```
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
```

```
**FILE**ID**PATLEX                                  J 5

PPPPPPPP      AAAAAA    TTTTTTTTTT  LL          EEEEEEEEEE  XX      XX
PPPPPPPP      AAAAAA    TTTTTTTTTT  LL          EEEEEEEEEE  XX      XX
PP      PP  AA      AA      TT      LL          EE          XX      XX
PP      PP  AA      AA      TT      LL          EE          XX      XX
PP      PP  AA      AA      TT      LL          EE            XX  XX
PP      PP  AA      AA      TT      LL          EE            XX  XX
PPPPPPPP    AA      AA      TT      LL          EEEEEEEE        XX
PPPPPPPP    AA      AA      TT      LL          EEEEEEEE        XX
PP          AAAAAAAAAA      TT      LL          EE            XX  XX
PP          AAAAAAAAAA      TT      LL          EE            XX  XX
PP          AA      AA      TT      LL          EE          XX      XX      ....
PP          AA      AA      TT      LL          EE          XX      XX      ....
PP          AA      AA      TT      LLLLLLLLLL  EEEEEEEEEE  XX      XX      ....
PP          AA      AA      TT      LLLLLLLLLL  EEEEEEEEEE  XX      XX      ....

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 PATLEX (
L 0                     %IF %VARIANT EQL 1
  0                     %THEN
  0                             ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE),
  0                     %FI
  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:     PATCH
  1 !
  1 ! ABSTRACT:     THIS MODULE CONTAINS A MARS SCANNER.
  1 !
  1 ! ENVIRONMENT:  STARLET, user mode, interrupts disabled.
  1 !
  1 ! AUTHOR: Carol Peters, CREATION DATE: 25 July 1977
  1 !
  1 ! MODIFIED BY:
  1 !
  1 !       V02-012 PCG0001         Peter George            02-FEB-1981
  1 !               Add require statement for LIB$:PATDEF.REQ
  1 !
  1 ! MODIFICATIONS:
  1 !  NO   DATE            PROGRAMMER              PURPOSE
  1 !  --   ----            ----------              -------
  1 !
  1 !  00   5-JAN-78        K.D. MORSE              ADAPT VERSION 15 FOR PATCH.
  1 !  01   24-JAN-78       K.D. MORSE              NO CHANGES FOR VERS 16.
  1 !  02   24-MAR-78       K.D. MORSE              NO CHANGES FOR VERS 17-18.
  1 !  03   14-APR-78       K.D. MORSE              NO CHANGES FOR VERS 19-20.
  1 !  04   25-APR-78       K.D. MORSE              CONVERT TO NATIVE COMPILER.
```

```
1 !  05  26-APR-78      K.D. MORSE           INCLUDE CODE TO HANDLE KEYWORDS
1 !                                          BEGINNING WITH A PERIOD.
1 !  06  02-MAY-78      K.D. MORSE           CHANGE RETURNED TOKEN TYPE FROM
1 !                                          ALPHA TO ALPHA_STR_TOKEN.
1 !  07  17-MAY-78      K.D. MORSE           NO CHANGES FOR VERS 21.
1 !  08  18-MAY-78      K.D. MORSE           NO CHANGES FOR VERS 22-23.
1 !                                          DBGLEX.B32 BECAME DBGMAR.B32.
1 !  09  18-MAY-78      K.D. MORSE           NO CHANGES FOR VERS 24.
1 !  10  13-JUN-78      K.D. MORSE           ADD FAO COUNT TO SIGNALS.
1 !  11  27-JUN-78      K.D. MORSE           NO CHANGES FOR VERS 25.
1 !
1 !--
```

```
;    71        0070  1 !
;    72        0071  1 ! TABLE OF CONTENTS:
;    73        0072  1 !
;    74        0073  1 FORWARD ROUTINE
;    75        0074  1         PAT$MAR_GET_LEX;                                  ! Extracts a MARS lexeme from the input buff
;    76        0075  1
;    77        0076  1 !
;    78        0077  1 ! INCLUDE FILES:
;    79        0078  1 !
;    80        0079  1
;    81        0080  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    82        0081  1 REQUIRE 'SRC$:VXSMAC.REQ';
;    83        0146  1 REQUIRE 'SRC$:PATPCT.REQ';
;    84        0186  1 REQUIRE 'SRC$:CHRKEY.REQ';
;    85        0247  1 REQUIRE 'SRC$:PATGEN.REQ';
;    86        0469  1 REQUIRE 'SRC$:PATTER.REQ';
;    87        0676  1 REQUIRE 'SRC$:SCALIT.REQ';
;    88        0742  1 REQUIRE 'LIB$:PATDEF.REQ';                        ! Defines literals
;    89        0796  1 REQUIRE 'LIB$:PATMSG.REQ';
;    90        0970  1 REQUIRE 'SRC$:SYSSER.REQ';
;  R1002  1      SWITCHES LIST (SOURCE);
;  R1003  1
;  R1004  1      EXTERNAL ROUTINE
;  R1005  1              PAT$fao_out;              ! formats a line and outputs to the terminal
;  R1006  1
;    91           1052  1
;    92           1053  1  !
;    93           1054  1  ! MACROS:
;    94           1055  1  !
;    95           1056  1
;    96           1057  1  !
;    97           1058  1  ! EQUATED SYMBOLS:
;    98           1059  1  !
;    99           1060  1
;   100           1061  1  !
;   101           1062  1  ! OWN STORAGE:
;   102           1063  1  !
;   103           1064  1
;   104           1065  1  !
;   105           1066  1  ! EXTERNAL REFERENCES:
;   106           1067  1  !
;   107           1068  1 EXTERNAL ROUTINE
;   108           1069  1         PAT$RADX_CONVRT;                                  ! Converts ASCII strings to binary numbers
;   109           1070  1
;   110           1071  1 EXTERNAL
;   111           1072  1         PAT$GB_DEF_MOD : VECTOR [, BYTE],                 ! Mode structure
;   112           1073  1         PAT$GB_MOD_PTR : REF VECTOR [, BYTE];             ! Holds current radix
```

```
1 GLOBAL ROUTINE PAT$MAR_GET_LEX (input_stg_desc, lexeme_stg_desc) =        ! gets a lexeme from input line
1
1 !++
1 ! Functional description:
1 !
1 !       Using the character pointer for the input line, extracts a lexeme
1 !       from the input line. A lexeme is defined as an operator, an
1 !       alphanumeric string, a numeric string, or an
1 !       illegal string. Blanks and comments are absorbed.
1 !
1 !       The lexeme is returned in the lexeme buffer in the
1 !       same form as in the input string, except for numeric
1 !       strings, in which case the string is converted to a
1 !       binary number and that is returned in the lexeme buffer.
1 !       A token equivalent of the lexeme is the value of the
1 !       routine.
1 !
1 ! Calling Sequence:
1 !
1 !       CALL get_MAR_lexeme (input_stg_desc.rt.dd, lexeme_stg_desc.rt.dv)
1 !
1 ! Formal parameters:
1 !
1 !       input_stg_desc  - string descriptor to the input buffer.
1 !       lexeme_stg_desc - varying string descriptor to the lexeme buffer
1 !
1 ! Implicit inputs:
1 !
1 !       The character mapping table, char_type_table, that maps each
1 !       ASCII character onto a dense list of equivalents.
1 !       The token_table, that maps operators onto their token equivalents.
1 !
1 ! Outputs:
1 !
1 !       input_stg_desc  - the field dsc$a_pointer is updated to point to
1 !                         the next byte to be read in the input stream.
1 !                         This byte is the delimiter of the lexeme found.
1 !                         The field dsc$w_length contains the length of
1 !                         the yet unread input line.
1 !       lexeme_stg_desc - the field dsc$w_length holds the actual length
1 !                         in bytes of the lexeme found. The lexeme buffer
1 !                         addressed by the field dsc$a_pointer holds the
1 !                         lexeme string or value.
1 !
1 ! Implicit outputs:
1 !
1 !       The ASCII representation of the lexeme is written into the
1 !       string addressed by the dsc$a_pointer field of lexeme_stg_desc.
1 !
1 ! Routine value:
1 !
1 !       the type of lexeme found, namely number, alpha string,
1 !       operator, keyword token, illegal.
1 !
1 ! Side effects:
1 !
1 !       none
```

```
1 !--
1
2 BEGIN
2
2 LITERAL
2         table_offset     = 9,
2         operator_max     = 28;
2
2 BIND
2         token_table = UPLIT BYTE (
2                                   op_paren_token,
2                                   cl_paren_token,
2                                   plus_token,
2                                   minus_token,
2                                   slash_token,
2                                   colon_token,
2                                   semi_colo_token,
2                                   quote_token,
2                                   up_arrow_token,
2                                   backslash_token,
2                                   at_sign_token,
2                                   period_token,
2                                   asterisk_token,
2                                   langle_token,
2                                   rangle_token,
2                                   comma_token,
2                                   equals_token,
2                                   lsquare_token,
2                                   rsquare_token,
2                                   hash_token
2                                                       ) : VECTOR [, BYTE];
2
2 LITERAL
2         max_state_index = 4,                                          ! index ranges from 0 to 4
2         invalid_state   = 0,                                          ! invalid character seen
2         alpha_state     = 1,                                          ! alphabetic string expected
2         numeric_state   = 2,                                          ! numeric string expected
2         eol_token_state = 3,                                          ! logical end of line or error seen
2         radix_state     = 4,                                          ! radix setting expected
2         unspec_state    = 5;                                          ! unspecified state, probably special charac
2
2 BIND
2         lex_type_tbl    = UPLIT (
2                                   mask (illegal),
2                                   mask (alpha, alpha_low, alpha_and_hex, alphalo_and_hex, period),
2                                   mask (numeric),
2                                   mask (ind_comment, end_of_line),
3                                   mask (up_arrow)
2                                                             ) : VECTOR;
2
2 BIND
2         lex_state_tbl   = UPLIT BYTE (
2                                         invalid_state,
2                                         alpha_state,
2                                         numeric_state,
2                                         eol_token_state,
2                                         radix_state
;  228         1188  2                                                         ) : VECTOR [, BYTE];
;  229         1189  2
;  230         1190  2 LITERAL
;  231         1191  2         radix_max       = 3;                                    ! maximum number of MARS radices
;  232         1192  2
;  233         1193  2 BIND
;  234         1194  2         radix_equiv_tbl = UPLIT BYTE (
;  235         1195  2                                         'B', binary_radix,
;  236         1196  2                                         'O', octal_radix,
;  237         1197  2                                         'D', decimal_radix,
;  238         1198  2                                         'X', hex_radix
;  239         1199  2                                                         ) : BLOCK [, WORD];
;  240         1200  2
;  241         1201  2 MACRO
;  242         1202  2         radix_char      = 0, 8, 0%,                             ! radix ASCII character
;  243         1203  2         radix_equiv     = 8, 8, 0%;                             ! radix equivalent
;  244         1204  2
;  245         1205  2 MAP
;  246         1206  2         input_stg_desc  : REF BLOCK [, BYTE],                   ! input string descriptor
;  247         1207  2         lexeme_stg_desc : REF BLOCK [, BYTE];                   ! lexeme string descriptor
;  248         1208  2
;  249         1209  2 LOCAL
;  250         1210  2         input_ptr,                                              ! character pointer for input
;  251         1211  2         lexeme_ptr,                                             ! character pointer for lexeme
;  252         1212  2         previous_radix,                                         ! current local radix
;  253         1213  2         state_index,                                            ! index into lex_state_tbl
;  254         1214  2         state,                                                  ! current state of lexical processor
;  255         1215  2         char,                                                   ! holds a single character
;  256         1216  2         count;                                                  ! counts characters used
;  257         1217  2
;  258         1218  2 LABEL
;  259         1219  2         alpha_block,                                            ! label for alpha case in the select
;  260         1220  2         radix_block;                                            ! label for up arrow case in the select
;  261         1221  2
;  262         1222  2 !++
;  263         1223  2 ! See whether there is any input line left. If not, signal internal error.
;  264         1224  2 !--
;  265         1225  2 IF .input_stg_desc [dsc$w_length] LSS 0
;  266         1226  2 THEN SIGNAL (PAT$_PARSEERR);
; INFO#252            L1:1225
; Test expression is always false
;  267         1227  2
;  268         1228  2 !++
;  269         1229  2 ! Make the string pointers into formal BLISS character pointers.
;  270         1230  2 !--
;  271         1231  2 input_ptr = ch$ptr (.input_stg_desc [dsc$a_pointer]);
;  272         1232  2 lexeme_ptr = ch$ptr (.lexeme_stg_desc [dsc$a_pointer]);
;  273         1233  2
;  274         1234  2 !++
;  275         1235  2 ! Save the radix in case it changes temporarily.
;  276         1236  2 !--
;  277         1237  2 previous_radix = .PAT$gb_mod_ptr [mode_radix];
;  278         1238  2 count = 0;
;  279         1239  2 REPEAT                                                          ! skip leading blanks
;  280         1240  3         BEGIN
;  281         1241  3         char = ch$rchar (.input_ptr);
;  282         1242  3         IF .char_type_table [.char] NEQ blanks
;  283       1243  3                THEN
;  284       1244  4                        BEGIN
;  285       1245  4                        input_stg_desc [dsc$w_length] = .input_stg_desc [dsc$w_length] - .count;
;  286       1246  4                        EXITLOOP
;  287       1247  4                        END
;  288       1248  3                ELSE
;  289       1249  4                        BEGIN
;  290       1250  4                        input_ptr = ch$plus (.input_ptr, 1);
;  291       1251  4                        count = .count + 1;
;  292       1252  3                        END;
;  293       1253  2                END;
;  294       1254  2
;  295       1255  2        !++
;  296       1256  2        ! Convert the mapping of the first significant character into a lexical state.
;  297       1257  2        ! This state drives the later CASE processing.
;  298       1258  2        !--
;  299       1259  2        state_index = 0;
;  300       1260  2        REPEAT
;  301       1261  3                BEGIN
;  302       1262  3                IF .lex_type_tbl [.state_index] ^ .char_type_table [.char] LSS 0
;  303       1263  3                THEN
;  304       1264  4                        BEGIN
;  305       1265  4                        state = .lex_state_tbl [.state_index];
;  306       1266  4                        EXITLOOP
;  307       1267  4                        END
;  308       1268  3                ELSE
;  309       1269  4                        BEGIN
;  310       1270  4                        state_index = .state_index + 1;
;  311       1271  4                        IF .state_index GTR max_state_index
;  312       1272  4                        THEN
;  313       1273  5                                BEGIN
;  314       1274  5                                state = unspec_state;
;  315       1275  5                                EXITLOOP
;  316       1276  4                                END;
;  317       1277  3                        END;
;  318       1278  2                END;
;  319       1279  2
;  320       1280  2        REPEAT CASE .state FROM 0 to max_state_index + 1 OF                 ! analyze current state
;  321       1281  2                SET
;  322       1282  2
;  323       1283  2                [invalid_state]:                                            ! if illegal, just signal
;  324       1284  2                        SIGNAL (PAT$_INVCHAR);
```

```
  2                         [alpha_state]:                                      ! alphanumeric string
  2     alpha_block:
  3                         BEGIN
  3                         !++
  3                         ! This is an alphanumeric string.  If the
  3                         ! character is a period, see whether the next character is
  3                         ! an alphabetic. If it is, this must be a logical operator
  3                         ! keyword, so allow the leading dot.  Otherwise, it is an error.
  3                         !--
  3                         LOCAL
  3                                   period_present;
  3
  3                         count = 0;
  3                         IF .char EQL asc_period
  3                         THEN
  4                                   BEGIN
  4
  4                                   LOCAL
  4                                             new_char;
  4
  4                                   new_char = ch$rchar (ch$plus (.input_ptr, 1));
P 5                                   IF NOT (oneof (.char_type_table [.new_char], alpha, alpha_low,
  5                                                    alpha_and_hex, alphalo_and_hex))
  4                                   THEN
  5                                             BEGIN
  5 !                                           IF .char_type_table [.new_char] EQL numeric
  5 !                                           THEN state = numeric_state
  5 !                                           ELSE state = unspec_state;
  5                                             STATE = UNSPEC_STATE;               ! DON'T ACCEPT NUMBERS WITH DECIMAL POINTS
  5                                             LEAVE alpha_block;
  5                                             END
  4                                   ELSE period_present = TRUE;
  4                                   END
  3                         ELSE period_present = FALSE;
  3
  3                         !++
  3                         ! Now read the input buffer until a non-alpha and non-numeric
  3                         ! character is encountered. Store each character found in the
  3                         ! buffer for the lexeme unless the length of that buffer is
  3                         ! expended.
  3                         !--
  3                         DO
  4                                   BEGIN
  5                                   IF (oneof (.char_type_table [.char], alpha_low, alphalo_and_hex))
  4                                   THEN char = .char - upper_case_dif;
  4                                   count = .count + 1;
  4                                   IF .count LEQ sym_max_length
  4                                   THEN ch$wchar_a (.char, lexeme_ptr);
  4                                   char = ch$a_rchar (input_ptr);
  4                                   END
  3                         WHILE
P 4                                   (oneof (.char_type_table [.char], alpha, alpha_low, numeric,
  3                                                    alpha_and_hex, alphalo_and_hex, period));
  3
  3                         !++
  3                         ! Now see whether the next character is a period
  3                         ! AND the string started with a period. In this case, store the
```

```
;  383        1342  3                 ! ending period also.
;  384        1343  3                 !--
;  385        1344  3 !               IF .char EQL asc_period AND .period_present
;  386        1345  3 !               THEN
;  387        1346  3 !                       BEGIN
;  388        1347  3 !                       count = .count + 1;
;  389        1348  3 !                       IF .count LEQ sym_max_length
;  390        1349  3 !                       THEN ch$wchar (.char, .lexeme_ptr);
;  391        1350  3 !                       input_ptr = ch$plus (.input_ptr, 1);
;  392        1351  3 !                       END;
;  393        1352  3
;  394        1353  3                 !++
;  395        1354  3                 ! Return the alpha_str_token lexeme.
;  396        1355  3                 !--
;  397        1356  3                 IF .count GTR sym_max_length
;  398        1357  3                 THEN
;  399        1358  4                         BEGIN
;  400        1359  4                         SIGNAL (PAT$_STGTRUNC);
;  401        1360  4                         lexeme_stg_desc [dsc$w_length] = sym_max_length;
;  402        1361  4                         END
;  403        1362  3                 ELSE lexeme_stg_desc [dsc$w_length] = .count;
;  404        1363  3                 input_stg_desc [dsc$a_pointer] = .input_ptr;
;  405        1364  3                 input_stg_desc [dsc$w_length] = .input_stg_desc [dsc$w_length] - .count;
;  406        1365  3                 RETURN alpha_str_token
;  407        1366  2                 END;
;  409        1367  2     [numeric_state]:                                        ! numeric string
;  410        1368  3             BEGIN
;  411        1369  3             !++
;  412        1370  3             ! Now read the input buffer until a non-numeric character is
;  413        1371  3             ! encountered. Ignore all leading zeroes unless a decimal point
;  414        1372  3             ! was present.  Store each character found in the buffer for
;  415        1373  3             ! the lexeme unless the length of that buffer is expended.
;  416        1374  3             !--
;  417        1375  3             count = 0;
;  418        1376  3             WHILE
;  419        1377  3                     .char EQL '0'
;  420        1378  3             DO
;  421        1379  4                     BEGIN
;  422        1380  4                     count = .count + 1;
;  423        1381  4                     char = ch$a_rchar (input_ptr);
;  424        1382  3                     END;
;  425        1383  3
;  426        1384  3             !++
;  427        1385  3             ! If the entire number was zero, put a single
;  428        1386  3             ! zero in the lexeme buffer and return.
;  429        1387  3             !--
;  430        1388  3             input_stg_desc [dsc$w_length] = .input_stg_desc [dsc$w_length] - .count;
;  431        1389  3             count = 0;
;  432        1390  3             IF .char_type_table [.char] NEQ numeric
;  433        1391  4                     AND NOT (oneof (.char_type_table [.char], alpha_and_hex, alphalo_and_hex))
;  434        1392  3             THEN
;  435        1393  4                     BEGIN
;  436        1394  4                     ch$wchar (0, .lexeme_ptr);
;  437        1395  4                     lexeme_stg_desc [dsc$w_length] = 1;
;  438        1396  4                     input_stg_desc [dsc$a_pointer] = .input_ptr;
;  439        1397  4                     RETURN digit_str_token
;  440        1398  3                     END;
;  441        1399  3
;  442        1400  3             !++
;  443        1401  3             ! This is the normal store and pick up next numeric character.
;  444        1402  3             !--
;  445        1403  3             DO
;  446        1404  4                     BEGIN
;  447        1405  4                     IF .char_type_table [.char] EQL alphalo_and_hex
;  448        1406  4                     THEN char = .char - upper_case_dif;
;  449        1407  4                     count = .count + 1;
;  450        1408  4                     IF .count GTR num_max_length
;  451        1409  4                     THEN
;  452        1410  5                             BEGIN
;  453        1411  5                             ch$move (num_max_length - 1,
;  454        1412  5                                     ch$plus (ch$ptr (.lexeme_stg_desc [dsc$a_pointer]), 1),
;  455        1413  5                                     ch$ptr (.lexeme_stg_desc [dsc$a_pointer]));
;  456        1414  5                             ch$wchar (.char, .lexeme_ptr-1);
;  457        1415  5                             END
;  458        1416  4                     ELSE ch$wchar_a (.char, lexeme_ptr);
;  459        1417  4                     char = ch$a_rchar (input_ptr);
;  460        1418  4                     END
;  461        1419  3             WHILE
;  462      P 1420  4                     (oneof (.char_type_table [.char], numeric,
;  463        1421  3                             alpha_and_hex, alphalo_and_hex));
;  464        1422  3
;  465        1423  3
;  466        1424  3                 !++
;  467        1425  3                 ! Convert the number, restore the old radix,
;  468        1426  3                 ! and return the numeric lexeme.
;  469        1427  3                 !--
;  470        1428  3                 PAT$radx_convrt (.lexeme_stg_desc [dsc$a_pointer],
;  471        1429  3                         .lexeme_stg_desc [dsc$a_pointer]);
;  472        1430  3                 PAT$gb_mod_ptr [mode_radix] = .previous_radix;
;  473        1431  3                 lexeme_stg_desc [dsc$w_length] = 4;
;  474        1432  3                 input_stg_desc [dsc$a_pointer] = .input_ptr;
;  475        1433  3                 input_stg_desc [dsc$w_length] = .input_stg_desc [dsc$w_length] - .count;
;  476        1434  3                 RETURN digit_str_token
;  477        1435  2         END;
;  479          1436  2         [eol_token_state]:                                      ! logical end of line
;  480          1437  3                 BEGIN
;  481          1438  3                 !++
;  482          1439  3                 ! The length of the input line should be set to zero here.
;  483          1440  3                 ! Reduce it one so that it is less than zero. This will cause
;  484          1441  3                 ! an error if this same input line ever comes back to the lex
;  485          1442  3                 ! routine.
;  486          1443  3                 !--
;  487          1444  3                 lexeme_stg_desc [dsc$w_length] = 0;
;  488          1445  3                 input_stg_desc [dsc$a_pointer] = ch$plus (.input_ptr, 1);
;  489          1446  3                 input_stg_desc [dsc$w_length] = .input_stg_desc [dsc$w_length] - 1;
;  490          1447  3                 RETURN eol_token
;  491          1448  2                 END;
;  493        1449 2          [radix_state]:                                          ! up arrow, quote, percent sign
;  494        1450 2
;  495        1451 3  radix_block:    BEGIN                                           ! MARS handling
;  496        1452 3
;  497        1453 3                  !++
;  498        1454 3                  ! An up arrow can occur as a standalone character meaning
;  499        1455 3                  ! previous location, or as a special character that indicates
;  500        1456 3                  ! radix. In the latter case, the up arrow is followed by one of
;  501        1457 3                  ! the letters 'B', 'O', or 'X', and then a numeric string
;  502        1458 3                  ! (without an intervening space).  First check for the letter.
;  503        1459 3                  !--
;  504        1460 3                  LOCAL
;  505        1461 3                          new_char;
;  506        1462 3
;  507        1463 3                  char = ch$rchar (ch$plus (.input_ptr, 1));
;  508        1464 4                  IF (oneof (.char_type_table [.char], alpha_low, alphalo_and_hex))
;  509        1465 3                  THEN char = .char - upper_case_dif;
;  510        1466 4                  IF NOT ((.char EQL 'B') OR (.char EQL 'O') OR (.char EQL 'D') OR (.char EQL 'X'))
;  511        1467 3                  THEN
;  512        1468 4                          BEGIN
;  513        1469 4                          !++
;  514        1470 4                          ! This is the single character meaning previous location.
;  515        1471 4                          ! Just update the string descriptors, write the up arrow
;  516        1472 4                          ! into the lexeme buffer, and return.
;  517        1473 4                          !--
;  518        1474 4                          char = asc_up_arrow;
;  519        1475 4                          state = unspec_state;
;  520        1476 4                          LEAVE radix_block;
;  521        1477 3                          END;
;  522        1478 3
;  523        1479 3                  !++
;  524        1480 3                  ! This looks like a radix indicator. If a number follows, it
;  525        1481 3                  ! must be. In this case, set the current mode according to the
;  526        1482 3                  ! radix encoding. Then leave this code block. The effect is that
;  527        1483 3                  ! on the next loop through the CASE expression, control will
;  528        1484 3                  ! stop at the numeric processing block.
;  529        1485 3                  !--
;  530        1486 3                  new_char = ch$rchar (ch$plus (.input_ptr, 2));
;  531      P 1487 4                  IF (oneof (.char_type_table [.new_char], numeric,
;  532        1488 4                                  alpha_and_hex, alphalo_and_hex))
;  533        1489 3                  THEN
;  534        1490 4                          BEGIN
;  535        1491 4                          input_ptr = ch$plus (.input_ptr, 2);
;  536        1492 4                          INCR index FROM 0 TO radix_max DO
;  537        1493 4                                  IF .char EQL .radix_equiv_tbl [.index, radix_char]
;  538        1494 4                                  THEN
;  539        1495 5                                          BEGIN
;  540        1496 5                                          PAT$gb_mod_ptr [mode_radix] =
;  541        1497 5                                                      .radix_equiv_tbl [.index, radix_equiv];
;  542        1498 5                                          EXITLOOP
;  543        1499 4                                          END;
;  544        1500 4                          char = .new_char;
;  545        1501 4                          input_stg_desc [dsc$w_length] = .input_stg_desc [dsc$w_length] - 2;
;  546        1502 4                          state = numeric_state;
;  547        1503 4                          LEAVE radix_block;
;  548        1504 4                          END
;  549        1505 3                  ELSE
;  550          1506  4                          BEGIN
;  551          1507  4                          !++
;  552          1508  4                          ! This is not a radix indicator after all. Just return
;  553          1509  4                          ! the up arrow.
;  554          1510  4                          !--
;  555          1511  4                          char = asc_up_arrow;
;  556          1512  4                          state = unspec_state;
;  557          1513  4                          LEAVE radix_block;
;  558          1514  3                          END;
;  559          1515  2                  END;
;  561          1516  2         [unspec_state]:                                             ! special character like operator or illegal
;  562          1517  3                 BEGIN
;  563          1518  3                 !++
;  564          1519  3                 ! Most likely, this is a single character operator. Write its
;  565          1520  3                 ! ASCII value into the lexeme buffer, and return its equivalent
;  566          1521  3                 ! token.
;  567          1522  3                 !--
;  568          1523  3                 IF .char_type_table [.char] GEQ table_offset
;  569          1524  3                         AND .char_type_table [.char] LEQ operator_max
;  570          1525  3                 THEN
;  571          1526  4                         BEGIN
;  572          1527  4
;  573          1528  4                         LOCAL
;  574          1529  4                                 index;
;  575          1530  4
;  576          1531  4                         index = table_offset;
;  577          1532  4                         REPEAT
;  578          1533  5                                 BEGIN
;  579          1534  5                                 IF .char_type_table [.char] EQL .index
;  580          1535  5                                 THEN
;  581          1536  6                                         BEGIN
;  582          1537  6                                         ch$wchar (.char, .lexeme_ptr);
;  583          1538  6                                         lexeme_stg_desc [dsc$w_length] = 1;
;  584          1539  6                                         input_stg_desc [dsc$a_pointer] = ch$plus (.input_ptr, 1);
;  585          1540  6                                         input_stg_desc [dsc$w_length] = .input_stg_desc [dsc$w_length] - 1;
;  586          1541  6                                         RETURN .token_table [.index - table_offset]
;  587          1542  6                                         END
;  588          1543  5                                 ELSE index = .index + 1;
;  589          1544  5                                 IF .index GTR operator_max
;  590          1545  5                                 THEN EXITLOOP;
;  591          1546  4                                 END;
;  592          1547  3                         END;
;  593          1548  3
;  594          1549  3                 !++
;  595          1550  3                 ! This doesn't seem to be anything about which we know.
;  596          1551  3                 ! SIGNAL invalid character.
;  597          1552  3                 !--
;  598          1553  3                 SIGNAL (PAT$_INVCHAR);
;  599          1554  2                 END;
;  600          1555  2
;  601          1556  2         TES;
;  602          1557  2
;  603          1558  1 END;                                                                ! end of get_mar_lexeme
; INFO#212              L1:1278
; Null expression appears in value-required context


                                                                       .TITLE  PATLEX
                                                                       .IDENT  \V04-000\

                                                                       .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

00 06 06 06 06 04 00 00 00 00 00 00 00 00 06 00000 P.AAA:   .BYTE   6, 0, 0, 0, 0, 0, 0, 0, 0, 4, 6, 6, 6, 6, - ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000F                   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, - ;
18 0B 15 0A 09 10 00 00 01 1C 10 05 04 00 00 0001E                   0, 0, 0, 0, 4, 5, 16, 28, 1, 0, 0, 16, 9, - ;
0F 0E 02 02 02 02 02 02 02 02 02 0D 14 0C 0002D                      10, 21, 11, 24, 12, 20, 13, 2, 2, 2, 2, - ;
01 01 01 01 03 03 03 03 03 03 13 00 17 19 16 0003C                   2, 2, 2, 2, 2, 2, 14, 15, 22, 25, 23, 0, - ;
```

```
01  01  01  01  01  01  01  01  01  01  01  01  01  01  01  0004B                   19, 3, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, -        ;
07  07  08  08  08  08  08  08  00  01  11  1B  12  1A  01  0005A                   1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -      ;
07  07  07  07  07  07  07  07  07  07  07  07  07  07  07  00069                   26, 18, 27, 17, 1, 0, 8, 8, 8, 8, 8, 8, -       ;
                             00  00  00  00  00  07  07  07  00078                   7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, -     ;
                                                                                    7, 7, 7, 7, 7, 7, 0, 0, 0, 0, 0                 ;
4F  44  3C  4B  3D  3E  53  4D  51  40  52  46  4C  3F  49  00080 P.AAB:   .BYTE    73, 63, 76, 70, 82, 64, 81, 77, 83, 62, -       ;
                                        43  50  45  42  41  0008F                   61, 75, 60, 68, 79, 65, 66, 69, 80, 67          ;
          00004000  06000000  20000000  51800800  80000000  00094 P.AAC:   .LONG    -2147483648, 1367345152, 536870912, -           ;
                                                                                    100663296, 16384                                ;
                                        04  03  02  01  00  000A8 P.AAD:   .BYTE    0, 1, 2, 3, 4                                   ;
                                                        42  000AD P.AAE:   .ASCII   \B\                                             ;
                                                        02  000AE          .BYTE    2                                               ;
                                                        4F  000AF          .ASCII   \O\                                             ;
                                                        08  000B0          .BYTE    8                                               ;
                                                        44  000B1          .ASCII   \D\                                             ;
                                                        0A  000B2          .BYTE    10                                              ;
                                                        58  000B3          .ASCII   \X\                                             ;
                                                        10  000B4          .BYTE    16                                              ;

                                                                  CHAR_TYPE_TABLE=    P.AAA
                                                                  TOKEN_TABLE=        P.AAB
                                                                  LEX_TYPE_TBL=       P.AAC
                                                                  LEX_STATE_TBL=      P.AAD
                                                                  RADIX_EQUIV_TBL=    P.AAE
                                                                           .EXTRN   PAT$FAO_OUT, PAT$RADX_CONVRT
                                                                           .EXTRN   PAT$GB_DEF_MOD, PAT$GB_MOD_PTR

                                                                           .PSECT   _PAT$CODE,NOWRT,2

                                                     0FFC 00000            .ENTRY   PAT$MAR_GET_LEX, Save R2,R3,R4,R5,R6,R7,R8,-; 1074
                                                                                    R9,R10,R11                                  ;
                          5E              04    C2 00002                   SUBL2    #4, SP                                      ;
                          57         04   AC    D0 00005                   MOVL     INPUT_STG_DESC, R7                          ; 1231
                                     04   A7    9F 00009                   PUSHAB   4(R7)                                       ;
                          58         00   BE    D0 0000C                   MOVL     @0(SP), INPUT_PTR                           ;
                          59         08   AC    D0 00010                   MOVL     LEXEME_STG_DESC, R9                         ; 1232
                                     04   A9    DD 00014                   PUSHL    4(R9)                                       ;
                          7E 00000000G    FF    9A 00017                   MOVZBL   @PAT$GB_MOD_PTR, PREVIOUS_RADIX             ; 1237
                                          5A    D4 0001E                   CLRL     COUNT                                       ; 1238
                          56              68    9A 00020 1$:               MOVZBL   (INPUT_PTR), CHAR                           ; 1241
                          04 00000000'EF46      91 00023                   CMPB     CHAR_TYPE_TABLE[CHAR], #4                   ; 1242
                                          05    13 0002B                   BEQL     2$                                          ;
                          67              5A    A2 0002D                   SUBW2    COUNT, (R7)                                 ; 1245
                                          06    11 00030                   BRB      3$                                          ; 1244
                                          58    D6 00032 2$:               INCL     INPUT_PTR                                   ; 1250
                                          5A    D6 00034                   INCL     COUNT                                       ; 1251
                                          E8    11 00036                   BRB      1$                                          ; 1238
                                          50    D4 00038 3$:               CLRL     STATE_INDEX                                 ; 1259
          51 00000000'EF40 00000000'EF46        78 0003A 4$:               ASHL     CHAR_TYPE_TABLE[CHAR], LEX_TYPE_TBL-        ; 1262
                                                                                    [STATE_INDEX], R1                           ;
                                          0B    18 00048                   BGEQ     5$                                          ;
                     0C      AE 00000000'EF40   9A 0004A                   MOVZBL   LEX_STATE_TBL[STATE_INDEX], STATE           ; 1265
                                          0B    11 00053                   BRB      7$                                          ; 1264
                                          50    D6 00055 5$:               INCL     STATE_INDEX                                 ; 1270
                          04              50    D1 00057                   CMPL     STATE_INDEX, #4                             ; 1271
                                          DE    15 0005A                   BLEQ     4$                                          ;
                     0C      AE           05    D0 0005C 6$:               MOVL     #5, STATE                                   ; 1274
```

```
                            05     00        0C   AE  CF 00060 7$:     CASEL    STATE, #0, #5                                 ; 1280
        0135              0090   000F           0200     00065 8$:     .WORD    37$-8$,-                                      ;
                                 01C8           0143     0006D                  9$-8$,-                                       ;
                                                                                17$-8$,-                                      ;
                                                                                26$-8$,-                                      ;
                                                                                27$-8$,-                                      ;
                                                                                34$-8$                                        ;
                                                01F1  31 00071         BRW      37$                                           ; 1284
                                                  5A  D4 00074 9$:     CLRL     COUNT                                         ; 1297
                                   2E             56  D1 00076         CMPL     CHAR, #46                                     ; 1298
                                                  18  12 00079         BNEQ     10$                                           ;
                                   50        01   A8  9A 0007B         MOVZBL   1(INPUT_PTR), NEW_CHAR                        ; 1305
                   50 51800000     8F  00000000'EF40  78 0007F         ASHL     CHAR_TYPE_TABLE[NEW_CHAR], #1367343104, R0    ; 1307
                                                  CE  18 0008C         BGEQ     6$                                            ;
                                   50             01  D0 0008E         MOVL     #1, PERIOD_PRESENT                            ; 1316
                                                  02  11 00091         BRB      11$                                           ; 1298
                                                  50  D4 00093 10$:    CLRL     PERIOD_PRESENT                                ; 1318
                                   51  00000000'EF46  9A 00095 11$:    MOVZBL   CHAR_TYPE_TABLE[CHAR], R1                     ; 1328
                   50 01800000     8F             51  78 0009D 12$:    ASHL     R1, #25165824, R0                             ;
                                                  03  18 000A5         BGEQ     13$                                           ;
                                   56             20  C2 000A7         SUBL2    #32, CHAR                                     ; 1329
                                                  5A  D6 000AA 13$:    INCL     COUNT                                         ; 1330
                                   1F             5A  D1 000AC         CMPL     COUNT, #31                                    ; 1331
                                                  07  14 000AF         BGTR     14$                                           ;
                            04     BE             56  90 000B1         MOVB     CHAR, @LEXEME_PTR                             ; 1332
                                             04   AE  D6 000B5         INCL     LEXEME_PTR                                    ;
                                                  58  D6 000B8 14$:    INCL     INPUT_PTR                                     ; 1333
                                   56             68  9A 000BA         MOVZBL   (INPUT_PTR), CHAR                             ;
                                   51  00000000'EF46  9A 000BD         MOVZBL   CHAR_TYPE_TABLE[CHAR], R1                     ; 1337
                   50 71800800     8F             51  78 000C5         ASHL     R1, #1904216064, R0                           ;
                                                  CE  19 000CD         BLSS     12$                                           ;
                                   1F             5A  D1 000CF         CMPL     COUNT, #31                                    ; 1356
                                                  12  15 000D2         BLEQ     15$                                           ;
                                       006D8033   8F  DD 000D4         PUSHL    #7176243                                      ; 1359
                     00000000G     00             01  FB 000DA         CALLS    #1, LIB$SIGNAL                                ;
                                   69             1F  B0 000E1         MOVW     #31, (R9)                                     ; 1360
                                                  03  11 000E4         BRB      16$                                           ; 1356
                                   69             5A  B0 000E6 15$:    MOVW     COUNT, (R9)                                   ; 1362
                            08     BE             58  D0 000E9 16$:    MOVL     INPUT_PTR, @8(SP)                             ; 1363
                                   67             5A  A2 000ED         SUBW2    COUNT, (R7)                                   ; 1364
                                   50        47   8F  9A 000F0         MOVZBL   #71, R0                                       ; 1365
                                                      04 000F4         RET                                                    ;
                                                  5A  D4 000F5 17$:    CLRL     COUNT                                         ; 1375
                                   30             56  D1 000F7 18$:    CMPL     CHAR, #48                                     ; 1377
                                                  09  12 000FA         BNEQ     19$                                           ;
                                                  5A  D6 000FC         INCL     COUNT                                         ; 1380
                                                  58  D6 000FE         INCL     INPUT_PTR                                     ; 1381
                                   56             68  9A 00100         MOVZBL   (INPUT_PTR), CHAR                             ;
                                                  F2  11 00103         BRB      18$                                           ; 1376
                                   67             5A  A2 00105 19$:    SUBW2    COUNT, (R7)                                   ; 1388
                                                  5A  D4 00108         CLRL     COUNT                                         ; 1389
                                   02  00000000'EF46  91 0010A         CMPB     CHAR_TYPE_TABLE[CHAR], #2                     ; 1390
                                                  1B  13 00112         BEQL     20$                                           ;
                   50 10800000     8F  00000000'EF46  78 00114         ASHL     CHAR_TYPE_TABLE[CHAR], #276824064, R0         ; 1391
                                                  0C  19 00121         BLSS     20$                                           ;
                                             04   BE  94 00123         CLRB     @LEXEME_PTR                                   ; 1394
                                   69             01  B0 00126         MOVW     #1, (R9)                                      ; 1395
```

```
                                08   BE            58  D0 00129        MOVL    INPUT_PTR, @8(SP)                           ; 1396
                                                   66  11 0012D        BRB     25$                                         ; 1397
                                     5B 00000000'EF46  9A 0012F 20$:   MOVZBL  CHAR_TYPE_TABLE[CHAR], R11                  ; 1405
                                     08            5B  D1 00137 21$:   CMPL    R11, #8                                     ;
                                                   03  12 0013A        BNEQ    22$                                         ;
                                     56            20  C2 0013C        SUBL2   #32, CHAR                                   ; 1406
                                                   5A  D6 0013F 22$:   INCL    COUNT                                       ; 1407
                                     14            5A  D1 00141        CMPL    COUNT, #20                                  ; 1408
                                                   13  15 00144        BLEQ    23$                                         ;
                                     50       04   A9  D0 00146        MOVL    4(R9), R0                                   ; 1412
                        60      01   A0            13  28 0014A        MOVC3   #19, 1(R0), (R0)                            ; 1413
                        50      04   AE            01  C3 0014F        SUBL3   #1, LEXEME_PTR, R0                          ; 1414
                                     60            56  90 00154        MOVB    CHAR, (R0)                                  ;
                                                   07  11 00157        BRB     24$                                         ; 1416
                                04   BE            56  90 00159 23$:   MOVB    CHAR, @LEXEME_PTR                           ;
                                              04   AE  D6 0015D        INCL    LEXEME_PTR                                  ;
                                                   58  D6 00160 24$:   INCL    INPUT_PTR                                   ; 1417
                                     56            68  9A 00162        MOVZBL  (INPUT_PTR), CHAR                           ;
                                     5B 00000000'EF46  9A 00165        MOVZBL  CHAR_TYPE_TABLE[CHAR], R11                  ; 1421
                        50 30800000  8F            5B  78 0016D        ASHL    R11, #813694976, R0                         ;
                                                   C0  19 00175        BLSS    21$                                         ;
                                              04   A9  DD 00177        PUSHL   4(R9)                                       ; 1429
                                              04   A9  DD 0017A        PUSHL   4(R9)                                       ; 1428
                           00000000G EF            02  FB 0017D        CALLS   #2, PAT$RADX_CONVRT                         ;
                           00000000G FF            6E  90 00184        MOVB    PREVIOUS_RADIX, @PAT$GB_MOD_PTR             ; 1430
                                     69            04  B0 0018B        MOVW    #4, (R9)                                    ; 1431
                                08   BE            58  D0 0018E        MOVL    INPUT_PTR, @8(SP)                           ; 1432
                                     67            5A  A2 00192        SUBW2   COUNT, (R7)                                 ; 1433
                                     50       48   8F  9A 00195 25$:   MOVZBL  #72, R0                                     ; 1434
                                                       04 00199        RET                                                 ;
                                                   69  B4 0019A 26$:   CLRW    (R9)                                        ; 1444
                                08   BE       01   A8  9E 0019C        MOVAB   1(R8), @8(SP)                               ; 1445
                                                   67  B7 001A1        DECW    (R7)                                        ; 1446
                                     50       63   8F  9A 001A3        MOVZBL  #99, R0                                     ; 1447
                                                       04 001A7        RET                                                 ;
                                     56       01   A8  9A 001A8 27$:   MOVZBL  1(INPUT_PTR), CHAR                          ; 1463
                        50 01800000  8F 00000000'EF46  78 001AC        ASHL    CHAR_TYPE_TABLE[CHAR], #25165824, R0        ; 1464
                                                   03  18 001B9        BGEQ    28$                                         ;
                                     56            20  C2 001BB        SUBL2   #32, CHAR                                   ; 1465
                           00000042  8F            56  D1 001BE 28$:   CMPL    CHAR, #66                                   ; 1466
                                                   1B  13 001C5        BEQL    29$                                         ;
                           0000004F  8F            56  D1 001C7        CMPL    CHAR, #79                                   ;
                                                   12  13 001CE        BEQL    29$                                         ;
                           00000044  8F            56  D1 001D0        CMPL    CHAR, #68                                   ;
                                                   09  13 001D7        BEQL    29$                                         ;
                           00000058  8F            56  D1 001D9        CMPL    CHAR, #88                                   ;
                                                   44  12 001E0        BNEQ    33$                                         ;
                                     51       02   A8  9A 001E2 29$:   MOVZBL  2(INPUT_PTR), NEW_CHAR                      ; 1486
                        50 30800000  8F 00000000'EF41  78 001E6        ASHL    CHAR_TYPE_TABLE[NEW_CHAR], #813694976, R0   ; 1488
                                                   31  18 001F3        BGEQ    33$                                         ;
                                     58            02  C0 001F5        ADDL2   #2, INPUT_PTR                               ; 1491
                                                   50  D4 001F8        CLRL    INDEX                                       ; 1493
                                        00000000'EF40  3F 001FA 30$:   PUSHAW  RADIX_EQUIV_TBL[INDEX]                      ;
          56            9E           08            00  ED 00201        CMPZV   #0, #8, @(SP)+, CHAR                        ;
                                                   0E  12 00206        BNEQ    31$                                         ;
                           00000000G FF 00000000'EF40  33 00208        CVTWB   RADIX_EQUIV_TBL+1[INDEX], @PAT$GB_MOD_PTR   ; 1497
                                                   04  11 00214        BRB     32$                                         ; 1495
```

```
                 E0      50               03  F3 00216 31$:    AOBLEQ  #3, INDEX, 30$                         ; 1493
                         56               51  D0 0021A 32$:    MOVL    NEW_CHAR, CHAR                         ; 1500
                         67               02  A2 0021D         SUBW2   #2, (R7)                               ; 1501
                   0C    AE               02  D0 00220         MOVL    #2, STATE                              ; 1502
                                          4C  11 00224         BRB     38$                                    ; 1503
                         56       5E      8F  9A 00226 33$:    MOVZBL  #94, CHAR                              ; 1511
                                        FE2F  31 0022A         BRW     6$                                     ; 1512
                         51 00000000'EF46     9A 0022D 34$:    MOVZBL  CHAR_TYPE_TABLE[CHAR], R1              ; 1523
                         09               51  91 00235         CMPB    R1, #9                                 ;
                                          2B  1F 00238         BLSSU   37$                                    ;
                         1C               51  91 0023A         CMPB    R1, #28                                ; 1524
                                          26  1A 0023D         BGTRU   37$                                    ;
                         50               09  D0 0023F         MOVL    #9, INDEX                              ; 1531
                         50               51  D1 00242 35$:    CMPL    R1, INDEX                              ; 1534
                                          17  12 00245         BNEQ    36$                                    ;
                   04    BE               56  90 00247         MOVB    CHAR, @LEXEME_PTR                      ; 1537
                         69               01  B0 0024B         MOVW    #1, (R9)                               ; 1538
                   08    BE       01      A8  9E 0024E         MOVAB   1(R8), @8(SP)                          ; 1539
                                          67  B7 00253         DECW    (R7)                                   ; 1540
                         50 00000000'EF40     9A 00255         MOVZBL  TOKEN_TABLE-9[INDEX], R0               ; 1541
                                              04 0025D         RET                                            ;
                                          50  D6 0025E 36$:    INCL    INDEX                                  ; 1543
                         1C               50  D1 00260         CMPL    INDEX, #28                             ; 1544
                                          DD  15 00263         BLEQ    35$                                    ;
                            006D80D2      8F  DD 00265 37$:    PUSHL   #7176402                               ; 1553
              00000000G  00               01  FB 0026B         CALLS   #1, LIB$SIGNAL                         ;
                                        FDEB  31 00272 38$:    BRW     7$                                     ; 1280

; Routine Size:  629 bytes,    Routine Base:  _PAT$CODE + 0000
;  605        1559  1 END                                                          ! End of module
;  606        1560  0 ELUDOM




                                                                 .EXTRN  LIB$SIGNAL

;                                  PSECT SUMMARY
;
;     Name                        Bytes                       Attributes
;
;  _PAT$PLIT                          181  NOVEC,NOWRT, RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(0)
;  _PAT$CODE                          629  NOVEC,NOWRT, RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                            Library Statistics
;
;                                            -------- Symbols --------          Pages          Processing
;     File                                   Total     Loaded   Percent         Mapped         Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1             18619         6         0         1000            00:01.9




; Information:   2
; Warnings:      0
; Errors:        0



;                                  COMMAND QUALIFIERS

;     BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/VARIANT:1/LIS=LIS$:PATLEX/OBJ=OBJ$:PATLEX MSRC$:PATLEX/UPDATE=(ENH$:PATLEX)

; Size:          629 code + 181 data bytes
; Run Time:         00:26.9
; Elapsed Time:     01:23.5
; Lines/CPU Min:     3478
; Lexemes/CPU-Min: 35021
; Memory Used:  282 pages
; Compilation Complete
```

0302 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PATPAR
LIS

PATMAC
LIS

PATMAI
LIS

PATLST
LIS

PATIO
LIS

PATLEX
LIS

PATMOD
LIS

PATMSG
LIS