
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCC HHH HHH
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCC HHH HHH
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCC 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 AAAAAAAAAAAAAAA TTT CCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCC HHH HHH

```
PPPPPPPP      AAAAAA    TTTTTTTTTT  PPPPPPPP      AAAAAA    RRRRRRRR
PPPPPPPP      AAAAAA    TTTTTTTTTT  PPPPPPPP      AAAAAA    RRRRRRRR
PP      PP   AA    AA       TT      PP      PP   AA    AA   RR      RR
PP      PP   AA    AA       TT      PP      PP   AA    AA   RR      RR
PP      PP   AA    AA       TT      PP      PP   AA    AA   RR      RR
PP      PP   AA    AA       TT      PP      PP   AA    AA   RR      RR
PPPPPPPP     AA    AA       TT      PPPPPPPP     AA    AA   RRRRRRRR
PPPPPPPP     AA    AA       TT      PPPPPPPP     AA    AA   RRRRRRRR
PP           AAAAAAAAAA     TT      PP           AAAAAAAAAA   RR  RR
PP           AAAAAAAAAA     TT      PP           AAAAAAAAAA   RR  RR
PP           AA    AA       TT      PP           AA    AA     RR    RR      ....
PP           AA    AA       TT      PP           AA    AA     RR    RR      ....
PP           AA    AA       TT      PP           AA    AA     RR      RR    ....
PP           AA    AA       TT      PP           AA    AA     RR      RR    ....

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
```

```
;     1          0001  0 MODULE PATPAR (
;     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  1 BEGIN
;     8          0008  1
;     9          0009  1 !*****************************************************************************
;    10          0010  1 !*                                                                           *
;    11          0011  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    12          0012  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    13          0013  1 !*  ALL RIGHTS RESERVED.                                                      *
;    14          0014  1 !*                                                                           *
;    15          0015  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    16          0016  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    17          0017  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    18          0018  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    19          0019  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    20          0020  1 !*  TRANSFERRED.                                                              *
;    21          0021  1 !*                                                                           *
;    22          0022  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    23          0023  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    24          0024  1 !*  CORPORATION.                                                              *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    27          0027  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*                                                                           *
;    30          0030  1 !*****************************************************************************
;    31          0031  1 !
;    32          0032  1 ! FACILITY:     PATCH
;    33          0033  1 !
;    34          0034  1 !++
;    35          0035  1 ! FUNCTIONAL DESCRIPTION:
;    36          0036  1 !
;    37          0037  1 !       Parser for MARS PATCH syntax
;    38          0038  1 !
;    39          0039  1 ! Version:      V02-025
;    40          0040  1 !
;    41          0041  1 ! History:
;    42          0042  1 !       Author:
;    43          0043  1 !               Carol Peters, 03 Jul 1976: Version 01
;    44          0044  1 !
;    45          0045  1 !       Modified by:
;    46          0046  1 !               Kathleen Morse, 13-Oct-77: Version X01.00
;    47          0047  1 !
;    48          0048  1 ! MODIFIED BY:
;    49          0049  1 !
;    50          0050  1 !       V03-002 MCN0151         Maria del C. Nasr       13-Feb-1984
;    51          0051  1 !               Add VOLATILE qualifier to local NEXT_TOKEN to eliminate
;    52          0052  1 !               information messages from the compiler.
;    53          0053  1 !
;    54          0054  1 !       V03-001 MTR0012         Mike Rhodes             16-Aug-1982
;    55          0055  1 !               Modify file names to remove duplicate file name useage
;    56          0056  1 !               between code and require files.
;    57          0057  1 !
```

```
;    58          0058  1 !       V02-025 MTR0002         Mike Rhodes             01-Oct-1981
;    59          0059  1 !               Modify the Macro SET_PATAREA_BIT and add a new macro
;    60          0060  1 !               SET_INIT_BIT to handle the /INITIALIZE qualifier.
;    61          0061  1 !               The macros set the appropriate context bits and
;    62          0062  1 !               will set the correct values of the address of the
;    63          0063  1 !               patch area descriptor and initial size value into
;    64          0064  1 !               the first element of the linked argument list.
;    65          0065  1 !
;    66          0066  1 !       V02-024 MTR0001         Mike Rhodes             20-AUG-1981
;    67          0067  1 !               Add HELP command.  This includes a new macro for
;    68          0068  1 !               saving a descriptor to the rest of the command line
;    69          0069  1 !               while faking the parser into believing its reached
;    70          0070  1 !               the end of the line (which causes the command to be
;    71          0071  1 !               executed).  The macro is called GET_HELP_TOPIC.
;    72          0072  1 !
;    73          0073  1 !       V02-023 PCG0001         Peter George            02-FEB-1981
;    74          0074  1 !               Add require statement for LIB$:PATDEF.REQ
;    75          0075  1 !
;    76          0076  1 ! Revision history:
;    77          0077  1 !
;    78          0078  1 !  NO   DATE            PROGRAMMER              PURPOSE
;    79          0079  1 !  --   ----            ----------              -------
;    80          0080  1 !
;    81          0081  1 !  00   18-OCT-77       K.D. MORSE              ADAPT VERSION 31 TO PATCH
;    82          0082  1 !  01   29-DEC-77       K. D. MORSE             ADD SET/SHOW MODULE/SCOPE CMDS. (44)
;    83          0083  1 !  02   4-JAN-78        K.D. MORSE              NO CHANGES FOR VERS 32-41.
;    84          0084  1 !                                               IN MACRO GET_QUOTED_STG, DON'T
;    85          0085  1 !                                               ACCEPT INPUT UNLESS EITHER
;    86          0086  1 !                                               INSTRUCTION MODE OR ASCII MODE
;    87          0087  1 !                                               IS SET. (42)
;    88          0088  1 !                                               NO CHANGES FOR 43,45.
;    89          0089  1 !                                               DON'T ALLOW DIVISION BY ZERO (46).
;    90          0090  1 !                                               NO CHANGES FOR 47.
;    91          0091  1 !                                               CHANGE PARSE STACK OFFSETS TO
;    92          0092  1 !                                               NAMES DEFINED IN PATMSG.REQ. (48)
;    93          0093  1 !                                               PLACE TYPE ON ARGUMENT LIST AS
;    94          0094  1 !                                               WELL AS EXPRESSION. (48)
;    95          0095  1 !                                               CHANGE EACH ELEMENT OF THE PARSE
;    96          0096  1 !                                               STACKS TO BE PAT$K_STELM_SIZ (48).
;    97          0097  1 !                                               FOLLOW A FINAL CALL TO BUILD_PATH
;    98          0098  1 !                                               WITH PLACING INTERGER TYPE ON
;    99          0099  1 !                                               SEMAN2. (49)
;   100          0100  1 !  03   24-JAN-78        K.D. MORSE              NO CHANGES FOR VERS 50.
;   101          0101  1 !  04   27-JAN-78        K.D. MORSE              ADD MACRO LINK_EXIT FOR THE
;   102          0102  1 !                                               REPLACE AND VERIFY COMMANDS.
;   103          0103  1 !  05   28-FEB-78        K.D. MORSE              CHANGE MACRO GET_QUOTED_STG
;   104          0104  1 !                                               TO ACCEPT ' AND " .  (51)
;   105          0105  1 !  06   01-MAR-78        K.D. MORSE              NO CHANGES FOR 52.
;   106          0106  1 !  07   24-MAR-78        K.D. MORSE              NO CHANGES FOR 53-54.
;   107          0107  1 !  08   07-APR-78        K.D. MORSE              INIT THE DELIMITER IN GET_QUOTED_STG (55).
;   108          0108  1 !                                               BUILD_PATH IS NOT THE FINAL WORD
;   109          0109  1 !                                               IN MACRO REDUCE_PATHNAME. (56)
;   110          0110  1 !  09   14-APR-78        K.D. MORSE              NO CHANGES FOR VERS. 57-59.
;   111          0111  1 !  10   18-APR-78        K.D. MORSE              ADD ACTION ROUTINES TO SET BIT
;   112          0112  1 !                                               LITERAL_BIT.
;   113          0113  1 !  11   18-APR-78        K.D. MORSE              ADD MACRO GET_FILE_SPEC FOR THE
;   114          0114  1 !                                               CREATE COMMAND.
```

```
;  115          0115  1 !  12   25-APR-78       K.D. MORSE          CONVERT TO NATIVE COMPILER.
;  116          0116  1 !  13   09-MAY-78       K.D. MORSE          ADD CHECK IN LINK_ARG_PAIR AND
;  117          0117  1 !                                           LINK_ARG TO CHECK THAT ASCII OR
;  118          0118  1 !                                           INSTRUCTION MODES ARE NOT SET.
;  119          0119  1 !                                           ADD LINK_VAL TO AVOID THE CHECK.
;  120          0120  1 !  14   16-MAY-78       K.D. MORSE          CHANGED CALLS TO PAT$SET_OVERS
;  121          0121  1 !                                           TO PASS VALUE INSTEAD OF STACK
;  122          0122  1 !                                           POINTER. (60)
;  123          0123  1 !                                           NO CHANGES FOR VERS 61.
;  124          0124  1 !                                           REMOVED (.STACK_PTR) FROM MACRO
;  125          0125  1 !                                           CALL TO "SET_DEC_OVERS". (62)
;  126          0126  1 !  15   18-MAY-78       K.D. MORSE          SET OVERRIDE_MODE IN "SET_DEC_OVERS"
;  127          0127  1 !                                           BEFORE SETTING DECIMAL_TOKEN. (63)
;  128          0128  1 !  16   18-MAY-78       K.D. MORSE          NO CHANGES FOR VERS 64.
;  129          0129  1 !  17   18-MAY-78       K.D. MORSE          NO CHANGES FOR VERS 65.
;  130          0130  1 !  18   18-MAY-78       K.D. MORSE          NO CHANGES FOR VERS 66.
;  131          0131  1 !  19   13-JUN-78       K.D. MORSE          ADD FAO COUNTS TO SIGNALS.
;  132          0132  1 !  20   16-JUN-78       K.D. MORSE          ALWAYS CALL PAT$SET_COMQUAL
;  133          0133  1 !                                           FOR CORRECT APPENDED PATCH
;  134          0134  1 !                                           COMMAND TEXT QUALIFIERS.
;  135          0135  1 !  21   21-JUN-78       K. D. MORSE         ADD PAT$_SYNTAX ERROR MESSAGE
;  136          0136  1 !                                           TO THE PAT$_INVCMD MESSAGE. (67)
;  137          0137  1 !  22   28-JUN-78       K.D. MORSE          NO CHANGES FOR VERS 68-72.
;  138          0138  1 !                                           ERROR HANDLING FOR DIGIT_TOKEN (73).
;  139          0139  1 !                                           CHANGE CALLING SEQUENCE FOR
;  140          0140  1 !                                           PAT$FIND_MODULE. (74)
;  141          0141  1 !                                           NO CHANGES FOR VERS 75-81.
;  142          0142  1 !
;  143          0143  1 !--
```

```
;   145          0144  1
;   146          0145  1 FORWARD ROUTINE
;   147          0146  1           MAR_REDUCTN,                                  ! MARS action routines
;   148          0147  1           PAT$PARS_A_LINE: NOVALUE;                     ! Global routine to parse an input line
;   149          0148  1
;   150          0149  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;   151          0150  1 REQUIRE 'SRC$:PREFIX.REQ';
;   152          0338  1 REQUIRE 'SRC$:PATPRE.REQ';
;   153          0501  1 REQUIRE 'SRC$:VXSMAC.REQ';
;   154          0566  1 REQUIRE 'LIB$:PATDEF.REQ';                              ! Defines literals
;   155          0620  1 REQUIRE 'LIB$:PATMSG.REQ';
;   156          0794  1 REQUIRE 'SRC$:PATRTS.REQ';
;   157          1890  1 REQUIRE 'SRC$:PATPCT.REQ';
;   158          1930  1 REQUIRE 'SRC$:PATGEN.REQ';
;   159          2152  1 REQUIRE 'SRC$:PATTER.REQ';
;   160          2359  1 REQUIRE 'SRC$:PATTOK.REQ';
;   161          2429  1 REQUIRE 'SRC$:BSTRUC.REQ';
;   162          2505  1 REQUIRE 'SRC$:LISTEL.REQ';
;   163          2547  1 REQUIRE 'SRC$:VXPALT.REQ';
;   164          2599  1 REQUIRE 'SRC$:SYSSER.REQ';
```

```
;  R2631  1     SWITCHES LIST (SOURCE);
;  R2632  1
;  R2633  1     EXTERNAL ROUTINE
;  R2634  1              PAT$fao_out;           ! formats a line and outputs to the terminal
;  R2635  1
```

```
;  165          2681  1 REQUIRE 'SRC$:PATACS.REQ';                                  ! Case labels for MARS syntax
;  166          2801  1 REQUIRE 'SRC$:PATTAB.REQ';                                 ! Parse tables for MARS
;  167          4714  1
;  168          4715  1 EXTERNAL ROUTINE
;  169          4716  1          PAT$ADD_ARG,                                       ! Adds arguments to the command argument lis
;  170          4717  1          PAT$BUILD_PATH,                                    ! Routine to build a path name
;  171          4718  1          PAT$FIND_MODULE,                                   ! Finds the RST address of a module
;  172          4719  1          PAT$FREEZ,                                         ! Allocate a block of free storage
;  173          4720  1          PAT$GET_A_TOKEN,                                   ! Get a single token from input buffer
;  174          4721  1          PAT$INIT_MODES,                                    ! Initializes input and output modes
;  175          4722  1          PAT$PERFORM_CMD,                                   ! Executes a complete command
;  176          4723  1          PAT$PROMPT_READ,                                   ! Prompts and reads a command line
;  177          4724  1          PAT$SET_COMQUAL,                                   ! Sets command qualifier indicators
;  178          4725  1          PAT$SET_MOD_LVL,                                   ! Resets modes to a certain level
;  179          4726  1          PAT$SET_OVERS,                                     ! Sets override or local modes
;  180          4727  1          PAT$TRANS_NAME;                                    ! Translates a name into a binary value
;  181          4728  1
;  182          4729  1 EXTERNAL
;  183          4730  1          PAT$GL_HELP_LIN : BLOCK [8,BYTE],                  ! Global descriptor to remainder of command
;  184          4731  1          PAT$GL_FLAGS,                                      ! CLI flags
;  185          4732  1          PAT$GB_MOD_PTR : REF VECTOR [, BYTE],              ! Current mode
;  186          4733  1          PAT$GL_COMQUAL: BITVECTOR,                         ! Contains the command qualifier indicators
;  187          4734  1          PAT$GL_CONTEXT: BITVECTOR,                         ! Context word
;  188          4735  1          PAT$GL_HEAD_LST,                                   ! Head of linked argument list
;  189          4736  1          PAT$GL_TAIL_LST,                                   ! Tail of linked argument list
;  190          4737  1          PAT$GL_SEMAN1: VECTOR,                             ! Semantic stack for tokens
;  191          4738  1          PAT$GL_SEMAN2: VECTOR;                             ! Semantic stack for string pointers
;  192          4739  1
;  193          4740  1 !
;  194          4741  1 ! OWN STORAGE
;  195          4742  1 !
;  196          4743  1 OWN
;  197          4744  1          QUOTE_INDIC;                                      ! Indicator if parameter was quoted string
```

```
;  199          4745  1 !++
;  200          4746  1 ! The following macros are simple actions to perform with reductions to
;  201          4747  1 ! the grammar.  They correspond to the action routines in PATACT.REQ.  Instead
;  202          4748  1 ! of calling global routines, these macros are simply expanded in line.
;  203          4749  1 !--
;  204          4750  1 MACRO
;  205          4751  1
;  206          4752  1 !++
;  207          4753  1 ! The first set of macros do arithmetic.
;  208          4754  1 !--
;  209          4755  1
;  210          4756  1         !++
;  211          4757  1         ! The ADDITION macro adds the value at the top of the stack
;  212          4758  1         ! to the value at the third position in the stack and places
;  213          4759  1         ! the result at the top of the stack.
;  214          4760  1         !--
;  215        M 4761  1         ADDITION (SEMSP) =
;  216          4762  1                 PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] + .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  217          4763  1
;  218          4764  1         !++
;  219          4765  1         ! The DIVISION macro divides the value at the top of the stack
;  220          4766  1         ! by the value at the third position in the stack and places
;  221          4767  1         ! the result at the top of the stack.
;  222          4768  1         !--
;  223        M 4769  1         DIVISION (SEMSP) =
;  224        M 4770  1                 BEGIN
;  225        M 4771  1                 IF .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO] EQL 0
;  226        M 4772  1                 THEN SIGNAL(PAT$_DIVZERO+MSG$K_WARN);
;  227        M 4773  1                 PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] / .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO];
;  228          4774  1                 END%,
;  229          4775  1
;  230          4776  1 ! ******** I BET THIS WILL NOT WORK DUE TO MAPPED ADDRESSES. *******
;  231          4777  1         !++
;  232          4778  1         ! The INDIRECTION macro considers the value at the second position
;  233          4779  1         ! in the stack to be an address. It takes the contents of that
;  234          4780  1         ! address and places it on the top of the stack.
;  235          4781  1         !--
;  236        M 4782  1         INDIRECTION (SEMSP) =
;  237          4783  1                 PAT$GL_SEMAN1 [SEMSP] = ..PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_ONE]%,
;  238          4784  1
;  239          4785  1         !++
;  240          4786  1         ! The MULTIPLICATION macro multiplies the value at the top of
;  241          4787  1         ! the stack to the value at the third position on the stack
;  242          4788  1         ! and places the result at the top of the stack.
;  243          4789  1         !--
;  244        M 4790  1         MULTIPLICATION (SEMSP) =
;  245          4791  1                 PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] * .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  246          4792  1
;  247          4793  1         !++
;  248          4794  1         ! The NEGATION macro negates the value found in the second
;  249          4795  1         ! position on the stack and places the result on the top of
;  250          4796  1         ! the stack.
;  251          4797  1         !--
;  252        M 4798  1         NEGATION (SEMSP) =
;  253          4799  1                 PAT$GL_SEMAN1 [SEMSP] = - .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_ONE]%,
;  254          4800  1
;  255          4801  1         !++
```

```
;  256          4802  1        ! The POSITIVE macro takes the value found in the second
;  257          4803  1        ! position on the stack and places it on the top of the stack.
;  258          4804  1        !--
;  259        M 4805  1        POSITIVE (SEMSP) =
;  260          4806  1                PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_ONE]%,
;  261          4807  1
;  262          4808  1        !++
;  263          4809  1        ! The REMOVE_PARENS macro takes the value found in the second
;  264          4810  1        ! position on the stack and places it on the top of the stack.
;  265          4811  1        !--
;  266        M 4812  1        REMOVE_PARENS (SEMSP) =
;  267          4813  1                PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_ONE]%,
;  268          4814  1
;  269          4815  1        !++
;  270          4816  1        ! The ARITH_SHIFT macro shifts the value at the top of the
;  271          4817  1        ! stack by the value found in the third position on the stack
;  272          4818  1        ! and places the result on the top of the stack.
;  273          4819  1        !--
;  274        M 4820  1        ARITH_SHIFT (SEMSP) =
;  275          4821  1                PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] ^ .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  276          4822  1
;  277          4823  1        !++
;  278          4824  1        ! The SUBTRACTION macro subtracts the value found in the third
;  279          4825  1        ! position on the stack from the value at the top of the stack
;  280          4826  1        ! and places the result on the top of the stack.
;  281          4827  1        !--
;  282        M 4828  1        SUBTRACTION (SEMSP) =
;  283          4829  1                PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] - .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  284          4830  1
;  285          4831  1        !++
;  286          4832  1        ! The COMPLEMENT macro applies the NOT operator to the value
;  287          4833  1        ! found in the second position on the stack and places the
;  288          4834  1        ! result on the top of the stack.
;  289          4835  1        !--
;  290        M 4836  1        COMPLEMENT (SEMSP) =
;  291          4837  1                PAT$GL_SEMAN1 [SEMSP] = NOT .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_ONE]%,
;  292          4838  1
;  293          4839  1        !++
;  294          4840  1        ! The LOGICAL_OR macro applies the OR operator to the values found
;  295          4841  1        ! in the first and third position on the stack and places the
;  296          4842  1        ! result on the top of the stack.
;  297          4843  1        !--
;  298        M 4844  1        LOGICAL_OR (SEMSP) =
;  299        M 4845  1                PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] OR
;  300          4846  1                                        .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  301          4847  1
;  302          4848  1        !++
;  303          4849  1        ! The LOGICAL_AND macro applies the AND operator to the values
;  304          4850  1        ! found in the first and third position on the stack and places
;  305          4851  1        ! the result on the top of the stack.
;  306          4852  1        !--
;  307        M 4853  1        LOGICAL_AND (SEMSP) =
;  308        M 4854  1                PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] AND
;  309          4855  1                                        .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  310          4856  1
;  311          4857  1        !++
;  312          4858  1        ! The EXTRACT_BITS macro extracts a bit field from the value
```

```
;  313          4859  1        ! on the top of the stack and places that bit field on the
;  314          4860  1        ! top of the stack. The starting bit number of the bit field
;  315          4861  1        ! is in the fifth position on the stack. The end position
;  316          4862  1        ! of the bit field is in the third position. After the value
;  317          4863  1        ! is extracted, the mode is reset to override level.
;  318          4864  1        !--
;  319        M 4865  1        EXTRACT_BITS (SEMSP) =
;  320        M 4866  1                BEGIN
;  321        M 4867  1
;  322        M 4868  1                LOCAL
;  323        M 4869  1                        VALUE : BLOCK [4, BYTE];
;  324        M 4870  1
;  325        M 4871  1                VALUE = .PAT$GL_SEMAN1 [SEMSP];
;  326        M 4872  1                IF (.PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO] GTR 31) OR
;  327        M 4873  1                   (.PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_FOUR] GTR 31) OR
;  328        M 4874  1                   (.PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_FOUR] GTR
;  329        M 4875  1                                .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO])
;  330        M 4876  1                THEN
;  331        M 4877  1                        SIGNAL (PAT$_EXTBIT+MSG$K_WARN);
;  332        M 4878  1                PAT$GL_SEMAN1 [SEMSP] = .VALUE [0, .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_FOUR],
;  333        M 4879  1                        .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO] -
;  334        M 4880  1                                .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_FOUR] + 1, 0];
;  335        M 4881  1                PAT$INIT_MODES (LOCAL_MODE, OVERRIDE_MODE);
;  336        M 4882  1                PAT$SET_MOD_LVL (OVERRIDE_MODE);
;  337          4883  1                END%,
```

```
;  339          4884  1 !++
;  340          4885  1 ! The next few macros put items into linked lists.
;  341          4886  1 !--
;  342          4887  1
;  343          4888  1         !++
;  344          4889  1         ! The LINK_ARG_PAIR macro calls the routine PAT$ADD_ARG to place
;  345          4890  1         ! the value at the top of the stack as a new link in the command
;  346          4891  1         ! argument list. If the new link is created successfully, then
;  347          4892  1         ! the value found at the third position on the stack is placed
;  348          4893  1         ! in the new link as well, at the position called LINK_ELEM_EXP2.
;  349          4894  1         !--
;  350        M 4895  1         LINK_ARG_PAIR (SEMSP) =
;  351        M 4896  1                 BEGIN
;  352        M 4897  1                 IF (.QUOTE_INDIC)
;  353        M 4898  1                 THEN
;  354        M 4899  1                         SIGNAL(PAT$_INVQUO+MSG$K_WARN);
;  355        M 4900  1                 PAT$ADD_ARG (.PAT$GL_SEMAN1 [SEMSP]);
;  356        M 4901  1                 LIST_ELEM_EXP2 (.PAT$GL_TAIL_IST) = .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO];
;  357          4902  1                 END%,
;  358          4903  1
;  359          4904  1         !++
;  360          4905  1         ! The LINK_ARG macro calls the routine PAT$ADD_ARG to create
;  361          4906  1         ! a new link in the command argument list, and to insert in that
;  362          4907  1         ! link the value found at the top of the stack.
;  363          4908  1         !--
;  364        M 4909  1         LINK_ARG (SEMSP) =
;  365        M 4910  1                 BEGIN
;  366        M 4911  1                 IF (.PAT$GB_MOD_PTR[MODE_INSTRUC] OR .PAT$GB_MOD_PTR[MODE_ASCII]) AND
;  367        M 4912  1                    (NOT .QUOTE_INDIC)
;  368        M 4913  1                 THEN
;  369        M 4914  1                         SIGNAL(PAT$_NOTQUO+MSG$K_WARN);
;  370        M 4915  1                 QUOTE_INDIC = FALSE;
;  371        M 4916  1                 PAT$ADD_ARG (.PAT$GL_SEMAN1 [SEMSP]);
;  372          4917  1                 END%,
;  373          4918  1
;  374          4919  1         !++
;  375          4920  1         ! The LINK_NUM macro calls the routine PAT$ADD_ARG to create
;  376          4921  1         ! a new link in the command argument list, and to insert in that
;  377          4922  1         ! link the location found at the top of the stack.  This macro expects
;  378          4923  1         ! an address and so no quoted string should have been given.
;  379          4924  1         !--
;  380        M 4925  1         LINK_NUM (SEMSP) =
;  381        M 4926  1                 BEGIN
;  382        M 4927  1                 IF (.QUOTE_INDIC)
;  383        M 4928  1                 THEN
;  384        M 4929  1                         SIGNAL(PAT$_INVQUO+MSG$K_WARN);
;  385        M 4930  1                 PAT$ADD_ARG (.PAT$GL_SEMAN1 [SEMSP]);
;  386          4931  1                 END%,
;  387          4932  1
;  388          4933  1         !++
;  389          4934  1         ! The LINK_EXP_NAME macro calls the routine PAT$ADD_ARG to create
;  390          4935  1         ! a new link in the command argument list and to place in that
;  391          4936  1         ! link the address value found at the top of the second parse
;  392          4937  1         ! stack. This address is the address of a string descriptor.
;  393          4938  1         ! If the new link is created successfully, the value found at
;  394          4939  1         ! the third position on the first parse stack is placed in the
;  395          4940  1         ! link in the position called LIST_ELEM_EXP2.
```

```
;  396          4941  1          !--
;  397        M 4942  1          LINK_EXP_NAME (SEMSP) =
;  398        M 4943  1                  BEGIN
;  399        M 4944  1                  IF (.PAT$GB_MOD_PTR[MODE_INSTRUC] OR .PAT$GB_MOD_PTR[MODE_ASCII]) AND
;  400        M 4945  1                     (NOT .QUOTE_INDIC)
;  401        M 4946  1                  THEN
;  402        M 4947  1                          SIGNAL(PAT$_NOTQUO+MSG$K_WARN);
;  403        M 4948  1                  QUOTE_INDIC = FALSE;
;  404        M 4949  1                  PAT$ADD_ARG (.PAT$GL_SEMAN2 [SEMSP]);
;  405        M 4950  1                  LIST_ELEM_EXP2 (.PAT$GL_TAIL_LST) = .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO];
;  406          4951  1                  END%,
;  407          4952  1
;  408          4953  1
;  409          4954  1          !++
;  410          4955  1          ! The LINK_NAME macro calls the routine PAT$ADD_ARG to create
;  411          4956  1          ! a new link in the command argument list, and to insert in that
;  412          4957  1          ! link the string descriptor address found at the top of the stack.
;  413          4958  1          !--
;  414        M 4959  1          LINK_NAME (SEMSP) =
;  415          4960  1                  PAT$ADD_ARG (.PAT$GL_SEMAN2 [SEMSP])%,
;  416          4961  1
;  417          4962  1
;  418          4963  1          !++
;  419          4964  1          ! The LINK_EXIT macro puts an EXIT_TOKEN into the second expression
;  420          4965  1          ! location of the last argument on the argument list.  The location
;  421          4966  1          ! of this argument is known via PAT$GL_TAIL_LST, which is updated
;  422          4967  1          ! whenever any argument is added.  Note that LINK_EXIT may only be
;  423          4968  1          ! called for arguments that are not ranges.
;  424          4969  1          !--
;  425        M 4970  1          LINK_EXIT =
;  426          4971  1                  LIST_ELEM_EXP2(.PAT$GL_TAIL_LST) = EXIT_TOKEN%,
;  427          4972  1
;  428          4973  1  !++
;  429          4974  1  ! The next macro does special translation of a module name.
;  430          4975  1  !--
;  431          4976  1
;  432          4977  1          !++
;  433          4978  1          ! The ADD_MODULE macro translates the ascii name of a module into
;  434          4979  1          ! the address of that module's record in the RST, and puts that
;  435          4980  1          ! address onto the top of the parse stack.
;  436          4981  1          !--
;  437        M 4982  1          ADD_MODULE (SEMSP) =
;  438          4983  1                  PAT$GL_SEMAN1 [SEMSP] = PAT$FIND_MODULE (LEX_STG_DESC, TRUE)%,
;  439          4984  1
;  440          4985  1          !++
;  441          4986  1          ! The SAVE_NAME macro allocates an area of free storage to hold
;  442          4987  1          ! the name of a symbol and a string descriptor to that symbol.
;  443          4988  1          ! This macro is called when a name occurs in a DEFINE command,
;  444          4989  1          ! and the name stored in free storage is used by PAT$DEFINE_SYM.
;  445          4990  1          ! The storage is freed by PAT$PERFORM_CMD.
;  446          4991  1          !
;  447          4992  1          ! All this string descriptor baggage seems rather cumbersome
;  448          4993  1          ! here. Maybe at some later point, the use of string descriptors
;  449          4994  1          ! in this case should be changed to counted strings.
;  450          4995  1          !--
;  451        M 4996  1          SAVE_NAME (SEMSP) =
;  452        M 4997  1                  BEGIN
```

```
;  453        M 4998  1
;  454        M 4999  1                   LOCAL
;  455        M 5000  1                         POINTER : REF VECTOR;
;  456        M 5001  1                   POINTER = PAT$FREEZ (((.LEX_STG_DESC [DSC$W_LENGTH] + 3) / A_LONGWORD) + 2);
;  457        M 5002  1                   CH$MOVE (.LEX_STG_DESC [DSC$W_LENGTH], .LEX_STG_DESC [DSC$A_POINTER],
;  458        M 5003  1                                 POINTER [2]);
;  459        M 5004  1                   POINTER [0] = .LEX_STG_DESC [DSC$W_LENGTH];
;  460        M 5005  1                   POINTER [1] = POINTER [2];
;  461        M 5006  1                   PAT$GL_SEMAN2 [SEMSP] = .POINTER;
;  462          5007  1                   END%,
;  463          5008  1
;  464          5009  1 !++
;  465          5010  1 ! The next set of macros manipulates mode settings.
;  466          5011  1 !--
;  467          5012  1
;  468          5013  1           !++
;  469          5014  1           ! The SET_OVERR_MODE macro calls the routine PAT$SET_OVERS to
;  470          5015  1           ! set the mode pointer to OVERRIDE level, and to set the mode
;  471          5016  1           ! according to the token found at the top of the stack.
;  472          5017  1           !--
;  473        M 5018  1           SET_OVERR_MODE (SEMSP) =
;  474        M 5019  1                   BEGIN
;  475        M 5020  1                   PAT$SET_OVERS (OVERRIDE_MODE, .PAT$GL_SEMAN1[SEMSP]);
;  476        M 5021  1                   PAT$SET_COMQUAL(SEMSP);
;  477          5022  1                   END%,
;  478          5023  1
;  479          5024  1           !++
;  480          5025  1           ! The SET_DEC_OVERS macro calls the routine PAT$SET_OVERS to
;  481          5026  1           ! set the mode pointer to LOCAL level, and to set the mode
;  482          5027  1           ! to decimal.
;  483          5028  1           !--
;  484        M 5029  1           SET_DEC_OVERS =
;  485        M 5030  1                   BEGIN
;  486        M 5031  1                   PAT$SET_MOD_LVL(OVERRIDE_MODE);
;  487        M 5032  1                   PAT$SET_OVERS (LOCAL_MODE, DECIMAL_TOKEN);
;  488          5033  1                   END%,
;  489          5034  1
;  490          5035  1           !++
;  491          5036  1           ! The SET_MODE_BIT macro turns on the default bit in the
;  492          5037  1           ! PATCH context word.
;  493          5038  1           !--
;  494        M 5039  1           SET_MODE_BIT =
;  495          5040  1                   PAT$GL_CONTEXT [MODE_BIT] = TRUE%,
;  496          5041  1
;  497          5042  1 !++
;  498          5043  1 ! The next macro is GET_QUOTED_STG. Its major failing is
;  499          5044  1 ! that it writes into the input stream. This could be solved by
;  500          5045  1 ! calling it with another argument, the string descriptor for a
;  501          5046  1 ! writable string. For the nonce, it writes into an otherwise pure
;  502          5047  1 ! stream.
;  503          5048  1 !--
;  504          5049  1
;  505          5050  1 !++
;  506          5051  1 ! The next macro collects a string that is enclosed in quotes.
;  507          5052  1 !--
;  508          5053  1
;  509          5054  1           !++
```

```
;  510          5055  1        ! The GET_QUOTED_STG macro reads characters from the input string.  It
;  511          5056  1        ! picks up the delimiting character (' or ") and reads until the next
;  512          5057  1        ! occurance of that character.  If none is found, the invalid command
;  513          5058  1        ! message is output, and end of line processing is done. Otherwise, the
;  514          5059  1        ! length of the string is placed in the position of the opening quote.
;  515          5060  1        ! A zero is placed in the position of the closing quote for end-of-line.
;  516          5061  1        ! If the current mode is instruction mode, the address of the string is
;  517          5062  1        ! placed on the top of the first parse stack.  Otherwise, the string
;  518          5063  1        ! must be ASCII, so it is reduced to four characters or less and placed
;  519          5064  1        ! on the top of stack. The parse string descriptor is updated to address
;  520          5065  1        ! the character after the closing quote.
;  521          5066  1        !--
;  522        M 5067  1        GET_QUOTED_STG (SEMSP) =
;  523        M 5068  1                BEGIN
;  524        M 5069  1
;  525        M 5070  1                MAP
;  526        M 5071  1                        PARSE_STG_DESC : REF BLOCK [, BYTE];
;  527        M 5072  1
;  528        M 5073  1                LOCAL
;  529        M 5074  1                        CHAR,
;  530        M 5075  1                        COUNT,
;  531        M 5076  1                        DELIMITER,
;  532        M 5077  1                        INPUT_PTR : REF VECTOR[,BYTE],
;  533        M 5078  1                        TEMP_PTR;
;  534        M 5079  1
;  535        M 5080  1                IF (NOT .PAT$GB_MOD_PTR[MODE_INSTRUC]) AND
;  536        M 5081  1                   (NOT .PAT$GB_MOD_PTR[MODE_ASCII])
;  537        M 5082  1                THEN
;  538        M 5083  1                        SIGNAL(PAT$_INVQUO+MSG$K_WARN);
;  539        M 5084  1                QUOTE_INDIC = TRUE;
;  540        M 5085  1                INPUT_PTR = CH$PLUS (.PARSE_STG_DESC [DSC$A_POINTER], -1); ! Point to delimiter
;  541        M 5086  1                TEMP_PTR = CH$PTR(.INPUT_PTR, 0);
;  542        M 5087  1                DELIMITER = CH$RCHAR_A(INPUT_PTR);
;  543        M 5088  1                COUNT = 0;
;  544        M 5089  1                REPEAT
;  545        M 5090  1                        BEGIN
;  546        M 5091  1                        CHAR = CH$RCHAR_A (INPUT_PTR);
;  547        M 5092  1                        IF (.CHAR EQL 0)                        ! Line always ends with zero
;  548        M 5093  1                        THEN
;  549        M 5094  1                                BEGIN
;  550        M 5095  1                                !++
;  551        M 5096  1                                ! This message has been made informational
;  552        M 5097  1                                ! instead of warning, to allow user typo's
;  553        M 5098  1                                ! of eliminating the closing quote on symbolic
;  554        M 5099  1                                ! instructions.  This will eliminate annoyance
;  555        M 5100  1                                ! of aborting the command after multiple input lines.
;  556        M 5101  1                                !--
;  557        M 5102  1                                SIGNAL (PAT$_MISSQUO+MSG$K_INFO);
;  558        M 5103  1                                CHAR = .DELIMITER;
;  559        M 5104  1                                END;
;  560        M 5105  1                        IF (.CHAR EQL .DELIMITER)
;  561        M 5106  1                        THEN
;  562        M 5107  1                                BEGIN
;  563        M 5108  1                                !++
;  564        M 5109  1                                ! Found a closing quote. Replace the opening
;  565        M 5110  1                                ! quote with the length of the quoted string.
;  566        M 5111  1                                ! Replace th closing quote with a zero.  This
```

```
;  567          M 5112  1                    ! is for forward referencing inside symbolic
;  568          M 5113  1                    ! instructions.
;  569          M 5114  1                    !--
;  570          M 5115  1                    CH$WCHAR (.COUNT, .TEMP_PTR);
;  571          M 5116  1                    INPUT_PTR[-1] = 0;
;  572          M 5117  1                    EXITLOOP
;  573          M 5118  1                    END
;  574          M 5119  1            ELSE
;  575          M 5120  1                    BEGIN
;  576          M 5121  1                    COUNT = .COUNT + 1;
;  577          M 5122  1                    END;
;  578          M 5123  1            END;
;  579          M 5124  1
;  580          M 5125  1        !++
;  581          M 5126  1        ! Quoted string found. Put the QTD_STG_TOKEN on the
;  582          M 5127  1        ! the first parse stack. Put the address of the string on
;  583          M 5128  1        ! the first parse stack.
;  584          M 5129  1        !--
;  585          M 5130  1        IF .PAT$GB_MOD_PTR [MODE_INSTRUC]
;  586          M 5131  1        THEN PAT$GL_SEMAN1 [SEMSP] = .TEMP_PTR
;  587          M 5132  1        ELSE
;  588          M 5133  1                BEGIN
;  589          M 5134  1
;  590          M 5135  1                MAP
;  591          M 5136  1                        TEMP_PTR : REF VECTOR [, BYTE];
;  592          M 5137  1
;  593          M 5138  1                LOCAL
;  594          M 5139  1                        VALUE : VECTOR [A_LONGWORD, BYTE];
;  595          M 5140  1
;  596          M 5141  1                VALUE = 0;
;  597          M 5142  1                INCR INDEX FROM 0 TO
;  598          M 5143  1                        (IF .TEMP_PTR [0] LEQ 4
;  599          M 5144  1                         THEN .TEMP_PTR [0] - 1
;  600          M 5145  1                         ELSE 3)
;  601          M 5146  1                DO VALUE [.INDEX] = .TEMP_PTR [.INDEX + 1];
;  602          M 5147  1                IF .TEMP_PTR [0] GTR .PAT$GB_MOD_PTR [MODE_LENGTH]
;  603          M 5148  1                THEN
;  604          M 5149  1                        SIGNAL (PAT$_STGTRUNC);
;  605          M 5150  1                PAT$GL_SEMAN1 [SEMSP] = .VALUE;
;  606          M 5151  1                END;
;  607          M 5152  1
;  608          M 5153  1        !++
;  609          M 5154  1        ! Now update the parse string descriptor so that the
;  610          M 5155  1        ! address of the buffer is the address of the character
;  611          M 5156  1        ! after the closing quote.
;  612          M 5157  1        !--
;  613          M 5158  1        PARSE_STG_DESC [DSC$A_POINTER] = CH$PTR (.INPUT_PTR, 0);
;  614          M 5159  1        PARSE_STG_DESC [DSC$W_LENGTH] = .PARSE_STG_DESC [DSC$W_LENGTH] - (.COUNT + 1);
;  615            5160  1        END%,
;  616            5161  1
```

```
;  618          5162  1 !++
;  619          5163  1 ! This macro collects a file specification.
;  620          5164  1 !--
;  621          5165  1
;  622          5166  1         !++
;  623          5167  1         ! The GET_FILE_SPEC macro reads from the input string starting
;  624          5168  1         ! at the character after the keyword CREATE until a character is
;  625          5169  1         ! found that is not a space, tab, or a zero. If such a character
;  626          5170  1         ! is found before the end of line is reached, then the position
;  627          5171  1         ! of the character before is noted for storage of the length of
;  628          5172  1         ! the file spec string. Then the string is read until a zero
;  629          5173  1         ! or carriage return character is encountered. The length
;  630          5174  1         ! of the string is stored in the previously saved position.
;  631          5175  1         ! Finally, the address of the file specification is placed in
;  632          5176  1         ! a new link of the command argument list by a call to the
;  633          5177  1         ! routine PAT$ADD_ARG. This address is the address of a counted
;  634          5178  1         ! string.
;  635          5179  1         !--
;  636        M 5180  1 GET_FILE_SPEC =
;  637        M 5181  1
;  638        M 5182  1                 BEGIN
;  639        M 5183  1
;  640        M 5184  1                 MAP
;  641        M 5185  1                         PARSE_STG_DESC : REF BLOCK [, BYTE];
;  642        M 5186  1
;  643        M 5187  1                 LOCAL
;  644        M 5188  1                         CHAR,
;  645        M 5189  1                         COUNT,
;  646        M 5190  1                         INPUT_PTR,
;  647        M 5191  1                         TEMP_PTR;
;  648        M 5192  1
;  649        M 5193  1                 INPUT_PTR = CH$PTR (.PARSE_STG_DESC [DSC$A_POINTER]);
;  650        M 5194  1                 CHAR = CH$RCHAR (.INPUT_PTR);
;  651        M 5195  1
;  652        M 5196  1
;  653        M 5197  1                 !++
;  654        M 5198  1                 ! This loop skips spaces and tabs that delimit the CREATE
;  655        M 5199  1                 ! verb and precede the file specification. At the end of
;  656        M 5200  1                 ! this loop, the character pointer INPUT_PTR is pointing
;  657        M 5201  1                 ! to the first character of the file specification.
;  658        M 5202  1                 !--
;  659        M 5203  1                 COUNT = 0;
;  660        M 5204  1                 REPEAT
;  661        M 5205  1                         BEGIN
;  662        M 5206  1
;  663        M 5207  1                         !++
;  664        M 5208  1                         ! The character we recognize as the beginning
;  665        M 5209  1                         ! of the file specification is the first
;  666        M 5210  1                         ! character after the delimiter of the CREATE verb.
;  667        M 5211  1                         !--
;  668        M 5212  1                         IF .CHAR NEQ ASC_SPACE AND .CHAR NEQ ASC_TAB
;  669        M 5213  1                         THEN EXITLOOP
;  670        M 5214  1                         ELSE
;  671        M 5215  1                                 BEGIN
;  672        M 5216  1                                 COUNT = .COUNT + 1;
;  673        M 5217  1                                 CHAR = CH$A_RCHAR (INPUT_PTR);
;  674        M 5218  1                                 END;
```

```
;  675          M 5219  1                END;
;  676          M 5220  1
;  677          M 5221  1            !++
;  678          M 5222  1            ! Check if there was no file name specified.
;  679          M 5223  1            ! If there was no file name, then PATCH wants to return
;  680          M 5224  1            ! successfully from this action routine.  The only reason
;  681          M 5225  1            ! this macro can contain a "RETURN TRUE" statement, is that
;  682          M 5226  1            ! MAR_REDUCTN merely returns after executing it.
;  683          M 5227  1            !--
;  684          M 5228  1            IF (.CHAR EQL 0)
;  685          M 5229  1            THEN
;  686          M 5230  1                    EXITLOOP(TRUE);
;  687          M 5231  1
;  688          M 5232  1            !++
;  689          M 5233  1            ! The beginning of the file specification has been
;  690          M 5234  1            ! found. If the delimiter was non-null, then
;  691          M 5235  1            ! collect the rest of the string.
;  692          M 5236  1            !--
;  693          M 5237  1            IF (.COUNT LEQ 0)
;  694          M 5238  1            THEN
;  695          M 5239  1                    BEGIN
;  696          M 5240  1                    IF (.TOKEN EQL DIGIT_STR_TOKEN) OR (.TOKEN EQL EOL_TOKEN)
;  697          M 5241  1                    THEN
;  698          M 5242  1                            SIGNAL (PAT$_INVCMD)
;  699          M 5243  1                    ELSE
;  700          M 5244  1                            BEGIN
;  701          M 5245  1                            SIGNAL(PAT$_SYNTAX+MSG$K_WARN, 1, LEX_STG_DESC);
;  702          M 5246  1                            RETURN;
;  703          M 5247  1                            END
;  704          M 5248  1                    END;
;  705          M 5249  1
;  706          M 5250  1            PARSE_STG_DESC [DSC$W_LENGTH] = .PARSE_STG_DESC [DSC$W_LENGTH] - .COUNT;
;  707          M 5251  1            TEMP_PTR = CH$PLUS (.INPUT_PTR, -1);
;  708          M 5252  1            COUNT = 0;
;  709          M 5253  1            REPEAT
;  710          M 5254  1                    BEGIN
;  711          M 5255  1                    IF .CHAR EQL 0
;  712          M 5256  1 !                          OR .CHAR EQL CARRIAGE_RET              !***line always ends in zero***
;  713          M 5257  1                    THEN
;  714          M 5258  1                            BEGIN
;  715          M 5259  1
;  716          M 5260  1                            !++
;  717          M 5261  1                            ! Found the end of the file specification.
;  718          M 5262  1                            ! Exit this loop after setting the count
;  719          M 5263  1                            ! byte of this counted string.
;  720          M 5264  1                            !--
;  721          M 5265  1                            IF (.COUNT LEQ 0)
;  722          M 5266  1                            THEN
;  723          M 5267  1                                    BEGIN
;  724          M 5268  1
;  725          M 5269  1                                    SIGNAL (PAT$_INVCMD);
;  726          M 5270  1                                    END;
;  727          M 5271  1                            CH$WCHAR (.COUNT, .TEMP_PTR);
;  728          M 5272  1                            EXITLOOP
;  729          M 5273  1                            END
;  730          M 5274  1                    ELSE
;  731          M 5275  1                            BEGIN
```

```
;   732        M 5276  1                        COUNT = .COUNT + 1;
;   733        M 5277  1                        CHAR = CH$A_RCHAR (INPUT_PTR);
;   734        M 5278  1                        END;
;   735        M 5279  1                END;
;   736        M 5280  1            !++
;   737        M 5281  1            ! Now put the address of the file specification into
;   738        M 5282  1            ! a new link in the command argument list.
;   739        M 5283  1            ! Also, increment the address of the parse string in
;   740        M 5284  1            ! the parse string descriptor to address the delimiting
;   741        M 5285  1            ! carriage return or null byte.
;   742        M 5286  1            !--
;   743        M 5287  1            PAT$ADD_ARG (.TEMP_PTR);
;   744        M 5288  1            PARSE_STG_DESC [DSC$A_POINTER] = .INPUT_PTR;
;   745        M 5289  1            PARSE_STG_DESC [DSC$W_LENGTH] = .PARSE_STG_DESC [DSC$W_LENGTH] - .COUNT;
;   746          5290  1            END%,
;   747          5291  1
;   748          5292  1  !++
;   749          5293  1  ! The macros below manipulate names of data specified in commands.
;   750          5294  1  !--
;   751          5295  1
;   752          5296  1        !++
;   753          5297  1        ! The TRANSLATE_NAME macro calls the routine PAT$TRANS_NAME to
;   754          5298  1        ! translate a token into a binary value. If the routine fails in
;   755          5299  1        ! the translation, a signal and unwind occurs.
;   756          5300  1        !--
;   757        M 5301  1        TRANSLATE_NAME (SEMSP) =
;   758          5302  1                PAT$TRANS_NAME (SEMSP, LEX_STG_DESC)%,
;   759          5303  1
;   760          5304  1  !++
;   761          5305  1  ! The next macro is called when a complete command has been collected.
;   762          5306  1  !--
;   763          5307  1
;   764          5308  1        !++
;   765          5309  1        ! The EXECUTE_CMD macro calls PAT$PERFORM_CMD and returns
;   766          5310  1        ! from PARSE_A_LINE if that routine returns a false value.
;   767          5311  1        !--
;   768        M 5312  1        EXECUTE_CMD (SEMSP) =
;   769        M 5313  1                IF NOT PAT$PERFORM_CMD (SEMSP)
;   770        M 5314  1                THEN
;   771          5315  1                        RETURN%,
;   772          5316  1
;   773          5317  1
;   774          5318  1  !++
;   775          5319  1  ! The next macros manipulate scope and pathnames.
;   776          5320  1  !--
;   777          5321  1
;   778          5322  1        !++
;   779          5323  1        ! The REDUCE_PATHNAME macro calls PAT$BUILD_PATH to convert a
;   780          5324  1        ! pathname into an equivalent value. If the conversion fails,
;   781          5325  1        ! then the symbol does not exist, a message is produced, and an
;   782          5326  1        ! UNWIND is done.
;   783          5327  1        !--
;   784        M 5328  1        REDUCE_PATHNAME (SEMSP) =
;   785          5329  1                PAT$BUILD_PATH (0, PAT$GL_SEMAN1 [SEMSP], TRUE)%,
;   786          5330  1
;   787          5331  1
;   788          5332  1        !++
```

```
;  789          5333  1          ! The macro SET_SCOPE_BIT turns on the appropriate context
;  790          5334  1          ! bit so that a path name will be saved.
;  791          5335  1          !--
;  792        M 5336  1          SET_SCOPE_BIT =
;  793          5337  1                  PAT$GL_CONTEXT [SCOPE_BIT] = TRUE%,
;  794          5338  1
;  795          5339  1  !++
;  796          5340  1  ! The next macro sets the context bit indicating the "SET PATCH_AREA" command.
;  797          5341  1  !
;  798          5342  1  ! The macro also tests to see if the address of the patch area descriptor was declared before
;  799          5343  1  ! the /INITIALIZE qualifier (if it was present).  If the /INITIALIZE qualifier came first, then
;  800          5344  1  ! move the initial size value into the second expression location in the linked argument list,
;  801          5345  1  ! followed by copying the address of the patch area descriptor into the first expression location.
;  802          5346  1  !--
;  803          5347  1
;  804        M 5348  1          SET_PATAREA_BIT =
;  805        M 5349  1                  BEGIN
;  806        M 5350  1                  PAT$GL_CONTEXT [PAT_AREA_BIT] = TRUE;
;  807        M 5351  1                  IF .PAT$GL_HEAD_LST NEQ .PAT$GL_TAIL_LST THEN
;  808        M 5352  1                          BEGIN
;  809        M 5353  1                          LIST_ELEM_EXP2 (.PAT$GL_HEAD_LST) = .LIST_ELEM_EXP1 (.PAT$GL_HEAD_LST);
;  810        M 5354  1                          LIST_ELEM_EXP1 (.PAT$GL_HEAD_LST) = .LIST_ELEM_EXP1 (.PAT$GL_TAIL_LST);
;  811        M 5355  1                          END;
;  812          5356  1                  END%,
;  813          5357  1
;  814          5358  1  !++
;  815          5359  1  ! The next macro sets the context bit indicating that the /INITIALIZE qualifier
;  816          5360  1  ! was present in the SET PATCH_AREA command.
;  817          5361  1  !
;  818          5362  1  ! The macro also checks to see if the /Initialize qualifier was specified after
;  819          5363  1  ! the patch area descriptor.  If this is the case, all we have to do is to copy
;  820          5364  1  ! the initial size value into the second expression location of the first element
;  821          5365  1  ! of the linked argument list.
;  822          5366  1  !--
;  823          5367  1
;  824        M 5368  1          SET_INIT_BIT (SEMSP) =
;  825        M 5369  1                  BEGIN
;  826        M 5370  1                  PAT$GL_CONTEXT [INIT_PAT_BIT] = TRUE;
;  827        M 5371  1                  PAT$GL_COMQUAL [INITIALIZE_QUAL] = TRUE;
;  828        M 5372  1                  IF .PAT$GL_HEAD_LST NEQ .PAT$GL_TAIL_LST THEN
;  829        M 5373  1                          LIST_ELEM_EXP2 (.PAT$GL_HEAD_LST) = .LIST_ELEM_EXP1 (.PAT$GL_TAIL_LST);
;  830          5374  1                  END%,
;  831          5375  1
;  832          5376  1  !++
;  833          5377  1  ! The next macro sets the context bit for the keyword "MODULE".
;  834          5378  1  !--
;  835          5379  1
;  836        M 5380  1          SET_MODULE_BIT =
;  837          5381  1                  PAT$GL_CONTEXT [MODULE_BIT] = TRUE%,
;  838          5382  1
;  839          5383  1  !++
;  840          5384  1  ! The next macro sets the context bit for the LITERAL qualifier.
;  841          5385  1  !--
;  842        M 5386  1          SET_LIT_BIT =
;  843          5387  1                  PAT$GL_CONTEXT[LITERAL_BIT] = TRUE%,
;  844          5388  1
;  845          5389  1  !++
```

```
;  846          5390  1 ! The next two macros set bits indicating what type of ECO command is
;  847          5391  1 ! to be executed.
;  848          5392  1 !--
;  849          5393  1
;  850          5394  1         !++
;  851          5395  1         ! The macro SET_NOT_ECO_BIT sets a bit to indicate the command
;  852          5396  1         ! was CHECK NOT ECO, i.e., check that the eco level bits
;  853          5397  1         ! are NOT set.
;  854          5398  1         !--
;  855        M 5399  1         SET_NOT_ECO_BIT (SEMSP) =
;  856        M 5400  1                 BEGIN
;  857        M 5401  1                 PAT$GL_CONTEXT [SET_NOT_ECO] = TRUE;
;  858        M 5402  1                 SET_DEC_OVERS;
;  859        M 5403  1                 END%,
;  860          5404  1
;  861          5405  1         !++
;  862          5406  1         ! The macro SET_ECO_BIT sets a context bit to indicate the command
;  863          5407  1         ! was setting ECO bits.
;  864          5408  1         !--
;  865        M 5409  1         SET_ECO_BIT (SEMSP) =
;  866        M 5410  1                 BEGIN
;  867        M 5411  1                 PAT$GL_CONTEXT [SET_ECO] = TRUE;
;  868        M 5412  1                 SET_DEC_OVERS;
;  869        M 5413  1                 END%,
;  870          5414  1
;  871          5415  1 !++
;  872          5416  1 ! The next five macros set the align context bits.
;  873          5417  1 !--
;  874          5418  1
;  875        M 5419  1         SET_BYTE_BIT =
;  876          5420  1                 PAT$GL_CONTEXT [ALIGN_BYTE] = TRUE%,
;  877          5421  1
;  878        M 5422  1         SET_LONG_BIT =
;  879          5423  1                 PAT$GL_CONTEXT [ALIGN_LONG] = TRUE%,
;  880          5424  1
;  881        M 5425  1         SET_PAGE_BIT =
;  882          5426  1                 PAT$GL_CONTEXT [ALIGN_PAGE] = TRUE%,
;  883          5427  1
;  884        M 5428  1         SET_QUAD_BIT =
;  885          5429  1                 PAT$GL_CONTEXT [ALIGN_QUAD] = TRUE%,
;  886          5430  1
;  887        M 5431  1         SET_WORD_BIT =
;  888          5432  1                 PAT$GL_CONTEXT [ALIGN_WORD] = TRUE%,
;  889          5433  1
;  890          5434  1 !++
;  891          5435  1 ! The next six macros perform a logical test between the two items
;  892          5436  1 ! and then place TRUE or FALSE on the top of the stack.
;  893          5437  1 !--
;  894          5438  1
;  895        M 5439  1         EQ_EXPR (SEMSP) =
;  896        M 5440  1                 PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] EQL
;  897          5441  1                                         .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  898          5442  1
;  899        M 5443  1         GE_EXPR (SEMSP) =
;  900        M 5444  1                 PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] GEQ
;  901          5445  1                                         .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  902          5446  1
```

```
;  903          M 5447  1          GT_EXPR (SEMSP) =
;  904          M 5448  1                  PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] GTR
;  905            5449  1                                          .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  906            5450  1
;  907          M 5451  1          LE_EXPR (SEMSP) =
;  908          M 5452  1                  PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] LEQ
;  909            5453  1                                          .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  910            5454  1
;  911          M 5455  1          LT_EXPR (SEMSP) =
;  912          M 5456  1                  PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] LSS
;  913            5457  1                                          .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
;  914            5458  1
;  915          M 5459  1          NE_EXPR (SEMSP) =
;  916          M 5460  1                  PAT$GL_SEMAN1 [SEMSP] = .PAT$GL_SEMAN1 [SEMSP] NEQ
;  917            5461  1                                          .PAT$GL_SEMAN1 [SEMSP + PAT$K_SPOS_TWO]%,
```

```
;  919          5462  1 !++
;  920          5463  1 ! This macro collects a HELP topic specification.
;  921          5464  1 !--
;  922          5465  1
;  923          5466  1         !++
;  924          5467  1         ! The GET_HELP_TOPIC macro reads from the input string starting
;  925          5468  1         ! at the character after the keyword HELP until a character is
;  926          5469  1         ! found that is not a space, tab, or a zero. If such a character
;  927          5470  1         ! is found before the end of line is reached, then the position
;  928          5471  1         ! of the character is noted for storage in the pointer field of
;  929          5472  1         ! PAT$GL_HELP_LIN [DSC$A_POINTER].  Then the string is read until
;  930          5473  1         ! a zero or carriage return character is encountered.
;  931          5474  1         !--
;  932          5475  1
;  933        M 5476  1 GET_HELP_TOPIC =
;  934        M 5477  1
;  935        M 5478  1                 BEGIN
;  936        M 5479  1
;  937        M 5480  1                 MAP
;  938        M 5481  1                         PARSE_STG_DESC : REF BLOCK [, BYTE];
;  939        M 5482  1
;  940        M 5483  1                 LOCAL
;  941        M 5484  1                         HELP_CHAR,
;  942        M 5485  1                         HELP_COUNT,
;  943        M 5486  1                         HELP_INPUT_PTR;
;  944        M 5487  1
;  945        M 5488  1                 HELP_INPUT_PTR = CH$PTR (.PARSE_STG_DESC [DSC$A_POINTER]);
;  946        M 5489  1                 HELP_CHAR = CH$RCHAR (.HELP_INPUT_PTR);
;  947        M 5490  1
;  948        M 5491  1
;  949        M 5492  1                 !++
;  950        M 5493  1                 ! This loop skips spaces and tabs that delimit the HELP
;  951        M 5494  1                 ! verb and precede the topic specification. At the end of
;  952        M 5495  1                 ! this loop, the character pointer HELP_INPUT_PTR is pointing
;  953        M 5496  1                 ! to the first character of the topic specification.
;  954        M 5497  1                 !--
;  955        M 5498  1
;  956        M 5499  1                 HELP_COUNT = 0;
;  957        M 5500  1                 REPEAT
;  958        M 5501  1                         BEGIN
;  959        M 5502  1
;  960        M 5503  1                         !++
;  961        M 5504  1                         ! The character we recognize as the beginning
;  962        M 5505  1                         ! of the topic specification is the first
;  963        M 5506  1                         ! character after the delimiter of the HELP verb.
;  964        M 5507  1                         !--
;  965        M 5508  1
;  966        M 5509  1                         IF .HELP_CHAR NEQ ASC_SPACE AND .HELP_CHAR NEQ ASC_TAB
;  967        M 5510  1                         THEN EXITLOOP
;  968        M 5511  1                         ELSE
;  969        M 5512  1                                 BEGIN
;  970        M 5513  1                                 HELP_COUNT = .HELP_COUNT + 1;
;  971        M 5514  1                                 HELP_CHAR = CH$A_RCHAR (HELP_INPUT_PTR);
;  972        M 5515  1                                 END;
;  973        M 5516  1                         END;
;  974        M 5517  1
;  975        M 5518  1                 !++
```

```
;  976        M 5519  1              ! Check if there was no HELP topic specified.
;  977        M 5520  1              ! If there was no topic, then PATCH wants to return
;  978        M 5521  1              ! successfully from this action routine.  The only reason
;  979        M 5522  1              ! this macro can contain a "RETURN TRUE" statement, is that
;  980        M 5523  1              ! MAR_REDUCTN merely returns after executing it.
;  981        M 5524  1              ! Stated another way...
;  982        M 5525  1              !       We will be leaving the current parse string pointing
;  983        M 5526  1              ! to the end of the line, hence the EOL token is going to
;  984        M 5527  1              ! cause us to call the real action routine, which can deal
;  985        M 5528  1              ! with not having any topic specified on the command line.
;  986        M 5529  1              !--
;  987        M 5530  1
;  988        M 5531  1              IF (.HELP_CHAR EQL 0)           !***line always ends in zero***
;  989        M 5532  1              THEN
;  990        M 5533  1                      EXITLOOP(TRUE);
;  991        M 5534  1
;  992        M 5535  1              !++
;  993        M 5536  1              ! With the HELP topic now in hand, we now reduce the remaining number of
;  994        M 5537  1              ! characters in the parse string.  Don't forget to set the global pointer
;  995        M 5538  1              ! PAT$GL_HELP_LIN [DSC$A_POINTER] to the beginning of the topic string.
;  996        M 5539  1              !--
;  997        M 5540  1
;  998        M 5541  1              PARSE_STG_DESC [DSC$W_LENGTH] = .PARSE_STG_DESC [DSC$W_LENGTH] - .HELP_COUNT;
;  999        M 5542  1              PAT$GL_HELP_LIN [DSC$A_POINTER] = .HELP_INPUT_PTR;
; 1000        M 5543  1
; 1001        M 5544  1              !++
; 1002        M 5545  1              ! Now we will busy ourselves with faking out the parser...
; 1003        M 5546  1              !       The reason for this action, is to allow the LBR$OUTPUT_HELP routine
; 1004        M 5547  1              ! to do its own parsing of the remainder of the command string.
; 1005        M 5548  1              ! Further, the parser doesn't quit easily.  Sooo, we kinda force it to
; 1006        M 5549  1              ! execute the command by simulating the end of line condition.
; 1007        M 5550  1              !       This is done by counting the number of characters remaining in the
; 1008        M 5551  1              ! parse string (between the beginning character of the topic string and the
; 1009        M 5552  1              ! end of line mark) then reducing the parse string count by that amount.
; 1010        M 5553  1              ! We also at this time update the length portion of the global length
; 1011        M 5554  1              ! PAT$GL_HELP_LIN [DSC$W_LENGTH].
; 1012        M 5555  1              !--
; 1013        M 5556  1
; 1014        M 5557  1              HELP_COUNT = 0;
; 1015        M 5558  1              REPEAT
; 1016        M 5559  1                      BEGIN
; 1017        M 5560  1
; 1018        M 5561  1                      IF .HELP_CHAR EQL 0             !***line always ends in zero***
; 1019        M 5562  1                      THEN
; 1020        M 5563  1                              EXITLOOP
; 1021        M 5564  1                      ELSE
; 1022        M 5565  1                              BEGIN
; 1023        M 5566  1                              HELP_COUNT = .HELP_COUNT + 1;
; 1024        M 5567  1                              HELP_CHAR = CH$A_RCHAR (HELP_INPUT_PTR);
; 1025        M 5568  1                              END;
; 1026        M 5569  1                      END;
; 1027        M 5570  1              !++
; 1028        M 5571  1              ! Now set the length of the Help topic, into the global
; 1029        M 5572  1              ! descriptor field PAT$GL_HELP_LIN [DSC$W_LENGTH].
; 1030        M 5573  1              ! Also, increment the address of the parse string in
; 1031        M 5574  1              ! the parse string descriptor to address the delimiting
; 1032        M 5575  1              ! carriage return or null byte.
```

```
;  1033          M 5576  1          !--
;  1034          M 5577  1
;  1035          M 5578  1          PAT$GL_HELP_LIN [DSC$W_LENGTH] = .HELP_COUNT;
;  1036          M 5579  1          PARSE_STG_DESC [DSC$A_POINTER] = .HELP_INPUT_PTR;
;  1037          M 5580  1          PARSE_STG_DESC [DSC$W_LENGTH] = .PARSE_STG_DESC [DSC$W_LENGTH] - .HELP_COUNT;
;  1038            5581  1          END%;
;  1039            5582  1
```

```
;  1041          5583  1 GLOBAL ROUTINE PAT$PARS_A_LINE (PARSE_STG_DESC) : NOVALUE =
;  1042          5584  1
;  1043          5585  1 !++
;  1044          5586  1 ! FUNCTIONAL DESCRIPTION:
;  1045          5587  1 !
;  1046          5588  1 !       Parses a line of PATCH commands and performs associated action routines.
;  1047          5589  1 !
;  1048          5590  1 ! CALLING SEQUENCE:
;  1049          5591  1 !
;  1050          5592  1 !       PAT$PARS_A_LINE ()
;  1051          5593  1 !
;  1052          5594  1 ! INPUTS:
;  1053          5595  1 !
;  1054          5596  1 !       PARSE_STG_DESC  - string descriptor of the buffer that holds
;  1055          5597  1 !                         the input string.
;  1056          5598  1 !
;  1057          5599  1 ! IMPLICIT INPUTS:
;  1058          5600  1 !
;  1059          5601  1 !       the parsing tables
;  1060          5602  1 !
;  1061          5603  1 ! OUTPUTS:
;  1062          5604  1 !
;  1063          5605  1 !       No value returned.
;  1064          5606  1 !       Outputs are the effects of the action routines called.
;  1065          5607  1 !
;  1066          5608  1 ! IMPLICIT OUTPUTS:
;  1067          5609  1 !
;  1068          5610  1 !       none
;  1069          5611  1 !
;  1070          5612  1 ! ROUTINE VALUE:
;  1071          5613  1 !
;  1072          5614  1 !       novalue
;  1073          5615  1 !
;  1074          5616  1 ! SIDE EFFECTS:
;  1075          5617  1 !
;  1076          5618  1 !       none
;  1077          5619  1 !--
;  1078          5620  2 BEGIN
;  1079          5621  2
;  1080          5622  2 MAP
;  1081          5623  2         PARSE_STG_DESC : REF BLOCK[,BYTE];
;  1082          5624  2
;  1083          5625  2 LOCAL
;  1084          5626  2         ACTION_TO_TAKE,                                     ! Action from PAT_ACT_TABLE
;  1085          5627  2         CUR_PARSE_STATE,                                    ! The state of the parse machine
;  1086          5628  2         J,                                                  ! Control variable
;  1087          5629  2         LEXEME_ADDR: VECTOR [CHS_PER_LEXEME, BYTE],         ! Buffer for string lexeme
;  1088          5630  2         LAHEAD_ADDR: VECTOR [CHS_PER_LEXEME, BYTE],         ! Buffer for a lookahead
;  1089          5631  2         LEX_STG_DESC: BLOCK [12, BYTE] ,                    ! String descriptor for lexemes
;  1090          5632  2         LAHEAD_STG_DESC: BLOCK [8, BYTE],                   ! Lookahead for one lexeme
;  1091          5633  2         LAST_STG_DESC : BLOCK[8,BYTE],                      ! Previous place in PARSE_STG_DESC
;  1092          5634  2         MATCH_TRANSIT,                                      ! Boolean, TRUE if matching transition is fo
;  1093          5635  2         NEXT_TOKEN : VOLATILE,                              ! Saves next token
;  1094          5636  2         OLD_J,                                              ! Saves state
;  1095          5637  2         OLD_TOKEN,                                          ! Saved token
;  1096          5638  2         PARSE_MORE,                                         ! Boolean, says whether to parse more
;  1097          5639  2         PARSE_STACK: VECTOR [MAX_STACK_PTR * PAT$K_STELM_SIZ, WORD], ! Parse state stack
```

```
;  1098          5640  2          REDUCE_WAS_LAST,                        ! TRUE if reduction was last action
;  1099          5641  2          SCAN_NEXT_SYM,                          ! Boolean, TRUE if next symbol needs to be s
;  1100          5642  2          STACK_POINTER,                          ! Stack INDEX
;  1101          5643  2          TOKEN,                                  ! holds latest token
;  1102          5644  2          TRANSIT_CODE;                           ! Transition code from PAT_STATE_TABLE
;  1103          5645  2
;  1104          5646  2 LABEL
;  1105          5647  2          MATCH_LOOP;
;  1106          5648  2
;  1107          5649  2 !++
;  1108          5650  2 ! Initialize control variables.  Get a token from the input string.
;  1109          5651  2 !--
;  1110          5652  2 OLD_TOKEN = 0;
;  1111          5653  2 REDUCE_WAS_LAST = FALSE;
;  1112          5654  2 STACK_POINTER = 0;
;  1113          5655  2 QUOTE_INDIC = FALSE;
;  1114          5656  2 CH$FILL (0,DSC$C_S_BLN,PAT$GL_HELP_LIN);
;  1115          5657  2 !++
;  1116          5658  2 ! Initialize the look-ahead string descriptor.
;  1117          5659  2 !--
;  1118          5660  2 LEX_STG_DESC [DSC$W_LENGTH] = 0;
;  1119          5661  2 LEX_STG_DESC [DSC$W_MAXLEN] = CHS_PER_LEXEME;
;  1120          5662  2 LEX_STG_DESC [DSC$A_POINTER] = LEXEME_ADDR;
;  1121          5663  2 LAHEAD_STG_DESC [DSC$A_POINTER] = LAHEAD_ADDR;
;  1122          5664  2
;  1123          5665  2 !++
;  1124          5666  2 ! Maintain a 'last' string descriptor which is always what 'PARSE_STG_DESC'
;  1125          5667  2 ! was before the last time it was changed.
;  1126          5668  2 !--
;  1127          5669  2 LAST_STG_DESC[DSC$W_LENGTH] = .PARSE_STG_DESC[DSC$W_LENGTH];
;  1128          5670  2 LAST_STG_DESC[DSC$A_POINTER] = .PARSE_STG_DESC[DSC$A_POINTER];
;  1129          5671  2
;  1130          5672  2 !++
;  1131          5673  2 ! Get the first token from the command line.
;  1132          5674  2 !--
;  1133          5675  2 TOKEN = PAT$GET_A_TOKEN (.PARSE_STG_DESC, LEX_STG_DESC);
;  1134          5676  2
;  1135          5677  2 !++
;  1136          5678  2 ! Initialize the parse control variables.
;  1137          5679  2 !--
;  1138          5680  2 CUR_PARSE_STATE = 0;
;  1139          5681  2 SCAN_NEXT_SYM = TRUE;
;  1140          5682  2 PARSE_MORE = TRUE;
;  1141          5683  2 MATCH_TRANSIT = FALSE;
;  1142          5684  2
;  1143          5685  2 !++
;  1144          5686  2 ! This is the main loop of the parser.  It continues
;  1145          5687  2 ! until the variable "PARSE_MORE" has a value of FALSE.
;  1146          5688  2 !--
;  1147          5689  2 DO
;  1148          5690  3          BEGIN
;  1149          5691  3          J = .CUR_PARSE_STATE;
;  1150          5692  3
;  1151          5693  3          !++
;  1152          5694  3          ! The following loop searches for a matching token
;  1153          5695  3          ! and exits when a match is found.
;  1154          5696  3          !--
```

```
;  1155     5697  3            DO
;  1156     5698  4 MATCH_LOOP:     BEGIN
;  1157     5699  4                 MATCH_TRANSIT = FALSE;
;  1158     5700  4                 TRANSIT_CODE = .PAT_STATE_TABLE [.J];
;  1159     5701  4
;  1160     5702  4                 !++
;  1161     5703  4                 ! See whether this transit code is an else code,
;  1162     5704  4                 ! and at the same time the token is a keyword
;  1163     5705  4                 ! token. In these circumstances, try passing
;  1164     5706  4                 ! through the loop again with the pretense
;  1165     5707  4                 ! that the token is a ALPHA_STR_TOKEN. The current
;  1166     5708  4                 ! token and state must be saved so that
;  1167     5709  4                 ! if ALPHA_STR_TOKEN does not make a valid sentence,
;  1168     5710  4                 ! then the effect of the else code can be reestablished.
;  1169     5711  4                 !--
;  1170     5712  5                 IF (.TRANSIT_CODE EQL ELSE_CODE) AND (.TOKEN LEQ KEYWORD_RANGE)
;  1171     5713  4                 THEN
;  1172     5714  5                         BEGIN
;  1173     5715  5                         OLD_TOKEN = .TOKEN;
;  1174     5716  5                         OLD_J = .J;
;  1175     5717  5                         J = .CUR_PARSE_STATE;
;  1176     5718  5                         TOKEN = ALPHA_STR_TOKEN;
;  1177     5719  5                         LEAVE MATCH_LOOP;
;  1178     5720  4                         END;
;  1179     5721  4
;  1180     5722  4                 !++
;  1181     5723  4                 ! Now allow for restoring the original token
;  1182     5724  4                 ! in the case that the newly inserted ALPHA_STR_TOKEN
;  1183     5725  4                 ! brought no better results.
;  1184     5726  4                 !--
;  1185     5727  5                 IF (.TRANSIT_CODE EQL ELSE_CODE) AND (.TOKEN EQL ALPHA_STR_TOKEN)
;  1186     5728  5                         AND (.OLD_TOKEN NEQ 0)
;  1187     5729  4                 THEN
;  1188     5730  5                         BEGIN
;  1189     5731  5                         TOKEN = .OLD_TOKEN;
;  1190     5732  5                         J = .OLD_J;
;  1191     5733  5                         OLD_TOKEN = 0;
;  1192     5734  4                         END;
;  1193     5735  4
;  1194     5736  4                 !++
;  1195     5737  4                 ! No special handling here. Just compare the
;  1196     5738  4                 ! token and the transit code.
;  1197     5739  4                 !--
;  1198     5740  4                 IF (.TRANSIT_CODE EQL .TOKEN) OR
;  1199     5741  5                    (.TRANSIT_CODE EQL ELSE_CODE)
;  1200     5742  4                 THEN
;  1201     5743  5                         BEGIN
;  1202     5744  5                         !++
;  1203     5745  5                         ! A match has been found, so the lexical string can be
;  1204     5746  5                         ! read or reduced.
;  1205     5747  5                         !--
;  1206     5748  5                         MATCH_TRANSIT = TRUE;
;  1207     5749  5                         ACTION_TO_TAKE = .PAT_ACT_TABLE [.J];
;  1208     5750  6                         IF (.ACTION_TO_TAKE NEQ ERROR_CODE)
;  1209     5751  5                         THEN
;  1210     5752  6                                 BEGIN
;  1211     5753  7                                 IF (.ACTION_TO_TAKE GTR - SCAN_CODE)
```

```
;  1212          5754  6                   THEN
;  1213          5755  6                           !++
;  1214          5756  6                           ! Nothing else to scan.  Perform the
;  1215          5757  6                           ! associated action routine for this
;  1216          5758  6                           ! lexical entity.
;  1217          5759  6                           !--
;  1218          5760  7                           BEGIN
;  1219          5761  7                           PARSE_STACK [.STACK_POINTER] = .CUR_PARSE_STATE;
;  1220          5762  7
;  1221          5763  7                           !++
;  1222          5764  7                           ! Put the token on the top of the parse stack
;  1223          5765  7                           !--
;  1224          5766  7                           IF NOT .REDUCE_WAS_LAST
;  1225          5767  7                           THEN PAT$GL_SEMANT [.STACK_POINTER] = .TOKEN;
;  1226          5768  6                           END;
;  1227          5769  6
;  1228          5770  7                   IF (.ACTION_TO_TAKE GEQ 0)
;  1229          5771  6                   THEN
;  1230          5772  6                           !++
;  1231          5773  6                           ! This is a read state.
;  1232          5774  6                           !--
;  1233          5775  7                           BEGIN
;  1234          5776  7                           CUR_PARSE_STATE = .ACTION_TO_TAKE;
;  1235          5777  7                           REDUCE_WAS_LAST = FALSE;
;  1236          5778  8                           IF (.STACK_POINTER GEQ (MAX_STACK_PTR * PAT$K_STELM_SIZ))
;  1237          5779  7                           THEN
;  1238          5780  8                                   BEGIN
;  1239          5781  8                                   SIGNAL (PAT$_PARSEERR)
;  1240          5782  8                                   END
;  1241          5783  7                           ELSE STACK_POINTER = .STACK_POINTER + PAT$K_SPOS_ONE;
;  1242          5784  7
;  1243          5785  7                           !++
;  1244          5786  7                           ! Now input the next token if more
;  1245          5787  7                           ! reading is necessary.
;  1246          5788  7                           !--
;  1247          5789  7                           IF .SCAN_NEXT_SYM
;  1248          5790  7                           THEN
;  1249          5791  8                                   BEGIN
;  1250        P 5792  8                                   ZEROCOR(.LAHEAD_STG_DESC[DSC$A_POINTER],
;  1251          5793  8                                           (.LAHEAD_STG_DESC[DSC$W_MAXLEN]/4));
;  1252          5794  8                                   LAHEAD_STG_DESC [DSC$W_LENGTH] =
;  1253          5795  8                                           .LEX_STG_DESC [DSC$W_LENGTH];
;  1254          5796  8                                   CH$MOVE (.LEX_STG_DESC [DSC$W_LENGTH],
;  1255          5797  8                                           .LEX_STG_DESC [DSC$A_POINTER],
;  1256          5798  8                                           .LAHEAD_STG_DESC [DSC$A_POINTER]);
;  1257          5799  8                                   LAST_STG_DESC[DSC$W_LENGTH] = .PARSE_STG_DESC[DSC$W_LENGTH];
;  1258          5800  8                                   LAST_STG_DESC[DSC$A_POINTER] = .PARSE_STG_DESC[DSC$A_POINTER
;  1259          5801  8                                   TOKEN = PAT$GET_A_TOKEN (.PARSE_STG_DESC, LEX_STG_DESC);
;  1260          5802  8                                   END
;  1261          5803  7                           ELSE
;  1262          5804  7                                   !++
;  1263          5805  7                                   ! If no more scanning is needed,
;  1264          5806  7                                   ! put the next token in the
;  1265          5807  7                                   ! variable "TOKEN".
;  1266          5808  8                                   BEGIN
;  1267          5809  8                                   SCAN_NEXT_SYM = TRUE;
;  1268          5810  8                                   TOKEN = .NEXT_TOKEN;
```

```
;  1269          5811  7                        END;
;  1270          5812  7
;  1271          5813  7                    !++
;  1272          5814  7                    ! This is the end of the read.
;  1273          5815  7                    !--
;  1274          5816  7                    END
;  1275          5817  6                ELSE
;  1276          5818  6                    !++
;  1277          5819  6                    ! This is a reduction.
;  1278          5820  6                    !--
;  1279          5821  6                    BEGIN
;  1280          5822  8                    IF (.ACTION_TO_TAKE LEQ - SCAN_CODE)
;  1281          5823  7                    THEN
;  1282          5824  7                        !++
;  1283          5825  7                        ! This is a noscan reduction,
;  1284          5826  7                        ! which takes an extra stack pop.
;  1285          5827  7                        !--
;  1286          5828  8                        BEGIN
;  1287          5829  8                        ACTION_TO_TAKE = .ACTION_TO_TAKE + SCAN_CODE;
;  1288          5830  8                        SCAN_NEXT_SYM = FALSE;
;  1289          5831  8                        STACK_POINTER = .STACK_POINTER - PAT$K_SPOS_ONE;
;  1290          5832  8                        NEXT_TOKEN = .TOKEN;
;  1291          5833  7                        END;
;  1292          5834  7
;  1293          5835  7                    ACTION_TO_TAKE = - .ACTION_TO_TAKE;
;  1294          5836  7                    STACK_POINTER = .STACK_POINTER -
;  1295          5837  7                            (.PAT_POP_TABLE [.ACTION_TO_TAKE] * PAT$K_STELM_SIZ);
;  1296          5838  7                    CUR_PARSE_STATE = .PARSE_STACK [.STACK_POINTER];
;  1297          5839  7                    TOKEN = .PAT_LHS_TABLE [.ACTION_TO_TAKE];
;  1298          5840  7                    REDUCE_WAS_LAST = TRUE;
;  1299          5841  7
;  1300          5842  7                    !++
;  1301          5843  7                    ! In here go the semantic actions for each reduction.
;  1302          5844  7                    !--
;  1303          5845  7                    IF NOT MAR_REDUCTN (.PAT_SEM_TABLE [.ACTION_TO_TAKE],
;  1304          5846  7                            .STACK_POINTER)
;  1305          5847  7                    THEN SELECTONE .PAT_SEM_TABLE [.ACTION_TO_TAKE] OF
;  1306          5848  7                        SET
;  1307          5849  7                        [PATGETHLP]:    GET_HELP_TOPIC;
;  1308          5850  7                        [PATDEFNAM]:    SAVE_NAME (.STACK_POINTER);
;  1309          5851  7                        [PATGETFIL]:    GET_FILE_SPEC;
;  1310          5852  7                        [PATOLDPMT]:    IF NOT PAT$PROMPT_READ(OLDPMT_TOKEN, .PARSE_
;  1311          5853  7                        [PATNEWPMT]:    IF NOT PAT$PROMPT_READ(NEWPMT_TOKEN, .PARSE_
;  1312          5854  7                        [PATLOCPMT]:    IF NOT PAT$PROMPT_READ(LOCPMT_TOKEN, .PARSE_
;  1313          5855  7                        [PATNAMPMT]:    IF NOT PAT$PROMPT_READ(NAMPMT_TOKEN, .PARSE_
;  1314          5856  7                        [PATEXPPMT]:    IF NOT PAT$PROMPT_READ(EXPPMT_TOKEN, .PARSE_
;  1315          5857  7                        [PATECOPMT]:    IF NOT PAT$PROMPT_READ(ECOPMT_TOKEN, .PARSE_
;  1316          5858  7                        [PATQUOTEC]:    GET_QUOTED_STG (.STACK_POINTER);
;  1317          5859  7                        [PATSAVMDL]:    ADD_MODULE (.STACK_POINTER);
;  1318          5860  7                        [PATSAVPAT]:    PAT$BUILD_PATH (LEX_STG_DESC, 0, TRUE);
;  1319          5861  7                        [PATTRNNAM]:    REDUCE_PATHNAME (.STACK_POINTER);
;  1320          5862  7                        [PATUQUNAM]:    TRANSLATE_NAME (.STACK_POINTER);
;  1321          5863  7                        [OTHERWISE]:    RETURN;
;  1322          5864  7                        TES;
;  1323          5865  7
;  1324          5866  6                    END;
;  1325          5867  6                END
```

```
;  1326     5868  6
;  1327     5869  6                    !++
;  1328     5870  6                    ! This is the ERROR_CODE processing.
;  1329     5871  6                    ! It is the else portion of the (.ACTION_TO_TAKE
;  1330     5872  6                    ! EQL ERROR_CODE) IF statement.
;  1331     5873  6                    !--
;  1332     5874  5                    ELSE
;  1333     5875  6                            BEGIN
;  1334     5876  7                            IF (.TOKEN NEQ DIGIT_STR_TOKEN)
;  1335     5877  6                            THEN
;  1336     5878  7                                    BEGIN
;  1337     5879  7                                    !++
;  1338     5880  7                                    ! Truncate the string to 10 characters
;  1339     5881  7                                    ! unless it is already shorter than 10.
;  1340     5882  7                                    !--
;  1341     5883  8                                    IF (.LAST_STG_DESC[DSC$W_LENGTH] GTR 10)
;  1342     5884  7                                    THEN
;  1343     5885  7                                            LAST_STG_DESC[DSC$W_LENGTH] = 10;
;  1344     5886  7                                    SIGNAL(PAT$_SYNTAX+MSG$K_WARN, 1, LAST_STG_DESC);
;  1345     5887  6                                    END;
;  1346     5888  7                            IF (.TOKEN NEQ EOL_TOKEN)
;  1347     5889  6                            THEN
;  1348     5890  6                                    SIGNAL(PAT$_INVCMD)
;  1349     5891  6                            ELSE
;  1350     5892  6                                    SIGNAL (PAT$_INVTOKEN+MSG$K_WARN, 1, LEX_STG_DESC);
;  1351     5893  6                            END
;  1352     5894  5                    END
;  1353     5895  4            ELSE
;  1354     5896  4                            !++
;  1355     5897  4                            ! This is the ELSE portion of the IF statement
;  1356     5898  4                            ! (.TRANSIT_CODE EQL .TOKEN) OR
;  1357     5899  4                            ! (.TRANSIT_CODE EQL ELSE_CODE)
;  1358     5900  4                            !--
;  1359     5901  5                            IF (.TRANSIT_CODE EQL CONT_CODE)
;  1360     5902  4                            THEN J = .PAT_ACT_TABLE [.J]
;  1361     5903  4                            ELSE J = .J + 1;
;  1362     5904  4                    END                                     ! End of MATCH_LOOP
;  1363     5905  3            UNTIL .MATCH_TRANSIT
;  1364     5906  3            END
;  1365     5907  3            !++
;  1366     5908  3            ! Continue processing until an action routine sets this
;  1367     5909  3            ! flag to false.
;  1368     5910  3            !--
;  1369     5911  2 WHILE .PARSE_MORE;
;  1370     5912  1 END;


                                                    .TITLE  PATPAR
                                                    .IDENT  \V04-000\

                                                    .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

0006 0005 0004 006E 006C 0002 0001 0068 0067 0066 00000 P.AAA:  .WORD   102, 103, 104, 1, 2, 108, 110, 4, 5, 6, -
000F 000E 0079 000A 000C 000D 0005 0009 0008 0007 00014                 7, 8, 9, 11, 13, 12, 10, 121, 14, 15, 16, -
270D 0051 0063 270C 0067 0064 270D 0003 0011 0010 00028                 17, 3, 9997, 100, 103, 9996, 99, 81, -
006A 270D 002C 002E 001F 0031 0015 0069 270D 0052 0003C                 9997, 82, 9997, 105, 21, 49, 31, 46, 44, -
0021 002D 270D 0047 008B 270D 0063 0047 008D 008B 00050                 9997, 106, 139, 141, 71, 99, 9997, 139, -
```

```
0047 0063 009A 008D 0098 006B 0052 270D 002F 0022 00064          71, 9997, 45, 33, 34, 47, 9997, 82, 107, -  ;
270C 009A 0099 270C 009A 270D 0041 270D 0012 270D 00078         152, 141, 154, 99, 71, 9997, 18, 9997, -  ;
007B 009E 006D 270C 008D 270C 008D 270C 009A 000A 0008C         65, 9997, 154, 9996, 153, 154, 9996, 10, -  ;
270D 0040 270C 0096 270D 0041 270D 0048 0063 0096 000A0         154, 9996, 141, 9996, 141, 9996, 109, -  ;
0040 007B 270C 0096 000A 270C 0096 009F 270C 0096 000B4         158, 123, 150, 99, 72, 9997, 65, 9997, -  ;
0096 270C 0040 007B 270C 007B 270C 0096 270D 0063 000C8         150, 9996, 64, 9997, 150, 9996, 159, 150, -  ;
006F 270C 006D 270C 0096 270D 0040 270C 007B 270C 000DC         9996, 10, 150, 9996, 123, 64, 99, 9997, -  ;
0042 270C 008B 270D 0041 270C 0063 008B 008D 008A 000F0         150, 9996, 123, 9996, 123, 64, 9996, 150, -  ;
003D 0029 0046 004C 0093 0092 0053 0091 008C 270D 00104         9996, 123, 9996, 64, 9997, 150, 9996, -  ;
004C 270D 0047 004D 003E 004B 0048 007F 0044 0094 00118         109, 9996, 111, 138, 141, 139, 99, 9996, -  ;
0092 270C 0092 270D 0013 002B 003D 0052 003C 0046 0012C         65, 9997, 139, 9996, 66, 9997, 140, 145, -  ;
270D 0044 0095 270C 0092 270C 0092 270C 0092 270C 00140         83, 146, 147, 76, 70, 41, 61, 148, 68, -  ;
008C 270D 004F 270C 0096 270D 0040 0097 270C 0096 00154         127, 72, 75, 62, 77, 71, 9997, 76, 70, -  ;
0092 270C 0092 270D 0047 270D 003E 270D 004F 270C 00168         60, 82, 61, 43, 19, 9997, 146, 9996, 146, -  ;
0090 270C 0092 270C 0092 270C 0092 270C 0092 270C 0017C         9996, 146, 9996, 146, 9996, 146, 9996, -  ;
270C 008C 270D 0063 0080 270C 008B 000A 270C 008B 00190         149, 68, 9997, 150, 9996, 151, 64, 9997, -  ;
270D 0042 270C 008D 270C 008C 270C 0080 270C 008D 001A4         150, 9996, 79, 9997, 140, 9996, 79, 9997, -  ;
270C 0063 008C 0088 007A 0052 0071 0070 270C 008C 001B8         62, 9997, 71, 9997, 146, 9996, 146, 9996, -  ;
002A 001B 009D 270C 0063 008C 0088 007A 0052 0071 001CC         146, 9996, 146, 9996, 146, 9996, 146, -  ;
0028 0030 0024 0014 0026 001C 001F 0031 0015 0016 001E0         9996, 144, 139, 9996, 10, 139, 9996, 128, -  ;
0082 270D 0063 0085 0042 270D 0025 0019 0027 002F 001F4         99, 9997, 140, 9996, 141, 9996, 128, -  ;
0089 0086 270C 008C 0089 270D 0041 270C 008C 0089 00208         9996, 140, 9996, 141, 9996, 66, 9997, -  ;
270C 0085 270C 0085 270C 008C 0089 000A 270C 008C 0021C         140, 9996, 112, 113, 82, 122, 136, 140, -  ;
0089 000A 270C 0089 0086 270C 0085 270C 008C 007A 00230         99, 9996, 113, 82, 122, 136, 140, 99, -  ;
0073 270C 0088 007A 0052 0073 0072 270C 009D 270C 00244         9996, 157, 27, 42, 22, 21, 49, 31, 28, -  ;
0080 0042 270C 002D 009D 00A0 270C 0088 007A 0052 00258         38, 20, 36, 48, 40, 47, 39, 25, 37, 9997, -  ;
0089 000A 270C 0089 0084 270D 0041 270C 0082 270C 0026C         66, 133, 99, 9997, 130, 137, 140, 9996, -  ;
0084 270C 0080 270C 007A 270C 0080 270C 0080 270C 00280         65, 9997, 137, 140, 9996, 134, 137, 140, -  ;
0052 0075 0074 270C 00A0 270C 0089 000A 270C 0089 00294         9996, 10, 137, 140, 9996, 133, 9996, 133, -  ;
0089 008F 0082 0052 0075 270C 0063 0089 008F 0082 002A8         9996, 122, 140, 9996, 133, 9996, 134, -  ;
0089 008E 270D 0041 270C 001E 009D 00A1 270C 0063 002BC         137, 9996, 10, 137, 9996, 157, 9996, 114, -  ;
00A1 270C 008F 270D 0063 008F 270C 0089 000A 270C 002D0         115, 82, 122, 136, 9996, 115, 82, 122, -  ;
0053 0091 008C 007A 0052 0076 0070 270C 001E 009D 002E4         136, 9996, 160, 157, 45, 9996, 66, 128, -  ;
0048 007F 0044 0094 003D 0029 0046 004C 0093 0092 002F8         9996, 130, 9996, 65, 9997, 132, 137, -  ;
0091 008C 007A 0052 0076 270D 0047 004D 003E 004B 0030C         9996, 10, 137, 9996, 128, 9996, 128, -  ;
007F 0044 0094 003D 0029 0046 004C 0093 0092 00320         9996, 122, 9996, 128, 9996, 132, 137, -  ;
008C 007A 270D 0041 270D 0047 004D 003E 004B 00334         9996, 10, 137, 9996, 160, 9996, 116, 117, -  ;
0041 270C 008C 270D 0040 270C 008C 270D 0040 270C 00348         82, 130, 143, 137, 99, 9996, 117, 82, -  ;
007A 0052 0077 270C 0088 007A 0052 0077 0070 270D 0035C         130, 143, 137, 99, 9996, 161, 157, 30, -  ;
0080 0041 0083 270C 0082 270C 0085 0042 270C 0088 00370         9996, 65, 9997, 142, 137, 9996, 10, 137, -  ;
000A 0089 0087 270C 0086 270C 000A 270C 0084 270C 00384         9996, 143, 99, 9997, 143, 9996, 161, 157, -  ;
0085 270C 007A 270C 000A 270C 0084 270C 0080 270C 00398         30, 9996, 112, 118, 82, 122, 140, 145, -  ;
0084 270C 0080 270C 000A 0089 0087 270C 0086 270C 003AC         83, 146, 147, 76, 70, 41, 61, 148, 68, -  ;
0078 270C 0088 007A 0052 0078 0070 270C 000A 270C 003C0         127, 72, 75, 62, 77, 71, 9997, 118, 82, -  ;
0083 270C 0089 270C 0085 0042 270C 0088 007A 0052 003D4         122, 140, 145, 83, 146, 147, 76, 70, 41, -  ;
0084 270C 0083 270C 0089 270C 000A 270C 0084 270C 003E8         61, 148, 68, 127, 72, 75, 62, 77, 71, -  ;
0083 270C 0089 270C 0085 270C 007A 270C 000A 270C 003FC         9997, 65, 9997, 122, 140, 9996, 64, 9997, -  ;
270C 0063 008C 007B 007A 270C 000A 270C 0084 270C 00410         140, 9996, 64, 9997, 140, 9996, 65, 9997, -  ;
009B 007C 270D 0017 002F 002D 0022 0021 270C 007A 00424         112, 119, 82, 122, 136, 9996, 119, 82, -  ;
009D 009C 270C 009D 270D 0041 270C 0063 009D 0080 00438         122, 136, 9996, 66, 133, 9996, 130, 9996, -  ;
006B 0052 270C 0080 270C 0080 270C 009D 000A 270C 0044C         131, 65, 128, 9996, 132, 9996, 10, 9996, -  ;
270C 0063 008C 0080 007A 0052 007D 270D 0012 270C 00460         134, 9996, 135, 137, 10, 9996, 128, 9996, -  ;
270C 0063 008C 0080 007A 007E 270D 0042 270D 0032 00474         132, 9996, 10, 9996, 122, 9996, 133, -  ;
270C 007A 270C 007A 270C 0063 008C 0080 007A 007D 00488         9996, 134, 9996, 135, 137, 10, 9996, 128, -  ;
270D 003E 270C 007F 270D 003E 270C 0047 0080 007F 0049C         9996, 132, 9996, 10, 9996, 112, 120, 82, -  ;
0088 007A 0052 0081 0070 270D 002F 002D 0022 0021 004B0         122, 136, 9996, 120, 82, 122, 136, 9996, -  ;
0082 270C 0085 0042 270C 0088 007A 0052 0081 270C 004C4         66, 133, 9996, 137, 9996, 131, 9996, 132, -  ;
```

```
0085  270C  007A  270C  0087  270C  0086  270D  0041  270C  004D8
270D  0017  270D  0029  0017  270C  0087  270C  0086  270C  004EC
                                                  0000  00500

00DE  0077  FFF2  0075  004D  0030  001E  001B  FFFE  0018  00502 P.AAB:   .WORD
0258  0214  020D  FFE1  01E3  01AF  FFE6  0175  0151  0125  00516
270F  FFFB  FFFC  0002  FFFD  FFFF  270F  027B  025D  FFD3  0052A
FFFA  270F  FF48  FF49  FF4A  FF4B  FF4C  0027  270F  0020  0053E
FFF8  FFF9  270F  FFC2  FFC3  270F  FFAA  FFC2  002D  FFC4  00552
FF85  FFAA  FF88  0043  003F  FFF6  003D  270F  FFF5  0035  00566
003B  004B  0046  003B  FF89  D87D  0041  270F  FFF7  270F  0057A
005C  0054  FFF4  002B  FF86  002B  FF87  003B  0049  FF8A  0058E
D85B  005A  0052  0058  D85D  0056  270F  FF61  FFA8  0071  005A2
0066  FF65  0052  0062  FF6A  0052  006A  005F  0052  FF67  005B6
006F  0064  006D  FF64  0064  FF63  0052  0068  270F  FFA8  005CA
FFF1  004E  FFF3  0052  FF66  D85A  0073  0064  FF62  0052  005DE
0083  002E  0081  D8AB  007F  002E  FFAA  00DA  00C7  007D  005F2
00A5  00A3  00A1  009F  FF95  FF9D  FF9E  0095  FFC6  270F  00606
009D  270F  FF5A  FF5C  FF5D  FF5E  FF5F  00B7  00B3  00A7  0061A
FF94  0087  FF9C  D891  00C5  00C3  00C1  00BF  00BD  00BB  0062E
D882  FF8D  00AA  0087  FF91  0087  FF92  0087  FF93  0087  00642
00B5  270F  FF8F  0052  00B1  270F  FF8C  00AF  0052  00AC  00656
FF9A  0087  FF9B  270F  FF5B  D852  00B9  270F  FF8E  0084  0066A
00CA  0087  FF96  0087  FF97  0087  FF98  0087  FF99  0087  0067E
0084  00D2  270F  FFAC  00D0  002E  00CD  FFB8  002E  00D4  00692
270F  00DC  002B  FFB6  0084  00D8  00CE  00D6  002B  FFB7  006A6
0084  FFAD  FFA0  0118  00FF  0123  FFEF  00E6  0084  FFC5  006BA
FF7D  FF7E  FF59  0084  FFAD  FFA0  0118  00FF  00ED  FFF0  006CE
FF73  FF74  FF75  FF76  FF77  FF78  FF79  FF7A  FF7B  FF7C  006E2
0107  270F  FFAB  010C  0103  270F  FF6F  FF70  FF71  FF72  006F6
0116  0110  0084  FFA5  FFC1  D8A4  0109  0084  FFA5  FFC0  0070A
0101  FFCA  0101  FFCB  0084  FFA5  0114  FFB1  0084  FFA5  0071E
0114  FFB0  0105  0116  0120  0101  011D  0084  FFA0  011B  00732
FFEE  00E3  0145  0134  014F  FFED  012B  00EE  FF58  0105  00746
013B  0137  00EE  FF51  FF52  FF54  00E3  0145  0134  0130  0075A
0141  FFB4  0105  0143  013E  D8A7  0109  0104  0139  00CE  0076E
014C  00CE  0149  0119  0147  00CE  FFCC  00CE  FFCD  0105  00782
0171  FFEB  0159  0131  FF53  0105  0141  FFB3  0105  0143  00796
FFC0  0166  0164  0160  FFEC  0105  FFA9  FFC0  0166  0164  007AA
016F  0169  D8A1  0109  00EE  FF4D  FF4E  FF50  0105  FFA9  007BE
FF4F  016D  FFBE  270F  FFA9  FFBF  0105  016C  FFAE  0105  007D2
FF9E  0095  01A9  FFBC  0123  01AD  018B  00EE  FF4D  FF4E  007E6
FF5F  00B7  00B3  00A7  00A5  00A3  00A1  009F  FF95  FF9D  007FA

                                                  9996, 10, 9996, 137, 9996, 131, 9996, -
                                                  132, 9996, 10, 9996, 122, 9996, 133, -
                                                  9996, 137, 9996, 131, 9996, 132, 9996, -
                                                  10, 9996, 122, 123, 140, 99, 9996, 122, -
                                                  9996, 33, 34, 45, 47, 23, 9997, 124, 155, -
                                                  128, 157, 99, 9996, 65, 9997, 157, 9996, -
                                                  156, 157, 9996, 10, 157, 9996, 128, 9996, -
                                                  128, 9996, 82, 107, 9996, 18, 9997, 125, -
                                                  82, 122, 128, 140, 99, 9996, 50, 9997, -
                                                  66, 9997, 126, 122, 128, 140, 99, 9996, -
                                                  125, 122, 128, 140, 99, 9996, 122, 9996, -
                                                  122, 9996, 127, 128, 71, 9996, 62, 9997, -
                                                  127, 9996, 62, 9997, 33, 34, 45, 47, -
                                                  9997, 112, 129, 82, 122, 136, 9996, 129, -
                                                  82, 122, 136, 9996, 66, 133, 9996, 130, -
                                                  9996, 65, 9997, 134, 9996, 135, 9996, -
                                                  122, 9996, 133, 9996, 134, 9996, 135, -
                                                  9996, 23, 41, 9997, 23, 9997, 0
                                                  24, -2, 27, 30, 48, 77, 117, -14, 119, -
                                                  222, 293, 337, 373, -26, 431, 483, -31, -
                                                  525, 532, 600, -45, 605, 635, 9999, -1, -
                                                  -3, 2, -4, -5, 9999, 32, 9999, 39, -180, -
                                                  -181, -182, -183, -184, 9999, -6, -60, -
                                                  45, -62, -86, 9999, -61, -62, 9999, -7, -
                                                  -8, 53, -11, 9999, 61, -10, 63, 67, -120, -
                                                  -86, -123, 9999, -9, 9999, 65, -10115, -
                                                  -119, 59, 70, 75, 59, -118, 73, 59, -121, -
                                                  43, -122, 43, -12, 84, 92, 113, -88, -
                                                  -159, 9999, 86, -10147, 88, 82, 90, -
                                                  -10149, -153, 82, 95, 106, 82, -150, 98, -
                                                  82, -155, 102, -88, 9999, 104, 82, -157, -
                                                  100, -156, 109, 100, 111, 82, -158, 100, -
                                                  115, -10150, -154, 82, -13, 78, -15, 125, -
                                                  199, 218, -86, 46, 127, -10069, 129, 46, -
                                                  131, 9999, -58, 149, -98, -99, -107, 159, -
                                                  161, 163, 165, 167, 179, 183, -161, -162, -
                                                  -163, -164, -166, 9999, 157, 187, 189, -
                                                  191, 193, 195, 197, -10095, -100, 135, -
                                                  -108, 135, -109, 135, -110, 135, -111, -
                                                  135, 170, -115, -10110, 172, 82, 175, -
                                                  -116, 9999, 177, 82, -113, 9999, 181, -
                                                  132, -114, 9999, 185, -10158, -165, 9999, -
                                                  -101, 135, -102, 135, -103, 135, -104, -
                                                  135, -105, 135, -106, 135, 202, 212, 46, -
                                                  -72, 205, 46, 208, -84, 9999, 210, 132, -
                                                  -73, 43, 214, 206, 216, 132, -74, 43, -
                                                  220, 9999, -59, 132, 230, -17, 291, 255, -
                                                  280, -96, -83, 132, -16, 237, 255, 280, -
                                                  -96, -83, 132, -167, -130, -131, -132, -
                                                  -133, -134, -135, -136, -137, -138, -139, -
                                                  -140, -141, -142, -143, -144, -145, 9999, -
                                                  259, 268, -85, 9999, 263, -64, -91, 132, -
                                                  265, -10076, -63, -91, 132, 272, 278, -
                                                  -91, 132, -79, 276, -91, 132, -53, 257, -
                                                  -54, 257, 283, -96, 132, 285, 257, 288, -
                                                  278, 261, -80, 276, 261, -168, 238, 299, -
                                                  -19, 335, 308, 325, 227, -18, 304, 308, -
```

```
0095  01A9  FFBC  00ED  01A0  D8D9  FF5A  FF5C  FF5D  FF5E  0080E
00B7  00B3  00A7  00A5  00A3  00A1  009F  FF95  FF9D  FF9E  00822
01A5  FFBD  D8DC  01A2  D8DB  FF5A  FF5C  FF5D  FF5E  FF5F  00836
01A2  0084  FFBA  D892  01AB  0084  FFBB  D892  01A7  0084  0084A
01BA  00ED  FFE5  00E3  01D3  01BA  0123  FFE4  01B5  D8DA  0085E
FFA7  0109  01C3  0104  01BF  0101  01C7  01BD  00E3  01D3  00872
FFA6  0114  01CD  010D  01C9  013F  FFD0  013C  01C5  00CE  00886
01D7  0119  01D5  013F  FFCF  013C  01D1  00CE  01CF  0105  0089A
01E1  00CE  01DF  0105  FFA6  0114  01DD  010D  01D9  0101  008AE
FFE3  00E3  0201  01EE  0123  FFE2  01E9  013F  FFCE  013C  008C2
01F5  0105  01F3  0101  01F9  01F1  00E3  0201  01EE  00ED  008D6
01FF  01C1  01FD  0105  01FB  013F  FFC9  013C  01F7  01C1  008EA
0209  0105  0207  0101  0205  0119  0203  013F  FFC8  013C  008FE
0084  FFA8  FFA0  0212  FFE0  013F  FFC7  013C  020B  01C1  00912
0220  FFDE  270F  FF6E  024E  0233  022E  021A  0119  FFDF  00926
022C  0227  00EE  FF80  DB76  0222  00EE  FFAC  FF7F  0224  0093A
FFDC  0231  00CE  FF81  00CE  FF82  00EE  022A  FF83  00EE  0094E
0084  FFAC  FFA0  024A  FFA4  023A  FFDB  270F  FFDD  0037  00962
0084  FFAC  FFA0  024C  FFA2  0244  270F  023E  270F  023C  00976
0119  FFA1  0119  FFA3  0084  FFAC  FFA0  024A  FFA4  FFDA  0098A
D8CA  00B9  0093  0256  D8CB  00B9  00CE  FF5A  0254  0252  0099E
0273  0268  0123  FFD1  0263  270F  FFD4  FFD5  FFD6  FFD7  009B2
026D  0101  026F  026B  00E3  0273  0268  00ED  FFD2  00E3  009C6
0277  0119  0275  01CA  FF56  010D  0271  D849  0109  0104  009DA
270F  FF6C  270F  027E  FF6D  01CA  FF55  010D  0279  0101  009EE
                                                  0000  00A02

00  01  01  01  02  03  01  01  03  01  01  01  00  01  00  00A04 P.AAC:  .BYTE
02  01  02  00  00  01  01  02  01  02  01  02  01  02  01  00A13
01  01  01  01  03  02  06  02  02  03  02  02  01  00  01  00A22
02  04  06  05  05  01  02  01  02  07  06  05  01  02  00  00A31
03  04  02  00  02  04  00  02  01  02  00  02  00  01  00  00A40
00  00  00  00  00  00  00  02  00  04  03  02  04  03  02  00A4F
02  02  02  02  02  00  00  00  00  01  00  01  00  00  00  00A5E
02  02  00  00  00  02  05  00  01  01  01  01  00  02  02  00A6D
00  00  00  00  00  00  02  01  02  02  00  00  01  02  00  00A7C
00  02  01  01  00  00  00  00  00  00  00  00  00  00  00  00A8B
00  00  00  00  00  00  03  04  01  02  02  04  00  02  02  00A9A
00  00  01  02  00  00  01  02  04  03  02  01  02  00  02  00AA9
                                      00  00  00  00  00  00AB8

325, 227, -172, -174, -175, 238, 311, -
315, 206, 313, 260, 265, -10073, 318, -
323, 261, -76, 321, 261, -51, 206, -52, -
206, 327, 281, 329, 206, 332, 323, 261, -
-77, 321, 261, -173, 305, 345, -21, 369, -
356, 358, -64, -87, 261, -20, 352, 356, -
358, -64, -87, 261, -176, -178, -179, -
238, 265, -10079, 361, 367, 261, -82, -
364, 261, -65, -87, 9999, -66, 365, -177, -
-178, -179, 238, 395, 429, 291, -68, 425, -
149, -98, -99, -107, 159, 161, 163, 165, -
167, 179, 183, -161, -162, -163, -164, -
-166, -10023, 416, 237, -68, 425, 149, -
-98, -99, -107, 159, 161, 163, 165, 167, -
179, 183, -161, -162, -163, -164, -166, -
-10021, 418, -10020, -67, 421, 132, 423, -
-10094, -69, 132, 427, -10094, -70, 132, -
418, -10022, 437, -28, 291, 442, 467, -
227, -27, 237, 442, 467, 227, 445, 455, -
257, 447, 260, 451, 265, -89, 206, 453, -
316, -48, 319, 457, 269, 461, 276, -90, -
261, 463, 206, 465, 316, -49, 319, 469, -
281, 471, 257, 473, 269, 477, 276, -90, -
261, 479, 206, 481, 316, -50, 319, 489, -
-30, 291, 494, 513, 227, -29, 237, 494, -
513, 227, 497, 505, 257, 499, 261, 501, -
449, 503, 316, -55, 319, 507, 261, 509, -
449, 511, 316, -56, 319, 515, 281, 517, -
257, 519, 261, 521, 449, 523, 316, -57, -
319, -32, 530, -96, -88, 132, -33, 281, -
538, 558, 563, 590, -146, 9999, -34, 544, -
548, -129, -84, 238, 546, -10122, -128, -
238, 551, 556, 238, -125, 554, 238, -126, -
206, -127, 206, 561, -36, 55, -35, 9999, -
-37, 570, -92, 586, -96, -84, 132, 572, -
9999, 574, 9999, 580, -94, 588, -96, -84, -
132, -38, -92, 586, -96, -84, 132, -93, -
281, -95, 281, 594, 596, -166, 206, 185, -
-10037, 598, 147, 185, -10038, -41, -42, -
-43, -44, 9999, 611, -47, 291, 616, 627, -
227, -46, 237, 616, 627, 227, 619, 623, -
257, 621, 260, 265, -10167, 625, 269, -
-170, 458, 629, 281, 631, 257, 633, 269, -
-171, 458, -147, 638, 9999, -148, 9999, 0

0, 1, 0, 1, 1, 1, 3, 1, 1, 3, 2, 1, 1, 1, -
0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 0, 0, 2, -
1, 2, 1, 0, 1, 2, 2, 3, 2, 2, 6, 2, 3, 1, -
1, 1, 1, 0, 2, 1, 5, 6, 7, 2, 1, 2, 1, 5, -
5, 6, 4, 2, 0, 1, 0, 2, 0, 2, 1, 2, 0, 4, -
2, 0, 2, 4, 3, 2, 3, 4, 2, 3, 4, 0, 2, 0, -
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, -
0, 0, 2, 2, 2, 2, 2, 2, 2, 0, 1, 1, 1, 1, -
0, 5, 2, 0, 0, 0, 2, 2, 0, 2, 1, 0, 0, 2, -
2, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -
0, 0, 0, 0, 0, 0, 1, 1, 2, 0, 2, 2, 0, 4, -
2, 2, 1, 4, 3, 0, 0, 0, 0, 0, 0, 2, 0, 2, -
1, 2, 3, 4, 2, 1, 0, 0, 2, 1, 0, 0, 0, 0, -
```

```
68 68 68 68 68 68 68 68 68 67 67 66 66 65 00 00ABD P.AAD:  .BYTE   0, 0, 0
68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 00ACC                0, 101, 102, 102, 103, 103, 104, 104, -
68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 00ADB                104, 104, 104, 104, 104, 104, 104, 104, -
8A 8A 78 78 78 86 86 84 84 77 77 77 68 68 68 00AEA                104, 104, 104, 104, 104, 104, 104, 104, -
90 90 6F 6F 76 76 76 76 8E 8E 82 82 8B 6A 6A 00AF9                104, 104, 104, 104, 104, 104, 104, 104, -
83 7B 8F 8D 85 80 88 75 75 71 71 71 73 73 73 00B08                104, 104, 104, 104, 104, 104, 104, 104, -
91 91 91 91 91 91 8C 8C 7A 7E 7E 7D 7D 89 87 00B17                119, 119, 119, -124, -124, -122, -122, -
98 6B 6B 97 95 93 93 93 92 92 92 92 92 91 91 00B26                120, 120, 120, -118, -118, 106, 106, -
9D 9D 9D 9D 9D 9B 9B 9C 9C 7C 7C 9A 99 99 98 00B35                -117, -126, -126, -114, -114, 118, 118, -
6D 6E 6C 79 9D 9D 9D 9D 9D 9D 9D 9D 9D 9D 9D 00B44                118, 118, 111, 111, -112, -112, 115, 115, -
94 94 94 94 94 96 9F 9F 9F 9F 9E 9E 9E 9E 6D 00B53                115, 113, 113, 113, 117, 117, -120, -128, -
A1 A1 74 74 A0 A0 72 72 81 81 81 70 70 7F 7F 00B62                -123, -115, -113, 123, -125, -121, -119, -
                                  00 69 69 69 69 69 00B71         125, 125, 126, 126, 122, -116, -116, -
                                                                  -111, -111, -111, -111, -111, -111, -111, -
                                                                  -111, -110, -110, -110, -110, -110, -109, -
                                                                  -109, -109, -107, -105, 107, 107, -104, -
                                                                  -104, -103, -103, -102, 124, 124, -100, -
                                                                  -100, -101, -101, -99, -99, -99, -99, -
                                                                  -99, -99, -99, -99, -99, -99, -99, -99, -
                                                                  -99, -99, -99, -99, 121, 108, 110, 109, -
                                                                  109, -98, -98, -98, -98, -97, -97, -97, -
                                                                  -97, -106, -108, -108, -108, -108, -108, -
                                                                  127, 127, 112, 112, -127, -127, -127, -
                                                                  114, 114, -96, -96, 116, 116, -95, -95, -
                                                                  105, 105, 105, 105, 105, 0
0E 3C 3C 2B 1E 1E 20 2C 3C 10 10 3C 3C 3C 3C 00B77 P.AAE:  .BYTE   60, 60, 60, 60, 16, 16, 60, 44, 32, 30, -
3C 3C 3C 14 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 00B86                30, 43, 60, 60, 14, 60, 60, 60, 60, 60, -
2B 2C 1E 20 2B 2B 3C 3C 1E 1E 20 3C 3C 3C 3C 00B95                60, 60, 60, 60, 60, 60, 20, 60, 60, 60, -
0F 1D 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 00BA4                60, 60, 60, 60, 32, 30, 30, 60, 60, 43, -
0F 0A 3C 3C 01 03 3C 3C 3C 3C 3C 3C 06 05 04 00BB3                43, 32, 30, 44, 43, 60, 60, 60, 60, 60, -
2E 1C 36 21 29 34 25 3C 3C 3C 3C 3C 3C 3C 3C 00BC2                60, 60, 60, 60, 60, 60, 60, 60, 29, 15, -
22 28 33 31 0B 3C 32 3C 38 1F 1F 2C 2C 35 2E 00BD1                4, 5, 6, 60, 60, 60, 60, 60, 60, 3, 1, -
3A 3C 3C 0D 0D 18 27 3C 1B 39 19 37 3C 13 30 00BE0                60, 60, 10, 15, 60, 60, 60, 60, 60, 60, -
3C 3C 3C 3C 3C 38 3B 38 2D 3C 3C 17 35 24 35 00BEF                60, 60, 37, 52, 41, 33, 54, 28, 46, 46, -
3C 08 0D 12 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 00BFE                53, 44, 44, 31, 31, 56, 60, 50, 60, 11, -
2A 32 32 32 26 32 01 02 38 2D 01 03 38 3B 3C 00C0D                49, 51, 40, 34, 48, 19, 60, 55, 25, 57, -
2F 3C 16 1A 2C 3C 16 1A 3C 3C 2E 16 1A 23 23 00C1C                27, 60, 39, 24, 13, 13, 60, 60, 58, 53, -
                                  09 07 0C 11 15 00C2B            36, 53, 23, 60, 60, 45, 56, 59, 56, 60, -
                                                                  60, 60, 60, 60, 60, 60, 60, 60, 60, 60, -
                                                                  60, 60, 60, 60, 60, 18, 13, 8, 60, 60, -
                                                                  59, 56, 3, 1, 45, 56, 2, 1, 50, 38, 50, -
                                                                  50, 50, 42, 35, 35, 26, 22, 46, 60, 60, -
                                                                  26, 22, 60, 44, 26, 22, 60, 47, 21, 17, -
                                                                  12, 7, 9

                                        .PSECT  _PAT$OWN,NOEXE,2

00000 QUOTE_INDIC:
                        .BLKB   4

      ISE$C_SIZE==          20
      TXT$C_SIZE==          4
      PAL$C_SIZE==          16
      ASD$C_SIZE==          9
      F  ($C_SIZE==         24
      PAT_STATE_TABLE==     P.AAA
```

```
                        PAT_ACT_TABLE==     P.AAB
                        PAT_POP_TABLE==     P.AAC
                        PAT_LHS_TABLE==     P.AAD
                        PAT_SEM_TABLE==     P.AAE
                                .EXTRN  PAT$FAO_OUT, PAT$ADD_ARG
                                .EXTRN  PAT$BUILD_PATH, PAT$FIND_MODULE
                                .EXTRN  PAT$FREEZ, PAT$GET_A_TOKEN
                                .EXTRN  PAT$INIT_MODES, PAT$PERFORM_CMD
                                .EXTRN  PAT$PROMPT_READ
                                .EXTRN  PAT$SET_COMQUAL
                                .EXTRN  PAT$SET_MOD_LVL
                                .EXTRN  PAT$SET_OVERS, PAT$TRANS_NAME
                                .EXTRN  PAT$GL_HELP_LIN
                                .EXTRN  PAT$GL_FLAGS, PAT$GB_MOD_PTR
                                .EXTRN  PAT$GL_COMQUAL, PAT$GL_CONTEXT
                                .EXTRN  PAT$GL_HEAD_LST
                                .EXTRN  PAT$GL_TAIL_LST
                                .EXTRN  PAT$GL_SEMAN1, PAT$GL_SEMAN2
                                .WEAK   ACCESS_CHECK

                                .PSECT  _PAT$CODE,NOWRT,2

                        0FFC 00000      .ENTRY  PAT$PARS_A_LINE, Save R2,R3,R4,R5,R6,R7,R8,-; 5583
                                                R9,R10,R11
                   5E    FF30   CE  9E 00002      MOVAB   -208(SP), SP                        ; 5653
                         1C     AE  7C 00007      CLRQ    REDUCE_WAS_LAST                     ; 5654
                                58  D4 0000A      CLRL    STACK_POINTER                       ; 5655
                00000000'       EF  D4 0000C      CLRL    QUOTE_INDIC                         ; 5656
      08        00        6E    00  2C 00012      MOVC5   #0, (SP), #0, #8, PAT$GL_HELP_LIN   ; 5660
                00000000G       EF     00017
                          BC    AD  B4 0001C      CLRW    LEX_STG_DESC                        ; 5661
                    C4    AD        19  B0 0001F      MOVW    #25, LEX_STG_DESC+8             ; 5662
                    C0    AD     E4 AD  9E 00023      MOVAB   LEXEME_ADDR, LEX_STG_DESC+4     ; 5663
                    B8    AD     C8 AD  9E 00028      MOVAB   LAHEAD_ADDR, LAHEAD_STG_DESC+4  ; 5669
                          5A     04 AC  D0 0002D      MOVL    PARSE_STG_DESC, R10             ; 5670
                    7C    AE        6A  B0 00031      MOVW    (R10), LAST_STG_DESC
                    18    AE     04 AA  9E 00035      MOVAB   4(R10), 24(SP)                  ; 5675
                    B0    AD     18 BE  D0 0003A      MOVL    @24(SP), LAST_STG_DESC+4
                          BC     AD 9F  00 0003F      PUSHAB  LEX_STG_DESC                    ; 5681
                                    5A  DD 00042      PUSHL   R10
                00000000G   EF        02  FB 00044      CALLS   #2, PAT$GET_A_TOKEN
                          5B        50  D0 0004B      MOVL    R0, TOKEN
                    10    AE        01  7D 0004E      MOVQ    #1, SCAN_NEXT_SYM               ; 5682
                    0C    AE        01  D0 00052      MOVL    #1, PARSE_MORE                  ; 5683
                             08     AE  D4 00056      CLRL    MATCH_TRANSIT                   ; 5691
                          59     14 AE  D0 00059  1$: MOVL    CUR_PARSE_STATE, J              ; 5699
                             08     AE  D4 0005D  2$: CLRL    MATCH_TRANSIT                   ; 5700
                    04    AE 00000000'EF49  3C 00060      MOVZWL  PAT_STATE_TABLE[J], TRANSIT_CODE ; 5712
                                    50  D4 00069      CLRL    R0
                0000270D    8F     04 AE  D1 0006B      CMPL    TRANSIT_CODE, #9997
                                    19  12 00073      BNEQ    3$
                                    50  D6 00075      INCL    R0
                          31        5B  D1 00077      CMPL    TOKEN, #49
                                    12  14 0007A      BGTR    3$
                    20    AE        5B  D0 0007C      MOVL    TOKEN, OLD_TOKEN                ; 5715
                          6E        59  D0 00080      MOVL    J, OLD_J                        ; 5716
                          59     14 AE  D0 00083      MOVL    CUR_PARSE_STATE, J              ; 5717
```

```
                    5B        47   8F  9A 00087        MOVZBL  #71, TOKEN                                  ; 5718
                                 0418  31 0008B        BRW     71$                                         ; 5719
                    18             50  E9 0008E 3$:    BLBC    R0, 4$                                      ; 5727
          00000047  8F             5B  D1 00091        CMPL    TOKEN, #71                                  ;
                                   0F  12 00098        BNEQ    4$                                          ;
                              20   AE  D5 0009A        TSTL    OLD_TOKEN                                   ; 5728
                                   0A  13 0009D        BEQL    4$                                          ;
                    5B        20   AE  D0 0009F        MOVL    OLD_TOKEN, TOKEN                            ; 5731
                    59             6E  D0 000A3        MOVL    OLD_J, J                                    ; 5732
                              20   AE  D4 000A6        CLRL    OLD_TOKEN                                   ; 5733
                    5B        04   AE  D1 000A9 4$:    CMPL    TRANSIT_CODE, TOKEN                         ; 5740
                                   06  13 000AD        BEQL    5$                                          ;
                    03             50  E8 000AF        BLBS    R0, 5$                                      ; 5741
                                 03DB  31 000B2        BRW     69$                                         ;
                08  AE             01  D0 000B5 5$:    MOVL    #1, MATCH_TRANSIT                           ; 5748
                    57 00000000'EF49  32 000B9        CVTWL   PAT_ACT_TABLE[J], ACTION_TO_TAKE            ; 5749
          0000270F  8F             57  D1 000C1        CMPL    ACTION_TO_TAKE, #9999                       ; 5750
                                   03  12 000C8        BNEQ    6$                                          ;
                                 0372  31 000CA        BRW     64$                                         ;
          FFFFD8F2  8F             57  D1 000CD 6$:    CMPL    ACTION_TO_TAKE, #-9998                      ; 5753
                                   12  15 000D4        BLEQ    7$                                          ;
             28 AE48        14   AE  B0 000D6        MOVW    CUR_PARSE_STATE, PARSE_STACK[STACK_POINTER] ; 5761
                    08        1C   AE  E8 000DC        BLBS    REDUCE_WAS_LAST, 7$                         ; 5766
          00000000GEF48            5B  D0 000E0        MOVL    TOKEN, PAT$GL_SEMAN1[STACK_POINTER]         ; 5767
                                   57  D5 000E8 7$:    TSTL    ACTION_TO_TAKE                              ; 5770
                                   64  19 000EA        BLSS    12$                                         ;
                14  AE             57  D0 000EC        MOVL    ACTION_TO_TAKE, CUR_PARSE_STATE             ; 5776
                              1C   AE  D4 000F0        CLRL    REDUCE_WAS_LAST                             ; 5777
                    28             58  D1 000F3        CMPL    STACK_POINTER, #40                          ; 5778
                                   0F  19 000F6        BLSS    8$                                          ;
                         006D8142  8F  DD 000F8        PUSHL   #7176514                                    ; 5781
          00000000G 00             01  FB 000FE        CALLS   #1, LIB$SIGNAL                              ;
                                   03  11 00105        BRB     9$                                          ; 5780
                    58             02  C0 00107 8$:    ADDL2   #2, STACK_POINTER                           ; 5783
                    37        10   AE  E9 0010A 9$:    BLBC    SCAN_NEXT_SYM, 10$                          ; 5789
                    50        BC   AD  3C 0010E        MOVZWL  LAHEAD_STG_DESC+8, R0                       ; 5793
                    50             04  C6 00112        DIVL2   #4, R0                                      ;
                    50             04  C4 00115        MULL2   #4, R0                                      ;
 50         00      6E             00  2C 00118        MOVC5   #0, (SP), #0, R0, @LAHEAD_STG_DESC+4        ;
                              B8   BD     0011D                                                           ;
                B4  AD        BC   AD  B0 0011F        MOVW    LAHEAD_STG_DESC+8, LAHEAD_STG_DESC          ; 5795
      B8   BD   C0  BD        BC   AD  28 00124        MOVC3   LEX_STG_DESC, @LEX_STG_DESC+4, -            ; 5798
                                                                @LAHEAD_STG_DESC+4                          ;
                7C  AE             6A  B0 0012B        MOVW    (R10), LAST_STG_DESC                        ; 5799
                B0  AD        18   BE  D0 0012F        MOVL    @24(SP), LAST_STG_DESC+4                    ; 5800
                              BC   AD  9F 00134        PUSHAB  LEX_STG_DESC                                ; 5801
                                   5A  DD 00137        PUSHL   R10                                         ;
          00000000G EF             02  FB 00139        CALLS   #2, PAT$GET_A_TOKEN                         ;
                    5B             50  D0 00140        MOVL    R0, TOKEN                                   ;
                                   08  11 00143        BRB     11$                                         ; 5789
                10  AE             01  D0 00145 10$:   MOVL    #1, SCAN_NEXT_SYM                           ; 5809
                    5B        78   AE  D0 00149        MOVL    NEXT_TOKEN, TOKEN                           ; 5810
                                 0356  31 0014D 11$:   BRW     71$                                         ; 5770
          FFFFD8F2  8F             57  D1 00150 12$:   CMPL    ACTION_TO_TAKE, #-9998                      ; 5822
                                   0F  14 00157        BGTR    13$                                         ;
                    57      270E   C7  9E 00159        MOVAB   9998(R7), ACTION_TO_TAKE                    ; 5829
                              10   AE  D4 0015E        CLRL    SCAN_NEXT_SYM                               ; 5830
```

```
                58          02  C2 00161          SUBL2   #2, STACK_POINTER                                  ; 5831
          78    AE          5B  D0 00164          MOVL    TOKEN, NEXT_TOKEN                                  ; 5832
                57          57  CE 00168 13$:     MNEGL   ACTION_TO_TAKE, ACTION_TO_TAKE                     ; 5835
                50 00000000'EF47  9A 0016B        MOVZBL  PAT_POP_TABLE[ACTION_TO_TAKE], R0                  ; 5837
                50          02  C4 00173          MULL2   #2, R0                                             ;
                58          50  C2 00176          SUBL2   R0, STACK_POINTER                                  ;
          14    AE    28 AE48  3C 00179          MOVZWL  PARSE_STACK[STACK_POINTER], CUR_PARSE_STATE        ; 5838
                5B 00000000'EF47  9A 0017F        MOVZBL  PAT_LHS_TABLE[ACTION_TO_TAKE], TOKEN               ; 5839
          1C    AE          01  D0 00187          MOVL    #1, REDUCE_WAS_LAST                                ; 5840
                            58  DD 0018B          PUSHL   STACK_POINTER                                      ; 5846
                7E 00000000'EF47  9A 0018D        MOVZBL  PAT_SEM_TABLE[ACTION_TO_TAKE], -(SP)               ; 5845
     00000000V  EF          02  FB 00195          CALLS   #2, MAR_REDUCTN                                    ;
                AE          50  E8 0019C          BLBS    R0, 11$                                            ;
                53 00000000'EF47  9A 0019F        MOVZBL  PAT_SEM_TABLE[ACTION_TO_TAKE], R3                  ; 5847
                14          53  91 001A7          CMPB    R3, #20                                            ; 5849
                            4C  12 001AA          BNEQ    20$                                                ;
                52    18    BE  D0 001AC          MOVL    @24(SP), HELP_INPUT_PTR                            ;
                51          62  9A 001B0          MOVZBL  (HELP_INPUT_PTR), HELP_CHAR                        ;
                            50  D4 001B3          CLRL    HELP_COUNT                                         ;
                20          51  D1 001B5 14$:     CMPL    HELP_CHAR, #32                                     ;
                            05  13 001B8          BEQL    15$                                                ;
                09          51  D1 001BA          CMPL    HELP_CHAR, #9                                      ;
                            09  12 001BD          BNEQ    16$                                                ;
                            50  D6 001BF 15$:     INCL    HELP_COUNT                                         ;
                            52  D6 001C1          INCL    HELP_INPUT_PTR                                     ;
                51          62  9A 001C3          MOVZBL  (HELP_INPUT_PTR), HELP_CHAR                        ;
                            ED  11 001C6          BRB     14$                                                ;
                            51  D5 001C8 16$:     TSTL    HELP_CHAR                                          ;
                            03  12 001CA          BNEQ    17$                                                ;
                          02DE  31 001CC          BRW     72$                                                ;
                6A          50  A2 001CF 17$:     SUBW2   HELP_COUNT, (R10)                                  ;
     00000000G  EF          52  D0 001D2          MOVL    HELP_INPUT_PTR, PAT$GL_HELP_LIN+4                  ;
                            50  D4 001D9          CLRL    HELP_COUNT                                         ;
                            51  D5 001DB 18$:     TSTL    HELP_CHAR                                          ;
                            09  13 001DD          BEQL    19$                                                ;
                            50  D6 001DF          INCL    HELP_COUNT                                         ;
                            52  D6 001E1          INCL    HELP_INPUT_PTR                                     ;
                51          62  9A 001E3          MOVZBL  (HELP_INPUT_PTR), HELP_CHAR                        ;
                            F3  11 001E6          BRB     18$                                                ;
     00000000G  EF          50  B0 001E8 19$:     MOVW    HELP_COUNT, PAT$GL_HELP_LIN                        ;
          18    BE          52  D0 001EF          MOVL    HELP_INPUT_PTR, @24(SP)                            ;
                6A          50  A2 001F3          SUBW2   HELP_COUNT, (R10)                                  ;
                            34  11 001F6          BRB     21$                                                ; 5847
                06          53  91 001F8 20$:     CMPB    R3, #6                                             ; 5850
                            32  12 001FB          BNEQ    22$                                                ;
                50    BC    AD  3C 001FD          MOVZWL  LEX_STG_DESC, R0                                   ;
                50          03  C0 00201          ADDL2   #3, R0                                             ;
                50          04  C6 00204          DIVL2   #4, R0                                             ;
                      02    A0  9F 00207          PUSHAB  2(R0)                                              ;
     00000000G  EF          01  FB 0020A          CALLS   #1, PAT$FREEZ                                      ;
                56          50  D0 00211          MOVL    R0, POINTER                                        ;
08    A6    C0  BD    BC    AD  28 00214          MOVC3   LEX_STG_DESC, @LEX_STG_DESC+4, 8(POINTER)          ;
                66    BC    AD  3C 0021B          MOVZWL  LEX_STG_DESC, (POINTER)                            ;
          04    A6    08    A6  9E 0021F          MOVAB   8(POINTER), 4(POINTER)                             ;
     00000000GEF48          56  D0 00224          MOVL    POINTER, PAT$GL_SEMAN2[STACK_POINTER]              ;
                          0277  31 0022C 21$:     BRW     71$                                                ; 5847
                0E          53  91 0022F 22$:     CMPB    R3, #14                                            ; 5851
```

```
                                  03  13 00232          BEQL    23$                                  ;
                                009A  31 00234          BRW     36$                                  ;
                  52        18    BE  D0 00237 23$:     MOVL    @24(SP), INPUT_PTR                   ;
                  55              62  9A 0023B          MOVZBL  (INPUT_PTR), CHAR                    ;
                                  54  D4 0023E          CLRL    COUNT                                ;
                  20              55  D1 00240 24$:     CMPL    CHAR, #32                            ;
                                  05  13 00243          BEQL    25$                                  ;
                  09              55  D1 00245          CMPL    CHAR, #9                             ;
                                  09  12 00248          BNEQ    26$                                  ;
                                  54  D6 0024A 25$:     INCL    COUNT                                ;
                                  52  D6 0024C          INCL    INPUT_PTR                            ;
                  55              62  9A 0024E          MOVZBL  (INPUT_PTR), CHAR                    ;
                                  ED  11 00251          BRB     24$                                  ;
                                  55  D5 00253 26$:     TSTL    CHAR                                 ;
                                  03  12 00255          BNEQ    27$                                  ;
                                0253  31 00257          BRW     72$                                  ;
                                  54  D5 0025A 27$:     TSTL    COUNT                                ;
                                  34  14 0025C          BGTR    30$                                  ;
        00000048  8F              5B  D1 0025E          CMPL    TOKEN, #72                           ;
                                  09  13 00265          BEQL    28$                                  ;
        00000063  8F              5B  D1 00267          CMPL    TOKEN, #99                           ;
                                  0F  12 0026E          BNEQ    29$                                  ;
                        006D80DA  8F  DD 00270 28$:     PUSHL   #7176410                             ;
        00000000G  00             01  FB 00276          CALLS   #1, LIB$SIGNAL                       ;
                                  13  11 0027D          BRB     30$                                  ;
                             BC   AD  9F 0027F 29$:     PUSHAB  LEX_STG_DESC                         ;
                                  01  DD 00282          PUSHL   #1                                   ;
                        006D82A8  8F  DD 00284          PUSHL   #7176872                             ;
        00000000G  00             03  FB 0028A          CALLS   #3, LIB$SIGNAL                       ;
                                      04 00291          RET                                          ;
                  6A              54  A2 00292 30$:     SUBW2   COUNT, (R10)                         ;
                  53        FF    A2  9E 00295          MOVAB   -1(R2), TEMP_PTR                     ;
                                  54  D4 00299          CLRL    COUNT                                ;
                                  55  D5 0029B 31$:     TSTL    CHAR                                 ;
                                  16  12 0029D          BNEQ    33$                                  ;
                                  54  D5 0029F          TSTL    COUNT                                ;
                                  0D  14 002A1          BGTR    32$                                  ;
                        006D80DA  8F  DD 002A3          PUSHL   #7176410                             ;
        00000000G  00             01  FB 002A9          CALLS   #1, LIB$SIGNAL                       ;
                  63              54  90 002B0 32$:     MOVB    COUNT, (TEMP_PTR)                    ;
                                  09  11 002B3          BRB     34$                                  ;
                                  54  D6 002B5 33$:     INCL    COUNT                                ;
                                  52  D6 002B7          INCL    INPUT_PTR                            ;
                  55              62  9A 002B9          MOVZBL  (INPUT_PTR), CHAR                    ;
                                  DD  11 002BC          BRB     31$                                  ;
                                  53  DD 002BE 34$:     PUSHL   TEMP_PTR                             ;
        00000000G  EF             01  FB 002C0          CALLS   #1, PAT$ADD_ARG                      ;
              18  BE              52  D0 002C7          MOVL    INPUT_PTR, @24(SP)                   ;
                  6A              54  A2 002CB          SUBW2   COUNT, (R10)                         ;
                                01D5  31 002CE 35$:     BRW     71$                                  ; 5847
                  29              53  91 002D1 36$:     CMPB    R3, #41                              ; 5852
                                  08  12 002D4          BNEQ    37$                                  ;
                                  5A  DD 002D6          PUSHL   R10                                  ;
                  7E        5B    8F  9A 002D8          MOVZBL  #91, -(SP)                           ;
                                  3F  11 002DC          BRB     42$                                  ;
                  34              53  91 002DE 37$:     CMPB    R3, #52                              ; 5853
                                  08  12 002E1          BNEQ    38$                                  ;
```

```
                                5A   DD 002E3          PUSHL   R10                                                                    ;
                      7E   5C   8F   9A 002E5          MOVZBL  #92, -(SP)                                                             ;
                                32   11 002E9          BRB     42$                                                                    ;
                      25        53   91 002EB 38$:     CMPB    R3, #37                                                                ; 5854
                                08   12 002EE          BNEQ    39$                                                                    ;
                                5A   DD 002F0          PUSHL   R10                                                                    ;
                      7E   5D   8F   9A 002F2          MOVZBL  #93, -(SP)                                                             ;
                                25   11 002F6          BRB     42$                                                                    ;
                      21        53   91 002F8 39$:     CMPB    R3, #33                                                                ; 5855
                                08   12 002FB          BNEQ    40$                                                                    ;
                                5A   DD 002FD          PUSHL   R10                                                                    ;
                      7E   5E   8F   9A 002FF          MOVZBL  #94, -(SP)                                                             ;
                                18   11 00303          BRB     42$                                                                    ;
                      36        53   91 00305 40$:     CMPB    R3, #54                                                                ; 5856
                                08   12 00308          BNEQ    41$                                                                    ;
                                5A   DD 0030A          PUSHL   R10                                                                    ;
                      7E   5F   8F   9A 0030C          MOVZBL  #95, -(SP)                                                             ;
                                0B   11 00310          BRB     42$                                                                    ;
                      1C        53   91 00312 41$:     CMPB    R3, #28                                                                ; 5857
                                11   12 00315          BNEQ    43$                                                                    ;
                                5A   DD 00317          PUSHL   R10                                                                    ;
                      7E   60   8F   9A 00319          MOVZBL  #96, -(SP)                                                             ;
00000000G             EF        02   FB 0031D 42$:     CALLS   #2, PAT$PROMPT_READ                                                    ;
                      A7        50   E8 00324          BLBS    R0, 35$                                                                ;
                                04   00327            RET                                                                            ;
                      2A        53   91 00328 43$:     CMPB    R3, #42                                                                ; 5858
                                03   13 0032B          BEQL    44$                                                                    ;
                              00B9   31 0032D          BRW     57$                                                                    ;
                   50 00000000G     EF   D0 00330 44$:     MOVL    PAT$GB_MOD_PTR, R0                                                     ;
                   11      03       A0   E8 00337          BLBS    3(R0), 45$                                                             ;
                   0D      04       A0   E8 0033B          BLBS    4(R0), 45$                                                             ;
                        006D8258    8F   DD 0033F          PUSHL   #7176792                                                               ;
00000000G          00               01   FB 00345          CALLS   #1, LIB$SIGNAL                                                         ;
00000000'          EF               01   D0 0034C 45$:     MOVL    #1, QUOTE_INDIC                                                        ;
53          18     BE               01   C3 00353          SUBL3   #1, @24(SP), INPUT_PTR                                                 ;
                   52               53   D0 00358          MOVL    INPUT_PTR, TEMP_PTR                                                    ;
                   56               83   9A 0035B          MOVZBL  (INPUT_PTR)+, DELIMITER                                                ;
                                    55   D4 0035E          CLRL    COUNT                                                                  ;
                   54               83   9A 00360 46$:     MOVZBL  (INPUT_PTR)+, CHAR                                                     ;
                                    10   12 00363          BNEQ    47$                                                                    ;
                        006D826B    8F   DD 00365          PUSHL   #7176811                                                               ;
00000000G          00               01   FB 0036B          CALLS   #1, LIB$SIGNAL                                                         ;
                   54               56   D0 00372          MOVL    DELIMITER, CHAR                                                        ;
                   56               54   D1 00375 47$:     CMPL    CHAR, DELIMITER                                                        ;
                                    08   12 00378          BNEQ    48$                                                                    ;
                   62               55   90 0037A          MOVB    COUNT, (TEMP_PTR)                                                      ;
                         FF         A3   94 0037D          CLRB    -1(INPUT_PTR)                                                          ;
                                    04   11 00380          BRB     49$                                                                    ;
                                    55   D6 00382 48$:     INCL    COUNT                                                                  ;
                                    DA   11 00384          BRB     46$                                                                    ;
                   51 00000000G     EF   D0 00386 49$:     MOVL    PAT$GB_MOD_PTR, R1                                                     ;
                   0A      03       A1   E9 0038D          BLBC    3(R1), 50$                                                             ;
00000000GEF        48               52   D0 00391          MOVL    TEMP_PTR, PAT$GL_SEMAN1[STACK_POINTER]                                 ;
                                    3E   11 00399          BRB     56$                                                                    ;
                         24         AE   D4 0039B 50$:     CLRL    VALUE                                                                  ;
                   04               62   91 0039E          CMPB    (TEMP_PTR), #4                                                         ;
                                    07   1A 003A1          BGTRU   51$                                                                    ;
```

```
                50              62  9A 003A3          MOVZBL  (TEMP_PTR), R0                                          ;
                                50  D7 003A6          DECL    R0                                                      ;
                                03  11 003A8          BRB     52$                                                     ;
                50              03  D0 003AA 51$:     MOVL    #3, R0                                                  ;
                54              01  CE 003AD 52$:     MNEGL   #1, INDEX                                               ;
                                07  11 003B0          BRB     54$                                                     ;
            24 AE44         01 A442 90 003B2 53$:     MOVB    1(INDEX)[TEMP_PTR], VALUE[INDEX]                        ;
F5              54              50  F3 003B9 54$:     AOBLEQ  R0, INDEX, 53$                                          ;
            01  A1              62  91 003BD          CMPB    (TEMP_PTR), 1(R1)                                       ;
                                0D  1B 003C1          BLEQU   55$                                                     ;
                        006D8033    8F DD 003C3       PUSHL   #7176243                                                ;
    00000000G   00              01  FB 003C9          CALLS   #1, LIB$SIGNAL                                          ;
    00000000GEF48           24  AE  D0 003D0 55$:     MOVL    VALUE, PAT$GL_SEMAN1[STACK_POINTER]                     ;
            18  BE              53  D0 003D9 56$:     MOVL    INPUT_PTR, @24(SP)                                      ;
                50              6A  3C 003DD          MOVZWL  (R10), R0                                               ;
                50              55  C2 003E0          SUBL2   COUNT, R0                                               ;
6A              50              01  A3 003E3          SUBW3   #1, R0, (R10)                                           ;
                                54  11 003E7          BRB     63$                                                     ; 5847
                17              53  91 003E9 57$:     CMPB    R3, #23                                                 ; 5859
                                16  12 003EC          BNEQ    58$                                                     ;
                                01  DD 003EE          PUSHL   #1                                                      ;
                            BC  AD  9F 003F0          PUSHAB  LEX_STG_DESC                                            ;
    00000000G   EF              02  FB 003F3          CALLS   #2, PAT$FIND_MODULE                                     ;
    00000000GEF48               50  D0 003FA          MOVL    R0, PAT$GL_SEMAN1[STACK_POINTER]                        ;
                                76  11 00402          BRB     67$                                                     ;
                23              53  91 00404 58$:     CMPB    R3, #35                                                 ; 5860
                                09  12 00407          BNEQ    59$                                                     ;
                                01  DD 00409          PUSHL   #1                                                      ;
                                7E  D4 0040B          CLRL    -(SP)                                                   ;
                            BC  AD  9F 0040D          PUSHAB  LEX_STG_DESC                                            ;
                                10  11 00410          BRB     60$                                                     ;
                26              53  91 00412 59$:     CMPB    R3, #38                                                 ; 5861
                                14  12 00415          BNEQ    61$                                                     ;
                                01  DD 00417          PUSHL   #1                                                      ;
                    00000000GEF48   DF 00419          PUSHAL  PAT$GL_SEMAN1[STACK_POINTER]                            ;
                                7E  D4 00420          CLRL    -(SP)                                                   ;
00000000G       EF              03  FB 00422 60$:     CALLS   #3, PAT$BUILD_PATH                                      ;
                                7B  11 00429          BRB     71$                                                     ;
                32              53  91 0042B 61$:     CMPB    R3, #50                                                 ; 5862
                                01  13 0042E          BEQL    62$                                                     ;
                                04  00430          RET                                                             ;
                            BC  AD  9F 00431 62$:     PUSHAB  LEX_STG_DESC                                            ;
                                58  DD 00434          PUSHL   STACK_POINTER                                           ;
00000000G       EF              02  FB 00436          CALLS   #2, PAT$TRANS_NAME                                      ;
                                67  11 0043D 63$:     BRB     71$                                                     ;
00000048        8F              5B  D1 0043F 64$:     CMPL    TOKEN, #72                                              ; 5876
                                1C  13 00446          BEQL    66$                                                     ;
                0A          7C  AE  B1 00448          CMPW    LAST_STG_DESC, #10                                      ; 5883
                                04  1B 0044C          BLEQU   65$                                                     ;
            7C  AE              0A  B0 0044E          MOVW    #10, LAST_STG_DESC                                      ; 5885
                            7C  AE  9F 00452 65$:     PUSHAB  LAST_STG_DESC                                           ; 5886
                                01  DD 00455          PUSHL   #1                                                      ;
                    006D82A8    8F  DD 00457          PUSHL   #7176872                                                ;
00000000G       00              03  FB 0045D          CALLS   #3, LIB$SIGNAL                                          ;
00000063        8F              5B  D1 00464 66$:     CMPL    TOKEN, #99                                              ; 5888
                                0F  13 0046B          BEQL    68$                                                     ;
                    006D80DA    8F  DD 0046D          PUSHL   #7176410                                                ; 5890
```

```
                00000000G  00          01  FB 00473          CALLS   #1, LIB$SIGNAL                          ;
                                       2A  11 0047A 67$:     BRB     71$                                     ;
                                  BC   AD  9F 0047C 68$:     PUSHAB  LEX_STG_DESC                            ; 5892
                                       01  DD 0047F          PUSHL   #1                                      ;
                           006D8270    8F  DD 00481          PUSHL   #7176816                                ;
                00000000G  00          03  FB 00487          CALLS   #3, LIB$SIGNAL                          ;
                                       16  11 0048E          BRB     71$                                     ; 5750
                0000270C   8F      04   AE  D1 00490 69$:     CMPL    TRANSIT_CODE, #9996                     ; 5901
                                       0A  12 00498          BNEQ    70$                                     ;
                           59 00000000'EF49  32 0049A        CVTWL   PAT_ACT_TABLE[J], J                     ; 5902
                                       02  11 004A2          BRB     71$                                     ;
                                       59  D6 004A4 70$:     INCL    J                                       ; 5903
                           03      08   AE  E8 004A6 71$:     BLBS    MATCH_TRANSIT, 72$                      ; 5905
                                     FBB0  31 004AA          BRW     2$                                      ;
                           03      0C   AE  E9 004AD 72$:     BLBC    PARSE_MORE, 73$                         ; 5911
                                     FBA5  31 004B1          BRW     1$                                      ;
                                           04 004B4 73$:     RET                                             ; 5912

; Routine Size:  1205 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
;  1372          5913  1 ROUTINE MAR_REDUCTN (ACTION_KEY, STACK_POINTER) =
;  1373          5914  1
;  1374          5915  1 !++
;  1375          5916  1 ! Functional description:
;  1376          5917  1 !
;  1377          5918  1 !       Does the action associated with a reduction. The action is chosen
;  1378          5919  1 !       based on the ACTION_KEY, which is the name of an action routine
;  1379          5920  1 !       as specified in the semantics table.
;  1380          5921  1 !
;  1381          5922  1 ! Calling sequence:
;  1382          5923  1 !
;  1383          5924  1 !       CALLS #2, MAR_REDUCTN
;  1384          5925  1 !
;  1385          5926  1 ! Inputs:
;  1386          5927  1 !
;  1387          5928  1 !       ACTION_KEY      - name of the action routine
;  1388          5929  1 !       STACK_POINTER   - top of stack in the context of the reduction
;  1389          5930  1 !
;  1390          5931  1 ! Implicit inputs:
;  1391          5932  1 !
;  1392          5933  1 !       The names of the two parse stacks, PAT$GL_SEMAN1 and
;  1393          5934  1 !       PAT$GL_SEMAN2.
;  1394          5935  1 !
;  1395          5936  1 ! Outputs:
;  1396          5937  1 !
;  1397          5938  1 !       TRUE if the action occurred and does not want to cause a return
;  1398          5939  1 !       from the parser.  Otherwise a FALSE.
;  1399          5940  1 !
;  1400          5941  1 !       none
;  1401          5942  1 !
;  1402          5943  1 ! Routine value:
;  1403          5944  1 !
;  1404          5945  1 !       TRUE or FALSE
;  1405          5946  1 !
;  1406          5947  1 ! Side effects:
;  1407          5948  1 !
;  1408          5949  1 !       The top of stack is often changed. Arguments are put into
;  1409          5950  1 !       linked lists, context values are altered.
;  1410          5951  1 !--
;  1411          5952  1
;  1412          5953  2 BEGIN
;  1413          5954  2
;  1414          5955  2 CASE .ACTION_KEY FROM 1 TO PATNONE OF
;  1415          5956  2
;  1416          5957  2         SET
;  1417          5958  2
;  1418          5959  2         [PATADDEXP]:    ADDITION (.STACK_POINTER);
;  1419          5960  2         [PATALIBYT]:    SET_BYTE_BIT;
;  1420          5961  2         [PATALILNG]:    SET_LONG_BIT;
;  1421          5962  2         [PATALINM0]:    LINK_NAME(.STACK_POINTER);
;  1422          5963  2         [PATALINM1]:    LINK_NAME(.STACK_POINTER + PAT$K_SPOS_ONE);
;  1423          5964  2         [PATALIPAG]:    SET_PAGE_BIT;
;  1424          5965  2         [PATALIQAD]:    SET_QUAD_BIT;
;  1425          5966  2         [PATALIWRD]:    SET_WORD_BIT;
;  1426          5967  2         [PATANDOPR]:    LOGICAL_AND (.STACK_POINTER);
;  1427          5968  2         [PATCHKNEC]:    SET_NOT_ECO_BIT (.STACK_POINTER);
;  1428          5969  2         [PATCOMLIN]:    EXECUTE_CMD (.STACK_POINTER);
```

```
2          [PATDEFONE]:    LINK_EXP_NAME (.STACK_POINTER + PAT$K_SPOS_ONE);
2          [PATDEFTWO]:    LINK_EXP_NAME (.STACK_POINTER + PAT$K_SPOS_TWO);
2          [PATDEFZER]:    LINK_EXP_NAME (.STACK_POINTER);
2          [PATDIVEXP]:    DIVISION (.STACK_POINTER);
2 !        [PATEQEXPR]:    EQ_EXPR (.STACK_POINTER);
2          [PATEXITOK]:    LINK_EXIT;
2          [PATEXPONE]:    LINK_ARG (.STACK_POINTER + PAT$K_SPOS_ONE);
2          [PATEXPTWO]:    LINK_ARG (.STACK_POINTER + PAT$K_SPOS_TWO);
2          [PATEXPZER]:    LINK_ARG (.STACK_POINTER);
2          [PATEXTBIT]:    EXTRACT_BITS (.STACK_POINTER);
2 !        [PATGEEXPR]:    GE_EXPR (.STACK_POINTER);
2 !        [PATGTEXPR]:    GT_EXPR (.STACK_POINTER);
2          [PATINDEXP]:    INDIRECTION (.STACK_POINTER);
2 !        [PATLEEXPR]:    LE_EXPR (.STACK_POINTER);
2 !        [PATLTEXPR]:    LT_EXPR (.STACK_POINTER);
2          [PATMULEXP]:    MULTIPLICATION (.STACK_POINTER);
2 !        [PATNEEXPR]:    NE_EXPR (.STACK_POINTER);
2          [PATNEGEXP]:    NEGATION (.STACK_POINTER);
2          [PATNOTOPR]:    COMPLEMENT (.STACK_POINTER);
2          [PATNUMONE]:    LINK_NUM (.STACK_POINTER + PAT$K_SPOS_ONE);
2          [PATNUMTWO]:    LINK_NUM (.STACK_POINTER + PAT$K_SPOS_TWO);
2          [PATNUMZER]:    LINK_NUM (.STACK_POINTER);
2          [PATOROPER]:    LOGICAL_OR (.STACK_POINTER);
2          [PATOVROP2]:    SET_OVERR_MODE (.STACK_POINTER + PAT$K_SPOS_TWO);
2          [PATOVROP1]:    SET_OVERR_MODE (.STACK_POINTER + PAT$K_SPOS_ONE);
2          [PATPOSEXP]:    POSITIVE (.STACK_POINTER);
2          [PATRANGE0]:    LINK_ARG_PAIR (.STACK_POINTER);
2          [PATRANGE1]:    LINK_ARG_PAIR (.STACK_POINTER + PAT$K_SPOS_ONE);
2          [PATRANGE2]:    LINK_ARG_PAIR (.STACK_POINTER + PAT$K_SPOS_TWO);
2          [PATREMPAR]:    REMOVE_PARENS (.STACK_POINTER);
2          [PATSETDEC]:    SET_DEC_OVERS;
2          [PATSETECO]:    SET_ECO_BIT (.STACK_POINTER);
2          [PATSETLIT]:    SET_LIT_BIT;
2          [PATSETMDL]:    SET_MODULE_BIT;
2          [PATSETMOD]:    SET_MODE_BIT;
2          [PATSETPAT]:    SET_PATAREA_BIT;
2          [PATSETINI]:    SET_INIT_BIT (.STACK_POINTER);
2          [PATSETSCO]:    SET_SCOPE_BIT;
2          [PATSHFEXP]:    ARITH_SHIFT (.STACK_POINTER);
2          [PATSUBEXP]:    SUBTRACTION (.STACK_POINTER);
2          [PATNONE]:      0;
2
2          [INRANGE, OUTRANGE]:
2                  RETURN FALSE;
2
2          TES;
2
2 RETURN TRUE
1 END;
; INFO#212                L1:5969
; Null expression appears in value-required context
```

```
                                                    0FFC 00000 MAR_REDUCTN:
```

```
                                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 5913
                        5B 00000000G  EF  9E 00002              MOVAB   PAT$ADD_ARG, R11                              ;
                        5A 00000000G  EF  9E 00009              MOVAB   PAT$GL_SEMAN2, R10                            ;
                        59 00000000G  EF  9E 00010              MOVAB   PAT$GL_TAIL_LST, R9                           ;
                        58 00000000G  EF  9E 00017              MOVAB   PAT$GB_MOD_PTR, R8                            ;
                        57 00000000G  EF  9E 0001E              MOVAB   PAT$GL_CONTEXT, R7                            ;
                        56 00000000G  00  9E 00025              MOVAB   LIB$SIGNAL, R6                                ;
                        55 00000000'  EF  9E 0002C              MOVAB   QUOTE_INDIC, R5                               ;
                        54 00000000G  EF  9E 00033              MOVAB   PAT$GL_SEMAN1, R4                             ;
                 3B     01        04  AC  CF 0003A              CASEL   ACTION_KEY, #1, #59                           ; 5955
0091            030D    02E7          02C5     0003F 1$:          .WORD   61$-1$,-                                      ;
00C4            00AA    03CE          009A     00047                    64$-1$,-                                      ;
008C            007A    00DA          00A5     0004F                    68$-1$,-                                      ;
00CA            0122    03CE          0343     00057                    5$-1$,-                                       ;
03CE            00B4    033F          00AF     0005F                    6$-1$,-                                       ;
0333            03CE    02A8          0086     00067                    88$-1$,-                                      ;
03CE            021D    0292          0238     0006F                    9$-1$,-                                       ;
0365            0389    035F          00FE     00077                    13$-1$,-                                      ;
016A            03CE    03B0          03CE     0007F                    8$-1$,-                                       ;
0145            01CD    03CE          03CE     00087                    16$-1$,-                                      ;
036A            03AA    03CE          03CE     0008F                    2$-1$,-                                       ;
0286            0359    0160          0250     00097                    4$-1$,-                                       ;
03CE            022C    03CE          03C0     0009F                    74$-1$,-                                      ;
026E            0333    03CE          01B1     000A7                    88$-1$,-                                      ;
03CA            025F    018D          0244     000AF                    22$-1$,-                                      ;
                                                                        14$-1$,-                                      ;
                                                                        10$-1$,-                                      ;
                                                                        73$-1$,-                                      ;
                                                                        11$-1$,-                                      ;
                                                                        88$-1$,-                                      ;
                                                                        3$-1$,-                                       ;
                                                                        58$-1$,-                                      ;
                                                                        88$-1$,-                                      ;
                                                                        71$-1$,-                                      ;
                                                                        46$-1$,-                                      ;
                                                                        57$-1$,-                                      ;
                                                                        44$-1$,-                                      ;
                                                                        88$-1$,-                                      ;
                                                                        19$-1$,-                                      ;
                                                                        76$-1$,-                                      ;
                                                                        81$-1$,-                                      ;
                                                                        77$-1$,-                                      ;
                                                                        88$-1$,-                                      ;
                                                                        84$-1$,-                                      ;
                                                                        88$-1$,-                                      ;
                                                                        29$-1$,-                                      ;
                                                                        88$-1$,-                                      ;
                                                                        88$-1$,-                                      ;
                                                                        41$-1$,-                                      ;
                                                                        25$-1$,-                                      ;
                                                                        88$-1$,-                                      ;
                                                                        88$-1$,-                                      ;
                                                                        82$-1$,-                                      ;
                                                                        79$-1$,-                                      ;
                                                                        48$-1$,-                                      ;
                                                                        27$-1$,-                                      ;
                                                                        75$-1$,-                                      ;
```

```
                                                                55$-1$,-                                  ;
                                                                86$-1$,-                                  ;
                                                                88$-1$,-                                  ;
                                                                45$-1$,-                                  ;
                                                                88$-1$,-                                  ;
                                                                38$-1$,-                                  ;
                                                                88$-1$,-                                  ;
                                                                71$-1$,-                                  ;
                                                                52$-1$,-                                  ;
                                                                47$-1$,-                                  ;
                                                                33$-1$,-                                  ;
                                                                50$-1$,-                                  ;
                                                                87$-1$                                    ;
                          5D  11 000B7          BRB     15$                                       ; 6013
          50        08    AC  D0 000B9 2$:      MOVL    STACK_POINTER, R0                         ; 5959
        6440        10 A440  C0 000BD           ADDL2   PAT$GL_SEMAN1+16[R0], PAT$GL_SEMAN1[R0]   ;
                          3B  11 000C3          BRB     12$                                       ;
          67        40    8F  88 000C5 3$:      BISB2   #64, PAT$GL_CONTEXT                       ; 5960
                          35  11 000C9          BRB     12$                                       ;
          67              04  88 000CB 4$:      BISB2   #4, PAT$GL_CONTEXT                        ; 5961
                          30  11 000CE          BRB     12$                                       ;
          50        08    AC  D0 000D0 5$:      MOVL    STACK_POINTER, R0                         ; 5962
                        6A40  DD 000D4          PUSHL   PAT$GL_SEMAN2[R0]                         ;
                          08  11 000D7          BRB     7$                                        ;
          50        08    AC  D0 000D9 6$:      MOVL    STACK_POINTER, R0                         ; 5963
                    08 AA40  DD 000DD           PUSHL   PAT$GL_SEMAN2+8[R0]                       ;
                        01DC  31 000E1 7$:      BRW     54$                                       ;
          67              20  88 000E4 8$:      BISB2   #32, PAT$GL_CONTEXT                       ; 5964
                          17  11 000E7          BRB     12$                                       ;
          67              08  88 000E9 9$:      BISB2   #8, PAT$GL_CONTEXT                        ; 5965
                          12  11 000EC          BRB     12$                                       ;
          67              10  88 000EE 10$:     BISB2   #16, PAT$GL_CONTEXT                       ; 5966
                          0D  11 000F1          BRB     12$                                       ;
          50        08    AC  D0 000F3 11$:     MOVL    STACK_POINTER, R0                         ; 5967
          51        10 A440  D2 000F7           MCOML   PAT$GL_SEMAN1+16[R0], R1                  ;
        6440              51  CA 000FC          BICL2   R1, PAT$GL_SEMAN1[R0]                     ;
                        0306  31 00100 12$:     BRW     87$                                       ;
          67              02  88 00103 13$:     BISB2   #2, PAT$GL_CONTEXT                        ; 5968
                        0279  31 00106          BRW     74$                                       ;
                    08    AC  DD 00109 14$:     PUSHL   STACK_POINTER                             ; 5969
00000000G EF              01  FB 0010C          CALLS   #1, PAT$PERFORM_CMD                       ;
          EA              50  E8 00113          BLBS    R0, 12$                                   ;
                        02F4  31 00116 15$:     BRW     88$                                       ;
          50              68  D0 00119 16$:     MOVL    PAT$GB_MOD_PTR, R0                        ; 5970
          04        03    A0  E8 0011C          BLBS    3(R0), 17$                                ;
          0C        04    A0  E9 00120          BLBC    4(R0), 18$                                ;
          09              65  E8 00124 17$:     BLBS    QUOTE_INDIC, 18$                          ;
              006D8260    8F  DD 00127          PUSHL   #7176800                                  ;
          66              01  FB 0012D          CALLS   #1, LIB$SIGNAL                            ;
                          65  D4 00130 18$:     CLRL    QUOTE_INDIC                               ;
          50        08    AC  D0 00132          MOVL    STACK_POINTER, R0                         ;
                    08 AA40  DD 00136           PUSHL   PAT$GL_SEMAN2+8[R0]                       ;
                        01FD  31 0013A          BRW     66$                                       ;
          50              68  D0 0013D 19$:     MOVL    PAT$GB_MOD_PTR, R0                        ; 5971
          04        03    A0  E8 00140          BLBS    3(R0), 20$                                ;
          0C        04    A0  E9 00144          BLBC    4(R0), 21$                                ;
          09              65  E8 00148 20$:     BLBS    QUOTE_INDIC, 21$                          ;
```

```
          006D8260   8F  DD 0014B          PUSHL   #7176800                                  ;
      66             01  FB 00151          CALLS   #1, LIB$SIGNAL                            ;
                     65  D4 00154 21$:     CLRL    QUOTE_INDIC                               ;
      50        08   AC  D0 00156          MOVL    STACK_POINTER, R0                         ;
                10 AA40  DD 0015A          PUSHL   PAT$GL_SEMAN2+16[R0]                      ;
                   01FF  31 0015E          BRW     70$                                       ;
      50             68  D0 00161 22$:     MOVL    PAT$GB_MOD_PTR, R0                        ; 5972
      04        03   A0  E8 00164          BLBS    3(R0), 23$                                ;
      0C        04   A0  E9 00168          BLBC    4(R0), 24$                                ;
      09             65  E8 0016C 23$:     BLBS    QUOTE_INDIC, 24$                          ;
          006D8260   8F  DD 0016F          PUSHL   #7176800                                  ;
      66             01  FB 00175          CALLS   #1, LIB$SIGNAL                            ;
                     65  D4 00178 24$:     CLRL    QUOTE_INDIC                               ;
      52        08   AC  D0 0017A          MOVL    STACK_POINTER, R2                         ;
                   6A42  DD 0017E          PUSHL   PAT$GL_SEMAN2[R2]                         ;
                   0193  31 00181          BRW     63$                                       ;
      52        08   AC  D0 00184 25$:     MOVL    STACK_POINTER, R2                         ; 5973
                10 A442  D5 00188          TSTL    PAT$GL_SEMAN1+16[R2]                      ;
                     09  12 0018C          BNEQ    26$                                       ;
          006D8248   8F  DD 0018E          PUSHL   #7176776                                  ;
      66             01  FB 00194          CALLS   #1, LIB$SIGNAL                            ;
    6442        10 A442  C6 00197 26$:     DIVL2   PAT$GL_SEMAN1+16[R2], PAT$GL_SEMAN1[R2]   ;
                     07  11 0019D          BRB     28$                                       ; 5955
      50             69  D0 0019F 27$:     MOVL    PAT$GL_TAIL_LST, R0                       ; 5975
 08   A0             0A  D0 001A2          MOVL    #10, 8(R0)                                ;
                   0260  31 001A6 28$:     BRW     87$                                       ;
      50             68  D0 001A9 29$:     MOVL    PAT$GB_MOD_PTR, R0                        ; 5976
      04        03   A0  E8 001AC          BLBS    3(R0), 30$                                ;
      0C        04   A0  E9 001B0          BLBC    4(R0), 31$                                ;
      09             65  E8 001B4 30$:     BLBS    QUOTE_INDIC, 31$                          ;
          006D8260   8F  DD 001B7          PUSHL   #7176800                                  ;
      66             01  FB 001BD          CALLS   #1, LIB$SIGNAL                            ;
                     65  D4 001C0 31$:     CLRL    QUOTE_INDIC                               ;
      50        08   AC  D0 001C2 32$:     MOVL    STACK_POINTER, R0                         ;
                08 A440  DD 001C6          PUSHL   PAT$GL_SEMAN1+8[R0]                       ;
                     21  11 001CA          BRB     37$                                       ;
      50             68  D0 001CC 33$:     MOVL    PAT$GB_MOD_PTR, R0                        ; 5977
      04        03   A0  E8 001CF          BLBS    3(R0), 34$                                ;
      0C        04   A0  E9 001D3          BLBC    4(R0), 35$                                ;
      09             65  E8 001D7 34$:     BLBS    QUOTE_INDIC, 35$                          ;
          006D8260   8F  DD 001DA          PUSHL   #7176800                                  ;
      66             01  FB 001E0          CALLS   #1, LIB$SIGNAL                            ;
                     65  D4 001E3 35$:     CLRL    QUOTE_INDIC                               ;
      50        08   AC  D0 001E5 36$:     MOVL    STACK_POINTER, R0                         ;
                10 A440  DD 001E9          PUSHL   PAT$GL_SEMAN1+16[R0]                      ;
                   00D0  31 001ED 37$:     BRW     54$                                       ;
      50             68  D0 001F0 38$:     MOVL    PAT$GB_MOD_PTR, R0                        ; 5978
      04        03   A0  E8 001F3          BLBS    3(R0), 39$                                ;
      0C        04   A0  E9 001F7          BLBC    4(R0), 40$                                ;
      09             65  E8 001FB 39$:     BLBS    QUOTE_INDIC, 40$                          ;
          006D8260   8F  DD 001FE          PUSHL   #7176800                                  ;
      66             01  FB 00204          CALLS   #1, LIB$SIGNAL                            ;
                     65  D4 00207 40$:     CLRL    QUOTE_INDIC                               ;
                   00AD  31 00209          BRW     53$                                       ;
      52        08   AC  D0 0020C 41$:     MOVL    STACK_POINTER, R2                         ; 5979
      53           6442  D0 00210          MOVL    PAT$GL_SEMAN1[R2], VALUE                  ;
      1F        10 A442  D1 00214          CMPL    PAT$GL_SEMAN1+16[R2], #31                 ;
```

```
                                                
```

```
                          02  DD 002EF          PUSHL   #2                                          ;
00000000G  EF             02  FB 002F1          CALLS   #2, PAT$SET_OVERS                           ;
                    02    A2  9F 002F8          PUSHAB  2(R2)                                       ;
00000000G  EF             01  FB 002FB 59$:     CALLS   #1, PAT$SET_COMQUAL                         ;
                          78  11 00302 60$:     BRB     72$                                         ; 5955
           09             65  E9 00304 61$:     BLBC    QUOTE_INDIC, 62$                            ; 5996
              006D8258    8F  DD 00307          PUSHL   #7176792                                    ;
           66             01  FB 0030D          CALLS   #1, LIB$SIGNAL                              ;
           52       08    AC  D0 00310 62$:     MOVL    STACK_POINTER, R2                           ;
                       6442  DD 00314          PUSHL   PAT$GL_SEMAN1[R2]                           ;
           6B             01  FB 00317 63$:     CALLS   #1, PAT$ADD_ARG                             ;
           50             69  D0 0031A          MOVL    PAT$GL_TAIL_LST, R0                         ;
     08    A0       10 A442  D0 0031D          MOVL    PAT$GL_SEMAN1+16[R2], 8(R0)                 ;
                       0081  31 00323          BRW     78$                                         ; 5955
           09             65  E9 00326 64$:     BLBC    QUOTE_INDIC, 65$                            ; 5997
              006D8258    8F  DD 00329          PUSHL   #7176792                                    ;
           66             01  FB 0032F          CALLS   #1, LIB$SIGNAL                              ;
           50       08    AC  D0 00332 65$:     MOVL    STACK_POINTER, R0                           ;
                    08 A440  DD 00336          PUSHL   PAT$GL_SEMAN1+8[R0]                         ;
           6B             01  FB 0033A 66$:     CALLS   #1, PAT$ADD_ARG                             ;
           51             69  D0 0033D          MOVL    PAT$GL_TAIL_LST, R1                         ;
           50       08    AC  D0 00340          MOVL    STACK_POINTER, R0                           ;
     08    A1       18 A440  D0 00344          MOVL    PAT$GL_SEMAN1+24[R0], 8(R1)                 ;
                          7A  11 0034A 67$:     BRB     80$                                         ; 5955
           09             65  E9 0034C 68$:     BLBC    QUOTE_INDIC, 69$                            ; 5998
              006D8258    8F  DD 0034F          PUSHL   #7176792                                    ;
           66             01  FB 00355          CALLS   #1, LIB$SIGNAL                              ;
           50       08    AC  D0 00358 69$:     MOVL    STACK_POINTER, R0                           ;
                    10 A440  DD 0035C          PUSHL   PAT$GL_SEMAN1+16[R0]                        ;
           6B             01  FB 00360 70$:     CALLS   #1, PAT$ADD_ARG                             ;
           51             69  D0 00363          MOVL    PAT$GL_TAIL_LST, R1                         ;
           50       08    AC  D0 00366          MOVL    STACK_POINTER, R0                           ;
     08    A1       20 A440  D0 0036A          MOVL    PAT$GL_SEMAN1+32[R0], 8(R1)                 ;
                          7B  11 00370          BRB     83$                                         ; 5955
           50       08    AC  D0 00372 71$:     MOVL    STACK_POINTER, R0                           ; 5999
         6440       08 A440  D0 00376          MOVL    PAT$GL_SEMAN1+8[R0], PAT$GL_SEMAN1[R0]      ;
                          7F  11 0037C 72$:     BRB     85$                                         ;
     02    A7             04  88 0037E 73$:     BISB2   #4, PAT$GL_CONTEXT+2                        ; 6001
                          02  DD 00382 74$:     PUSHL   #2                                          ;
00000000G  EF             01  FB 00384          CALLS   #1, PAT$SET_MOD_LVL                         ;
                          16  DD 0038B          PUSHL   #22                                         ;
                          03  DD 0038D          PUSHL   #3                                          ;
00000000G  EF             02  FB 0038F          CALLS   #2, PAT$SET_OVERS                           ;
                          71  11 00396          BRB     87$                                         ; 5955
     03    A7             02  88 00398 75$:     BISB2   #2, PAT$GL_CONTEXT+3                        ; 6002
                          6B  11 0039C          BRB     87$                                         ;
           67       80    8F  88 0039E 76$:     BISB2   #128, PAT$GL_CONTEXT                        ; 6003
                          65  11 003A2          BRB     87$                                         ;
           67             01  88 003A4 77$:     BISB2   #1, PAT$GL_CONTEXT                          ; 6004
                          60  11 003A7 78$:     BRB     87$                                         ;
     02    A7             08  88 003A9 79$:     BISB2   #8, PAT$GL_CONTEXT+2                        ; 6005
           50 00000000G   EF  D0 003AD          MOVL    PAT$GL_HEAD_LST, R0                         ;
           51             69  D0 003B4          MOVL    PAT$GL_TAIL_LST, R1                         ;
           51             50  D1 003B7          CMPL    R0, R1                                      ;
                          4D  13 003BA          BEQL    87$                                         ;
     08    A0       04    A0  D0 003BC          MOVL    4(R0), 8(R0)                                ;
     04    A0       04    A1  D0 003C1          MOVL    4(R1), 4(R0)                                ;
```

```
                                                41  11 003C6 80$:    BRB     87$                                             ; 5955
                                    02   A7     02  88 003C8 81$:    BISB2   #2, PAT$GL_CONTEXT+2                            ; 6006
                              00000000G  EF     10  88 003CC         BISB2   #16, PAT$G[_COMQUAL+1                           ;
                                         51 00000000G EF  D0 003D3    MOVL    PAT$GL_HEAD_LST, R1                             ;
                                         50          69  D0 003DA    MOVL    PAT$GL_TAIL_LST, R0                             ;
                                         50          51  D1 003DD    CMPL    R1, R0                                          ;
                                                     27  13 003E0    BEQL    87$                                             ;
                                    08   A1     04  A0  D0 003E2    MOVL    4(R0), 8(R1)                                    ;
                                                20  11 003E7         BRB     87$                                             ; 5955
                                    02   A7     01  88 003E9 82$:    BISB2   #1, PAT$GL_CONTEXT+2                            ; 6007
                                                1A  11 003ED 83$:    BRB     87$                                             ;
                                         50     08  AC  D0 003EF 84$:    MOVL    STACK_POINTER, R0                           ; 6008
                                                10 A440 DF 003F3         PUSHAL  PAT$G[_SEMAN1+16[R0]                        ;
                              6440           6440     9E  78 003F7         ASHL    @(SP)+, PAT$GL_SEMAN1[R0], PAT$GL_SEMAN1-   ;
                                                                                   [R0]                                        ;
                                                0A  11 003FD 85$:    BRB     87$                                             ;
                                         50     08  AC  D0 003FF 86$:    MOVL    STACK_POINTER, R0                           ; 6009
                                       6440     10 A440 C2  00403         SUBL2   PAT$G[_SEMAN1+16[R0], PAT$GL_SEMAN1[R0]      ;
                                         50           01  D0 00409 87$:    MOVL    #1, R0                                      ; 6017
                                                          04  0040C         RET                                                 ;
                                                     50   D4  0040D 88$:    CLRL    R0                                          ; 6018
                                                          04  0040F         RET                                                 ;

; Routine Size:  1040 bytes,    Routine Base:  _PAT$CODE + 04B5
```

```
; 1479          6019  1 END
; 1480          6020  0 ELUDOM



                                                                    .EXTRN  LIB$SIGNAL

;                                  PSECT SUMMARY
;
;       Name                    Bytes                        Attributes
;
;   _PAT$PLIT                    3120  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(0)
;   _PAT$OWN                        4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   _PAT$CODE                    2245  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   .  ABS  .                       0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)



;                              Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1        18619         8         0      1000         00:01.7



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


;                                  COMMAND QUALIFIERS

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

; Size:          2245 code + 3124 data bytes
; Run Time:         01:10.1
; Elapsed Time:     03:31.4
; Lines/CPU Min:     5155
; Lexemes/CPU-Min: 29406
; Memory Used:  536 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PATPAR
LIS

PATMAC
LIS

PATMAI
LIS

PATLST
LIS

PATIO
LIS

PATLEX
LIS

PATMOD
LIS

PATMSG
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PATREB
LIS

PATSCA
LIS

PATSIO
LIS

PATRST
LIS

PATSPA
LIS

PATSSV
LIS