  If this file is a terminal on which we
1 !       have PUT access, call FOR$$DO_NML_OUTPUT to dump the group name
1 !       and variable names in the current namelist group.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = DUMP_NAMES ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May list namelist group on terminal.
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;               ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     GLOBAL REGISTER
2         CCB = 11: REF $FOR$CCB_DECL;
2
2     CCB = .AP [NML$A_CCB];      ! Load CCB register
2
2     !+
2     ! If we are on a terminal with PUT access, list the namelist group.
2     !-
2
3         BEGIN
3         BIND
3             FAB = CCB: REF $FOR$FAB_CCB_STRUCT,
; 2931          2965  3            FAB_DEV = FAB [FAB$L_DEV]: BLOCK [4, BYTE];
; 2932          2966  3
; 2933          2967  3        IF .FAB_DEV [DEV$V_TRM] AND .FAB [FAB$V_PUT]
; 2934          2968  3        THEN
; 2935          2969  4            BEGIN
; 2936          2970  4            FOR$$REC_WSNO ();   ! Start output record
; 2937          2971  4            FOR$$DO_NML_OUTPUT (1);      ! Dump names only
; 2938          2972  3            END;
; 2939          2973  2        END;
; 2940          2974  2
; 2941          2975  2    RETURN 1;
; 2942          2976  2
; 2943          2977  1    END;



                                        083C 00000 DUMP_NAMES:
                                                         .WORD   Save R2,R3,R4,R5,R11                    ; 2909
                         5B        40   AC  D0 00002     MOVL    64(AP), CCB                             ; 2956
                         50      0084   CB  9E 00006     MOVAB   132(R11), R0                            ; 2965
                 13      60             02  E1 0000B     BBC     #2, (R0), 1$                            ; 2967
                         0F        5A   AB  E9 0000F     BLBC    90(FAB), 1$                             ;
                            00000000G   00  16 00013     JSB     FOR$$REC_WSNO                           ; 2970
                                        01  DD 00019     PUSHL   #1                                      ; 2971
                  00000000G  00         01  FB 0001B     CALLS   #1, FOR$$DO_NML_OUTPUT                  ;
                             50         01  D0 00022 1$: MOVL    #1, R0                                  ; 2975
                                            04 00025     RET                                             ; 2977

; Routine Size:  38 bytes.    Routine Base:  _FOR$CODE + 0770
```

```
1 %SBTTL 'DUMP_VALUES - Respond to "=?" inquiry'
1 ROUTINE DUMP_VALUES =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       LIB$TPARSE action routine which is called when '=?' is seen
1 !       in place of a variable.  If this file is a terminal on which we
1 !       have PUT access, call FOR$$DO_NML_OUTPUT to dump the group name
1 !       and variable names and values in the current namelist group.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       status = DUMP_VALUES ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP      Points to PARAM_BLOCK
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       1
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May list namelist group on terminal.
1 !
1 !--
1
2     BEGIN
2
2     BUILTIN
2         AP;                 ! Argument pointer points to parameter block
2
2     MAP
2         AP: REF BLOCK [, BYTE] FIELD (NML$FIELDS);
2
2     GLOBAL REGISTER
2         CCB = 11: REF $FOR$CCB_DECL;
2
2     CCB = .AP [NML$A_CCB];       ! Load CCB register
2
2     !+
2     ! If we are on a terminal with PUT access, list the namelist group.
2     !-
2
3         BEGIN
3         BIND
3             FAB = CCB: REF $FOR$FAB_CCB_STRUCT,
; 3002          3035  3               FAB_DEV = FAB [FAB$L_DEV]: BLOCK [4, BYTE];
; 3003          3036  3
; 3004          3037  3           IF .FAB_DEV [DEV$V_TRM] AND .FAB [FAB$V_PUT]
; 3005          3038  3           THEN
; 3006          3039  4               BEGIN
; 3007          3040  4               FOR$$REC_WSNO ();    ! Start output record
; 3008          3041  4               FOR$$DO_NML_OUTPUT (0);       ! Dump names and values
; 3009          3042  3               END;
; 3010          3043  2           END;
; 3011          3044  2
; 3012          3045  2       RETURN 1;
; 3013          3046  2
; 3014          3047  1       END;




                                                  083C 00000 DUMP_VALUES:
                                                                   .WORD   Save R2,R3,R4,R5,R11                          ; 2979
                            5B        40   AC  D0 00002           MOVL    64(AP), CCB                                    ; 3026
                            50      0084   CB  9E 00006           MOVAB   132(R11), R0                                   ; 3035
                 13         60             02  E1 0000B           BBC     #2, (R0), 1$                                   ; 3037
                            0F        5A   AB  E9 0000F           BLBC    90(FAB), 1$                                    ;
                               00000000G   00  16 00013           JSB     FOR$$REC_WSNO                                  ; 3040
                                           7E  D4 00019           CLRL    -(SP)                                          ; 3041
                 00000000G  00             01  FB 0001B           CALLS   #1, FOR$$DO_NML_OUTPUT                         ;
                            50             01  D0 00022 1$:       MOVL    #1, R0                                         ; 3045
                                               04 00025           RET                                                    ; 3047

; Routine Size:  38 bytes,    Routine Base:  _FOR$CODE + 0796
; 3016          3048  1 END                                              ! End of module FOR$$NML_TABLES
; 3017          3049  1
; 3018          3050  0 ELUDOM



;                               PSECT SUMMARY
;
;      Name                          Bytes                    Attributes
;
;  _LIB$KEY0$                            0  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  _LIB$STATE$                        1050  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  _FOR$CODE                          1980  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        37         0        581        00:01.1
;  _$255$DUA28:[FORRTL.OBJ]FORLIB.L32;1          711       216        30         52        00:00.5
;  _$255$DUA28:[FORRTL.OBJ]RTLLIB.L32;1           36         0         0          8        00:00.1
;  _$255$DUA28:[SYSLIB]TPAMAC.L32;1               42        27        64         14        00:00.1



;                               COMMAND QUALIFIERS

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

; Size:          1980 code + 1050 data bytes
; Run Time:         01:58.7
; Elapsed Time:     04:10.6
; Lines/CPU Min:     1541
; Lexemes/CPU-Min: 73012
; Memory Used:  417 pages
; Compilation Complete
```

0181 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0182 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY