```
VVV         VVV  MMM         MMM     SSSSSSSSSSSSS LLL                 IIIIIIIII     BBBBBBBBBBBB
VVV         VVV  MMM         MMM     SSSSSSSSSSSSS LLL                 IIIIIIIII     BBBBBBBBBBBB
VVV         VVV  MMM         MMM     SSSSSSSSSSSSS LLL                 IIIIIIIII     BBBBBBBBBBBB
VVV         VVV  MMMMMM   MMMMMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMMMMM   MMMMMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMMMMM   MMMMMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMM   MMM   MMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMM   MMM   MMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMM   MMM   MMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMM         MMM     SSSSSSSSS     LLL                    III        BBBBBBBBBBBB
VVV         VVV  MMM         MMM     SSSSSSSSS     LLL                    III        BBBBBBBBBBBB
VVV         VVV  MMM         MMM     SSSSSSSSS     LLL                    III        BBBBBBBBBBBB
VVV         VVV  MMM         MMM              SSS  LLL                    III        BBB         BBB
VVV         VVV  MMM         MMM              SSS  LLL                    III        BBB         BBB
VVV         VVV  MMM         MMM              SSS  LLL                    III        BBB         BBB
   VVV   VVV     MMM         MMM              SSS  LLL                    III        BBB         BBB
   VVV   VVV     MMM         MMM              SSS  LLL                    III        BBB         BBB
   VVV   VVV     MMM         MMM              SSS  LLL                    III        BBB         BBB
      VVV        MMM         MMM  SSSSSSSSSSSSS    LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB
      VVV        MMM         MMM  SSSSSSSSSSSSS    LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB
      VVV        MMM         MMM  SSSSSSSSSSSSS    LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB
```

```
**FILE**ID**TPARSE                                                    H 16




TTTTTTTTTT  PPPPPPPP      AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
TTTTTTTTTT  PPPPPPPP      AAAAAA    RRRRRRRR      SSSSSSSS  EEEEEEEEEE
    TT      PP      PP  AA      AA  RR      RR  SS          EE
    TT      PP      PP  AA      AA  RR      RR  SS          EE
    TT      PP      PP  AA      AA  RR      RR  SS          EE
    TT      PP      PP  AA      AA  RR      RR  SS          EE
    TT      PPPPPPPP    AA      AA  RRRRRRRR      SSSSSS    EEEEEEEE
    TT      PPPPPPPP    AA      AA  RRRRRRRR      SSSSSS    EEEEEEEE
    TT      PP          AAAAAAAAAA  RR  RR            SS  EE
    TT      PP          AAAAAAAAAA  RR  RR            SS  EE
    TT      PP          AA      AA  RR    RR          SS  EE              ....
    TT      FP          AA      AA  RR    RR          SS  EE              ....
    TT      PP          AA      AA  RR      RR  SSSSSSSS    EEEEEEEEEE      ....
    TT      PP          AA      AA  RR      RR  SSSSSSSS    EEEEEEEEEE      ....


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

```
0 MODULE LIB$TPARSE (
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1
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  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:  System Library
1 !
1 ! ABSTRACT:
1 !
1 !       TPARSE is a general purpose state table driven parser. Its
1 !       general design is that of a finite state parser; however,
1 !       some of its features allow non-deterministic parsing and
1 !       limited use as a push-down parser. The input string is parsed
1 !       by interpreting the transitions in the user suppled state
1 !       table; user supplied action routines are called as indicated
1 !       in the state table to provide the semantics associated with
1 !       the table specified syntax.
1 !
1 ! ENVIRONMENT:
1 !
1 !       Native node star processor; no operating system facilities are
1 !       needed. Minimum of 21 longwords of stack required.
1 !
1 !--
1 !
1 !
1 ! AUTHOR:  Andrew C. Goldstein, CREATION DATE:  14-Oct-1976  13:55
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-004 ACG0396         Andrew C. Goldstein,    31-Jan-1984  22:10
```

```
1 !                 Add multi-national characters to alphabetics; add a
1 !                 character data type for hex digits
1 !
1 !         V03-003 ACG0392         Andrew C. Goldstein,    19-Jan-1984  21:32
1 !                 Add match-all identifier to ident parsing; also add
1 !                 TPA$_FILESPEC item.
1 !
1 !         V03-002 ACG0351         Andrew C. Goldstein,    22-Aug-1983  15:42
1 !                 Add wild cards to UIC and IDENT parsing; disallow explicit
1 !                 -1 values for group and member
1 !
1 !         V03-001 ACG0345         Andrew C. Goldstein,    29-Jul-1983  15:20
1 !                 Add UIC and identifier token types
1 !
1 !         V02-009 ACG0263         Andrew C. Goldstein,    8-Feb-1982  23:35
1 !                 Fix token pointer after subexpression parse failure
1 !
1 !         V02-008 ACG0203         Andrew C. Goldstein,    28-Apr-1981  15:44
1 !                 Propagate return status past subexpression calls
1 !
1 !         V02-007 ACG0155         Andrew C. Goldstein,    12-Mar-1980  10:04
1 !                 Allow subexpressions to change the command string
1 !
1 !         1-006   ACG26405        Andrew C. Goldstein,    6-Dec-1979  20:14
1 !                 Change string count in control block to word
1 !
1 !         1-005   ACG0088         Andrew C. Goldstein,    4-Dec-1979  17:22
1 !                 Propagate action routine status through subexpression returns
1 !
1 !         V0004   ACG0019         Andrew C. Goldstein,  26-Mar-1978  16:27
1 !         Add return of action routine status; fix trailing blanks proolem
1 !         Change PSECTS to new procedure library standards
1 !
1 !**
1
1
1 LIBRARY 'SYS$LIBRARY:STARLET.L32';
1 LIBRARY 'SYS$LIBRARY:TPAMAC.L32';
1
1
1
1 PSECT
1         CODE            = _LIB$CODE (SHARE, PIC),
1         PLIT            = _LIB$CODE (SHARE, PIC);
1
1 LINKAGE
1         L_GETSTRING     = CALL (STANDARD, STANDARD)
1                         : GLOBAL (CHAR_COUNT = 6);
1
1 FORWARD ROUTINE
1         LIB$TPARSE,                     ! main parser routine
1         GETSTRING       : L_GETSTRING,  ! extract a basic string token
1         PARSE_FILESPEC,                 ! parse filespec string
1         PARSE_IDENT;                    ! parse identifier string
1
1
1 EXTERNAL ROUTINE
```

```
;  115        0115  1        LIB$CVT_DTB      : ADDRESSING_MODE (GENERAL),     ! decimal to binary conversion
;  116        0116  1        LIB$CVT_OTB      : ADDRESSING_MODE (GENERAL),     ! octal to binary conversion
;  117        0117  1        LIB$CVT_HTB      : ADDRESSING_MODE (GENERAL);     ! hex to binary conversion
```

```
  1 !
  1 ! Local macros and symbol definitions
  1 !
  1
  1 LITERAL
  1
  1 ! Character codes
  1 !
  1       SPACE           = %O'40',        ! space character
  1       TERMINATOR      = %O'377',       ! keyword string terminator
  1       TAB             = %O'11',        ! tab character
  1
  1 ! String types (input to GETSTRING routine)
  1 !
  1       SPACES          = 0,
  1       NUMERIC         = 1,
  1       HEXNUMERIC      = 2,
  1       ALPHANUMERIC    = 3,
  1       SYMBOL          = 4,
  1
  1 ! Token types
  1 !
  1       $FILESPEC       = %X'FF' AND TPA$_FILESPEC,     ! filespec expression
  1       $UIC            = %X'FF' AND TPA$_UIC,          ! UIC expression
  1       $IDENT          = %X'FF' AND TPA$_IDENT,        ! identifier expression
  1       $ANY            = %X'FF' AND TPA$_ANY,          ! any single character
  1       $ALPHA          = %X'FF' AND TPA$_ALPHA,        ! any alphabetic character
  1       $DIGIT          = %X'FF' AND TPA$_DIGIT,        ! any numeric character
  1       $STRING         = %X'FF' AND TPA$_STRING,       ! any alphanumeric string
  1       $SYMBOL         = %X'FF' AND TPA$_SYMBOL,       ! any symbol constituent set string
  1       $BLANK          = %X'FF' AND TPA$_BLANK,        ! any string of spaces and tabs
  1       $DECIMAL        = %X'FF' AND TPA$_DECIMAL,      ! decimal number
  1       $OCTAL          = %X'FF' AND TPA$_OCTAL,        ! octal number
  1       $HEX            = %X'FF' AND TPA$_HEX,          ! hexadecimal number
  1       $LAMBDA         = %X'FF' AND TPA$_LAMBDA,       ! empty string
  1       $EOS            = %X'FF' AND TPA$_EOS,          ! end of string
  1       $SUBEXPR        = %X'FF' AND TPA$_SUBEXPR,      ! subexpression
  1       NULL_MATCH      = TPA$_LAMBDA,  ! codes geq match null strings
  1       HIGH_ASCII      = 255,          ! highest ASCII character code
  1       KEYWORD         = 256,          ! start of keyword codes
  1       HIGH_KEYWORD    = 475,          ! highest keyword code
  1       LOW_SPECIAL     = $FILESPEC,    ! first special type code
  1       HIGH_SPECIAL    = $SUBEXPR;     ! last special type code
  1
  1 ! Macros to determine character types
  1 !
  1
  1 MACRO
M 1       IS_ALPHABETIC (CHAR) = (SELECTONE CHAR OF
M 1                               SET
M 1                               [%ASCII 'A' TO %ASCII 'Z',
M 1                                %ASCII 'a' TO %ASCII 'z',
M 1                                %X'C0' TO %X'CF',
M 1                                %X'D1' TO %X'DD',
M 1                                %X'DF' TO %X'EF',
M 1                                %X'F1' TO %X'FD']: 1;
M 1                               [OTHERWISE]: 0;
```

```
                                 TES
                                )%,

        IS_NUMERIC (CHAR)     - (SELECTONE CHAR OF
                                 SET
                                 [%ASCII '0' TO %ASCII '9']: 1,
                                 [OTHERWISE]: 0;
                                 TES
                                )%,

        IS_HEX (CHAR)         = (SELECTONE CHAR OF
                                 SET
                                 [%ASCII '0' TO %ASCII '9',
                                  %ASCII 'A' TO %ASCII 'F',
                                  %ASCII 'a' TO %ASCII 'f']: 1;
                                 [OTHERWISE]: 0;
                                 TES
                                )%,

        IS_SYMBOL (CHAR)       = (SELECTONE CHAR OF
                                 SET
                                 [%ASCII '_', %ASCII '$']: 1;
                                 [OTHERWISE]: 0;
                                 TES
                                )%,

        IS_SPACE (CHAR)       = (SELECTONE CHAR OF
                                 SET
                                 [SPACE, TAB]: 1;
                                 [OTHERWISE]: 0;
                                 TES
                                )%;

! Macros to fetch state table entries, for both data items and self
! relative addresses.
!

MACRO
        GET_BYTE (POINTER) =
                (POINTER = .POINTER + 1; .(.POINTER-1)<0,8>)
                %,

        GET_WORD (POINTER) =
                (POINTER = .POINTER + 2; .(.POINTER-2)<0,16>)
                %,

        GET_LONG (POINTER) =
                (POINTER = .POINTER + 4; .(.POINTER-4)<0,32>)
                %,

        REL_WORD (POINTER) =
                (POINTER = .POINTER + 2; .POINTER + .(.POINTER-2)<0,16,1>)
                %,

        REL_LONG (POINTER) =
                (POINTER = .POINTER + 4; .POINTER + .(.POINTER-4)<0,32,1>)
                %;
```

```

! Structure definition for the transition length table. It consists of 64
! entries, indexed by the 6 transition option bits. Each entry is a 4 bit
! nibble giving the number of words that are used by that transition
! option mask.
!

STRUCTURE
        NIBBLEVECTOR [I; N] =
                [(N+1)/2]
                NIBBLEVECTOR<I*4, 4, 0>;

! Macro to generate nibbles in a PLIT expression.
!

MACRO
        NIBBLE [A, B] =
                BYTE (B^4 + A)
                %;

! The transition length table.
!

BIND
        LENGTH_TABLE    = UPLIT (NIBBLE (0,1,1,2,2,3,3,4,2,3,3,4,4,5,5,6,
                                         2,3,3,4,4,5,5,6,4,5,5,6,6,7,7,8,
                                         2,3,3,4,4,5,5,6,4,5,5,6,6,7,7,8,
                                         4,5,5,6,6,7,7,8,6,7,7,8,8,9,9,10
                                         ))
                        : NIBBLEVECTOR;
```

```
1 GLOBAL ROUTINE LIB$TPARSE (STATE_VECTOR, START_STATE, KEYTAB) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is the main parser routine.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       LIB$TPARSE (ARG1, ARG2, ARG3)
1 !
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1 = address of state vector, containing:
1 !               options longword
1 !                   bit 0 set to match blanks and tabs
1 !                         clear to ignore blanks and tabs
1 !                   bit 1 set to allow minimum abbreviation of keywords
1 !                         clear to use match count
1 !                   high byte = keyword match count (0 = exact)
1 !               string descriptor of string to be parsed
1 !               data made available to action routines:
1 !                   string descriptor of matching token
1 !                   single character token
1 !                   numerical value of numeric token
1 !       ARG2 = address of starting state in state table
1 !       ARG3 = address of keyword table
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       string descriptor pointed to by ARG1 updated to indicate
1 !       string not processed by the parser
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       1 if successful parse
1 !       LIB$_SYNTAXERR if unsuccessful parse
1 !       LIB$_INVTYPE if state table is invalid
1 !       value of action routine rejecting transition
1 !
1 ! SIDE EFFECTS:
1 !       none except as produced by user's action routines
1 !
1 !--
1
1
2 BEGIN
2
2 LOCAL
2       STATE,                          ! state table pointer
2       LAST_SPACE,                     ! character count of last string of spaces
2       ACT_STATUS      : BLOCK [4, BYTE], ! status return of action routine
```

```
 2              TYPE            : BLOCK [4, BYTE]; ! syntax type of current transition
 2
 2
 2      MAP
 2              STATE_VECTOR    : REF BLOCK [,BYTE], ! user state vector
 2              KEYTAB  : REF VECTOR[,WORD,SIGNED];  ! keyword table
 2
 2      BUILTIN
 2              TESTBITSC;
 2
 2
 2      ! TPARSE data area available to action routines.
 2      !
 2      MACRO
 2              STATE_LENGTH    = STATE_VECTOR[TPA$L_COUNT]%,    ! length of user supplied state vector
 2              STRING_COUNT    = (STATE_VECTOR[TPA$L_STRINGCNT])<0,16>%, ! byte count of string being parsed
 2              STRING_POINTER  = STATE_VECTOR[TPA$L_STRINGPTR]%, ! address of string being parsed
 2              TOKEN_COUNT     = STATE_VECTOR[TPA$L_TOKENCNT]%, ! byte count of current token
 2              TOKEN_POINTER   = STATE_VECTOR[TPA$L_TOKENPTR]%, ! address of current token
 2              STATE_CHAR      = STATE_VECTOR[TPA$L_CHAR]%,     ! current single character token
 2              STATE_NUMBER    = STATE_VECTOR[TPA$L_NUMBER]%,   ! numerical value of number token
 2              STATE_PARAM     = STATE_VECTOR[TPA$L_PARAM]%,    ! action routine parameter from state table
 2              MCOUNT          = STATE_VECTOR[TPA$B_MCOUNT]%,   ! match abbreviation count
 2              SPACE_FLAG      = STATE_VECTOR[TPA$V_BLANKS]%,   ! process spaces explicitly
 2              ABBRFM_FLAG     = STATE_VECTOR[TPA$V_ABBRFM]%,   ! allow first match abbreviations
 2              ABBREV_FLAG     = STATE_VECTOR[TPA$V_ABBREV]%,   ! allow minimal abbreviations
 2              AMBIG_FLAG      = STATE_VECTOR[TPA$V_AMBIG]%;    ! ambiguous keyword in this state
 2
 2
 2      !
 2      ! Contents of the type byte - code and flags.
 2      !
 2      MACRO
 2              TYPECODE        = 0,0,9,0%,     ! full token type code
 2              TYPEBYTE        = 0,0,8,0%,     ! token type byte
 2              OPTION_BITS     = 0,11,6,0%,    ! transition option flags
 2              CODEFLAG        = 0,8,1,0%,     ! transition is special code
 2              EXTRAFLAG       = 0,9,1,0%,     ! extra type byte present
 2              LASTFLAG        = 0,10,1,0%,    ! last transition in state
 2              EXTFLAG         = 0,11,1,0%,    ! type extension present
 2              TRANFLAG        = 0,12,1,0%,    ! transition target present
 2              MASKFLAG        = 0,13,1,0%,    ! bitmask present
 2              ADDRFLAG        = 0,14,1,0%,    ! data address present
 2              ACTFLAG         = 0,15,1,0%,    ! action routine present
 2              PARMFLAG        = 0,16,1,0%;    ! parameter longword present
 2
 2      GLOBAL REGISTER
 2              CHAR_COUNT      = 6;            ! character count in string token
 2
 2      EXTERNAL LITERAL
 2              LIB$_SYNTAXERR,                 ! syntax error status
 2              LIB$_INVTYPE;                   ! invalid state table
 2
 2
 2      !+
 2      !
 2      ! Entry initialization consists of loading the starting state
```

```
2 ! into the state pointer. Then we proceed into the main
2 ! loop that attempts to match transitions in the state table to
2 ! the current string contents.
2 !
2 !-
2
2
2 IF .STATE_LENGTH LSSU TPA$K_COUNT0 THEN RETURN SS$_INSFARG; ! check minimum length of state vector
2
2 AMBIG_FLAG = 0;
2 ACT_STATUS = 0;
2 STATE = .START_STATE;
2
2 WHILE 1 DO
3 BEGIN
3
3 ! The following horrendous expression attempts to match the token type
3 ! of the current transition to the current string position. The outer
3 ! IF is a condition which, if true, causes the current transition
3 ! (just evaluated) to be skipped by the code at the end of this loop. The
3 ! inner IF causes the action routine of the transition to be called if
3 ! true (if the transition matches).
3 !
3 IF
4     BEGIN
4     IF
5         BEGIN
5         TYPE = GET_WORD (STATE);         ! get basic type code
5         IF .TYPE[EXTRAFLAG]              ! and extra byte if present
5         THEN TYPE<16,8> = GET_BYTE (STATE);
5
5         LAST_SPACE = 0;
5
5         IF NOT .SPACE_FLAG AND .TYPE[TYPECODE] NEQ TPA$_LAMBDA
5         THEN
6             BEGIN
6             GETSTRING (STRING_COUNT, SPACES);
6             LAST_SPACE = .CHAR_COUNT;
6             STRING_COUNT = .STRING_COUNT - .CHAR_COUNT;  ! update string pointer
6             STRING_POINTER = .STRING_POINTER + .CHAR_COUNT;
5             END;
5
5         CHAR_COUNT = 0;                  ! init matching string descriptor
5         TOKEN_POINTER = .STRING_POINTER;
5
5         IF .TYPE[TYPECODE] LSSU NULL_MATCH AND .STRING_COUNT EQL 0
5         THEN 0                     ! no match if at end
5         ELSE
5
5             SELECTONEU .TYPE[TYPECODE] OF
5             SET
5
5 ! Single characters are matched by token types whose numerical value is
5 ! the ASCII code of the character.
5 !
5
```

```
5             [0 TO HIGH_ASCII]:  ! single ASCII character
5                 IF .TYPE[TYPEBYTE] EQL CH$RCHAR (.STRING_POINTER)
5                 THEN
6                     BEGIN
6                     STATE_CHAR = CH$RCHAR(.STRING_POINTER);
6                     CHAR_COUNT = 1
6                     END
5                 ELSE 0;
5
5 ! Keyworus are matched by token types whose bits 0-6 contain the keyword
5 ! number. A keyword token may be either (1) an exact match or
5 ! (2) abbreviated to some number of characters fixed for the call or
5 ! (3) arbitrarily abbreviated (such that the first match wins) or
5 ! (4) abbreviated to the minimum which is locally unambiguous. Condition
5 ! (4) is tested for ambiguity by finding the next entry in the keyword table
5 ! and matching it against the token string. The keyword strings for each
5 ! state are padded with a filler to prevent false matches across states.
5 !
5
5
5             [KEYWORD TO HIGH_KEYWORD]:  ! keyword match
5                 IF NOT .AMBIG_FLAG
5                 AND GETSTRING (STRING_COUNT, SYMBOL)
5                 THEN
6                     BEGIN
6                     LOCAL KEY;
6                     KEY = .KEYTAB + .KEYTAB[.TYPE[TYPEBYTE]];
6                     IF CH$EQL (.CHAR_COUNT, .TOKEN_POINTER,
6                         .CHAR_COUNT, .KEY, 0)
6                     THEN
7                         BEGIN
8                         IF (CH$RCHAR (.KEY + .CHAR_COUNT) EQL TERMINATOR
9                             OR (.MCOUNT NEQ 0 AND .CHAR_COUNT GEQU .MCOUNT)
8                             )
7                         THEN 1
7                         ELSE IF .ABBRFM_FLAG
7                         THEN 1
7                         ELSE IF .ABBREV_FLAG
7                         THEN
8                             BEGIN
8                             KEY = 1 + CH$FIND_CH (65535, .KEY+.CHAR_COUNT, TERMINATOR);
8                             IF CH$NEQ (.CHAR_COUNT, .TOKEN_POINTER,
8                                         .CHAR_COUNT, .KEY, 0)
8                             THEN 1
8                             ELSE
9                                 BEGIN
9                                 AMBIG_FLAG = 1;
9                                 0
9                                 END
8                             END
7                         ELSE 0
7                         END
6                     ELSE 0
6                     END
5                 ELSE 0;
5
5 ! All other token types are special cases, representing commonly occurring
```

```
5 ! composites and other useful artifacts.
5 !
5
5               [OTHERWISE]:                    ! all other types
5                   CASE .TYPE[TYPEBYTE] FROM LOW_SPECIAL TO HIGH_SPECIAL OF
5                   SET
5
5                   [$LAMBDA]:                  ! empty string
5                       1;
5
5                   [$EOS]:                     ! end of input
5                       .STRING_COUNT EQL 0;
5
5                   [$ANY]:                     ! any single character
6                       BEGIN
6                       STATE_CHAR = CH$RCHAR (.STRING_POINTER);
6                       CHAR_COUNT = 1
5                       END;
5
5                   [$ALPHA]:                   ! alphabetic
6                       IF IS_ALPHABETIC (CH$RCHAR (.STRING_POINTER))
6                       THEN (STATE_CHAR = CH$RCHAR (.STRING_POINTER);
6                             CHAR_COUNT = 1)
5                       ELSE 0;
5
5                   [$DIGIT]:                   ! single digit
6                       IF IS_NUMERIC (CH$RCHAR (.STRING_POINTER))
6                       THEN (STATE_CHAR = CH$RCHAR (.STRING_POINTER);
6                             CHAR_COUNT = 1)
5                       ELSE 0;
5
5                   [$STRING]:                  ! alphanumeric string
5                       GETSTRING (STRING_COUNT, ALPHANUMERIC);
5
5                   [$SYMBOL]:                  ! symbol constituent set string
5                       GETSTRING (STRING_COUNT, SYMBOL);
5
5                   [$BLANK]:                   ! blanks or tabs
5                       GETSTRING (STRING_COUNT, SPACES);
5
5                   [$DECIMAL]:                 ! decimal number
5                       IF GETSTRING (STRING_COUNT, NUMERIC)
5                       THEN LIB$CVT_DTB (.CHAR_COUNT, .TOKEN_POINTER, STATE_NUMBER)
5                       ELSE 0;
5
5                   [$OCTAL]:                   ! octal number
5                       IF GETSTRING (STRING_COUNT, NUMERIC)
5                       THEN LIB$CVT_OTB (.CHAR_COUNT, .TOKEN_POINTER, STATE_NUMBER)
5                       ELSE 0;
5
5                   [$HEX]:                     ! hexa-decimal number
5                       IF GETSTRING (STRING_COUNT, HEXNUMERIC)
5                       THEN LIB$CVT_HTB (.CHAR_COUNT, .TOKEN_POINTER, STATE_NUMBER)
5                       ELSE 0;
5
5                   [$FILESPEC, $UIC, $IDENT, $SUBEXPR]: ! complex expressions
6                       BEGIN
```

```
;  549         0547  6                     LOCAL STATUS, SAVECOUNT, SAVEPOINTER;
;  550         0548  6                     SAVECOUNT = .STRING_COUNT;  ! save current position
;  551         0549  6                     SAVEPOINTER = .STRING_POINTER;
;  552         0550  7                     STATUS = (SELECTONEU .TYPE[TYPEBYTE] OF
;  553         0551  7                         SET
;  554         0552  7                         [$FILESPEC]:
;  555         0553  7                             PARSE_FILESPEC (.STATE_VECTOR);
;  556         0554  7                         [$UIC]:
;  557         0555  7                             PARSE_IDENT (.STATE_VECTOR, 0);
;  558         0556  7                         [$IDENT]:
;  559         0557  7                             PARSE_IDENT (.STATE_VECTOR, 1);
;  560         0558  7                         [$SUBEXPR]:
;  561         0559  7                             LIB$TPARSE (.STATE_VECTOR,
;  562         0560  7                                          .STATE + 2 + .(.STATE)<0,16,1>,
;  563         0561  7                                          .KEYTAB);
;  564         0562  6                         TES);
;  565         0563  6                     IF .STATUS NEQ LIB$_SYNTAXERR
;  566         0564  6                     THEN ACT_STATUS = .STATUS;
;  567         0565  6                     TOKEN_POINTER = .SAVEPOINTER;
;  568         0566  6                     CHAR_COUNT = .STRING_POINTER - .SAVEPOINTER;
;  569         0567  6                     IF .STATUS
;  570         0568  6                     THEN
;  571         0569  7                         BEGIN
;  572         0570  7                         STRING_COUNT = .STRING_COUNT + .CHAR_COUNT;
;  573         0571  7                         STRING_POINTER = .SAVEPOINTER;
;  574         0572  7                         1
;  575         0573  7                         END
;  576         0574  6                     ELSE
;  577         0575  7                         BEGIN
;  578         0576  7                         STRING_COUNT = .SAVECOUNT;
;  579         0577  7                         STRING_POINTER = .SAVEPOINTER;
;  580         0578  7                         0
;  581         0579  7                         END
;  582         0580  5                     END;
;  583         0581  5
;  584         0582  5                 [INRANGE]: RETURN LIB$_INVTYPE; ! just for completeness
;  585         0583  5
;  586         0584  5                 [OUTRANGE]: RETURN LIB$_INVTYPE;
;  587         0585  5
;  588         0586  5                 TES;                        ! end of special types case
;  589         0587  5
;  590         0588  5             TES                             ! end of select on .TYPE
;  591         0589  5         END                                 ! end of inner condition (token match)
;  592         0590  5
;  593         0591  5 !+
;  594         0592  5 !
;  595         0593  5 ! The type code in this transition matches the current string,
;  596         0594  5 ! which is now described by the global string descriptor. Call
;  597         0595  5 ! the user's action routine, if it exists, and if it returns true,
;  598         0596  5 ! gobble the string and take the transition. Note that we set R0
;  599         0597  5 ! to 1 going into the call, making it easier for the routine to
;  600         0598  5 ! return success.
;  601         0599  5 !
;  602         0600  5 !-
;  603         0601  5
;  604         0602  4     THEN
;  605         0603  5         BEGIN
```

```
5              STRING_COUNT = .STRING_COUNT - .CHAR_COUNT;  ! update string pointer
5              STRING_POINTER = .STRING_POINTER + .CHAR_COUNT;
5              TOKEN_COUNT = .CHAR_COUNT;
5                                               ! skip extension if present
5              IF TESTBITSC (TYPE[EXTFLAG]) THEN STATE = .STATE + 2;
5
5              IF TESTBITSC (TYPE[PARMFLAG])    ! set parameter longword if present
5              THEN STATE_PARAM = GET_LONG (STATE);
5
5              IF
6                  BEGIN
6                  IF TESTBITSC (TYPE[ACTFLAG])
6                  THEN
7                      BEGIN
7                      BUILTIN R0, CALLG;
7                      LOCAL STATUS;
7                      R0 = 1;
7                      STATUS = CALLG (.STATE_VECTOR, REL_LONG (STATE));  ! call action routine
7                      IF .STATUS NEQ 0
7                      THEN ACT_STATUS = .STATUS;
7                      .STATUS
7                      END
6                  ELSE 1
6                  END
6
6  ! Either there was no action routine, or the action routine has returned
6  ! success; we take the transition. First we get the data address, if present.
6  ! If present, store whatever data is called for: the mask, if supplied, or
6  ! type dependent data if not - either the matching character, the number
6  ! value, or the string descriptor of the matching string.
6  !
6
5              THEN
6                  BEGIN
6                  IF .TYPE[ADDRFLAG]
6                  THEN
7                      BEGIN
7                      LOCAL ADDRESS;
7                      ADDRESS = REL_LONG (STATE);
7                      IF .TYPE[MASKFLAG]
7                      THEN
8                          .ADDRESS = ..ADDRESS OR GET_LONG (STATE)
7                      ELSE
8                          BEGIN
8                          IF NOT .TYPE[CODEFLAG]
8                          THEN
8                              (.ADDRESS)<0,8> = .STATE_CHAR
8                          ELSE
9                              BEGIN
9                              CASE .TYPE[TYPEBYTE] FROM LOW_SPECIAL TO HIGH_SPECIAL OF
9                              SET
9
9                              [$ANY, $ALPHA, $DIGIT]:
9                                      (.ADDRESS)<0,8> = .STATE_CHAR;
9
9                              [$DECIMAL, $OCTAL, $HEX, $UIC, $IDENT]:
9                                      .ADDRESS = .STATE_NUMBER;
```

```
 9
 9                         [INRANGE, OUTRANGE]:
10                                 BEGIN
10                                 (.ADDRESS) = .TOKEN_COUNT;
10                                 (.ADDRESS+4) = .TOKEN_POINTER;
 9                                 END;
 9
 9                         TES;
 8                         END;
 7                     END;
 6                 END;
 6
 6 ! Take the transition. If an explicit target exists, follow it. -1 means
 6 ! exit with success, -2 means exit with failure. If no explicit target exists,
 6 ! skip transitions until we reach the beginning of the next state.
 6 !
 6
 6             AMBIG_FLAG = 0;
 6             ACT_STATUS = 0;
 6             IF .TYPE[TRANFLAG]
 6             THEN                            ! take the transition
 6                 IF .(.STATE)<0,16,1> EQL TPA$_EXIT  ! TPA$_EXIT means exit
 6                 THEN RETURN 1
 6                 ELSE IF .(.STATE)<0,16,1> EQL TPA$_FAIL  ! TPA$_FAIL means exit
 6                 THEN RETURN LIB$_SYNTAXERR
 6
 7                 ELSE STATE = REL_WORD (STATE)
 6             ELSE                            ! default to next state
 6                 UNTIL .TYPE[LASTFLAG] DO
 7                     BEGIN
 7                     TYPE = GET_WORD (STATE);
 7                     IF .TYPE[EXTRAFLAG]
 7                     THEN TYPE<16,8> = GET_BYTE (STATE);
 7                                             ! skip optional components
 7                     STATE = .STATE + 2 * .LENGTH_TABLE[.TYPE[OPTION_BITS]];
 6                     END;
 6             0
 6             END
 6
 6 ! The action routine has rejected the transition. Make like it never matched.
 6 !
 6
 5         ELSE
 6             BEGIN                           ! return characters to string
 6             STRING_COUNT = .STRING_COUNT + .CHAR_COUNT;
 6             STRING_POINTER = .STRING_POINTER - .CHAR_COUNT;
 6                                             ! skip the rest of the transition
 6             1
 6             END
 5         END
 5
 4     ELSE
 4         1
 4     END                                     ! end of outer condition
 4
 4 !+
 4 !
```

```
;  720          0718  4 ! If the transition does not match, we execute this code. It skips
;  721          0719  4 ! the current transition to set up to try the next one in the state.
;  722          0720  4 ! If this was the last transition in the state, a syntax error has
;  723          0721  4 ! occurred and TPARSE returns the value 0.
;  724          0722  4 !
;  725          0723  4 !-
;  726          0724  4
;  727          0725  3 THEN
;  728          0726  4     BEGIN
;  729          0727  4     STATE = .STATE + 2 * .LENGTH_TABLE[.TYPE[OPTION_BITS]];
;  730          0728  4     IF .TYPE[LASTFLAG]
;  731          0729  4     THEN
;  732          0730  5         BEGIN
;  733          0731  5         GETSTRING (STRING_COUNT, SPACES);
;  734          0732  5         GETSTRING (STRING_COUNT, SYMBOL);
;  735          0733  5         TOKEN_COUNT = .CHAR_COUNT;
;  736          0734  5         IF .TOKEN_COUNT EQL 0 AND .STRING_COUNT NEQ 0
;  737          0735  5         THEN TOKEN_COUNT = .TOKEN_COUNT + 1;
;  738          0736  6         RETURN (
;  739          0737  6             IF .ACT_STATUS[STS$V_COND_ID] NEQ 0
;  740          0738  6             THEN .ACT_STATUS
;  741          0739  5             ELSE LIB$_SYNTAXERR);
;  742          0740  4         END;
;  743          0741  4
;  744          0742  4     STRING_COUNT = .STRING_COUNT + .LAST_SPACE; ! return flushed spaces
;  745          0743  4     STRING_POINTER = .STRING_POINTER - .LAST_SPACE;
;  746          0744  4     LAST_SPACE = 0;
;  747          0745  4     END
;  748          0746  4
;  749          0747  3 END                                        ! end of outside loop
;  750          0748  3
;  751          0749  1 END;                                       ! end of routine TPARSE


                                                               .TITLE  LIB$TPARSE
                                                               .IDENT  \V04-000\

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

                                            10  00000 P.AAA:   .BYTE   16
                                            21  00001          .BYTE   33
                                            32  00002          .BYTE   50
                                            43  00003          .BYTE   67
                                            32  00004          .BYTE   50
                                            43  00005          .BYTE   67
                                            54  00006          .BYTE   84
                                            65  00007          .BYTE   101
                                            32  00008          .BYTE   50
                                            43  00009          .BYTE   67
                                            54  0000A          .BYTE   84
                                            65  0000B          .BYTE   101
                                            54  0000C          .BYTE   84
                                            65  0000D          .BYTE   101
                                            76  0000E          .BYTE   118
                                            87  0000F          .BYTE   -121
                                            32  00010          .BYTE   50
                                            43  00011          .BYTE   67
```

```
                                          54  00012            .BYTE    84                                  ;
                                          65  00013            .BYTE    101                                 ;
                                          54  00014            .BYTE    84                                  ;
                                          65  00015            .BYTE    101                                 ;
                                          76  00016            .BYTE    118                                 ;
                                          87  00017            .BYTE    -121                                ;
                                          54  00018            .BYTE    84                                  ;
                                          65  00019            .BYTE    101                                 ;
                                          76  0001A            .BYTE    118                                 ;
                                          87  0001B            .BYTE    -121                                ;
                                          76  0001C            .BYTE    118                                 ;
                                          87  0001D            .BYTE    -121                                ;
                                          98  0001E            .BYTE    -104                                ;
                                          A9  0001F            .BYTE    -87                                 ;

                                                     LENGTH_TABLE=        P.AAA
                                                               .EXTRN   LIB$CVT_DTB, LIB$CVT_OTB
                                                               .EXTRN   LIB$CVT_HTB, LIB$_SYNTAXERR
                                                               .EXTRN   LIB$_INVTYPE

                                        0FFC  00000            .ENTRY   LIB$TPARSE, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 0262
                                                                        R10,R11                             ;
                         5E         08  C2  00002              SUBL2    #8, SP                              ;
                         50    04   AC  D0  00005              MOVL     STATE_VECTOR, R0                    ; 0383
                         08         60  D1  00009              CMPL     (R0), #8                            ;
                                    06  1E  0000C              BGEQU    1$                                  ;
                         50  0114   8F  3C  0000E              MOVZWL   #276, R0                            ;
                                        04  00013              RET                                          ;
                    06   A0         01  8A  00014  1$:         BICB2    #1, 6(R0)                           ; 0385
                                    55  D4  00018              CLRL     ACT_STATUS                          ; 0386
                         58    08   AC  D0  0001A              MOVL     START_STATE, STATE                  ; 0387
                         54         88  3C  0001E  2$:         MOVZWL   (STATE)+, TYPE                      ; 0404
               08        54         09  E1  00021              BBC      #9, TYPE, 3$                        ; 0405
                                    58  D6  00025              INCL     STATE                               ; 0406
      54       08        10    FF   A8  F0  00027              INSV     -1(STATE), #16, #8, TYPE            ;
                               04   AE  D4  0002D  3$:         CLRL     LAST_SPACE                          ; 0408
                         57    04   AC  D0  00030              MOVL     STATE_VECTOR, R7                    ; 0410
                         6E    04   A7  9E  00034              MOVAB    4(R7), (SP)                         ;
                         21    00   BE  E8  00038              BLBS     @0(SP), 4$                          ;
000001F6  8F   54        09         00  ED  0003C              CMPZV    #0, #9, TYPE, #502                  ;
                                    16  13  00045              BEQL     4$                                  ;
                                    7E  D4  00047              CLRL     -(SP)                               ; 0413
                               08   A7  9F  00049              PUSHAB   8(R7)                               ;
                  0000V  CF         02  FB  0004C              CALLS    #2, GETSTRING                       ;
                     04  AE         56  D0  00051              MOVL     CHAR_COUNT, LAST_SPACE              ; 0414
                     08  A7         56  A2  00055              SUBW2    CHAR_COUNT, 8(R7)                   ; 0415
                     0C  A7         56  C0  00059              ADDL2    CHAR_COUNT, 12(R7)                  ; 0416
                                    56  D4  0005D  4$:         CLRL     CHAR_COUNT                          ; 0419
                         5B    14   A7  9E  0005F              MOVAB    20(R7), R11                         ;
                         59    0C   A7  9E  00063              MOVAB    12(R7), R9                          ; 0420
                         6B         69  D0  00067              MOVL     (R9), (R11)                         ;
000001F6  8F   54        09         00  ED  0006A              CMPZV    #0, #9, TYPE, #502                  ; 0422
                                    08  1E  00073              BGEQU    5$                                  ;
                               08   A7  B5  00075              TSTW     8(R7)                               ;
                                    03  12  00078              BNEQ     5$                                  ;
                                  02EF  31  0007A              BRW      64$                                 ;
000000FF  8F   54        09         00  ED  0007D  5$:         CMPZV    #0, #9, TYPE, #255                  ; 0433
```

```
                                                09  1A 00086          BGTRU   6$                                          ;
                             00    B9           54  91 00088          CMPB    TYPE, @0(R9)                                ; 0434
                                                79  12 0008C          BNEQ    9$                                          ;
                                              00AB  31 0008E          BRW     14$                                         ; 0437
00000100    8F            54           09       00  ED 00091 6$:      CMPZV   #0, #9, TYPE, #256                          ; 0453
                                                6E  1F 0009A          BLSSU   10$                                         ;
000001DB    8F            54           09       00  ED 0009C          CMPZV   #0, #9, TYPE, #475                          ;
                                                63  1A 000A5          BGTRU   10$                                         ;
                          5B     00    BE       10  E0 000A7          BBS     #16, @0(SP), 9$                             ; 0454
                                                04  DD 000AC          PUSHL   #4                                          ; 0455
                                          08    A7  9F 000AE          PUSHAB  8(R7)                                       ;
                              0000V    CF       02  FB 000B1          CALLS   #2, GETSTRING                               ;
                                       4E       50  E9 000B6          BLBC    R0, 9$                                      ;
                                       50       54  9A 000B9          MOVZBL  TYPE, R0                                    ; 0459
                                       5A    0C BC40 32 000BC         CVTWL   @KEYTAB[R0], KEY                            ;
                                       5A    0C   AC  C0 000C1        ADDL2   KEYTAB, KEY                                 ;
                          6A     00    BB       56  29 000C5          CMPC3   CHAR_COUNT, @0(R11), (KEY)                  ; 0460
                                                3B  12 000CA          BNEQ    9$                                          ;
                                 FF    8F     664A  91 000CC          CMPB    (CHAR_COUNT)[KEY], #255                     ; 0464
                                                67  13 000D1          BEQL    13$                                         ;
                                          07    A7  95 000D3          TSTB    7(R7)                                       ; 0465
                                                08  13 000D6          BEQL    7$                                          ;
            56       07   A7           08       00  ED 000D8          CMPZV   #0, #8, 7(R7), CHAR_COUNT                   ;
                                                5A  1B 000DE          BLEQU   13$                                         ;
                          55     00    BE       02  E0 000E0 7$:      BBS     #2, @0(SP), 13$                             ; 0468
                          1D     00    BE       01  E1 000E5          BBC     #1, @0(SP), 9$                              ; 0470
                        664A   FFFF    8F   FF  8F  3A 000EA          LOCC    #255, #65535, (CHAR_COUNT)[KEY]             ; 0473
                                                02  12 000F2          BNEQ    8$                                          ;
                                                51  D4 000F4          CLRL    R1                                          ;
                                       5A   01  A1  9E 000F6 8$:      MOVAB   1(R1), KEY                                  ;
                          6A     00    BB       56  29 000FA          CMPC3   CHAR_COUNT, @0(R11), (KEY)                  ; 0474
                                                39  12 000FF          BNEQ    13$                                         ;
      00    BE            01           10       01  F0 00101          INSV    #1, #16, #1, @0(SP)                         ; 0479
                                              0262  31 00107 9$:      BRW     64$                                         ;
                          0E     EA    8F       54  8F 0010A 10$:     CASEB   TYPE, #234, #14                             ; 0494
        002D            0108         0108     0108     0010F 11$:     .WORD   38$-11$,-                                   ;
        009E            009A         0080     0034     00117                  38$-11$,-                                   ;
        00E8            00CB         00AE     00A2     0011F                  38$-11$,-                                   ;
                        0108         0026     017E     00127                  14$-11$,-                                   ;
                                                                              15$-11$,-                                   ;
                                                                              21$-11$,-                                   ;
                                                                              29$-11$,-                                   ;
                                                                              30$-11$,-                                   ;
                                                                              31$-11$,-                                   ;
                                                                              33$-11$,-                                   ;
                                                                              34$-11$,-                                   ;
                                                                              35$-11$,-                                   ;
                                                                              48$-11$,-                                   ;
                                                                              12$-11$,-                                   ;
                                                                              38$-11$                                     ;
                                 50 00000000G   8F  D0 0012D          MOVL    #LIB$_INVTYPE, R0                           ; 0584
                                                04 00134              RET                                                 ;
                                          08    A7  B5 00135 12$:     TSTW    8(R7)                                       ; 0501
                                                CD  12 00138          BNEQ    9$                                          ;
                                                6B  11 0013A 13$:     BRB     28$                                         ;
                                 18    A7   00  B9  9A 0013C 14$:     MOVZBL  @0(R9), 24(R7)                              ; 0505
                                                61  11 00141          BRB     27$                                         ; 0506
            50     00   B9  9A 00143 15$:    MOVZBL  @0(R9), R0                          ; 0510
      41    8F          50  91 00147         CMPB    R0, #65                             ;
                        06  1F 0014B         BLSSU   16$                                 ;
      5A    8F          50  91 0014D         CMPB    R0, #90                             ;
                        4D  1B 00151         BLEQU   26$                                 ;
      61    8F          50  91 00153 16$:    CMPB    R0, #97                             ;
                        06  1F 00157         BLSSU   17$                                 ;
      7A    8F          50  91 00159         CMPB    R0, #122                            ;
                        41  1B 0015D         BLEQU   26$                                 ;
      C0    8F          50  91 0015F 17$:    CMPB    R0, #192                            ;
                        06  1F 00163         BLSSU   18$                                 ;
      CF    8F          50  91 00165         CMPB    R0, #207                            ;
                        35  1B 00169         BLEQU   26$                                 ;
      D1    8F          50  91 0016B 18$:    CMPB    R0, #209                            ;
                        06  1F 0016F         BLSSU   19$                                 ;
      DD    8F          50  91 00171         CMPB    R0, #221                            ;
                        29  1B 00175         BLEQU   26$                                 ;
      DF    8F          50  91 00177 19$:    CMPB    R0, #223                            ;
                        06  1F 0017B         BLSSU   20$                                 ;
      EF    8F          50  91 0017D         CMPB    R0, #239                            ;
                        1D  1B 00181         BLEQU   26$                                 ;
      F1    8F          50  91 00183 20$:    CMPB    R0, #241                            ;
                        0D  1F 00187         BLSSU   22$                                 ;
      FD    8F          50  91 00189         CMPB    R0, #253                            ;
                        0F  11 0018D         BRB     25$                                 ;
            50     00   B9  9A 0018F 21$:    MOVZBL  @0(R9), R0                          ; 0516
            30          50  91 00193         CMPB    R0, #48                             ;
                        03  1E 00196 22$:    BGEQU   24$                                 ;
                      01D1  31 00198 23$:    BRW     64$                                 ;
            39          50  91 0019B 24$:    CMPB    R0, #57                             ;
                        F8  1A 0019E 25$:    BGTRU   23$                                 ;
      18    A7          50  D0 001A0 26$:    MOVL    R0, 24(R7)                          ; 0517
            56          01  D0 001A4 27$:    MOVL    #1, CHAR_COUNT                      ; 0518
                        6C  11 001A7 28$:    BRB     37$                                 ;
                        03  DD 001A9 29$:    PUSHL   #3                                  ; 0522
                        06  11 001AB         BRB     32$                                 ;
                        04  DD 001AD 30$:    PUSHL   #4                                  ; 0525
                        02  11 001AF         BRB     32$                                 ;
                        7E  D4 001B1 31$:    CLRL    -(SP)                               ; 0528
                   08   A7  9F 001B3 32$:    PUSHAB  8(R7)                               ;
   0000V    CF          02  FB 001B6         CALLS   #2, GETSTRING                       ;
                        55  11 001BB         BRB     36$                                 ;
                        01  DD 001BD 33$:    PUSHL   #1                                  ; 0531
                   08   A7  9F 001BF         PUSHAB  8(R7)                               ;
   0000V    CF          02  FB 001C2         CALLS   #2, GETSTRING                       ;
            CE          50  E9 001C7         BLBC    R0, 23$                             ;
                   1C   A7  9F 001CA         PUSHAB  28(R7)                              ; 0532
                        6B  DD 001CD         PUSHL   (R11)                               ;
                        56  DD 001CF         PUSHL   CHAR_COUNT                          ;
00000000G   00          03  FB 001D1         CALLS   #3, LIB$CVT_DTB                     ;
                        38  11 001D8         BRB     36$                                 ;
                        01  DD 001DA 34$:    PUSHL   #1                                  ; 0536
                   08   A7  9F 001DC         PUSHAB  8(R7)                               ;
   0000V    CF          02  FB 001DF         CALLS   #2, GETSTRING                       ;
            B1          50  E9 001E4         BLBC    R0, 23$                             ;
                   1C   A7  9F 001E7         PUSHAB  28(R7)                              ; 0537
                        6B  DD 001EA         PUSHL   (R11)                               ;
```

```
LIB$TPARSE                                     B  2
V04-000                               16-Sep-1984 02:20:18    VAX-11 Bliss-32 V4.0-742            Page  19
                                      14-Sep-1984 13:35:03    [VMSLIB.SRC]TPARSE.B32,1                 (3)

                                      56  DD 001EC          PUSHL   CHAR_COUNT                          ;
           00000000G  00              03  FB 001EE          CALLS   #3, [LIB$CVT_OTB                    ;
                                      1B  11 001F5          BRB     36$                                 ;
                                      02  DD 001F7 35$:     PUSHL   #2                                  ; 0541
                                08    A7  9F 001F9          PUSHAB  8(R7)                               ;
                0000V CF              02  FB 001FC          CALLS   #2, GETSTRING                       ;
                      94              50  E9 00201          BLBC    R0, 23$                             ;
                                1C    A7  9F 00204          PUSHAB  28(R7)                              ; 0542
                                      6B  DD 00207          PUSHL   (R11)                               ;
                                      56  DD 00209          PUSHL   CHAR_COUNT                          ;
           00000000G  00              03  FB 0020B          CALLS   #3, [LIB$CVT_HTB                    ;
                      75              50  E9 00212 36$:     BLBC    R0, 47$                             ;
                                      76  11 00215 37$:     BRB     48$                                 ;
                      53        08    A7  3C 00217 38$:     MOVZWL  8(R7), SAVECOUNT                    ; 0548
                      52              69  D0 0021B          MOVL    (R9), SAVEPOINTER                   ; 0549
                   EA 8F              54  91 0021E          CMPB    TYPE, #234                          ; 0552
                                      09  12 00222          BNEQ    39$                                 ;
                                      57  DD 00224          PUSHL   R7                                  ; 0553
                0000V CF              01  FB 00226          CALLS   #1, PARSE_FILESPEC                  ;
                                      37  11 0022B          BRB     44$                                 ;
                   EB 8F              54  91 0022D 39$:     CMPB    TYPE, #235                          ; 0554
                                      04  12 00231          BNEQ    40$                                 ;
                                      7E  D4 00233          CLRL    -(SP)                               ; 0555
                                      08  11 00235          BRB     41$                                 ;
                   EC 8F              54  91 00237 40$:     CMPB    TYPE, #236                          ; 0556
                                      0B  12 0023B          BNEQ    42$                                 ;
                                      01  DD 0023D          PUSHL   #1                                  ; 0557
                                      57  DD 0023F 41$:     PUSHL   R7                                  ;
                0000V CF              02  FB 00241          CALLS   #2, PARSE_IDENT                     ;
                                      1C  11 00246          BRB     44$                                 ;
                   F8 8F              54  91 00248 42$:     CMPB    TYPE, #248                          ; 0558
                                      05  13 0024C          BEQL    43$                                 ;
                      50              01  CE 0024E          MNEGL   #1, STATUS                          ;
                                      11  11 00251          BRB     44$                                 ;
                                0C    AC  DD 00253 43$:     PUSHL   KEYTAB                              ; 0561
                      51              68  32 00256          CVTWL   (STATE), R1                         ; 0560
                                02 A148   9F 00259          PUSHAB  2(R1)[STATE]                        ;
                                      57  DD 0025D          PUSHL   R7                                  ; 0559
                 FD9C CF              03  FB 0025F          CALLS   #3, LIB$TPARSE                      ;
           00000000G  8F              50  D1 00264 44$:     CMPL    STATUS, #LIB$_SYNTAXERR             ; 0563
                                      03  13 0026B          BEQL    45$                                 ;
                      55              50  D0 0026D          MOVL    STATUS, ACT_STATUS                  ; 0564
                      6B              52  D0 00270 45$:     MOVL    SAVEPOINTER, (R11)                  ; 0565
     56               69              52  C3 00273          SUBL3   SAVEPOINTER, (R9), CHAR_COUNT       ; 0566
                      09              50  E9 00277          BLBC    STATUS, 46$                         ; 0567
                   08 A7              56  A0 0027A          ADDW2   CHAR_COUNT, 8(R7)                   ; 0570
                      69              52  D0 0027E          MOVL    SAVEPOINTER, (R9)                   ; 0571
                                      0A  11 00281          BRB     48$                                 ;
                   08 A7              53  B0 00283 46$:     MOVW    SAVECOUNT, 8(R7)                    ; 0576
                      69              52  D0 00287          MOVL    SAVEPOINTER, (R9)                   ; 0577
                                    00DF  31 0028A 47$:     BRW     64$                                 ;
                   08 A7              56  A2 0028D 48$:     SUBW2   CHAR_COUNT, 8(R7)                   ; 0604
                      69              56  C0 00291          ADDL2   CHAR_COUNT, (R9)                    ; 0605
                   10 A7              56  D0 00294          MOVL    CHAR_COUNT, 16(R7)                  ; 0606
     03               54              0B  E5 00298          BBCC    #11, TYPE, 49$                      ; 0608
                      58              02  C0 0029C          ADDL2   #2, STATE                           ;
     04               54              10  E5 0029F 49$:     BBCC    #16, TYPE, 50$                      ; 0610
```

```
                          20     A7          88  D0 002A3        MOVL    (STATE)+, 32(R7)                  : 0611
               18                54          0F  E5 002A7 50$:   BBCC    #15, TYPE, 52$                    : 0615
                                 50          01  D0 002AB        MOVL    #1, R0                            : 0620
                                 58          04  C0 002AE        ADDL2   #4, STATE                         : 0621
                           FC  B848          67  FA 002B1        CALLG   (R7), @-4(STATE)[STATE]           :
                                             50  D5 002B6        TSTL    STATUS                            : 0622
                                             03  13 002B8        BEQL    51$                               :
                                 55          50  D0 002BA        MOVL    STATUS, ACT_STATUS                : 0623
                                 03          50  E8 002BD 51$:   BLBS    STATUS, 52$                       : 0624
                                           009D  31 002C0        BRW     63$                               :
               48                54          0E  E1 002C3 52$:   BBC     #14, TYPE, 58$                    : 0638
               51                58          88  C1 002C7        ADDL3   (STATE)+, STATE, ADDRESS          : 0642
               05                54          0D  E1 002CB        BBC     #13, TYPE, 53$                    : 0643
                                 61          88  C8 002CF        BISL2   (STATE)+, (ADDRESS)               : 0645
                                             3B  11 002D2        BRB     58$                               :
                                 50      04  AC  D0 002D4 53$:   MOVL    STATE_VECTOR, R0                  : 0650
               29                54          08  E1 002D8        BBC     #8, TYPE, 56$                     : 0648
               0E          EA    8F          54  8F 002DC        CASEB   TYPE, #234, #14                   : 0653
     0024    002A              002A        001E     002E1 54$:   .WORD   55$-54$,-                         :
     001E    001E              0024        0024     002E9                57$-54$,-                         :
     002A    002A              002A        001E     002F1                57$-54$,-                         :
             001E              001E        001E     002F9                56$-54$,-                         :
                                                                         56$-54$,-                         :
                                                                         56$-54$,-                         :
                                                                         55$-54$,-                         :
                                                                         55$-54$,-                         :
                                                                         55$-54$,-                         :
                                                                         57$-54$,-                         :
                                                                         57$-54$,-                         :
                                                                         57$-54$,-                         :
                                                                         55$-54$,-                         :
                                                                         55$-54$,-                         :
                                                                         55$-54$                           :
                                 61      10  A0  7D 002FF 55$:   MOVQ    16(R0), (ADDRESS)                 : 0664
                                             0A  11 00303        BRB     58$                               : 0653
                                 61      18  A0  90 00305 56$:   MOVB    24(R0), (ADDRESS)                 : 0657
                                             04  11 00309        BRB     58$                               :
                                 61      1C  A0  D0 0030B 57$:   MOVL    28(R0), (ADDRESS)                 : 0660
                                 50      04  AC  D0 0030F 58$:   MOVL    STATE_VECTOR, R0                  : 0671
                          06     A0          01  8A 00313        BICB2   #1, 6(R0)                         : 0678
                                             55  D4 00317        CLRL    ACT_STATUS                        : 0679
               1B                54          0C  E1 00319        BBC     #12, TYPE, 61$                    : 0680
                        FFFF     8F          68  B1 0031D        CMPW    (STATE), #-1                      : 0682
                                             04  12 00322        BNEQ    59$                               :
                                 50          01  D0 00324        MOVL    #1, R0                            : 0683
                                                 04 00327        RET                                       :
                        FFFE     8F          68  B1 00328 59$:   CMPW    (STATE), #-2                      : 0684
                                             7F  13 0032D        BEQL    66$                               :
                                 50          88  32 0032F        CVTWL   (STATE)+, R0                      : 0687
                                 58          50  C0 00332        ADDL2   R0, STATE                         :
                                           FCE6  31 00335 60$:   BRW     2$                                : 0682
               F9                54          0A  E0 00338 61$:   BBS     #10, TYPE, 60$                    : 0689
                                 54          88  3C 0033C        MOVZWL  (STATE)+, TYPE                    : 0691
               08                54          09  E1 0033F        BBC     #9, TYPE, 62$                     : 0692
                                             58  D6 00343        INCL    STATE                             : 0693
       54      08                10      FF  A8  F0 00345        INSV    -1(STATE), #16, #8, TYPE          :
       51      54                06          0B  EF 0034B 62$:   EXTZV   #11, #6, TYPE, R1                 : 0695
```

```
                                    51          04  C4 00350           MULL2    #4, R1                              ;
         50      FC87   CF          04          51  EF 00353           EXTZV    R1, #4, LENGTH_TABLE, R0            ;
                                    58        6840  3E 0035A           MOVAW    (STATE)[R0], STATE                  ;
                                                D8  11 0035E           BRB      61$                                 ; 0689
                                    50      04  AC  D0 00360 63$:      MOVL     STATE_VECTOR, R0                    ; 0704
                             08     A0          56  A0 00364           ADDW2    CHAR_COUNT, 8(R0)                   ; 0705
                             0C     A0          56  C2 00368           SUBL2    CHAR_COUNT, 12(R0)                  ; 0706
         51             54          06          0B  EF 0036C 64$:      EXTZV    #11, #6, TYPE, R1                   ; 0727
                                    51          04  C4 00371           MULL2    #4, R1                              ;
         50      FC66   CF          04          51  EF 00374           EXTZV    R1, #4, LENGTH_TABLE, R0            ;
                                    58        6840  3E 0037B           MOVAW    (STATE)[R0], STATE                  ;
                        39          54          0A  E1 0037F           BBC      #10, TYPE, 68$                      ; 0728
                                                7E  D4 00383           CLRL     -(SP)                               ; 0731
                                    52      04  AC  D0 00385           MOVL     STATE_VECTOR, R2                    ;
                                            08  A2  9F 00389           PUSHAB   8(R2)                               ;
                          0000V     CF          02  FB 0038C           CALLS    #2, GETSTRING                       ;
                                                04  DD 00391           PUSHL    #4                                  ; 0732
                                            08  A2  9F 00393           PUSHAB   8(R2)                               ;
                          0000V     CF          02  FB 00396           CALLS    #2, GETSTRING                       ;
                             10     A2          56  D0 0039B           MOVL     CHAR_COUNT, 16(R2)                  ; 0733
                                                08  12 0039F           BNEQ     65$                                 ; 0734
                                            08  A2  B5 003A1           TSTW     8(R2)                               ;
                                                03  13 003A4           BEQL     65$                                 ;
                                            10  A2  D6 003A6           INCL     16(R2)                              ; 0735
         00             55          19          03  ED 003A9 65$:      CMPZV    #3, #25, ACT_STATUS, #0             ; 0737
                                                04  13 003AE 66$:      BEQL     67$                                 ;
                                    50          55  D0 003B0           MOVL     ACT_STATUS, R0                      ; 0738
                                                04  003B3              RET                                          ;
                                    50 00000000G  8F  D0 003B4 67$:    MOVL     #LIB$_SYNTAXERR, R0                 ; 0737
                                                04  003BB              RET                                          ; 0736
                                    50      04  AC  D0 003BC 68$:      MOVL     STATE_VECTOR, R0                    ; 0740
                             08     A0      04  AE  A0 003C0           ADDW2    LAST_SPACE, 8(R0)                   ; 0742
                             0C     A0      04  AE  C2 003C5           SUBL2    LAST_SPACE, 12(R0)                  ; 0743
                                            04  AE  D4 003CA           CLRL     LAST_SPACE                          ; 0744
                                              FC4E  31 003CD           BRW      2$                                  ; 0390

; Routine Size:  976 bytes,    Routine Base:  _LIB$CODE + 0020
```

```
1 ROUTINE GETSTRING (STRING, TYPE) : L_GETSTRING =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !         This routine extracts a string of the indicated type from
1 !         the front of the string being parsed.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !         GETSTRING (ARG1, ARG2)
1 !
1 !
1 ! INPUT PARAMETERS:
1 !         ARG1 = address of string descriptor of source string
1 !         ARG2 = string type code
1 !
1 ! IMPLICIT INPUTS:
1 !         NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !         NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !         NONE
1 !
1 ! ROUTINE VALUE:
1 !         1 if string is not empty
1 !         0 if string is null
1 !
1 ! SIDE EFFECTS:
1 !         NONE
1 !
1 !--
1
1
2 BEGIN
2
2 MAP
2         STRING: REF VECTOR;
2
2 EXTERNAL REGISTER
2         CHAR_COUNT      = 6;            ! character count of found string
2
2 !+
2 !
2 ! To extract the string we simply scan through the input string
2 ! until we hit a character that is not in the class.
2 ! We count the characters in the global string count.
2 !
2 !-
2
2 CHAR_COUNT = 0;
2
2 WHILE .(STRING[0])<0,16> GTRU .CHAR_COUNT
3 AND (
;  810          0807  4      (    IF .TYPE EQL SPACES
;  811          0808  5           THEN IS_SPACE (CH$RCHAR (.STRING[1] + .CHAR_COUNT))
;  812          0809  4           ELSE 0)
;  813          0810  4       OR  (IF .TYPE GEQU NUMERIC
;  814          0811  5           THEN IS_NUMERIC (CH$RCHAR (.STRING[1] + .CHAR_COUNT))
;  815          0812  4           ELSE 0)
;  816          0813  4       OR  (IF .TYPE GEQU HEXNUMERIC
;  817          0814  5           THEN IS_HEX (CH$RCHAR (.STRING[1] + .CHAR_COUNT))
;  818          0815  4           ELSE 0)
;  819          0816  4       OR  (IF .TYPE GEQU ALPHANUMERIC
;  820          0817  5           THEN IS_ALPHABETIC (CH$RCHAR (.STRING[1] + .CHAR_COUNT))
;  821          0818  4           ELSE 0)
;  822          0819  4       OR  (IF .TYPE EQL SYMBOL
;  823          0820  5           THEN IS_SYMBOL (CH$RCHAR (.STRING[1] + .CHAR_COUNT))
;  824          0821  4           ELSE 0)
;  825          0822  3      )
;  826          0823  2 DO (CHAR_COUNT = .CHAR_COUNT + 1);
;  827          0824  2
;  828          0825  2 RETURN .CHAR_COUNT GTRU 0;
;  829          0826  2
;  830          0827  1 END;                                        ! end of routine GETSTRING



                                              0004 00000 GETSTRING:
                                                                 .WORD   Save R2                                    ; 0750
                                          56  D4 00002           CLRL    CHAR_COUNT                                 ; 0803
                            51       04   AC  D0 00004           MOVL    STRING, R1                                 ; 0805
      56           61       10            00  ED 00008 1$:       CMPZV   #0, #16, (R1), CHAR_COUNT                  ;
                                          03  1A 0000D           BGTRU   2$                                         ;
                                        00C0  31 0000F           BRW     17$                                        ;
                            52       08   AC  D0 00012 2$:       MOVL    TYPE, R2                                   ; 0807
                                          0F  12 00016           BNEQ    3$                                         ;
                            50       04 B146  9A 00018           MOVZBL  @4(R1)[CHAR_COUNT], R0                     ; 0808
                            09            50  91 0001D           CMPB    R0, #9                                     ;
                                          7C  13 00020           BEQL    11$                                        ;
                            20            50  91 00022           CMPB    R0, #32                                    ;
                                          77  13 00025           BEQL    11$                                        ;
                                          52  D5 00027 3$:       TSTL    R2                                         ; 0810
                                          0F  13 00029           BEQL    4$                                         ;
                            50       04 B146  9A 0002B           MOVZBL  @4(R1)[CHAR_COUNT], R0                     ; 0811
                            30            50  91 00030           CMPB    R0, #48                                    ;
                                          05  1F 00033           BLSSU   4$                                         ;
                            39            50  91 00035           CMPB    R0, #57                                    ;
                                          7C  1B 00038           BLEQU   14$                                        ;
                            02            52  D1 0003A 4$:       CMPL    R2, #2                                     ; 0813
                                          27  1F 0003D           BLSSU   7$                                         ;
                            50       04 B146  9A 0003F           MOVZBL  @4(R1)[CHAR_COUNT], R0                     ; 0814
                            30            50  91 00044           CMPB    R0, #48                                    ;
                                          05  1F 00047           BLSSU   5$                                         ;
                            39            50  91 00049           CMPB    R0, #57                                    ;
                                          7F  1B 0004C           BLEQU   16$                                        ;
                       41   8F            50  91 0004E 5$:       CMPB    R0, #65                                    ;
                                          06  1F 00052           BLSSU   6$                                         ;
                       46   8F            50  91 00054           CMPB    R0, #70                                    ;
```

```
                                73  1B 00058        BLEQU   16$                                     ;
        61    8F                50  91 0005A  6$:   CMPB    R0, #97                                 ;
                                06  1F 0005E        BLSSU   7$                                      ;
        66    8F                50  91 00060        CMPB    R0, #102                                ;
                                67  1B 00064        BLEQU   16$                                     ;
              03                52  D1 00066  7$:   CMPL    R2, #3                                  ; 0816
                                4D  1F 00069        BLSSU   15$                                     ;
              50          04 B146  9A 0006B        MOVZBL  @4(R1)[CHAR_COUNT], R0                  ; 0817
        41    8F                50  91 00070        CMPB    R0, #65                                 ;
                                06  1F 00074        BLSSU   8$                                      ;
        5A    8F                50  91 00076        CMPB    R0, #90                                 ;
                                51  1B 0007A        BLEQU   16$                                     ;
        61    8F                50  91 0007C  8$:   CMPB    R0, #97                                 ;
                                06  1F 00080        BLSSU   9$                                      ;
        7A    8F                50  91 00082        CMPB    R0, #122                                ;
                                45  1B 00086        BLEQU   16$                                     ;
        C0    8F                50  91 00088  9$:   CMPB    R0, #192                                ;
                                06  1F 0008C        BLSSU   10$                                     ;
        CF    8F                50  91 0008E        CMPB    R0, #207                                ;
                                39  1B 00092        BLEQU   16$                                     ;
        D1    8F                50  91 00094 10$:   CMPB    R0, #209                                ;
                                06  1F 00098        BLSSU   12$                                     ;
        DD    8F                50  91 0009A        CMPB    R0, #221                                ;
                                2D  1B 0009E 11$:   BLEQU   16$                                     ;
        DF    8F                50  91 000A0 12$:   CMPB    R0, #223                                ;
                                06  1F 000A4        BLSSU   13$                                     ;
        EF    8F                50  91 000A6        CMPB    R0, #239                                ;
                                21  1B 000AA        BLEQU   16$                                     ;
        F1    8F                50  91 000AC 13$:   CMPB    R0, #241                                ;
                                06  1F 000B0        BLSSU   15$                                     ;
        FD    8F                50  91 000B2        CMPB    R0, #253                                ;
                                15  1B 000B6 14$:   BLEQU   16$                                     ;
              04                52  D1 000B8 15$:   CMPL    R2, #4                                  ; 0819
                                15  12 000BB        BNEQ    17$                                     ;
              50          04 B146  9A 000BD        MOVZBL  @4(R1)[CHAR_COUNT], R0                  ; 0820
              24                50  91 000C2        CMPB    R0, #36                                 ;
                                06  13 000C5        BEQL    16$                                     ;
        5F    8F                50  91 000C7        CMPB    R0, #95                                 ;
                                05  12 000CB        BNEQ    17$                                     ;
                                56  D6 000CD 16$:   INCL    CHAR_COUNT                              ; 0823
                              FF36  31 000CF        BRW     1$                                      ;
                                50  D4 000D2 17$:   CLRL    R0                                      ; 0825
                                56  D5 000D4        TSTL    CHAR_COUNT                              ;
                                02  13 000D6        BEQL    18$                                     ;
                                50  D6 000D8        INCL    R0                                      ;
                                    04 000DA 18$:   RET                                             ; 0827

; Routine Size:  219 bytes,    Routine Base:  _LIB$CODE + 03F0
```

```
!
! Complex type parsing routines. The following routines parse complex
! token types by recursing through TPARSE, using local parse tables for
! the individual types.
!

!
! Declare TPARSE block passed as arg list.
!

MACRO
        TPARSE_ARGS =
                BUILTIN AP;
                BIND TPARSE_BLOCK = AP : REF $BBLOCK;
                %;

!
! TPARSE block extension usage by following routines:
!

MACRO
        TEMP_NUMBER     = TPA$K_LENGTH0+00, 0, 32, 0%,
        TEMP_CHAR       = TPA$K_LENGTH0+04, 0,  8, 0%;

LITERAL
        LOCAL_STORAGE   = 8;

FORWARD
        LIB$$UIC_STATE_TABLE    : VECTOR [0],
        LIB$$UIC_KEY_TABLE      : VECTOR [0],
        LIB$$IDENT_STATE_TABLE  : VECTOR [0],
        LIB$$IDENT_KEY_TABLE    : VECTOR [0];
```

```
  1 ROUTINE PARSE_FILESPEC (TPARSE_BLOCK) =
  1
  1 !++
  1 !
  1 ! FUNCTIONAL DESCRIPTION:
  1 !
  1 !       This routine parses a file specifier string, using the $FILESCAN
  1 !       service.
  1 !
  1 ! CALLING SEQUENCE:
  1 !       PARSE_FILESPEC (TPARSE_BLOCK)
  1 !
  1 ! INPUT PARAMETERS:
  1 !       TPARSE_BLOCK: address of TPARSE control block
  1 !
  1 ! IMPLICIT INPUTS:
  1 !       NONE
  1 !
  1 ! OUTPUT PARAMETERS:
  1 !       NONE
  1 !
  1 ! IMPLICIT OUTPUTS:
  1 !       NONE
  1 !
  1 ! ROUTINE VALUE:
  1 !       status of parse operation
  1 !
  1 ! SIDE EFFECTS:
  1 !       string descriptors in TPARSE block modified if successful
  1 !
  1 !--
  1
  2 BEGIN
  2
  2 MAP
  2         TPARSE_BLOCK    : REF $BBLOCK;  ! TPARSE control block arg
  2
  2 LOCAL
  2         FSCAN_LIST      : $BBLOCK [12]; ! descriptor list for $FILESCAN
  2
  2
  2 ! Set up the filescan list arguments and call it.
  2 !
  2
  2 FSCAN_LIST[ATR$W_TYPE] = FSCN$_FILESPEC;
  2 FSCAN_LIST+8 = 0;
P 2 IF NOT $FILESCAN (SRCSTR = TPARSE_BLOCK[TPA$L_STRINGCNT],
  3                   VALUELST = FSCAN_LIST)
  2 THEN RETURN 0;
  2
  2 ! If successful, update the string pointers in the user's block.
  2 !
  2
  2 TPARSE_BLOCK[TPA$L_STRINGCNT] = .TPARSE_BLOCK[TPA$L_STRINGCNT]
  2                               - .FSCAN_LIST[ATR$W_SIZE];
  2 TPARSE_BLOCK[TPA$L_STRINGPTR] = .TPARSE_BLOCK[TPA$L_STRINGPTR]
  2                               + .FSCAN_LIST[ATR$W_SIZE];
```

```
;  922         0917  2
;  923         0918  2 1
;  924         0919  1 END;                                   ! End of routine PARSE_FILESPEC


                                                                    .EXTRN  SYS$FILESCAN

                                              0004 00000 PARSE_FILESPEC:
                                                                    .WORD   Save R2                                 ; 0860
                         5E           0C  C2 00002                  SUBL2   #12, SP                                 ;
                  02     AE           01  B0 00005                  MOVW    #1, FSCAN_LIST+2                        ; 0904
                                08    AE  D4 00009                  CLRL    FSCAN_LIST+8                            ; 0905
                                      7E  D4 0000C                  CLRL    -(SP)                                   ; 0907
                                04    AE  9F 0000E                  PUSHAB  FSCAN_LIST                              ;
                         52     04    AC  D0 00011                  MOVL    TPARSE_BLOCK, R2                        ;
                                08    A2  9F 00015                  PUSHAB  8(R2)                                   ;
           00000000G     00           03  FB 00018                  CALLS   #3, SYS$FILESCAN                        ;
                         03           50  E8 0001F                  BLBS    R0, 1$                                  ;
                                      50  D4 00022                  CLRL    R0                                      ; 0908
                                          04 00024                  RET                                             ;
                         50           6E  3C 00025 1$:              MOVZWL  FSCAN_LIST, R0                          ; 0914
                  08     A2           50  C2 00028                  SUBL2   R0, 8(R2)                               ;
                         50           6E  3C 0002C                  MOVZWL  FSCAN_LIST, R0                          ; 0916
                  0C     A2           50  C0 0002F                  ADDL2   R0, 12(R2)                              ;
                         50           01  D0 00033                  MOVL    #1, R0                                  ; 0919
                                          04 00036                  RET                                             ;

; Routine Size:  55 bytes,    Routine Base:  _LIB$CODE + 04CB
```

```
1 ROUTINE PARSE_IDENT (TPARSE_BLOCK, GENERAL) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine parses an identifier string, leaving the binary value
1 !       in the TPA$L_NUMBER field of the tparse block.
1 !
1 ! CALLING SEQUENCE:
1 !       PARSE_IDENT (TPARSE_BLOCK, GENERAL)
1 !
1 ! INPUT PARAMETERS:
1 !       TPARSE_BLOCK: address of TPARSE control block
1 !       GENERAL: 0 to parse UIC only
1 !                1 to parse general identifier
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       status of parse operation
1 !
1 ! SIDE EFFECTS:
1 !       string descriptors in TPARSE block modified if successful
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       TPARSE_BLOCK    : REF $BBLOCK;  ! TPARSE control block arg
2
2 LOCAL
2       STATUS,                         ! general status value
2       LOCAL_BLOCK     : $BBLOCK [TPA$K_LENGTH0+LOCAL_STORAGE];
2                                       ! local TPARSE control block
2
2 ! Set up the local TPARSE control block by copying the user's.
2 ! Then do the parse.
2 !
2
2 CH$MOVE (TPA$K_LENGTH0, .TPARSE_BLOCK, LOCAL_BLOCK);
2 LOCAL_BLOCK[TPA$L_COUNT] = TPA$K_COUNT0;
2 LOCAL_BLOCK[TPA$V_BLANKS] = 1;
3 STATUS = (
3     IF .GENERAL
3     THEN LIB$TPARSE (LOCAL_BLOCK, LIB$$IDENT_STATE_TABLE, LIB$$IDENT_KEY_TABLE)
3     ELSE LIB$TPARSE (LOCAL_BLOCK, LIB$$UIC_STATE_TABLE, LIB$$UIC_KEY_TABLE)
2     );
2
```

```
0977  2 ! If successful, update the string pointers in the user's block.
0978  2 !
0979  2
0980  2 IF .STATUS
0981  2 THEN
0982  3     BEGIN
0983  3     TPARSE_BLOCK[TPA$L_STRINGCNT] = .LOCAL_BLOCK[TPA$L_STRINGCNT];
0984  3     TPARSE_BLOCK[TPA$L_STRINGPTR] = .LOCAL_BLOCK[TPA$L_STRINGPTR];
0985  3     TPARSE_BLOCK[TPA$L_NUMBER] = .LOCAL_BLOCK[TPA$L_NUMBER];
0986  2     END;
0987  2
0988  2 .STATUS
0989  1 END;                                  ! End of routine PARSE_IDENT



                                   007C 00000 PARSE_IDENT:
                                                      .WORD   Save R2,R3,R4,R5,R6                      ; 0920
               5E             2C  C2 00002            SUBL2   #44, SP                                  ;
               56       04    AC  D0 00005            MOVL    TPARSE_BLOCK, R6                         ; 0968
     6E        66             24  28 00009            MOVC3   #36, (R6), LOCAL_BLOCK                   ;
               6E             08  D0 0000D            MOVL    #8, LOCAL_BLOCK                          ; 0969
          04   AE             01  88 00010            BISB2   #1, LOCAL_BLOCK+4                        ; 0970
               0A       08    AC  E9 00014            BLBC    GENERAL, 1$                              ; 0972
                     0000V    CF  9F 00018            PUSHAB  LIB$$IDENT_KEY_TABLE                     ; 0973
                     0000V    CF  9F 0001C            PUSHAB  LIB$$IDENT_STATE_TABLE                   ;
                              08  11 00020            BRB     2$                                       ;
                     0000V    CF  9F 00022 1$:        PUSHAB  LIB$$UIC_KEY_TABLE                       ; 0974
                     0000V    CF  9F 00026            PUSHAB  LIB$$UIC_STATE_TABLE                     ;
                        08    AE  9F 0002A 2$:        PUSHAB  LOCAL_BLOCK                              ;
        FAEC   CF             03  FB 0002D            CALLS   #3, LIB$TPARSE                           ;
               0A             50  E9 00032            BLBC    STATUS, 3$                               ; 0980
          08   A6       08    AE  7D 00035            MOVQ    LOCAL_BLOCK+8, 8(R6)                     ; 0983
          1C   A6       1C    AE  D0 0003A            MOVL    LOCAL_BLOCK+28, 28(R6)                   ; 0985
                                  04 0003F 3$:        RET                                              ; 0989

; Routine Size:  64 bytes,    Routine Base:  _LIB$CODE + 0502
;  997          0990  1 !
;  998          0991  1 ! Action routines to parse UIC's and identifiers.
;  999          0992  1 !
; 1000          0993  1
; 1001          0994  1 !
; 1002          0995  1 ! Remember opening bracket.
; 1003          0996  1 !
; 1004          0997  1 ROUTINE SET_BRACKET =
; 1005          0998  2 BEGIN
; 1006          0999  2 TPARSE_ARGS;
; 1007          1000  2
; 1008          1001  2 TPARSE_BLOCK[TEMP_CHAR] = .TPARSE_BLOCK[TPA$B_CHAR];
; 1009          1002  2 1
; 1010          1003  1 END;



                                            0000 00000 SET_BRACKET:
                                                              .WORD   Save nothing                              ; 0997
                        28   AC        18   AC  90 00002      MOVB    24(TPARSE_BLOCK), 40(TPARSE_BLOCK)       ; 1001
                             50             01  D0 00007      MOVL    #1, R0                                    ; 1003
                                                04 0000A      RET                                               ;

; Routine Size:  11 bytes,    Routine Base:  _LIB$CODE + 0542


; 1011          1004  1
; 1012          1005  1 !
; 1013          1006  1 ! Check closing bracket against opening bracket.
; 1014          1007  1 !
; 1015          1008  1 ROUTINE CHECK_BRACKET =
; 1016          1009  2 BEGIN
; 1017          1010  2 TPARSE_ARGS;
; 1018          1011  2
; 1019          1012  2 $ASSUME ('<'-'>', EQL, '['-']');
; 1020          1013  3 .TPARSE_BLOCK[TPA$B_CHAR] EQL .TPARSE_BLOCK[TEMP_CHAR] + (']'-'[')
; 1021          1014  3
; 1022          1015  1 END;



                                            0000 00000 CHECK_BRACKET:
                                                              .WORD   Save nothing                              ; 1008
                             51        28   AC  9A 00002      MOVZBL  40(TPARSE_BLOCK), R1                      ; 1013
                             51             02  C0 00006      ADDL2   #2, R1                                    ;
                                            50  D4 00009      CLRL    R0                                        ;
        51       18   AC     08             00  ED 0000B      CMPZV   #0, #8, 24(TPARSE_BLOCK), R1              ;
                                            02  12 00011      BNEQ    1$                                        ;
                                            50  D6 00013      INCL    R0                                        ;
                                                04 00015 1$:  RET                                               ; 1015

; Routine Size:  22 bytes,    Routine Base:  _LIB$CODE + 054D
; 1023          1016  1
; 1024          1017  1 !
; 1025          1018  1 ! Check and save group number.
; 1026          1019  1 !
; 1027          1020  1 ROUTINE STORE_GROUP =
; 1028          1021  2 BEGIN
; 1029          1022  2 TPARSE_ARGS;
; 1030          1023  2
; 1031          1024  2 (TPARSE_BLOCK[TEMP_NUMBER])<16,16> = .TPARSE_BLOCK[TPA$L_NUMBER];
; 1032          1025  2 .TPARSE_BLOCK[TPA$L_NUMBER] LSSU 16383
; 1033          1026  2
; 1034          1027  1 END;



                                                   0000 00000 STORE_GROUP:
                                                                       .WORD    Save nothing                              ; 1020
                   26    AC        1C    AC    B0 00002               MOVW     28(TPARSE_BLOCK), 38(TPARSE_BLOCK)        ; 1024
                                         50    D4 00007               CLRL     R0                                        ; 1025
             00003FFF    8F        1C    AC    D1 00009               CMPL     28(TPARSE_BLOCK), #16383                  ;
                                         02    1E 00011               BGEQU    1$                                        ;
                                         50    D6 00013               INCL     R0                                        ;
                                               04 00015 1$:           RET                                                ; 1027

; Routine Size:  22 bytes,    Routine Base:  _LIB$CODE + 0563


; 1035          1028  1
; 1036          1029  1 !
; 1037          1030  1 ! Store wild card group number.
; 1038          1031  1 !
; 1039          1032  1 ROUTINE WILD_GROUP =
; 1040          1033  2 BEGIN
; 1041          1034  2 TPARSE_ARGS;
; 1042          1035  2
; 1043          1036  2 (TPARSE_BLOCK[TEMP_NUMBER])<16,16> = 16383;
; 1044          1037  2
; 1045          1038  2 1
; 1046          1039  1 END;



                                                   0000 00000 WILD_GROUP:
                                                                       .WORD    Save nothing                              ; 1032
                   26    AC      3FFF    8F    B0 00002               MOVW     #16383, 38(TPARSE_BLOCK)                  ; 1036
                         50              01    D0 00008               MOVL     #1, R0                                    ; 1039
                                               04 0000B               RET                                                ;

; Routine Size:  12 bytes,    Routine Base:  _LIB$CODE + 0579


; 1047          1040  1
; 1048          1041  1 !
; 1049          1042  1 ! Check and save member number.
; 1050          1043  1 !
; 1051          1044  1 ROUTINE STORE_MEMBER =
; 1052          1045  2 BEGIN
; 1053          1046  2 TPARSE_ARGS;
; 1054          1047  2
; 1055          1048  2 IF .TPARSE_BLOCK[TPA$L_NUMBER] GEQU 65535
; 1056          1049  2 THEN RETURN 0;
; 1057          1050  2 (TPARSE_BLOCK[TPA$L_NUMBER])<16,16> = .(TPARSE_BLOCK[TEMP_NUMBER])<16,16>;
; 1058          1051  2
; 1059          1052  2 1
; 1060          1053  1 END;



                                            0000 00000 STORE_MEMBER:
                                                              .WORD    Save nothing                                   ; 1044
                         0000FFFF   8F       1C    AC  D1 00002         CMPL     28(TPARSE_BLOCK), #65535              ; 1048
                                                   03  1F 0000A         BLSSU    1$                                    ;
                                                   50  D4 0000C         CLRL     R0                                    ; 1049
                                                       04 0000E         RET                                            ;
                               1E   AC       26    AC  B0 0000F 1$:     MOVW     38(TPARSE_BLOCK), 30(TPARSE_BLOCK)     ; 1050
                                    50             01  D0 00014         MOVL     #1, R0                                ; 1053
                                                       04 00017         RET                                            ;

; Routine Size:  24 bytes,    Routine Base:  _LIB$CODE + 0585


; 1061          1054  1
; 1062          1055  1 !
; 1063          1056  1 ! Store wild card member number.
; 1064          1057  1 !
; 1065          1058  1 ROUTINE WILD_MEMBER =
; 1066          1059  2 BEGIN
; 1067          1060  2 TPARSE_ARGS;
; 1068          1061  2
; 1069          1062  2 (TPARSE_BLOCK[TPA$L_NUMBER])<16,16> = .(TPARSE_BLOCK[TEMP_NUMBER])<16,16>;
; 1070          1063  2 (TPARSE_BLOCK[TPA$L_NUMBER])<0,16> = 65535;
; 1071          1064  2
; 1072          1065  2 1
; 1073          1066  1 END;



                                            0000 00000 WILD_MEMBER:
                                                              .WORD    Save nothing                                   ; 1058
                               1E   AC       26    AC  B0 00002         MOVW     38(TPARSE_BLOCK), 30(TPARSE_BLOCK)     ; 1062
                               1C   AC             01  AE 00007         MNEGW    #1, 28(TPARSE_BLOCK)                  ; 1063
                                    50             01  D0 0000B         MOVL     #1, R0                                ; 1066
                                                       04 0000E         RET                                            ;

; Routine Size:  15 bytes,    Routine Base:  _LIB$CODE + 059D
; 1074          1067  1
; 1075          1068  1 !
; 1076          1069  1 ! Store full wild card identifier.
; 1077          1070  1 !
; 1078          1071  1 ROUTINE ALL_WILD =
; 1079          1072  2 BEGIN
; 1080          1073  2 TPARSE_ARGS;
; 1081          1074  2
; 1082          1075  2 TPARSE_BLOCK[TPA$L_NUMBER] = UIC$K_MATCH_ALL;
; 1083          1076  2
; 1084          1077  2 1
; 1085          1078  1 END;



                                              0000 00000 ALL_WILD:
                                                                .WORD    Save nothing                                  ; 1071
                              1C    AC            01   CE 00002         MNEGL    #1, 28(TPARSE_BLOCK)                          ; 1075
                                    50            01   D0 00006         MOVL     #1, R0                                        ; 1078
                                                       04 00009         RET                                                    ;

; Routine Size:  10 bytes,    Routine Base:  _LIB$CODE + 05AC


; 1086          1079  1
; 1087          1080  1 !
; 1088          1081  1 ! Convert single identifier name to numeric value.
; 1089          1082  1 !
; 1090          1083  1 ROUTINE SINGLE_NAME =
; 1091          1084  2 BEGIN
; 1092          1085  2 TPARSE_ARGS;
; 1093          1086  2
; 1094        P 1087  2 $ASCTOID (NAME = TPARSE_BLOCK[TPA$L_TOKENCNT],
; 1095          1088  3          ID = TPARSE_BLOCK[TPA$L_NUMBER])
; 1096          1089  3
; 1097          1090  1 END;



                                                                        .EXTRN   SYS$ASCTOID

                                              0000 00000 SINGLE_NAME:
                                                                .WORD    Save nothing                                  ; 1083
                                                  7E   D4 00002         CLRL     -(SP)                                         ; 1088
                                           1C     AC   9F 00004         PUSHAB   28(TPARSE_BLOCK)                              ;
                                           10     AC   9F 00007         PUSHAB   16(TPARSE_BLOCK)                              ;
                        00000000G  00             03   FB 0000A         CALLS    #3, SYS$ASCTOID                               ;
                                                       04 00011         RET                                                    ; 1090

; Routine Size:  18 bytes,    Routine Base:  _LIB$CODE + 05B6


; 1098          1091  1
; 1099          1092  1 !
; 1100          1093  1 ! Convert and save group name.
; 1101          1094  1 !
; 1102          1095  1 ROUTINE GROUP_NAME =
; 1103          1096  2 BEGIN
; 1104          1097  2 TPARSE_ARGS;
; 1105          1098  2
; 1106          1099  2 LOCAL
; 1107          1100  2         STATUS;
; 1108          1101  2
; 1109        P 1102  2 STATUS = $ASCTOID (NAME = TPARSE_BLOCK[TPA$L_TOKENCNT],
; 1110          1103  2                    ID = TPARSE_BLOCK[TEMP_NUMBER]);
; 1111          1104  2 IF NOT .STATUS THEN RETURN .STATUS;
; 1112          1105  2 .(TPARSE_BLOCK[TEMP_NUMBER])<16,16> LEQU 16383 AND
; 1113          1106  2 .(TPARSE_BLOCK[TEMP_NUMBER])<00,16> EQLU 65535
; 1114          1107  2
; 1115          1108  1 END;



                                              0004 00000 GROUP_NAME:
                                                                .WORD   Save R2                          ; 1095
                                         7E  D4 00002           CLRL    -(SP)                            ; 1103
                                  24     AC  9F 00004           PUSHAB  36(TPARSE_BLOCK)                 ;
                                  10     AC  9F 00007           PUSHAB  16(TPARSE_BLOCK)                 ;
                      00000000G  00      03  FB 0000A           CALLS   #3, SYS$ASCTOID                  ;
                                 1E      50  E9 00011           BLBC    STATUS, 3$                       ; 1104
                                         51  D4 00014           CLRL    R1                               ; 1105
                 3FFF    8F       26     AC  B1 00016           CMPW    38(TPARSE_BLOCK), #16383         ;
                                         02  1A 0001C           BGTRU   1$                               ;
                                         51  D6 0001E           INCL    R1                               ;
                                         50  D4 00020 1$:       CLRL    R0                               ; 1106
                 FFFF    8F       24     AC  B1 00022           CMPW    36(TPARSE_BLOCK), #65535         ;
                                         02  12 00028           BNEQ    2$                               ;
                                         50  D6 0002A           INCL    R0                               ;
                         52              51  D2 0002C 2$:       MCOML   R1, R2                           ;
                         50              52  CA 0002F           BICL2   R2, R0                           ;
                                             04 00032 3$:       RET                                      ; 1108

; Routine Size:  51 bytes,    Routine Base:  _LIB$CODE + 05C8


; 1116          1109  1
; 1117          1110  1 !
; 1118          1111  1 ! Convert and check member name.
; 1119          1112  1 !
; 1120          1113  1 ROUTINE MEMBER_NAME =
; 1121          1114  2 BEGIN
; 1122          1115  2 TPARSE_ARGS;
; 1123          1116  2
; 1124          1117  2 LOCAL
; 1125          1118  2         STATUS;
; 1126          1119  2
; 1127        P 1120  2 STATUS = $ASCTOID (NAME = TPARSE_BLOCK[TPA$L_TOKENCNT],
; 1128          1121  2                    ID = TPARSE_BLOCK[TPA$L_NUMBER]);
; 1129          1122  2 IF NOT .STATUS THEN RETURN .STATUS;
; 1130          1123  2 .(TPARSE_BLOCK[TEMP_NUMBER])<16,16> EQL
; 1131          1124  2 .(TPARSE_BLOCK[TPA$[_NUMBER])<16,16>
; 1132          1125  2
; 1133          1126  1 END;



                                                       0000 00000 MEMBER_NAME:
                                                                          .WORD   Save nothing                                    : 1113
                                                    7E  D4 00002          CLRL    -(SP)                                           : 1121
                                           1C       AC  9F 00004          PUSHAB  28(TPARSE_BLOCK)                                :
                                           10       AC  9F 00007          PUSHAB  16(TPARSE_BLOCK)                                :
                        00000000G  00               03  FB 0000A          CALLS   #3, SYS$ASCTOID                                 :
                                   0B               50  E9 00011          BLBC    STATUS, 1$                                      : 1122
                                                    50  D4 00014          CLRL    R0                                              : 1124
                              1E   AC      26       AC  B1 00016          CMPW    38(TPARSE_BLOCK), 30(TPARSE_BLOCK)              :
                                                    02  12 0001B          BNEQ    1$                                              :
                                                    50  D6 0001D          INCL    R0                                              :
                                                        04 0001F 1$:      RET                                                     : 1126

; Routine Size:  32 bytes,    Routine Base:  _LIB$CODE + 05FB
```

```
!
! State table to parse UIC string.
!

$INIT_STATE (LIB$$UIC_STATE_TABLE, LIB$$UIC_KEY_TABLE, _LIB$PARSE);

$STATE  (,
        ('[',,SET_BRACKET),
        ('<',,SET_BRACKET),
        ('%',HEX_FORMAT)
        );

$STATE  (,
        ('*',,WILD_GROUP),
        (TPA$_OCTAL,,STORE_GROUP)
        );

$STATE  (,
        (',')
        );

$STATE  (,
        ('*',,WILD_MEMBER),
        (TPA$_OCTAL,,STORE_MEMBER)
        );

$STATE  (,
        (']',TPA$_EXIT,CHECK_BRACKET),
        ('>',TPA$_EXIT,CHECK_BRACKET)
        );

$STATE  (HEX_FORMAT,
        ('X')
        );

$STATE  (,
        (TPA$_HEX,TPA$_EXIT)
        );

!
! State table to parse UIC or identifier string.
!

$INIT_STATE (LIB$$IDENT_STATE_TABLE, LIB$$IDENT_KEY_TABLE, _LIB$PARSE);

$STATE  (,
        ('*',TPA$_EXIT,ALL_WILD),
        (TPA$_SYMBOL,TPA$_EXIT,SINGLE_NAME),
        ('[',,SET_BRACKET),
        ('<',,SET_BRACKET),
        ('%',HEX_FORMAT)
        );

$STATE  (,
        ((GROUP_MEMBER)),
        ('*',,ALL_WILD),
        (TPA$_SYMBOL,,SINGLE_NAME)
; 1192            1184  1          );
; 1193            1185  1
; 1194          P 1186  1 $STATE   (,
; 1195          P 1187  1          (']',TPA$_EXIT,CHECK_BRACKET),
; 1196          P 1188  1          ('>',TPA$_EXIT,CHECK_BRACKET)
; 1197            1189  1          );
; 1198            1190  1
; 1199          P 1191  1 $STATE   (GROUP_MEMBER,
; 1200          P 1192  1          ('*',,WILD_GROUP),
; 1201          P 1193  1          (TPA$_OCTAL,,STORE_GROUP),
; 1202          P 1194  1          (TPA$_SYMBOL,,GROUP_NAME)
; 1203            1195  1          );
; 1204            1196  1
; 1205          P 1197  1 $STATE   (,
; 1206          P 1198  1          (',')
; 1207            1199  1          );
; 1208            1200  1
; 1209          P 1201  1 $STATE   (,
; 1210          P 1202  1          ('*',TPA$_EXIT,WILD_MEMBER),
; 1211          P 1203  1          (TPA$_OCTAL,TPA$_EXIT,STORE_MEMBER),
; 1212          P 1204  1          (TPA$_SYMBOL,TPA$_EXIT,MEMBER_NAME)
; 1213            1205  1          );
; 1214            1206  1
; 1215            1207  1 END
; 1216            1208  0 ELUDOM


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

                              00000 LIB$$UIC_STATE_TABLE::
                                          .BLKB   0
                        805B  00000 ;TPA$TYPE
                                    U.2:      .WORD   -32677                          ;
                   00000000*  00002 ;TPA$ACTION
                                    U.3:      .LONG   <<SET_BRACKET-U.3>-4>           ;
                        803C  00006 ;TPA$TYPE
                                    U.4:      .WORD   -32708                          ;
                   00000000*  00008 ;TPA$ACTION
                                    U.5:      .LONG   <<SET_BRACKET-J.5>-4>           ;
                        1425  0000C ;TPA$TYPE
                                    U.6:      .WORD   5157                            ;
                       0000*  0000E ;TPA$TARGET
                                    U.8:      .WORD   <<U.7-U.8>-2>                   ;
                        802A  00010 ;TPA$TYPE
                                    U.9:      .WORD   -32726                          ;
                   00000000*  00012 ;TPA$ACTION
                                    U.10:     .LONG   <<WILD_GROUP-U.10>-4>           ;
                        85F4  00016 ;TPA$TYPE
                                    U.11:     .WORD   -31244                          ;
                   00000000*  00018 ;TPA$ACTION
                                    U.12:     .LONG   <<STORE_GROUP-U.12>-4>          ;
                        042C  0001C ;TPA$TYPE
                                    U.13:     .WORD   1068                            ;
                        802A  0001E ;TPA$TYPE
                                    U.14:     .WORD   -32726                          ;
                   00000000*  00020 ;TPA$ACTION
```

```
                       U.15:     .LONG    <<WILD_MEMBER-U.15>-4>                ;
      85F4  00024 ;TPA$TYPE
                       U.16:     .WORD    -31244                                ;
  00000000* 00026 ;TPA$ACTION
                       U.17:     .LONG    <<STORE_MEMBER-U.17>-4>               ;
      905D  0002A ;TPA$TYPE
                       U.18:     .WORD    -28579                                ;
  00000000* 0002C ;TPA$ACTION
                       U.19:     .LONG    <<CHECK_BRACKET-U.19>-4>              ;
      FFFF  00030 ;TPA$TARGET
                       U.20:     .WORD    -1                                    ;
      943E  00032 ;TPA$TYPE
                       U.21:     .WORD    -27586                                ;
  00000000* 00034 ;TPA$ACTION
                       U.22:     .LONG    <<CHECK_BRACKET-U.22>-4>              ;
      FFFF  00038 ;TPA$TARGET
                       U.23:     .WORD    -1                                    ;
            0003A ;HEX_FORMAT
                       U.7:      .BLKB    0
      0458  0003A ;TPA$TYPE
                       U.24:     .WORD    1112                                  ;
      15F5  0003C ;TPA$TYPE
                       U.25:     .WORD    5621                                  ;
      FFFF  0003E ;TPA$TARGET
                       U.26:     .WORD    -1                                    ;
            00040 LIB$$IDENT_STATE_TABLE::
                                 .BLKB    0
      902A  00040 ;TPA$TYPE
                       U.28:     .WORD    -28630                                ;
  00000000* 00042 ;TPA$ACTION
                       U.29:     .LONG    <<ALL_WILD-U.29>-4>                   ;
      FFFF  00046 ;TPA$TARGET
                       U.30:     .WORD    -1                                    ;
      91F1  00048 ;TPA$TYPE
                       U.31:     .WORD    -28175                                ;
  00000000* 0004A ;TPA$ACTION
                       U.32:     .LONG    <<SINGLE_NAME-U.32>-4>                ;
      FFFF  0004E ;TPA$TARGET
                       U.33:     .WORD    -1                                    ;
      805B  00050 ;TPA$TYPE
                       U.34:     .WORD    -32677                                ;
  00000000* 00052 ;TPA$ACTION
                       U.35:     .LONG    <<SET_BRACKET-U.35>-4>                ;
      803C  00056 ;TPA$TYPE
                       U.36:     .WORD    -32708                                ;
  00000000* 00058 ;TPA$ACTION
                       U.37:     .LONG    <<SET_BRACKET-U.37>-4>                ;
      1425  0005C ;TPA$TYPE
                       U.38:     .WORD    5157                                  ;
     0000*  0005E ;TPA$TARGET
                       U.39:     .WORD    <<U.7-U.39>-2>                        ;
      09F8  00060 ;TPA$TYPE
                       U.40:     .WORD    2552                                  ;
     0000*  00062 ;TPA$SUBEXP
                       U.42:     .WORD    <<U.41-U.42>-2>                       ;
      802A  00064 ;TPA$TYPE
                       U.43:     .WORD    -32726                                ;
```

```
00000000* 00066 ;TPA$ACTION
                U.44:    .LONG    <<ALL_WILD-U.44>-4>                        ;
     85F1 0006A ;TPA$TYPE
                U.45:    .WORD    -31247                                     ;
00000000* 0006C ;TPA$ACTION
                U.46:    .LONG    <<SINGLE_NAME-U.46>-4>                     ;
     905D 00070 ;TPA$TYPE
                U.47:    .WORD    -28579                                     ;
00000000* 00072 ;TPA$ACTION
                U.48:    .LONG    <<CHECK_BRACKET-U.48>-4>                   ;
     FFFF 00076 ;TPA$TARGET
                U.49:    .WORD    -1                                         ;
     943E 00078 ;TPA$TYPE
                U.50:    .WORD    -27586                                     ;
00000000* 0007A ;TPA$ACTION
                U.51:    .LONG    <<CHECK_BRACKET-U.51>-4>                   ;
     FFFF 0007E ;TPA$TARGET
                U.52:    .WORD    -1                                         ;
          00080 ;GROUP_MEMBER
                U.41:    .BLKB    0
     802A 00080 ;TPA$TYPE
                U.53:    .WORD    -32726                                     ;
00000000* 00082 ;TPA$ACTION
                U.54:    .LONG    <<WILD_GROUP-U.54>-4>                      ;
     81F4 00086 ;TPA$TYPE
                U.55:    .WORD    -32268                                     ;
00000000* 00088 ;TPA$ACTION
                U.56:    .LONG    <<STORE_GROUP-U.56>-4>                     ;
     85F1 0008C ;TPA$TYPE
                U.57:    .WORD    -31247                                     ;
00000000* 0008E ;TPA$ACTION
                U.58:    .LONG    <<GROUP_NAME-U.58>-4>                      ;
     042C 00092 ;TPA$TYPE
                U.59:    .WORD    1068                                       ;
     902A 00094 ;TPA$TYPE
                U.60:    .WORD    -28630                                     ;
00000000* 00096 ;TPA$ACTION
                U.61:    .LONG    <<WILD_MEMBER-U.61>-4>                     ;
     FFFF 0009A ;TPA$TARGET
                U.62:    .WORD    -1                                         ;
     91F4 0009C ;TPA$TYPE
                U.63:    .WORD    -28172                                     ;
00000000* 0009E ;TPA$ACTION
                U.64:    .LONG    <<STORE_MEMBER-U.64>-4>                    ;
     FFFF 000A2 ;TPA$TARGET
                U.65:    .WORD    -1                                         ;
     95F1 000A4 ;TPA$TYPE
                U.66:    .WORD    -27151                                     ;
00000000* 000A6 ;TPA$ACTION
                U.67:    .LONG    <<MEMBER_NAME-U.67>-4>                     ;
     FFFF 000AA ;TPA$TARGET
                U.68:    .WORD    -1                                         ;

                         .PSECT   _LIB$PARSE_KEY0,NOWRT,  SHR,  PIC,1

          00000 LIB$$UIC_KEY_TABLE::
                         .BLKB    0
```

```
                                          00000 ;TPA$KEY0
                                                U.1:      .BLKB   0
                                          00000 LIB$$IDENT_KEY_TABLE::
                                                          .BLKB   0
                                          00000 ;TPA$KEY0
                                                U.27:     .BLKB   0


;                          PSECT SUMMARY
;
;       Name                     Bytes                       Attributes
;
;  _LIB$CODE                      1563  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _LIB$PARSE_KEY0                   0  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  _LIB$PARSE_STATE                172  NOVEC,NOWRT,  RD .  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)




;                       Library Statistics
;
;                                            -------- Symbols --------      Pages     Processing
;       File                                 Total    Loaded   Percent      Mapped    Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1          9776        28         0        581       00:01.0
;  _$255$DUA28:[SYSLIB]TPAMAC.L32;1             42        33        78         14       00:00.1



;                          COMMAND QUALIFIERS

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

; Size:          1531 code + 204 data bytes
; Run Time:         00:41.3
; Elapsed Time:     01:26.4
; Lines/CPU Min:     1753
; Lexemes/CPU-Min: 48464
; Memory Used:  341 pages
; Compilation Complete
```

0437 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYSGMSG
LIS

SCRINPUT
LIS

SCRMISC
LIS

SCRREQ
LIS

SETPRIV
LIS

SHRMSG
LIS

TPARSE
LIS

SCRLIB
LIS

SCRVECTOR
LIS

SSMSG
LIS

0438 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

VMSVECTOR
LIS

VMSRTL

VMSRTL
MAP