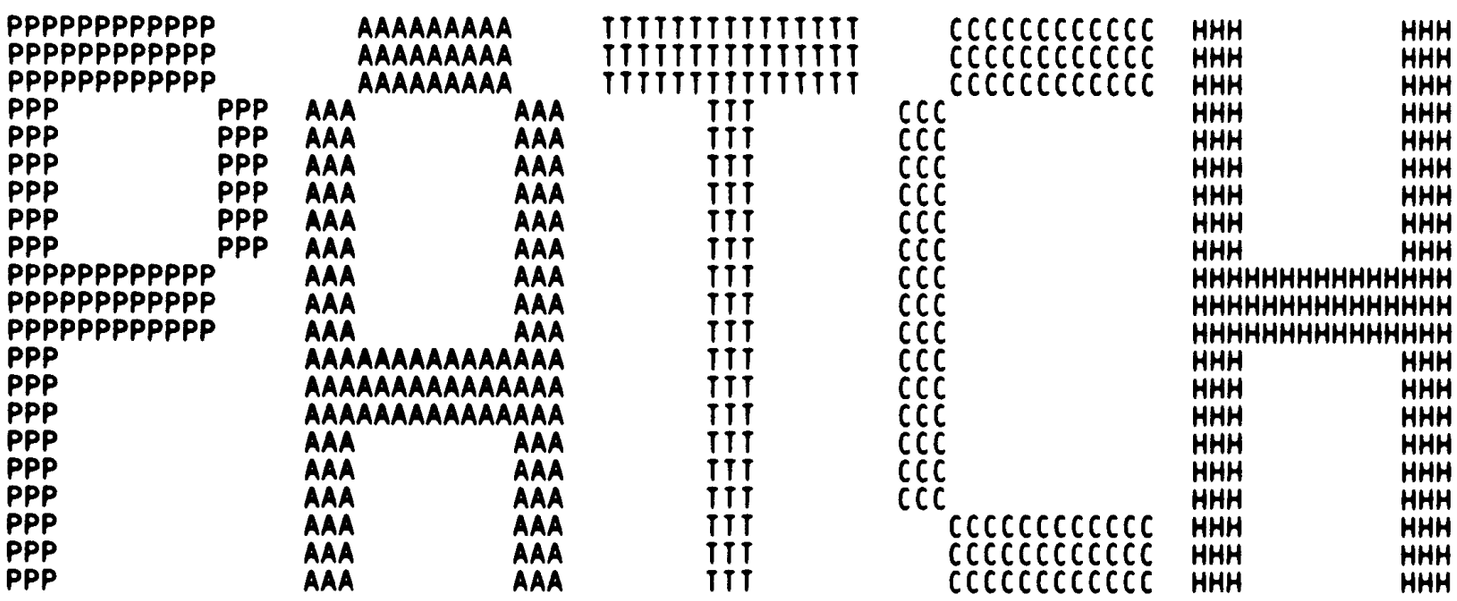
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPPPPPPPPPPP AAAAAAAAAAAAAAAA TTT CCC HHHHHHHHHHHHHHHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCCC HHH HHH

```
PPPPPPPP      AAAAAA    TTTTTTTTTT    SSSSSSSS    CCCCCCCC    AAAAAA
PPPPPPPP     AAAAAA     TTTTTTTTTT   SSSSSSSS    CCCCCCCC    AAAAAA
PP      PP   AA    AA       TT      SS          CC          AA    AA
PP      PP   AA    AA       TT      SS          CC          AA    AA
PP      PP   AA    AA       TT      SS          CC          AA    AA
PP      PP   AA    AA       TT      SS          CC          AA    AA
PPPPPPPP     AA    AA       TT        SSSSSS    CC          AA    AA
PPPPPPPP     AA    AA       TT        SSSSSS    CC          AA    AA
PP           AAAAAAAAAA     TT              SS  CC          AAAAAAAAAA
PP           AAAAAAAAAA     TT              SS  CC          AAAAAAAAAA
PP           AA    AA       TT              SS  CC          AA    AA      ....
PP           AA    AA       TT              SS  CC          AA    AA      ....
PP           AA    AA       TT      SSSSSSSS      CCCCCCCC  AA    AA      ....
PP           AA    AA       TT      SSSSSSSS      CCCCCCCC  AA    AA      ....

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
```

PAT
V04

```
;     1          0001  0 MODULE PATSCA (                                       ! Lexical scanner for PATCH
;     2        L 0002  0                 %IF %VARIANT EQL 1
;     3          0003  0                 %THEN
;     4          0004  0                         ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE),
;     5          0005  0                 %FI
;     6          0006  0                 IDENT = 'V04-000'
;     7          0007  0                 ) =
;     8          0008  1 BEGIN
;     9          0009  1
;    10          0010  1 !
;    11          0011  1 !*****************************************************************************
;    12          0012  1 !*                                                                           *
;    13          0013  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    14          0014  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    15          0015  1 !*  ALL RIGHTS RESERVED.                                                      *
;    16          0016  1 !*                                                                           *
;    17          0017  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    18          0018  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    19          0019  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    20          0020  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    21          0021  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    22          0022  1 !*  TRANSFERRED.                                                              *
;    23          0023  1 !*                                                                           *
;    24          0024  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    25          0025  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    26          0026  1 !*  CORPORATION.                                                              *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    29          0029  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    30          0030  1 !*                                                                           *
;    31          0031  1 !*                                                                           *
;    32          0032  1 !*****************************************************************************
;    33          0033  1 !
;    34          0034  1
;    35          0035  1 !++
;    36          0036  1 ! FACILITY:     PATCH
;    37          0037  1 !
;    38          0038  1 ! ABSTRACT:
;    39          0039  1 !
;    40          0040  1 !       This module contains the routine called by the parser to get a
;    41          0041  1 !       token from the input line. It also contains the routine that
;    42          0042  1 !       translates an alpha string into a keyword token.
;    43          0043  1 !
;    44          0044  1 ! ENVIRONMENT:  STARLET, user mode, interrupts disabled.
;    45          0045  1 !
;    46          0046  1 ! AUTHOR: Carol Peters, CREATION DATE: 13 September 1977
;    47          0047  1 !
;    48          0048  1 ! MODIFIED BY:
;    49          0049  1 !
;    50          0050  1 !       Kathleen Morse, 20 October 1977 : Version X01.00
;    51          0051  1 !
;    52          0052  1 ! Revision History:
;    53          0053  1 !
;    54          0054  1 !  NO   DATE            PROGRAMMER            PURPOSE
;    55          0055  1 !  --   ----            ----------            -------
;    56          0056  1 !
;    57          0057  1 !  00   20-OCT-77       K.D. MORSE            ADAPT VERSION 2 FOR PATCH
```

```
;    58          0058  1 !  01   4-JAN-78    K.D. MORSE      NO CHANGES FOR VERS 3.
;    59          0059  1 !  02   24-JAN-78   K.D. MORSE      NO CHANGES FOR VERS 4.
;    60          0060  1 .  03   25-APR-78   K.D. MORSE      CONVERT TO NATIVE COMPILER.
;    61          0061  1 !  04   01-MAY-78   K.D. MORSE      CHANGE ALPHA TO ALPHA_STR_TOKEN.
;    62          0062  1 .  05   18-MAY-78   K.D. MORSE      PAT$GET_A_TOKEN SETS MODE_LEVEL
;    63          0063  1 !                                   TO LOCAL BEFORE CALLING THE
;    64          0064  1 !                                   LEXICAL ROUTINE. (05)
;    65          0065  1 !                                   NO CHANGES FOR VERS 6.
;    66          0066  1 .  06   18-MAY-78   K.D. MORSE      NO CHANGES FOR VERS 7-8.
;    67          0067  1 .  07   13-JUN-78   K.D. MORSE      ADD FAO COUNTS TO SIGNALS.
;    68          0068  1 !
;    69          0069  1 !--
```

```
;    71          0070  1 !
;    72          0071  1 ! TABLE OF CONTENTS:
;    73          0072  1 !
;    74          0073  1 FORWARD ROUTINE
;    75          0074  1         PAT$GET_A_TOKEN,                    ! Extracts a token from the input buffer
;    76          0075  1         TRANS_LEXEME;                       . Translates a lexeme into a token
;    77          0076  1
;    78          0077  1 !
;    79          0078  1 ! INCLUDE FILES:
;    80          0079  1 !
;    81          0080  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    82          0081  1 REQUIRE 'SRC$:VXSMAC.REQ';
;    83          0146  1 REQUIRE 'SRC$:PATPCT.REQ';
;    84          0186  1 REQUIRE 'SRC$:PATGEN.REQ';
;    85          0408  1 REQUIRE 'SRC$:PATTER.REQ';
;    86          0615  1 REQUIRE 'SRC$:SCALIT.REQ';
;    87          0681  1 REQUIRE 'SRC$:SYSSER.REQ';
```

```
;  R0713  1     SWITCHES LIST (SOURCE);
;  R0714  1
;  R0715  1     EXTERNAL ROUTINE
;  R0716  1              PAT$fao_out;           ! formats a line and outputs to the terminal
;  R0717  1
```

```
;    88          0763  1 REQUIRE 'SRC$:PATKEY.REQ';
;    89          0896  1
;    90          0897  1 !
;    91          0898  1 ! MACROS:
;    92          0899  1 !
;    93          0900  1
;    94          0901  1 !
;    95          0902  1 ! EQUATED SYMBOLS:
;    96          0903  1 !
;    97          0904  1
;    98          0905  1 !
;    99          0906  1 ! OWN STORAGE:
;   100          0907  1 !
;   101          0908  1
;   102          0909  1 !
;   103          0910  1 ! EXTERNAL REFERENCES:
;   104          0911  1 !
;   105          0912  1
;   106          0913  1 EXTERNAL ROUTINE
;   107          0914  1         PAT$SET_MOD_LVL,                        ! Routine to set the mode level
;   108          0915  1         PAT$MAR_GET_LEX;                        ! Lexical analyzer
;   109          0916  1
;   110          0917  1 EXTERNAL
;   111          0918  1         PAT$GL_KEYW_TBL: VECTOR;                ! Holds keywords for current syntax
```

```
;  113     0919  1 GLOBAL ROUTINE PAT$GET_A_TOKEN (INPUT_STG_DESC, LEXEME_STG_DESC) =      ! Gets a token from input line                ; R
;  114     0920  1
;  115     0921  1 !++
;  116     0922  1 ! Functional description:
;  117     0923  1 !
;  118     0924  1 !       Extracts a lexeme from the input stream by calling the routine
;  119     0925  1 !       PAT$GET_MAR_LEXEME.  Translates the lexeme to a token (sometimes a
;  120     0926  1 !       null operation). Returns the token as the routine value and
;  121     0927  1 !       the ASCII string representing the token in the character string
;  122     0928  1 !       pointed to by the pointer field of lexeme_stg_desc. Also returns
;  123     0929  1 !       the actual length of the ASCII string of the lexeme in the
;  124     0930  1 !       length field of lexeme_stg_desc.
;  125     0931  1 !
;  126     0932  1 !       The pointer to the input buffer is updated and rewritten into
;  127     0933  1 !       the pointer field of INPUT_STG_DESC.
;  128     0934  1 !
;  129     0935  1 ! Calling sequence:
;  130     0936  1 !
;  131     0937  1 !       CALL PAT$GET_A_TOKEN (INPUT_STG_DESC.rt.dd, LEXEME_STG_DESC.rt.dv)
;  132     0938  1 !
;  133     0939  1 ! Inputs:
;  134     0940  1 !
;  135     0941  1 !       INPUT_STG_DESC  - string descriptor to the input buffer.
;  136     0942  1 !       LEXEME_STG_DESC - varying string descriptor to the lexeme buffer.
;  137     0943  1 !
;  138     0944  1 ! Implicit inputs:
;  139     0945  1 !
;  140     0946  1 !       none
;  141     0947  1 !
;  142     0948  1 ! Outputs:
;  143     0949  1 !
;  144     0950  1 !       none
;  145     0951  1 !
;  146     0952  1 ! Implicit outputs:
;  147     0953  1 !
;  148     0954  1 !       none
;  149     0955  1 !
;  150     0956  1 ! Routine value:
;  151     0957  1 !
;  152     0958  1 !       an encoded representation of the token found.
;  153     0959  1 !
;  154     0960  1 ! Side effects:
;  155     0961  1 !
;  156     0962  1 !       The mode level is set to local.
;  157     0963  1 !--
;  158     0964  1
;  159     0965  2 BEGIN
;  160     0966  2
;  161     0967  2 MAP
;  162     0968  2         LEXEME_STG_DESC: REF BLOCK [, BYTE];                    ! Descriptor of lexeme string
;  163     0969  2
;  164     0970  2 LOCAL
;  165     0971  2         LEXEME_TYPE;                                            ! Type of lexeme found
;  166     0972  2
;  167     0973  2 !++
;  168     0974  2 ! Fill the lexeme buffer with zeroes.
;  169     0975  2 !--
```

```
;  170          0976  2 ZEROCOR (.LEXEME_STG_DESC [DSC$A_POINTER], (.LEXEME_STG_DESC [DSC$W_MAXLEN] / 4));
;  171          0977  2 PAT$SET_MOD_LVL (LOCAL_MODE);                          ! Set mode level to local
;  172          0978  2 LEXEME_TYPE = PAT$MAR_GET_LEX (.INPUT_STG_DESC, .LEXEME_STG_DESC);
;  173          0979  3 IF (.LEXEME_TYPE EQL ALPHA_STR_TOKEN)
;  174          0980  2 THEN
;  175          0981  2         RETURN TRANS_LEXEME (.LEXEME_STG_DESC)
;  176          0982  2 ELSE
;  177          0983  2         RETURN .LEXEME_TYPE;
;  178          0984  1 END;                                                    ! End of PAT$GET_A_TOKEN
```

```
                                                        .TITLE  PATSCA
                                                        .IDENT  \V04-000\

                                                        .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                                    05  02  01  00000 P.AAA:   .BYTE   1, 2, 5
                                4E  47  49  4C  41  00003       .ASCII  \ALIGN\
                                    03  03  12  00008           .BYTE   18, 3, 3
                                    4C  4C  41  0000B           .ASCII  \ALL\
                                    03  02  13  0000E           .BYTE   19, 2, 3
                                    44  4E  41  00011           .ASCII  \AND\
                                    05  02  14  00014           .BYTE   20, 2, 5
                                49  49  43  53  41  00017       .ASCII  \ASCII\
                                    04  01  15  0001C           .BYTE   21, 1, 4
                                    45  54  59  42  0001F       .ASCII  \BYTE\
                                    06  02  02  00023           .BYTE   2, 2, 6
                            4C  45  43  4E  41  43  00026       .ASCII  \CANCEL\
                                    05  02  03  0002C           .BYTE   3, 2, 5
                                4B  43  45  48  43  0002F       .ASCII  \CHECK\
                                    06  02  04  00034           .BYTE   4, 2, 6
                            45  54  41  45  52  43  00037       .ASCII  \CREATE\
                                    07  01  07  0003D           .BYTE   7, 1, 7
                                54  49  53  4F  50  45  44  00040   .ASCII  \DEPOSIT\
                                    07  03  16  00047           .BYTE   22, 3, 7
                                4C  41  4D  49  43  45  44  0004A   .ASCII  \DECIMAL\
                                    06  03  05  00051           .BYTE   5, 3, 6
                                    45  4E  49  46  45  44  00054   .ASCII  \DEFINE\
                                    06  03  06  0005A           .BYTE   6, 3, 6
                                    45  54  45  4C  45  44  0005D   .ASCII  \DELETE\
                                    07  01  09  00063           .BYTE   9, 1, 7
                                45  4E  49  4D  41  58  45  00066   .ASCII  \EXAMINE\
                                    03  02  17  0006D           .BYTE   23, 2, 3
                                    4F  43  45  00070           .ASCII  \ECO\
                                    08  02  08  00073           .BYTE   8, 2, 8
                            45  54  41  55  4C  41  56  45  00076   .ASCII  \EVALUATE\
                                    04  03  0A  0007E           .BYTE   10, 3, 4
                                    54  49  58  45  00081       .ASCII  \EXIT\
                                    07  02  19  00085           .BYTE   25, 2, 7
                                53  4C  41  42  4F  4C  47  00088   .ASCII  \GLOBALS\
                                    04  03  0B  0008F           .BYTE   11, 3, 4
                                    50  4C  45  48  00092       .ASCII  \HELP\
                                    0B  01  1B  00096           .BYTE   27, 1, 11
4C  41  4D  49  43  45  44  41  58  45  48  00099               .ASCII  \HEXADECIMAL\
                                    0A  04  32  000A4           .BYTE   50, 4, 10
    45  5A  49  4C  41  49  54  49  4E  49  000A7               .ASCII  \INITIALIZE\
                                    0B  01  1C  000B1           .BYTE   28, 1, 11
```

```
                4E  4F  49  54  43  55  52  54  53  4E  49  000B4          .ASCII  \INSTRUCTION\
                                                06  04  0C  000BF          .BYTE   12, 4, 6
                                            54  52  45  53  4E  49  000C2  .ASCII  \INSERT\
                                                07  02  1E  000C8          .BYTE   30, 2, 7
                                        4C  41  52  45  54  49  4C  000CB  .ASCII  \LITERAL\
                                                04  02  1F  000D2          .BYTE   31, 2, 4
                                                    47  4E  4F  4C  000D5  .ASCII  \LONG\
                                                04  01  21  000D9          .BYTE   33, 1, 4
                                                    45  44  4F  4D  000DC  .ASCII  \MODE\
                                                06  04  22  000E0          .BYTE   34, 4, 6
                                            45  4C  55  44  4F  4D  000E3  .ASCII  \MODULE\
                                                07  03  24  000E9          .BYTE   36, 3, 7
                                        49  49  43  53  41  4F  4E  000EC  .ASCII  \NOASCII\
                                                09  03  25  000F3          .BYTE   37, 3, 9
                            53  4C  41  42  4F  4C  47  4F  4E  000F6  .ASCII  \NOGLOBALS\
                                                0D  03  26  000FF          .BYTE   38, 3, 13
        4E  4F  49  54  43  55  52  54  53  4E  49  4F  4E  00102  .ASCII  \NOINSTRUCTION\
                                                07  04  27  0010F          .BYTE   39, 4, 7
                                        45  50  4F  43  53  4F  4E  00112  .ASCII  \NOSCOPE\
                                                09  04  28  00119          .BYTE   40, 4, 9
                            53  4C  4F  42  4D  59  53  4F  4E  0011C  .ASCII  \NOSYMBOLS\
                                                03  03  29  00125          .BYTE   41, 3, 3
                                                    54  4F  4E  00128  .ASCII  \NOT\
                                                05  02  2A  0012B          .BYTE   42, 2, 5
                                            4C  41  54  43  4F  0012E  .ASCII  \OCTAL\
                                                02  02  2B  00133          .BYTE   43, 2, 2
                                                        52  4F  00136  .ASCII  \OR\
                                                04  03  2C  00138          .BYTE   44, 3, 4
                                                    45  47  41  50  0013B  .ASCII  \PAGE\
                                                0A  03  2D  0013F          .BYTE   45, 3, 10
    41  45  52  41  5F  48  43  54  41  50  00142  .ASCII  \PATCH_AREA\
                                                04  01  2E  0014C          .BYTE   46, 1, 4
                                                    44  41  55  51  0014F  .ASCII  \QUAD\
                                                07  02  0D  00153          .BYTE   13, 2, 7
                                        45  43  41  4C  50  45  52  00156  .ASCII  \REPLACE\
                                                05  02  2F  0015D          .BYTE   47, 2, 5
                                            45  50  4F  43  53  00160  .ASCII  \SCOPE\
                                                03  02  0E  00165          .BYTE   14, 2, 3
                                                    54  45  53  00168  .ASCII  \SET\
                                                04  02  0F  0016B          .BYTE   15, 2, 4
                                                    57  4F  48  53  0016E  .ASCII  \SHOW\
                                                07  02  30  00172          .BYTE   48, 2, 7
                                        53  4C  4F  42  4D  59  53  00175  .ASCII  \SYMBOLS\
                                                06  01  10  0017C          .BYTE   16, 1, 6
                                            45  54  41  44  50  55  0017F  .ASCII  \UPDATE\
                                                06  01  11  00185          .BYTE   17, 1, 6
                                            59  46  49  52  45  56  00188  .ASCII  \VERIFY\
                                                04  01  31  0018E          .BYTE   49, 1, 4
                                                    44  52  4F  57  00191  .ASCII  \WORD\
                                                        00  00195          .BYTE   0

                                                        KEYWORD_TABLE=      P.AAA
                                                                .EXTRN  PAT$FAO_OUT, PAT$SET_MOD_LVL
                                                                .EXTRN  PAT$MAR_GET_LEX
                                                                .EXTRN  PAT$GL_KEYW_TBL

                                                                .PSECT  _PAT$CODE,NOWRT,2
```

```
                                                 007C 00000        .ENTRY  PAT$GET_A_TOKEN, Save R2,R3,R4,R5,R6        ; 0919
                                          56     08   AC   D0 00002        MOVL    LEXEME_STG_DESC, R6                 ; 0976
                                          50     08   A6   3C 00006        MOVZWL  8(R6), R0                           ;
                                          50          04   C6 0000A        DIVL2   #4, R0                              ;
                                          50          04   C4 0000D        MULL2   #4, R0                              ;
                50              00        6E          00   2C 00010        MOVC5   #0, (SP), #0, R0, @4(R6)            ;
                                                  04  B6       00015
                                                      03   DD 00017        PUSHL   #3                                  ; 0977
                                00000000G     EF            01   FB 00019  CALLS   #1, PAT$SET_MOD_LVL                 ;
                                                            56   DD 00020  PUSHL   R6                                  ; 0978
                                                      04    AC   DD 00022  PUSHL   INPUT_STG_DESC                      ;
                                00000000G     EF            02   FB 00025  CALLS   #2, PAT$MAR_GET_LEX                 ;
                                00000047      8F            50   D1 0002C  CMPL    LEXEME_TYPE, #71                    ; 0979
                                                            09   12 00033  BNEQ    1$                                  ;
                                                            56   DD 00035  PUSHL   R6                                  ; 0981
                                00000000V     EF            01   FB 00037  CALLS   #1, TRANS_LEXEME                   ;
                                                               04 0003E 1$: RET                                         ; 0984

; Routine Size:  63 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
;  180     0985  1 ROUTINE TRANS_LEXEME (LEXEME_STG_DESC) =                    ! Translates a name into a keyword token
;  181     0986  1
;  182     0987  1 !++
;  183     0988  1 ! Functional description:
;  184     0989  1 !
;  185     0990  1 !       Maps an alphabetic string onto an element in the keyword table.
;  186     0991  1 !       If the alphabetic string does not match a keyword, then the
;  187     0992  1 !       token "alpha_str_token" is returned. If the alphabetic string does match
;  188     0993  1 !       a keyword, then the token for the keyword is abstracted from
;  189     0994  1 !       the keyword table and returned.
;  190     0995  1 !
;  191     0996  1 ! Calling sequence:
;  192     0997  1 !
;  193     0998  1 !       CALL TRANS_LEXEME (LEXEME_STG_DESC.rt.dv)
;  194     0999  1 !
;  195     1000  1 ! Inputs:
;  196     1001  1 !
;  197     1002  1 !       LEXEME_STG_DESC - varying string descriptor for lexeme string
;  198     1003  1 !
;  199     1004  1 ! Implicit inputs:
;  200     1005  1 !
;  201     1006  1 !       The keyword table for the PATCH language.
;  202     1007  1 !
;  203     1008  1 ! Outputs:
;  204     1009  1 !
;  205     1010  1 !       none
;  206     1011  1 !
;  207     1012  1 ! Implicit outputs:
;  208     1013  1 !
;  209     1014  1 !       none
;  210     1015  1 !
;  211     1016  1 ! Routine value:
;  212     1017  1 !
;  213     1018  1 !       The token for the keyword that matches the ASCII string,
;  214     1019  1 !       or the "alpha_str_token" token, if no keyword matches the string.
;  215     1020  1 !
;  216     1021  1 ! Side effects:
;  217     1022  1 !
;  218     1023  1 !       none
;  219     1024  1 !--
;  220     1025  1
;  221     1026  2 BEGIN
;  222     1027  2
;  223     1028  2 MAP
;  224     1029  2         LEXEME_STG_DESC: REF BLOCK [, BYTE];                ! Lexeme string descriptor
;  225     1030  2
;  226     1031  2 LOCAL
;  227     1032  2         KEYWORD_ENTRY : REF VECTOR [, BYTE];                ! Address of a keyword record;
;  228     1033  2
;  229     1034  2 KEYWORD_ENTRY = KEYWORD_TABLE [0];
;  230     1035  2 REPEAT
;  231     1036  3         BEGIN
;  232     1037  3         !++
;  233     1038  3         ! If the length of the keyword is at least as long as the lexeme found,
;  234     1039  3         ! and the lexeme found is at least as long as the abbreviation of the
;  235     1040  3         ! keyword, then try to match the strings.
;  236     1041  3         !--
```

```
;  237          1042  3          IF (.KEYWORD_ENTRY [KWORD_LENGTH] GEQ .LEXEME_STG_DESC [DSC$W_LENGTH]) AND
;  238          1043  4             (.LEXEME_STG_DESC [DSC$W_LENGTH] GEQ .KEYWORD_ENTRY [KWORD_ABBREV])
;  239          1044  3          THEN
;  240          1045  4                  BEGIN
;  241          1046  4                  !++
;  242          1047  4                  ! If a keyword match is found, return the token equivalent.
;  243          1048  4                  !--
;  244          1049  4                  IF CH$EQL (.LEXEME_STG_DESC [DSC$W_LENGTH],
;  245          1050  4                             CH$PTR (.LEXEME_STG_DESC [DSC$A_POINTER]),
;  246          1051  4                             .LEXEME_STG_DESC [DSC$W_LENGTH],
;  247          1052  4                             CH$PTR (KEYWORD_ENTRY [KWORD_NAME]))
;  248          1053  4                  THEN
;  249          1054  4                      RETURN .KEYWORD_ENTRY [KWORD_TOKEN];
;  250          1055  3                  END;
;  251          1056  3
;  252          1057  3          !++
;  253          1058  3          ! Keyword did not match. Advance the table pointer to
;  254          1059  3          ! point to the next entry. If the first byte of this
;  255          1060  3          ! next entry is zero, conclude that the table is
;  256          1061  3          ! exhausted, and just return the alpha_str_token.
;  257          1062  3          !--
;  258          1063  3          KEYWORD_ENTRY = KEYWORD_ENTRY [0]
;  259          1064  3                        + .KEYWORD_ENTRY [KWORD_LENGTH] + KWORD_OVERHEAD;
;  260          1065  4          IF (.KEYWORD_ENTRY [KWORD_TOKEN] EQL 0)
;  261          1066  3          THEN
;  262          1067  3                  RETURN ALPHA_STR_TOKEN;
;  263          1068  2          END;
;  264          1069  1 END;
; INFO#212                 L1:1034
; Null expression appears in value-required context


                                                007C 00000 TRANS_LEXEME:
                                                           .WORD   Save R2,R3,R4,R5,R6                          ; 0985
                                    54 00000000'   EF  9E 00002            MOVAB   KEYWORD_TABLE, KEYWORD_ENTRY ; 1034
                                    56         04  AC  D0 00009            MOVL    LEXEME_STG_DESC, R6          ; 1042
                                    55         02  A4  9A 0000D 1$:        MOVZBL  2(KEYWORD_ENTRY), R5         ;
                                    55             66  B1 00011            CMPW    (R6), R5                     ;
                                                   15  1A 00014            BGTRU   2$                           ;
                                    50         01  A4  9A 00016            MOVZBL  1(KEYWORD_ENTRY), R0         ; 1043
                                    66             50  B1 00019            CMPW    R0, (R6)                     ;
                                                   0C  1A 0001D            BGTRU   2$                           ;
                          03   A4        04   B6        66  29 0001F        CMPC3   (R6), @4(R6), 3(KEYWORD_ENTRY) ; 1052
                                                   04  12 00025            BNEQ    2$                           ;
                                    50             64  9A 00027            MOVZBL  (KEYWORD_ENTRY), R0          ; 1054
                                                       04 0002A            RET                                  ;
                                    54         03 A544  9E 0002B 2$:        MOVAB   3(R5)[KEYWORD_ENTRY], KEYWORD_ENTRY ; 1064
                                                   64  95 00030            TSTB    (KEYWORD_ENTRY)              ; 1065
                                                   D9  12 00032            BNEQ    1$                           ;
                                    50         47  8F  9A 00034            MOVZBL  #71, R0                      ; 1067
                                                       04 00038            RET                                  ; 1069

; Routine Size:  57 bytes,    Routine Base:  _PAT$CODE + 003F
```

```
;  266          1070  1 END
;  267          1071  0 ELUDOM                                        ! End of module
```

PSECT SUMMARY

Name	Bytes	Attributes
_PAT$PLIT	406	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(0)
_PAT$CODE	120	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
_255DUA28:[SYSLIB]LIB.L32;1	18619	3	0	1000	00:01.9

```
; Information:  1
; Warnings:     0
; Errors:       0


;                                       COMMAND QUALIFIERS

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

; Size:          120 code + 406 data bytes
; Run Time:        00:10.6
; Elapsed Time:    00:47.8
; Lines/CPU Min:    6690
; Lexemes/CPU-Min: 20576
; Memory Used:  99 pages
; Compilation Complete
```

PATREB
LIS

PATSCA
LIS

PATSIO
LIS

PATRST
LIS

PATSPA
LIS

PATSSV
LIS