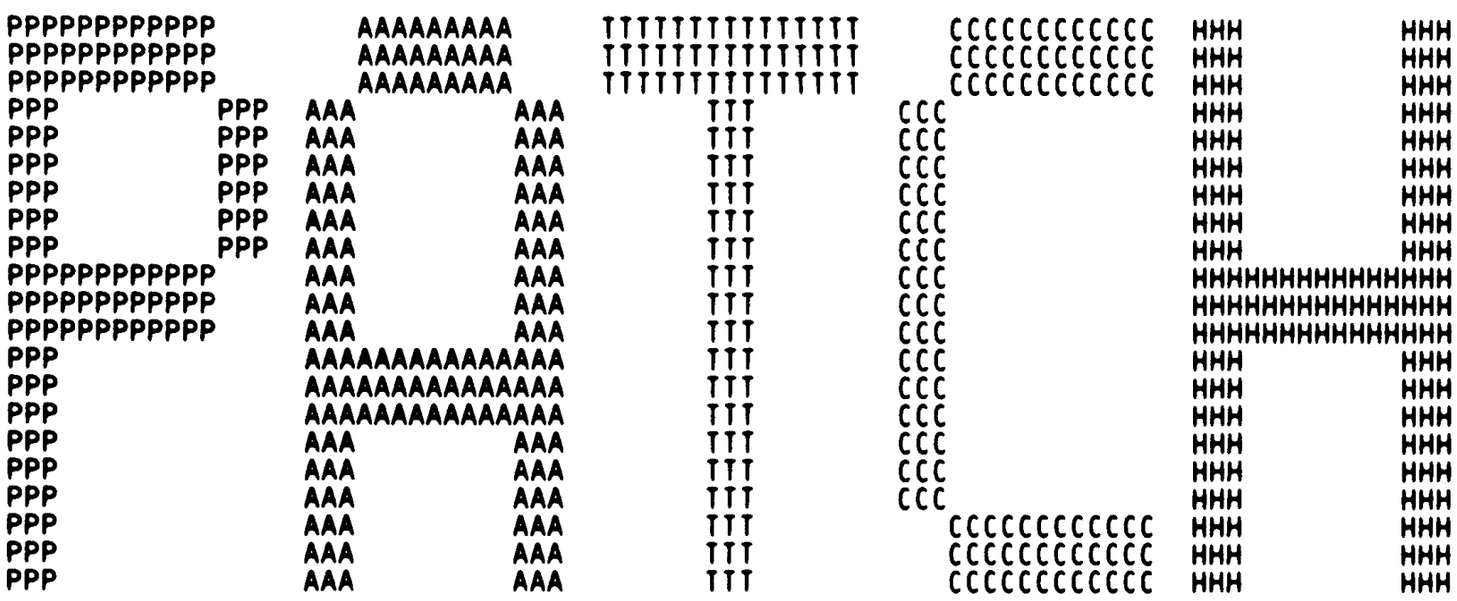
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPPPPPPPPPPP AAAAAAAAAAAAAAAA TTT CCC HHHHHHHHHHHHHHHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCCC HHH HHH

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


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 PATSSV (
;     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 !       CALLS TO STARLET OPERATING SYSTEM I/O SERVICES.  INCLUDE
;    38          0038  1 !       ASSIGNMENT AND DEASSIGNMENT OF CHANNELS, READS, WRITES.
;    39          0039  1
;    40          0040  1 ! Version:      V02-018
;    41          0041  1
;    42          0042  1 ! History:
;    43          0043  1 !       Author:
;    44          0044  1 !               Carol Peters, 21 Sep 1976: Version 01
;    45          0045  1 !
;    46          0046  1 !       Modified by:
;    47          0047  1 !               Kevin Pammett, 15-dec-77: Version 25
;    48          0048  1 !
;    49          0049  1 !       V03-001 MTR0012         Mike Rhodes             16-Aug-1982
;    50          0050  1 !               Modify file names to remove duplicate file name useage
;    51          0051  1 !               between code and require files.
;    52          0052  1 !
;    53          0053  1 !       V02-018 PCG0001         Peter George            02-FEB-1981
;    54          0054  1 !               Add require statement for LIB$:PATDEF.REQ
;    55          0055  1 !
;    56          0056  1 !       V0117   CNH0016         Chris Hume               4-Oct-1979     14:00
;    57          0057  1 !               Fix PAT$OUT_SYM_VAL to properly insert a leading zero where
```

```
;    58          0058  1 !               required to distinguish hex constants from symb ls.  This cures
;    59          0059  1 !               several anomalies.  (PATMAI.B32 02.26)
;    60          0060  1 !
;    61          0061  1 ! Revision History:
;    62          0062  1 !
;    63          0063  1 ! NO     DATE            PROGRAMMER          PURPOSE
;    64          0064  1 ! --     ----            ----------          -------
;    65          0065  1 !
;    66          0066  1 ! 00     20-DEC-77       K.D. MORSE          ADAPT VERSION 25 FOR PATCH.
;    67          0067  1 !  01    27-DEC-77       K.D. MORSE          CHANGE PAT$OUT_TYP_VAL CALLS TO
;    68          0068  1 !                                            FOR$CVT ... TO CALL PAT$GET_VALUE.
;    69          0069  1 !  02    2-JAN-78        K.D. MORSE          ADD CHECKS FOR IMAGE WITH NO SYMBOLS.
;    70          0070  1 !  03    4-JAN-78        K.D. MORSE          FOR OUTPUTTING COMPLEX NUMBERS,
;    71          0071  1 !                                            SURROUND THEM IN PARENTHESIS, AND
;    72          0072  1 !                                            REMOVE THE SPURIOUS QUOTES
;    73          0073  1 !                                            AROUND THE COMMA. (26)
;    74          0074  1 !                                            NO CHANGES FOR 27.
;    75          0075  1 !  04    28-FEB-78       K.D. MORSE          NO CHANGES FOR 28-29.
;    76          0076  1 !  05    02-MAR-78       K.D. MORSE          NO CHANGES FOR 30-31.
;    77          0077  1 !  06    17-MAR-78       K.D. MORSE          ADD CODE TO OUTPUT A LEADING 0
;    78          0078  1 !                                            FOR NEGATIVE HEX NUMBERS TO
;    79          0079  1 !                                            PAT$OUT_NUM_VAL.  THE CODE WAS
;    80          0080  1 !                                            TO DO THIS IN PAT$INS_DECODE
;    81          0081  1 !                                            WAS INCORRECT AND THERFORE REMOVED.
;    82          0082  1 !  07    24-MAR-78       K.D. MORSE          NO CHANGES FOR VERS 32-34.
;    83          0083  1 !  08    30-MAR-78       K.D. MORSE          ADD "PC-RELATIVE" ARRAYS.
;    84          0084  1 !  09    07-APR-78       K.D. MORSE          ADD PAT$PV TO CS, AND CHANGE
;    85          0085  1 !                                            PAT$PRINT_PATR TO USE IT (35).
;    86          0086  1 !                                            OUT_SYM_VAL NOW CONSIDERS
;    87          0087  1 !                                            DEFINE SYMBOLS (FOR EXACT
;    88          0088  1 !                                            MATCHES ONLY) (36).
;    89          0089  1 !                                            NO CHANGES FOR 37.
;    90          0090  1 !  10    25-APR-78       K.D. MORSE          CONVERT TO NATIVE COMPILER.
;    91          0091  1 !  11    17-MAY-78       K.D. MORSE          SHOW MODU NOW SKIPS THE GLOBAL
;    92          0092  1 !                                            MC. (38)
;    93          0093  1 !                                            SHOW MODU PRINTS ERROR MESSAGE
;    94          0094  1 !                                            IF THERE IS NO DST. (39)
;    95          0095  1 !  12    18-MAY-78       K.D. MORSE          NO CHANGES FOR VERS 40-41.
;    96          0096  1 !  13    13-JUN-78       K.D. MORSE          ADD FAO COUNTS TO SIGNALS.
;    97          0097  1 !  14    27-JUN-78       K.D. MORSE          ADD LEADING ZEROS FOR HEX
;    98          0098  1 !                                            OUTPUT IN OUT_SYM_VAL. (42)
;    99          0099  1 !  15    28-JUN-78       K.D. MORSE          NO CHANGES FOR VERS 43-48.
;   100          0100  1 !  16    29-JUN-78       K.D. MORSE          NO CHANGES FOR VERS 49.
;   101          0101  1 !                                            SHOW MODULE NOW PRINTS
;   102          0102  1 !                                            ALL NUMBERS AS DECIMAL (50).
;   103          0103  1 !                                            NO CHANGES FOR VERS 51.
;   104          0104  1 !
;   105          0105  1 !--
```

```
;  107          0106  1 !
;  108          0107  1 ! TABLE OF CONTENTS:
;  109          0108  1 !
;  110          0109  1
;  111          0110  1 FORWARD ROUTINE
;  112          0111  1         PAT$PV_TO_CS : NOVALUE,                 ! Encode a pathvector into a buffer.
;  113          0112  1         PAT$FAO_PUT : NOVALUE,                  ! Stuff output into current output buffer.
;  114          0113  1         PAT$MODULE_SIZE,                        ! Estimate the RST size of a module.
;  115          0114  1         PAT$OUT_NUM_VAL : NOVALUE,              ! Output of numeric values.
;  116          0115  1         PAT$OUT_PUT : NOVALUE,                  ! Actually write out a line to tty
;  117          0116  1         PAT$OUT_SYM_VAL : NOVALUE,              ! Output values either i  numeric
;  118          0117  1                                                 ! or in symbolic form.
;  119          0118  1         PAT$SHOW_MODULE : NOVALUE,              ! List off the module chain.
;  120          0119  1         PAT$SHOW_SCOPE : NOVALUE,               ! Print out the current CSP.
;  121          0120  1         PAT$PRINT_PATH : NOVALUE;               ! Print out a symbol pathname
;  122          0121  1
;  123          0122  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;  124          0123  1 REQUIRE 'SRC$:VXSMAC.REQ';
;  125          0188  1 REQUIRE 'LIB$:PATDEF.REQ';                      ! Defines literals
;  126          0242  1 REQUIRE 'LIB$:PATMSG.REQ';
;  127          0416  1 REQUIRE 'SRC$:PATPCT.REQ';
;  128          0456  1 REQUIRE 'SRC$:PATGEN.REQ';
;  129          0678  1 REQUIRE 'SRC$:SYSLIT.REQ';
;  130          0728  1 REQUIRE 'SRC$:SYSSER.REQ';
```

```
;  R0760  1     SWITCHES LIST (SOURCE);
;  R0761  1
;  R0762  1     EXTERNAL ROUTINE
;  R0763  1              PAT$fao_out;            ! formats a line and outputs to the terminal
;  R0764  1
```

```
;  131          0810  1 REQUIRE 'SRC$:PATRTS.REQ';
;  132          1906  1 REQUIRE 'SRC$:BSTRUC.REQ';
;  133          1982  1 REQUIRE 'SRC$:DLLNAM.REQ';
;  134          2040  1 REQUIRE 'SRC$:PREFIX.REQ';
;  135          2228  1 REQUIRE 'SRC$:PATPRE.REQ';
;  136          2391  1
;  137          2392  1 EXTERNAL ROUTINE
;  138          2393  1          PAT$FIND_VAL,                          ! VAL_TO_SYM on define symbols
;  139          2394  1          PAT$GET_BOUNDS : NOVALUE,              ! Understand array descriptors
;  140          2395  1          PAT$GET_VALUE : NOVALUE,               ! Takes unmapped address and returns content
;  141          2396  1          PAT$ADD_NT_T_PV,                       ! Built pathname vectors
;  142          2397  1          PAT$REPORT_FREE,                       ! Bytes remaining in free storage
;  143          2398  1          PAT$VAL_TO_SYM,                        ! Translate values to their
;  144          2399  1                                                 ! symbolic equivalent
;  145          2400  1          PAT$SYMBOL_VALU,
;  146          2401  1          SYS$FAOL: ADDRESSING_MODE (ABSOLUTE);  ! System service to do formatted output
;  147          2402  1
;  148          2403  1 EXTERNAL
;  149          2404  1          PAT$GL_SYMTBPTR,                       ! Pointer to default symbol table
;  150          2405  1          PAT$GL_OLDLABLS,                       ! Pointer to old labels symbol table
;  151          2406  1          PAT$GL_SYMHEAD,                        ! Pointer to user-defined symbol table
;  152          2407  1          PAT$GB_SYMBOLS,                        ! Indicator if image has symbols
;  153          2408  1          PAT$CP_OUT_STR: REF VECTOR[, BYTE],    ! Points into current output buffer
;  154          2409  1          PAT$GB_MOD_PTR: REF VECTOR[, BYTE],    ! Points to current I/O mode settings
;  155          2410  1          PAT$GL_BUF_SIZ,                        ! Holds current character count in output bu
;  156          2411  1          PAT$GL_CSP_PTR  : ref pathname_vector, ! The current scope position (CSP)
;  157          2412  1          PAT$GL_LAST_VAL,                       ! The last value displayed
;  158          2413  1          PAT$GL_MC_PTR   : REF MC_RECORD,       ! Pointer to the module chain (MC).
;  159          2414  1          PAT$GL_NEXT_LOC;                       ! Next location to display
```

```
;  161          2415  1 GLOBAL ROUTINE PAT$FAO_PUT( STRING, ARGUMENTS ) : NOVALUE =
;  162          2416  1
;  163          2417  1 !++
;  164          2418  1 ! FUNCTIONAL DESCRIPTION:
;  165          2419  1 !
;  166          2420  1 !       Do just what $FAO does, only here we work in co-operation with a
;  167          2421  1 !       global character buffer into which we are encoding arbitrary lines
;  168          2422  1 !       of output.
;  169          2423  1 !
;  170          2424  1 !       All console output done within PATCH should use this routine to build
;  171          2425  1 !       output lines.  The only other I/O call which PATCH should be making
;  172          2426  1 !       (for the console) is to PAT$OUT_PUT, which simply says "put out whatever
;  173          2427  1 !       PAT$FAL_PUT built".
;  174          2428  1 !
;  175          2429  1 ! CALLING SEQUENCE:
;  176          2430  1 !
;  177          2431  1 !       PAT$FAO_PUT ();
;  178          2432  1 !
;  179          2433  1 ! INPUTS:
;  180          2434  1 !
;  181          2435  1 !       STRING    - A counted string which contains the directives for $FAO.
;  182          2436  1 !       ARGUMENTS - The arguments for $FAO.
;  183          2437  1 !
;  184          2438  1 ! IMPLICIT INPUTS:
;  185          2439  1 !
;  186          2440  1 !       PAT$CP_OUT_STR  - Pointer to where we are in the current output buffer.
;  187          2441  1 !       PAT$GL_BUF_SIZ  - Count of characters in output buffer.
;  188          2442  1 !
;  189          2443  1 ! OUTPUTS:
;  190          2444  1 !
;  191          2445  1 !       The $FAO output is put into the output buffer.
;  192          2446  1 !
;  193          2447  1 ! IMPLICIT OUTPUTS:
;  194          2448  1 !
;  195          2449  1 !       The global character pointer is incremented so that it points
;  196          2450  1 !       (as always) to the next available place in the output buffer.
;  197          2451  1 !       The buffer count variable is incremented by the size of this string.
;  198          2452  1 !
;  199          2453  1 ! ROUTINE VALUE:
;  200          2454  1 !
;  201          2455  1 !       none
;  202          2456  1 !
;  203          2457  1 ! SIDE EFFECTS:
;  204          2458  1 !
;  205          2459  1 !       none
;  206          2460  1 !--
;  207          2461  1
;  208          2462  2 BEGIN
;  209          2463  2 MAP
;  210          2464  2         STRING : REF VECTOR[,BYTE];
;  211          2465  2 LOCAL
;  212          2466  2         INP_DESC : VECTOR[2],                   ! Input desc for $FAO
;  213          2467  2         OUT_DESC : VECTOR[2],                   ! Output desc for $FAO
;  214          2468  2         STR_SIZE : WORD;                        ! $FAO returns output size here
;  215          2469  2
;  216          2470  2 !++
;  217          2471  2 ! Build the descriptors that $FAO wants, ask it to do the encoding.
```

```
;  218          2472  2 ! copying the output into our global output buffer, and finally update the
;  219          2473  2 ! global pointer to the next free character position in the buffer.
;  220          2474  2 !
;  221          2475  2 INP_DESC[0] = .STRING[0];
;  222          2476  2 INP_DESC[1] = STRING[1];
;  223          2477  2 OUT_DESC[0] = TTY_OUT_WIDTH - 1 - .PAT$GL_BUF_SIZ;
;  224          2478  2 OUT_DESC[1] = .PAT$CP_OUT_STR;
;  225          2479  2 SYS$FAOL( INP_DESC, STR_SIZE, OUT_DESC, ARGUMENTS );
;  226          2480  2 PAT$CP_OUT_STR = .PAT$CP_OUT_STR + .STR_SIZE;
;  227          2481  2 PAT$GL_BUF_SIZ = .PAT$GL_BUF_SIZ + .STR_SIZE;
;  228          2482  1 END;


                                                        .TITLE  PATSSV
                                                        .IDENT  \V04-000\

                                                ISE$C_SIZE==        20
                                                TXT$C_SIZE==        4
                                                PAL$C_SIZE==        16
                                                ASD$C_SIZE==        9
                                                FWR$C_SIZE==        24
                                                        .EXTRN  PAT$FAO_OUT, PAT$FIND_VAL
                                                        .EXTRN  PAT$GET_BOUNDS, PAT$GET_VALUE
                                                        .EXTRN  PAT$ADD_NT_T_PV
                                                        .EXTRN  PAT$REPORT_FREE
                                                        .EXTRN  PAT$VAL_TO_SYM, PAT$SYMBOL_VALU
                                                        .EXTRN  SYS$FAOL, PAT$GL_SYMTBPTR
                                                        .EXTRN  PAT$GL_OLDLABLS
                                                        .EXTRN  PAT$GL_SYMHEAD, PAT$GB_SYMBOLS
                                                        .EXTRN  PAT$CP_OUT_STR, PAT$GB_MOD_PTR
                                                        .EXTRN  PAT$GL_BUF_SIZ, PAT$GL_CSP_PTR
                                                        .EXTRN  PAT$GL_LAST_VAL
                                                        .EXTRN  PAT$GL_MC_PTR, PAT$GL_NEXT_LOC
                                                        .WEAK   ACCESS_CHECK

                                                        .PSECT  _PAT$CODE,NOWRT,2

                                        000C 00000      .ENTRY  PAT$FAO_PUT, Save R2,R3                 ; 2415
                            53 00000000G   EF   9E 00002 MOVAB   PAT$GL_BUF_SIZ, R3                      ;
                            52 00000000G   EF   9E 00009 MOVAB   PAT$CP_OUT_STR, R2                      ;
                            5E             14   C2 00010 SUBL 2  #20, SP                                ;
                        0C  AE      04     BC   9A 00013 MOVZBL @STRING, INP_DESC                       ; 2475
                    10  AE      04 AC      01   C1 00018 ADDL3  #1, STRING, INP_DESC+4                  ; 2476
                    04  AE 00000083 8F      63   C3 0001E SUBL3  PAT$GL_BUF_SIZ, #131, OUT_DESC         ; 2477
                        08  AE           62   D0 00027 MOVL   PAT$CP_OUT_STR, OUT_DESC+4               ; 2478
                                    08 AC  9F 0002B PUSHAB  ARGUMENTS                                    ; 2479
                                    08 AE  9F 0002E PUSHAB  OUT_DESC                                     ;
                                    08 AE  9F 00031 PUSHAB  STR_SIZE                                     ;
                                    18 AE  9F 00034 PUSHAB  INP_DESC                                     ;
                    00000000G   9F      04   FB 00037 CALLS  #4, @#SYS$FAOL                           ;
                            50          6E   3C 0003E MOVZWL STR_SIZE, R0                            ; 2480
                            62          50   C0 00041 ADDL2  R0, PAT$CP_OUT_STR                       ;
                            50          6E   3C 00044 MOVZWL STR_SIZE, R0                            ; 2481
                            63          50   C0 00047 ADDL2  R0, PAT$GL_BUF_SIZ                      ;
                                           04 0004A     RET                                          ; 2482

; Routine Size:  75 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
;  230          2483  1 GLOBAL ROUTINE PAT$MODULE_SIZE( MC_PTR ) =
;  231          2484  1
;  232          2485  1 !++
;  233          2486  1 ! Functional Description:
;  234          2487  1 !
;  235          2488  1 !       Given an MC pointer to a given module, estimate how many bytes of RST
;  236          2489  1 !       storage would be required to SET (initialize) this module into the RST.
;  237          2490  1 !
;  238          2491  1 ! Calling Sequence:
;  239          2492  1 !
;  240          2493  1 !       The routine is simply called to pick up the returned value.  The reason
;  241          2494  1 !       why it is global is because it is called in PAT$SHOW_MODULE, of
;  242          2495  1 !       PATSSV.B32, and in ADD_MODULE, of PATBLD.B32.
;  243          2496  1 !
;  244          2497  1 ! Formal Parameters:
;  245          2498  1 !
;  246          2499  1 !       MC_PTR  -An MC pointer which tells us all we need
;  247          2500  1 !                to know about the indicated module.
;  248          2501  1 !
;  249          2502  1 ! Implicit Inputs:
;  250          2503  1 !
;  251          2504  1 !       The RST storage we consider is independent of GLOBAL
;  252          2505  1 !       considerations.  This means that in all likelihood we overestimate.
;  253          2506  1 !       (We guarantee NOT to underestimate in any case.)
;  254          2507  1 !
;  255          2508  1 !       How the storage manager works, and how we
;  256          2509  1 !       use it to allocate all RST storage.
;  257          2510  1 !
;  258          2511  1 ! Implicit Outputs:
;  259          2512  1 !
;  260          2513  1 !       none.
;  261          2514  1 !
;  262          2515  1 ! Return Value:
;  263          2516  1 !
;  264          2517  1 !       The number of bytes required to add the module
;  265          2518  1 !       to the Runtime Symbol Table data structures.
;  266          2519  1 !
;  267          2520  1 !--
;  268          2521  1
;  269          2522  2 BEGIN
;  270          2523  2
;  271          2524  2 MAP
;  272          2525  2         MC_PTR : REF MC_RECORD;
;  273          2526  2
;  274          2527  2 LOCAL
;  275          2528  2         TOTAL_BYTES;
;  276          2529  2
;  277          2530  2 !++
;  278          2531  2 ! Macro to calculate the total number of bytes
;  279          2532  2 ! taken for so-called vector storage (which
;  280          2533  2 ! is how we allocate NTs, LVTs, and SATs).
;  281          2534  2 !--
;  282          2535  2 MACRO
;  283          2536  2
;  284        M 2537  2         VEC_STORAGE_FOR( RECORDS, SIZE ) = (%UPVAL +              ! The storage manager's overhead,
;  285          2538  2                                             ( (RST_UNITS( RECORDS * SIZE )) * %UPVAL ))%; ! Rounded up vector st
;  286          2539  2
```

```
;  287          2540  2 !++
;  288          2541  2 ! Tally up the total number of bytes needed to add the NTs, LVTs, and
;  289          2542  2 ! SATs for the indicated module.  First initialize counter.
;  290          2543  2 !--
;  291          2544  2 TOTAL_BYTES = 0;
;  292          2545  2
;  293          2546  2 !++
;  294          2547  2 ! Although it is not true, we must assume that NT entries are fixed-size.
;  295          2548  2 !--
;  296          2549  2 TOTAL_BYTES = VEC_STORAGE_FOR( .MC_PTR [MC_NAMES], RST_NT_SIZE );
;  297          2550  2
;  298          2551  2 !++
;  299          2552  2 ! Add on that taken for SATs and LVTs.
;  300          2553  2 !--
;  301          2554  2 TOTAL_BYTES = .TOTAL_BYTES
;  302          2555  2               + VEC_STORAGE_FOR( .MC_PTR [MC_STATICS], RST_SAT_SIZE )
;  303          2556  2               + VEC_STORAGE_FOR( .MC_PTR [MC_LITERALS], RST_LVT_SIZE );
;  304          2557  2
;  305          2558  2 !++
;  306          2559  2 ! Return the size.
;  307          2560  2 !--
;  308          2561  2 RETURN(.TOTAL_BYTES);
;  309          2562  1 END;


                                                        .EXTRN  PAT$GL_RST_BEGN

                                                 000C 00000     .ENTRY  PAT$MODULE_SIZE, Save R2,R3                     ; 2483
                                              50   D4 00002     CLRL    TOTAL_BYTES                                     ; 2544
                52      04    AC 00000000G    EF   C1 00004     ADDL3   PAT$GL_RST_BEGN, MC_PTR, R2                     ; 2549
                51      08    A2              2C   C5 0000D     MULL3   #44, 8(R2), R1                                  ;
                              51              03   C0 00012     ADDL2   #3, R1                                          ;
                              51              04   C6 00015     DIVL2   #4, R1                                          ;
                50            51              02   78 00018     ASHL    #2, R1, TOTAL_BYTES                             ;
                              50              04   C0 0001C     ADDL2   #4, TOTAL_BYTES                                 ;
                51      31    A2              0A   C5 0001F     MULL3   #10, 49(R2), R1                                 ; 2555
                              51              03   C0 00024     ADDL2   #3, R1                                          ;
                              51              04   C6 00027     DIVL2   #4, R1                                          ;
                              53          6041   DE 0002A     MOVAL   (TOTAL_BYTES)[R1], R3                           ; 2556
                52      35    A2              06   C5 0002E     MULL3   #6, 53(R2), R2                                  ;
                              51      03      A2   9E 00033     MOVAB   3(R2), R1                                       ;
                              51              04   C6 00037     DIVL2   #4, R1                                          ;
                              50      08 A341      DE 0003A     MOVAL   8(R3)[R1], TOTAL_BYTES                          ;
                                                   04 0003F     RET                                                     ; 2562

; Routine Size:  64 bytes,    Routine Base:  _PAT$CODE + 004B
```

```
;  311     2563  1 GLOBAL ROUTINE PAT$OUT_NUM_VAL ( VALUE, NEW_LENGTH, NEW_RADIX, ZERO_FILL) : NOVALUE =
;  312     2564  1
;  313     2565  1 !++
;  314     2566  1 ! Functional Description:
;  315     2567  1 !
;  316     2568  1 !       Format the given value according to the current output mode settings.
;  317     2569  1 !       The formatted value is inserted into the output buffer described by
;  318     2570  1 !       PAT$CP_OUT_STR and PAT$GL_BUF_SIZ.  If the value is formatted in
;  319     2571  1 !       hexadecimal and negative and ZERO_FILL is TRUE, then a leading zero
;  320     2572  1 !       is output.
;  321     2573  1 !
;  322     2574  1 ! Calling Sequence:
;  323     2575  1 !
;  324     2576  1 !       PAT$OUT_NUM_VAL (VALUE, NEW_LENGTH, NEW_RADIX, ZERO_FILL)
;  325     2577  1 !
;  326     2578  1 ! Inputs:
;  327     2579  1 !
;  328     2580  1 !       VALUE - The actual value we are to write out.
;  329     2581  1 !       NEW_LENGTH - Either 0, or the MODE_LENGTH we should use.
;  330     2582  1 !                    (0 => use current LENGTH, non-zero allows
;  331     2583  1 !                     it to be overridden.)
;  332     2584  1 !                    (The literal, NO_OVERRIDE = 0, is used for this.)
;  333     2585  1 !       NEW_RADIX - Either 0, or the MODE_RADIX we should use.
;  334     2586  1 !       ZERO_FILL - Indicator if leading zero should be output
;  335     2587  1 !                       (TRUE = yes, FALSE = no)
;  336     2588  1 !
;  337     2589  1 ! Implicit Inputs:
;  338     2590  1 !
;  339     2591  1 !       PAT$GB_MOD_PTR - points to the current mode data structure.
;  340     2592  1 !       PAT$CP_OUT_STR - points into the output buffer at the first
;  341     2593  1 !                        place in the buffer which is available.
;  342     2594  1 !       PAT$GL_BUF_SIZ - holds the current number of characters in
;  343     2595  1 !                        the buffer.
;  344     2596  1 !
;  345     2597  1 ! Outputs:
;  346     2598  1 !
;  347     2599  1 !       The (numeric) character representation
;  348     2600  1 !       of the value is encoded into the output buffer.
;  349     2601  1 !
;  350     2602  1 ! Implicit Outputs:
;  351     2603  1 !
;  352     2604  1 !       The buffer pointer, PAT$CP_OUT_STR, is incremented.
;  353     2605  1 !       The buffer count, PAT$GL_BUF_SIZ, is incremented.
;  354     2606  1 !
;  355     2607  1 ! Routine Value:
;  356     2608  1 !
;  357     2609  1 !       NOVALUE.
;  358     2610  1 !
;  359     2611  1 ! Side Effects.
;  360     2612  1 !
;  361     2613  1 !       Negative hex numbers may be preceded by a leading 0.
;  362     2614  1 !--
;  363     2615  1
;  364     2616  2 BEGIN
;  365     2617  2
;  366     2618  2 LOCAL
;  367     2619  2         USE_LENGTH,                                     ! The MODE_LENGTH we actually use
```

```
;  368          2620  2          USE_RADIX;                                          ! The MODE_RADIX we actually use
;  369          2621  2
;  370          2622  2 OWN
;  371          2623  2          FORMAT : VECTOR[4,BYTE]                             . Build format string here
;  372          2624  2                   INITIAL( BYTE( %ASCIC '!??' ) );
;  373          2625  2
;  374          2626  2 !++
;  375          2627  2 ! Assume that we are passed an override for the LENGTH and RADIX,
;  376          2628  2 ! but revert to the globally-set one if this is untrue.
;  377          2629  2 !--
;  378          2630  3 IF ((USE_LENGTH = .NEW_LENGTH) EQL 0)
;  379          2631  2 THEN
;  380          2632  2          USE_LENGTH = .PAT$GB_MOD_PTR  [MODE_LENGTH];
;  381          2633  3 IF ((USE_RADIX = .NEW_RADIX) EQL 0)
;  382          2634  2 THEN
;  383          2635  2          USE_RADIX = .PAT$GB_MOD_PTR  [MODE_RADIX];
;  384          2636  2
;  385          2637  2 !++
;  386          2638  2 ! Now just build the required 2-character format string descriptor,
;  387          2639  2 ! based upon the current setting of the mode length and radix.
;  388          2640  2 !
;  389          2641  2 ! We assume signed, for decimal output, and take longword hexadecimal either
;  390          2642  2 ! when that is explicitly the case, or when some unknown length or radix is indicated.
;  391          2643  2 !--
;  392          2644  3 FORMAT[2] = ( SELECTONE .USE_RADIX OF
;  393          2645  3                   SET
;  394          2646  3                   [OCTAL_RADIX]:          'O';
;  395          2647  3                   [DECIMAL_RADIX]:        'S';
;  396          2648  3                   [OTHERWISE]:            'X';
;  397          2649  3                   TES
|;  398          2650  2               );
;  399          2651  2
;  400          2652  3 FORMAT[3] = ( SELECTONE .USE_LENGTH OF
;  401          2653  3                   SET
;  402          2654  3                   [BYTE_LENGTH]:  'B';
;  403          2655  3                   [WORD_LENGTH]:  'W';
;  404          2656  3                   [OTHERWISE]:    'L';
;  405          2657  3                   TES
|;  406          2658  2               );
;  407          2659  2
;  408          2660  2 !++
;  409          2661  2 ! Now check if the output radix is hexadecimal and if a leading zero is needed.
;  410          2662  2 ! This is required to make PATCH output acceptable as input.  Hexadecimal
;  411          2663  2 ! numbers require the leading character to be a numeric.
;  412          2664  2 !--
;  413          2665  2 IF (.FORMAT[2] EQLU 'X') AND (.ZERO_FILL) AND
;  414          2666  3    (.(VALUE)<0, .USE_LENGTH*8, 1> GTRA %X'9FFFFFFF')
;  415          2667  2 THEN
;  416          2668  3          BEGIN
;  417          2669  3          PAT$CP_OUT_STR[0] = '0';
;  418          2670  3          PAT$CP_OUT_STR = .PAT$CP_OUT_STR + 1;
;  419          2671  3          PAT$GL_BUF_SIZ = .PAT$GL_BUF_SIZ + 1;
;  420          2672  2          END;
;  421          2673  2
;  422          2674  2 PAT$FAO_PUT ( FORMAT, .VALUE );
;  423          2675  2
;  424          2676  1 END;
```

```
                                                .PSECT  _PAT$OWN,NOEXE,2

     3F  3F  21  03  00000 FORMAT:              .ASCII  <3>\!??\                                ;

                                                .PSECT  _PAT$CODE,NOWRT,2

                       0004 00000               .ENTRY  PAT$OUT_NUM_VAL, Save R2                ; 2563
52 00000000'    EF   9E 00002                   MOVAB   FORMAT+2, R2                            ;
50          08  AC   D0 00009                   MOVL    NEW_LENGTH, USE_LENGTH                  ; 2630
                0B   12 0000D                   BNEQ    1$                                      ;
51 00000000G    EF   D0 0000F                   MOVL    PAT$GB_MOD_PTR, R1                      ; 2632
50          01  A1   9A 00016                   MOVZBL  1(R1), USE_LENGTH                       ;
51          0C  AC   D0 0001A 1$:               MOVL    NEW_RADIX, USE_RADIX                    ; 2633
                07   12 0001E                   BNEQ    2$                                      ;
51 00000000G    FF   9A 00020                   MOVZBL  @PAT$GB_MOD_PTR, USE_RADIX              ; 2635
08                51   D1 00027 2$:             CMPL    USE_RADIX, #8                           ; 2646
                06   12 0002A                   BNEQ    3$                                      ;
51          4F  8F   9A 0002C                   MOVZBL  #79, R1                                 ;
                0F   11 00030                   BRB     5$                                      ;
0A                51   D1 00032 3$:             CMPL    USE_RADIX, #10                          ; 2647
                06   12 00035                   BNEQ    4$                                      ;
51          53  8F   9A 00037                   MOVZBL  #83, R1                                 ;
                04   11 0003B                   BRB     5$                                      ;
51          58  8F   9A 0003D 4$:               MOVZBL  #88, R1                                 ; 2648
62                51   90 00041 5$:             MOVB    R1, FORMAT+2                            ; 2644
01                50   D1 00044                 CMPL    USE_LENGTH, #1                          ; 2654
                06   12 00047                   BNEQ    6$                                      ;
51          42  8F   9A 00049                   MOVZBL  #66, R1                                 ;
                0F   11 0004D                   BRB     8$                                      ;
02                50   D1 0004F 6$:             CMPL    USE_LENGTH, #2                          ; 2655
                06   12 00052                   BNEQ    7$                                      ;
51          57  8F   9A 00054                   MOVZBL  #87, R1                                 ;
                04   11 00058                   BRB     8$                                      ;
51          4C  8F   9A 0005A 7$:               MOVZBL  #76, R1                                 ; 2656
          01    A2   51   90 0005E 8$:           MOVB    R1, FORMAT+3                            ; 2652
          58    8F        62   91 00062         CMPB    FORMAT+2, #88                           ; 2665
                29   12 00066                   BNEQ    9$                                      ;
25          10  AC   E9 00068                   BLBC    ZERO_FILL, 9$                           ;
50                08   C4 0006C                 MULL2   #8, R0                                  ; 2666
51      04  AC        50   00   EE 0006F         EXTV    #0, R0, VALUE, R1                       ;
          9FFFFFFF    8F        51   D1 00075   CMPL    R1, #-1610612737                        ;
                13   1B 0007C                   BLEQU   9$                                      ;
          00000000G FF        30   90 0007E     MOVB    #48, @PAT$CP_OUT_STR                    ; 2669
                00000000G EF  D6 00085          INCL    PAT$CP_OUT_STR                          ; 2670
                00000000G EF  D6 0008B          INCL    PAT$GL_BUF_SIZ                          ; 2671
                      04  AC  DD 00091 9$:      PUSHL   VALUE                                   ; 2674
                      FE  A2  9F 00094          PUSHAB  FORMAT                                  ;
              FED9    CF        02   FB 00097   CALLS   #2, PAT$FAO_PUT                         ;
                                  04 0009C      RET                                             ; 2676

; Routine Size:  157 bytes,    Routine Base:  _PAT$CODE + 008B
```

```
;  426          2677  1 GLOBAL ROUTINE PAT$OUT_PUT ( BUFFER ) : NOVALUE =
;  427          2678  1
;  428          2679  1 !++
;  429          2680  1 ! Functional Description:
;  430          2681  1 !
;  431          2682  1 !       Cause the current output buffer to be actually
;  432          2683  1 !       output to the console.
;  433          2684  1 !
;  434          2685  1 ! Calling Sequence:
;  435          2686  1 !
;  436          2687  1 !       PAT$OUT_PUT ()
;  437          2688  1 !
;  438          2689  1 ! Inputs:
;  439          2690  1 !
;  440          2691  1 !       BUFFER - Pointer to the beginning of the current output
;  441          2692  1 !                buffer.  this is supposed to be a counted
;  442          2693  1 !                string except that noone has supplied the count yet.
;  443          2694  1 !                ie we expect that the actual string to be printed
;  444          2695  1 !                starts in byte BUFFER+1.
;  445          2696  1 !
;  446          2697  1 ! Implicit Inputs:
;  447          2698  1 !
;  448          2699  1 !       PAT$CP_OUT_STR - Points into this output buffer at the first
;  449          2700  1 !                        place in the buffer which is NOT to be printed.
;  450          2701  1 !       PAT$GL_BUF_SIZ - Holds count of characters in buffer
;  451          2702  1 !
;  452          2703  1 ! Outputs:
;  453          2704  1 !
;  454          2705  1 !       The string is printed, exactly as it is in the
;  455          2706  1 !       buffer - ie, it should contain whatever carriage control
;  456          2707  1 !       you want.
;  457          2708  1 !
;  458          2709  1 ! Implicit Outputs:
;  459          2710  1 !
;  460          2711  1 !       The buffer pointer, PAT$CP_OUT_STR, is set to zero.  This will
;  461          2712  1 !       help routines know if they are to use their own buffer or
;  462          2713  1 !       if one is already partially built.
;  463          2714  1 ! ***   I don't think we have ever actually used the above
;  464          2715  1 !       'feature', and feel that this routine ought, instead, set
;  465          2716  1 !       up for more PAT$FAO_PUT calls on the current buffer when we
;  466          2717  1 !       are called to output it.
;  467          2718  1 !
;  468          2719  1 ! Routine Value:
;  469          2720  1 !
;  470          2721  1 !       NOVALUE.
;  471          2722  1 !
;  472          2723  1 ! Side Effects:
;  473          2724  1 !
;  474          2725  1 !       none.
;  475          2726  1 !--
;  476          2727  1
;  477          2728  2 BEGIN
;  478          2729  2
;  479          2730  2 MAP
;  480          2731  2         BUFFER : REF VECTOR[,BYTE];                     ! Points to output buffer
;  481          2732  2
;  482          2733  2 !++
```

```
;  483          2734  2 ! Fill in the count, and pass it to QIO.
;  484          2735  2 !--
;  485          2736  2 BUFFER [0] = .PAT$GL_BUF_SIZ;
;  486          2737  2 $FAO_TT_OUT ( '!AC', .BUFFER );
;  487          2738  2 PAT$CP_OUT_STR = 0;
;  488          2739  1 END;


                                                                .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                                                    03  00000 P.AAA:   .BYTE   3
                                                43  41  21  00001       .ASCII  \!AC\                                   ;

                                                                .PSECT  _PAT$CODE,NOWRT,2

                                                        0000 00000      .ENTRY  PAT$OUT_PUT, Save nothing                ; 2677
                                    04    BC 00000000G  EF  90 00002    MOVB    PAT$GL_BUF_SIZ, @BUFFER                 ; 2736
                                                   04   AC  DD 0000A    PUSHL   BUFFER                                  ; 2737
                                               00000000'  EF  9F 0000D    PUSHAB  P.AAA                                   ;
                                00000000G      EF         02  FB 00013    CALLS   #2, PAT$FAO_OUT                         ;
                                               00000000G  EF  D4 0001A    CLRL    PAT$CP_OUT_STR                          ; 2738
                                                          04 00020      RET                                             ; 2739

; Routine Size:  33 bytes,    Routine Base:  _PAT$CODE + 0128
```

```
;  490          2740  1 GLOBAL ROUTINE PAT$OUT_SYM_VAL ( VALUE, NEW_LENGTH, NEW_RADIX ) : NOVALUE =
;  491          2741  1
;  492          2742  1 !++
;  493          2743  1 ! Functional Description:
;  494          2744  1 !
;  495          2745  1 !       Write out the given value according to the current
;  496          2746  1 !       mode settings - including the current mode setting
;  497          2747  1 !       for the [NO]SYMBOLS flag.  (By 'write out', we mean
;  498          2748  1 !       'encode into the output buffer').
;  499          2749  1 !
;  500          2750  1 ! Calling Sequence:
;  501          2751  1 !
;  502          2752  1 !       PAT$OUT_SYM_VAL ()
;  503          2753  1 !
;  504          2754  1 ! Inputs:
;  505          2755  1 !
;  506          2756  1 !       VALUE - The actual value we are to write out.
;  507          2757  1 !       NEW_LENGTH - Either 0, or the MODE_LENGTH we should use.
;  508          2758  1 !                    (0 => use current LENGTH, non-zero allows
;  509          2759  1 !                     it to be overridden.)
;  510          2760  1 !                    (The literal, NO_OVERRIDE = 0, is used for this.)
;  511          2761  1 !       NEW_RADIX - Either 0, or the MODE_RADIX we should use.
;  512          2762  1 !
;  513          2763  1 ! Implicit Inputs:
;  514          2764  1 !
;  515          2765  1 !       PAT$GB_MOD_PTR - Points to the current mode data structure.
;  516          2766  1 !       PAT$CP_OUT_STR - Points into the output buffer at the first
;  517          2767  1 !                        place in the buffer which is available.
;  518          2768  1 !       PAT$GL_BUF_SIZ - Holds the current number of characters in
;  519          2769  1 !                        the buffer.
;  520          2770  1 !
;  521          2771  1 ! Outputs:
;  522          2772  1 !
;  523          2773  1 !       The symbolic or numeric (or symbolic+residue) representation
;  524          2774  1 !       of the value is encoded into the output buffer.
;  525          2775  1 !
;  526          2776  1 ! Implicit Outputs:
;  527          2777  1 !
;  528          2778  1 !       The buffer pointer, PAT$CP_OUT_STR, is incremented.
;  529          2779  1 !       The buffer count, PAT$GL_BUF_SIZ, is incremented.
;  530          2780  1 !
;  531          2781  1 ! Routine Value:
;  532          2782  1 !
;  533          2783  1 !       NOVALUE.
;  534          2784  1 !
;  535          2785  1 ! Side Effects.
;  536          2786  1 !       The VALUE is printed out.
;  537          2787  1 !--
;  538          2788  1
;  539          2789  2 BEGIN
;  540          2790  2
;  541          2791  2 LOCAL
;  542          2792  2                 ARRAY_DESC_ADDR,                    ! The address of an array descriptor,
;  543          2793  2                                                     ! if we symbolize an array reference.
;  544          2794  2                 NT_PTR : REF NT_RECORD,             ! A pointer to the name table (NT) entry
;  545          2795  2                                                     ! whic  corresponds to the symbol which we
;  546          2796  2                                                     ! deal with if we do symbolic output.
```

```
;  547     2797  2                 REAL_VALUE,                                   ! Used to recover the actual or real value
;  548     2798  2                                                               ! of the symbol we may base our symbolic
;  549     2799  2                                                               ! output off.
;  550     2800  2                 PATH_VEC : PATHNAME_VECTOR;                   ! Symbol path vector used to contain the
;  551     2801  2                                                               ! whole pathname assocaited with the
;  552     2802  2                                                               ! symbol pointed to by NT_PTR, above.
;  553     2803  2
;  554     2804  2 !++
;  555     2805  2 ! First, see if the value is an exact match to a DEFine type symbol.  Note that
;  556     2806  2 ! these symbols have nothing to do with the so-called RST and so much be handled
;  557     2807  2 ! specially.  First look in the current label symbol table, then in the old
;  558     2808  2 ! label table, and lastly in the user-defined symbol table.
;  559     2809  2 !--
;  560     2810  3 IF (.PAT$GB_MOD_PTR[MODE_SYMBOLS])
;  561     2811  2 THEN
;  562     2812  3         BEGIN
;  563     2813  3         LOCAL
;  564     2814  3                 TEMP_SYMTB,
;  565     2815  3                 INDEX;                                                                  ; R
;  566     2816  3
;  567     2817  3         INDEX = PAT$FIND_VAL(.VALUE,TRUE);
;  568     2818  4         IF (.INDEX EQL 0)
;  569     2819  3         THEN
;  570     2820  4                 BEGIN
;  571     2821  4                 TEMP_SYMTB = .PAT$GL_SYMTBPTR;                ! Remember current label table
;  572     2822  4                 PAT$GL_SYMTBPTR = .PAT$GL_OLDLABLS;           ! Use old labels symbol table
;  573     2823  4                 INDEX = PAT$FIND_VAL(.VALUE,TRUE);
;  574     2824  5                 IF (.INDEX EQL 0)
;  575     2825  4                 THEN
;  576     2826  5                         BEGIN
;  577     2827  5                         PAT$GL_SYMTBPTR = .PAT$GL_SYMHEAD;    ! Use user-defined symbol table
;  578     2828  5                         INDEX = PAT$FIND_VAL(.VALUE,TRUE);
;  579     2829  4                         END;
;  580     2830  4                 PAT$GL_SYMTBPTR = .TEMP_SYMTB;                ! Restor current label table
;  581     2831  3                 END;
;  582     2832  4         IF (.INDEX NEQ 0)
;  583     2833  3         THEN
;  584     2834  4                 BEGIN
;  585     2835  4                 !++
;  586     2836  4                 ! Found an exact match.  Print this and return an OK status.
;  587     2837  4                 !--
;  588     2838  4                 PAT$FAO_PUT(UPLIT(%ASCIC '!AD'),.SYM_CHCOUNT(.INDEX),SYM_NAME(.INDEX));
;  589     2839  4                 RETURN;
;  590     2840  3                 END;
;  591     2841  2         END;
;  592     2842  2 !++
;  593     2843  2 ! Even though PAT$VAL_TO_SYM 'initializes' NT_PTR when we pass on its address,
;  594     2844  2 ! we must still initialize it to zero because otherwise the top half of the
;  595     2845  2 ! longword never gets cleared.  (PAT$VAL_TO_SYM thinks, and rightly so,
;  596     2846  2 ! that NT_PTRs are only 2 bytes long, the problem is that BLISS doesn't realize
;  597     2847  2 ! this when it passes .NT_PTR on as an actual parameter later on.)
;  598     2848  2 !--
;  599     2849  2 NT_PTR = 0;
;  600     2850  2 ARRAY_DESC_ADDR = 0;
;  601     2851  2
;  602     2852  2 !++
;  603     2853  2 ! First, see if we should even attempt symbolic output.
```

```
;  604          2854  2 !--
;  605          2855  3 IF (.PAT$GB_MOD_PTR [MODE_SYMBOLS]) AND (.PAT$GB_SYMBOLS)
;  606          2856  2 THEN
;  607          2857  2         !++
;  608          2858  2         ! Next, ask VAL_TO_SYM for the NT-pointer which
;  609          2859  2         ! corresponds to the symbol we should use.
;  610          2860  2         ! The 'FALSE' says that we don't want LVTs
;  611          2861  2         ! (literals) to be considered.
;  612          2862  2         !--
;  613          2863  3         IF (PAT$VAL_TO_SYM( .VALUE, NT_PTR, FALSE))
;  614          2864  2         THEN
;  615          2865  2                 !++
;  616          2866  2                 ! Then, to be able and output "symbol+offset",
;  617          2867  2                 ! we must be able to recover the actual value
;  618          2868  2                 ! of the indicated symbol.
;  619          2869  2                 !--
;  620          2870  3                 IF (PAT$SYMBOL_VALU( .NT_PTR, REAL_VALUE))
;  621          2871  2                 THEN
;  622          2872  3                         BEGIN
;  623          2873  3                         !++
;  624          2874  3                         ! See if the value picked up by PAT$SYMBOL_VALU is
;  625          2875  3                         ! really the address which is bound to the
;  626          2876  3                         ! symbol, or if it is the address of a descriptor.
;  627          2877  3                         ! Note that DST_PTR is invalid for NTs which
;  628          2878  3                         ! were created only for global symbols.
;  629          2879  3                         !--
;  630          2880  4                         IF (.NT_PTR [NT_UP_SCOPE] NEQ 0)
;  631          2881  3                         THEN
;  632          2882  4                                 BEGIN
;  633          2883  4                                 LOCAL
;  634          2884  4                                         BOUNDS : ARRAY_BNDS_DESC,
;  635          2885  4                                         DST_RECRD : REF DST_RECORD;
;  636          2886  4
;  637          2887  4                                 !++
;  638          2888  4                                 ! Pick up the DST pointer so that we can see
;  639          2889  4                                 ! if access is via descriptor.
;  640          2890  4                                 !--
;  641          2891  4                                 DST_RECRD = .NT_PTR [NT_DST_PTR];
;  642          2892  4                                 IF (.DST_RECRD [DSTR_ACCES_TYPE] EQL ACCS_DESCRIPTOR) AND
;  643          2893  6                                    ( ( (.DST_RECRD[DSTR_ACCES_BASD] EQL ACCS_BASDIR) AND
;  644          2894  5                                        (.DST_RECRD[DSTR_ACCES_BREG] EQL 15) ) OR
;  645          2895  6                                      ( (.DST_RECRD[DSTR_ACCES_BASD] EQL 0) AND
;  646          2896  5                                        (.DST_RECRD[DSTR_ACCES_BREG] EQL 0) ) )
;  647          2897  4                                 THEN
;  648          2898  5                                         BEGIN
;  649          2899  5                                         !++
;  650          2900  5                                         ! REAL_VALUE, returned above, is artually
;  651          2901  5                                         ! the address of a descriptor.  Modify this
;  652          2902  5                                         ! value to be the beginning virtual address
;  653          2903  5                                         ! of the array.
;  654          2904  5                                         !--
;  655          2905  5                                         ARRAY_DESC_ADDR = .REAL_VALUE;
;  656          2906  5                                         PAT$GET_BOUNDS( .ARRAY_DESC_ADDR, BOUNDS);
;  657          2907  5                                         REAL_VALUE = .BOUNDS [ARRAY_ADDRESS];
;  658          2908  4                                         END;
;  659          2909  3                 END;
;  660          2910  3
```

```
;  661     2911  3                !++
;  662     2912  3                ! For unbounded symbols, we don't
;  663     2913  3                ! allow "symbol+offset" unless "offset"
;  664     2914  3                ! is less than a maximum value.
;  665     2915  3                !--
;  666     2916  5                IF (((.VALUE - .REAL_VALUE) LSSA RST_MAX_OFFSET)
;  667     2917  4                   OR (.NT_PTR [NT_IS_BOUNDED]))
;  668     2918  3                THEN
;  669     2919  4                        BEGIN                                                   ; R
;  670     2920  4                        !++
;  671     2921  4                        ! We have all we need now, so just
;  672     2922  4                        ! encode the characters into the
;  673     2923  4                        ! output stream. To do this we
;  674     2924  4                        ! must first build a pathname vector
;  675     2925  4                        ! to correspond to the NT scope chain we
;  676     2926  4                        ! implicitly now have.
;  677     2927  4                        !--
;  678     2928  4                        PAT$ADD_NT_T_PV( .NT_PTR, PATH_VEC );
;  679     2929  4
;  680     2930  4                        !++
;  681     2931  4                        ! Then we simply print this vector.
;  682     2932  4                        !--
;  683     2933  4                        PAT$PRINT_PATH( PATH_VEC );
;  684     2934  4
;  685     2935  4                        !++
;  686     2936  4                        ! See if there is any residue - i.e. see if
;  687     2937  4                        ! "symbol[(...)]" is sufficient, or if we must
;  688     2938  4                        ! output "symbol[(...)]+offset".
;  689     2939  4                        !--
;  690     2940  5                        IF (.VALUE EQL .REAL_VALUE)
;  691     2941  4                        THEN
;  692     2942  4                                RETURN;                 ! No residue -> we're done.
;  693     2943  4
;  694     2944  4                        !++
;  695     2945  4                        ! The residue (offset) should
;  696     2946  4                        ! be printed in numeric form, preceeded
;  697     2947  4                        ! by a '+'.
;  698     2948  4                        !--
;  699     2949  4                        PAT$FAO_PUT(UPLIT(%ASCIC '+'));
;  700     2950  4                        VALUE = .VALUE-.REAL_VALUE;
;  701     2951  4
;  702     2952  4                        !++
;  703     2953  4                        ! We should be able to simply
;  704     2954  4                        ! drop out of this part of the
;  705     2955  4                        ! code and have this done by PAT$OUT_NUM_VAL.
;  706     2956  4                        ! Until FAO can print HEX without the leading
;  707     2957  4                        ! 0s, we must do the following...
;  708     2958  4                        !--
;  709     2959  5                        IF (.PAT$GB_MOD_PTR [MODE_RADIX] EQL HEX_RADIX)
;  710     2960  4                        THEN
;  711     2961  5                                BEGIN
;  712     2962  5                                !++
;  713     2963  5                                ! Special cludge to ignore leading 0s.
;  714     2964  5                                ! But to then put exactly 1 leading 0
;  715     2965  5                                ! when the rest begins with A-F.
;  716     2966  5                                !--
;  717     2967  5                                LOCAL
```

```
;  718          2968  5                        CLUDGE_SIZE,    ! # of digits for FAO.
;  719          2969  5                        CLUDGE_VAL;     ! First non-zero digit.
;  720          2970  5
;  721          2971  5                    !++
;  722          2972  5                    ! We must always print at least 1 digit.
;  723          2973  5                    !--
;  724          2974  5                    DECRU I FROM 7
;  725          2975  5                    DO
;  726          2976  6                        BEGIN
;  727          2977  6                        CLUDGE_SIZE = .I+1;
;  728          2978  6                        IF (CLUDGE_VAL=.VALUE<.I*4,4,0>) NEQ 0
;  729          2979  6                        THEN
;  730          2980  6                            EXITLOOP;
;  731          2981  5                        END;
;  732          2982  5
;  733          2983  5                    IF .CLUDGE_VAL GEQU %X'A'
;  734          2984  5                    THEN
;  735          2985  5                        PAT$FAO_PUT(UPLIT(%ASCIC'0'));
;  736          2986  5                    PAT$FAO_PUT( UPLIT( %ASCIC '!#XL' ),
;  737          2987  5                        .CLUDGE_SIZE,.VALUE);
;  738          2988  5                    RETURN;
;  739          2989  4                    END;
;  740          2990  3                END;
;  741          2991  2            END;
;  742          2992  2
;  743          2993  2 !++
;  744          2994  2 ! We didn't try to do symbolic output, or we failed completely at it, or
;  745          2995  2 ! it went OK but we still have a 'residue'  value to output.
;  746          2996  2 !--
;  747          2997  2 PAT$OUT_NUM_VAL( .VALUE, .NEW_LENGTH, .NEW_RADIX, TRUE);
;  748          2998  1 END;


                                                .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                                 44  41  21  03  00004 P.AAB:  .ASCII  <3>\!AD\                                    ;
                                 00  00  2B  01  00008 P.AAC:  .ASCII  <1>\+\<0><0>                                ;
                                 00  00  30  01  0000C P.AAD:  .ASCII  <1>\0\<0><0>                                ;
                             00  00  00  4C  58  23  21  04  00010 P.AAE:  .ASCII  <4>\!#XL\<0><0><0>                   ;


                                                .PSECT  _PAT$CODE,NOWRT,2

                                         03FC 00000         .ENTRY  PAT$OUT_SYM_VAL, Save R2,R3,R4,R5,R6,R7,R8,-  ; 2740
                                                                    R9                                              ;
                                 59 00000000G   EF  9E 00002        MOVAB   PAT$GL_RST_BEGN, R9                     ;
                                 58 00000000G   EF  9E 00009        MOVAB   PAT$GB_MOD_PTR, R8                      ;
                                 57 00000000G   EF  9E 00010        MOVAB   PAT$FIND_VAL, R7                        ;
                                 56     FE9C    CF  9E 00017        MOVAB   PAT$FAO_PUT, R6                         ;
                                 55 00000000'   EF  9E 0001C        MOVAB   P.AAB, R5                               ;
                                 54 00000000G   EF  9E 00023        MOVAB   PAT$GL_SYMTBPTR, R4                     ;
                                 5E             3C  C2 0002A        SUBL2   #60, SP                                 ;
                                 50             68  D0 0002D        MOVL    PAT$GB_MOD_PTR, R0                      ; 2810
                                 44       02    A0  E9 00030        BLBC    2(R0), 3$                               ;
                                               01  DD 00034        PUSHL   #1                                      ; 2817
```

```
                        04   AC  DD 00036          PUSHL   VALUE                                                          ;
67                           02  FB 00039          CALLS   #2, PAT$FIND_VAL                                             ;
                             50  D5 0003C          TSTL    INDEX                                                        ; 2818
                             28  12 0003E          BNEQ    2$                                                           ;
52                           64  D0 00040          MOVL    PAT$GL_SYMTBPTR, TEMP_SYMTB                                  ; 2821
64 00000000G                 EF  D0 00043          MOVL    PAT$GL_OLDLABLS, PAT$GL_SYMTBPTR                             ; 2822
                             01  DD 0004A          PUSHL   #1                                                           ; 2823
                        04   AC  DD 0004C          PUSHL   VALUE                                                        ;
67                           02  FB 0004F          CALLS   #2, PAT$FIND_VAL                                             ;
                             50  D5 00052          TSTL    INDEX                                                        ; 2824
                             0F  12 00054          BNEQ    1$                                                           ;
64 00000000G                 EF  D0 00056          MOVL    PAT$GL_SYMHEAD, PAT$GL_SYMTBPTR                              ; 2827
                             01  DD 0005D          PUSHL   #1                                                           ; 2828
                        04   AC  DD 0005F          PUSHL   VALUE                                                        ;
67                           02  FB 00062          CALLS   #2, PAT$FIND_VAL                                             ;
64                           52  D0 00065 1$:      MOVL    TEMP_SYMTB, PAT$GL_SYMTBPTR                                  ; 2830
                             50  D5 00068 2$:      TSTL    INDEX                                                        ; 2832
                             0C  13 0006A          BEQL    3$                                                           ;
                        0D   A0  9F 0006C          PUSHAB  13(INDEX)                                                    ; 2838
7E                      0C   A0  9A 0006F          MOVZBL  12(INDEX), -(SP)                                             ;
                             55  DD 00073          PUSHL   R5                                                           ;
                          00EF   31 00075          BRW     13$                                                          ;
                             6E  D4 00078 3$:      CLRL    NT_PTR                                                       ; 2849
                             52  D4 0007A          CLRL    ARRAY_DESC_ADDR                                              ; 2850
50                           68  D0 0007C          MOVL    PAT$GB_MOD_PTR, R0                                           ; 2855
03                      02   A0  E8 0007F          BLBS    2(R0), 5$                                                    ;
                          00E5   31 00083 4$:      BRW     14$                                                          ;
F6 00000000G                 EF  E9 00086 5$:      BLBC    PAT$GB_SYMBOLS, 4$                                           ;
                             7E  D4 0008D          CLRL    -(SP)                                                        ; 2863
                        04   AE  9F 0008F          PUSHAB  NT_PTR                                                       ;
                        04   AC  DD 00092          PUSHL   VALUE                                                        ;
                00000000G    EF  03  FB 00095      CALLS   #3, PAT$VAL_TO_SYM                                           ;
                             E4  50  E9 0009C      BLBC    R0, 4$                                                       ;
                        04   AE  9F 0009F          PUSHAB  REAL_VALUE                                                   ; 2870
                        04   AE  DD 000A2          PUSHL   NT_PTR                                                       ;
                00000000G    EF  02  FB 000A5      CALLS   #2, PAT$SYMBOL_VALU                                          ;
                             D4  50  E9 000AC      BLBC    R0, 4$                                                       ;
                50           6E  69  C1 000AF      ADDL3   PAT$GL_RST_BEGN, NT_PTR, R0                                  ; 2880
                        08   A0  B5 000B3          TSTW    8(R0)                                                        ;
                             3E  13 000B6          BEQL    8$                                                           ;
                             50  04  A0 DO 000B8    MOVL    4(R0), DST_RECRD                                             ; 2891
02          02   A0          02      00 ED 000BC    CMPZV   #0, #2, 2(DST_RECRD), #2                                     ; 2892
                             32  12 000C2          BNEQ    8$                                                           ;
02          02   A0          02      02 ED 000C4    CMPZV   #2, #2, 2(DST_RECRD), #2                                     ; 2893
                             08  12 000CA          BNEQ    6$                                                           ;
0F          02   A0          04      04 ED 000CC    CMPZV   #4, #4, 2(DST_RECRD), #15                                    ; 2894
                             0D  13 000D2          BEQL    7$                                                           ;
                             0C  02  A0 93 000D4 6$:      BITB    2(DST_RECRD), #12                                    ; 2895
                             1C  12 000D8          BNEQ    8$                                                           ;
                        F0   8F  02  A0 93 000DA    BITB    2(DST_RECRD), #240                                           ; 2896
                             15  12 000DF          BNEQ    8$                                                           ;
                             52  04  AE D0 000E1 7$:      MOVL    REAL_VALUE, ARRAY_DESC_ADDR                          ; 2905
                                 08  AE 9F 000E5      PUSHAB  BOUNDS                                                       ; 2906
                                 52  DD 000E8          PUSHL   ARRAY_DESC_ADDR                                              ;
                00000000G        EF  02 FB 000EA      CALLS   #2, PAT$GET_BOUNDS                                           ;
                        04       AE  08 AE D0 000F1    MOVL    BOUNDS, REAL_VALUE                                           ; 2907
                50      04       AC  04 AE C3 000F6 8$:      SUBL3   REAL_VALUE, VALUE, R0                                ; 2916
```

```
                    00000100   8F              50  D1 000FC          CMPL    R0, #256                                        ;
                                                09  1F 00103          BLSSU   9$                                              ;
                         50    6E              69  C1 00105          ADDL3   PAT$GL_RST_BEGN, NT_PTR, R0                     ; 2917
                                           03   A0  95 00109          TSTB    3(R0)                                           ;
                                                5D  18 0010C          BGEQ    14$                                             ;
                                           10   AE  9F 0010E 9$:      PUSHAB  PATH_VEC                                        ; 2928
                                           04   AE  DD 00111          PUSHL   NT_PTR                                          ;
                    00000000G   EF              02  FB 00114          CALLS   #2, PAT$ADD_NT_T_PV                             ;
                                           10   AE  9F 0011B          PUSHAB  PATH_VEC                                        ; 2933
                    00000000V   EF              01  FB 0011E          CALLS   #1, PAT$PRINT_PATH                              ;
                          04    AE        04   AC  D1 00125          CMPL    VALUE, REAL_VALUE                               ; 2940
                                                4D  13 0012A          BEQL    15$                                             ;
                                           04   A5  9F 0012C          PUSHAB  P.AAC                                           ; 2949
                                                66  01 FB 0012F       CALLS   #1, PAT$FAO_PUT                                 ;
                          04    AC        04   AE  C2 00132          SUBL2   REAL_VALUE, VALUE                               ; 2950
                                           10   00  B8 91 00137      CMPB    @PAT$GB_MOD_PTR, #16                            ; 2959
                                                2E  12 0013B          BNEQ    14$                                             ;
                                                50  07 D0 0013D       MOVL    #7, I                                           ; 2974
                                                52  01 A0 9E 00140 10$: MOVAB  1(R0), CLUDGE_SIZE                              ; 2977
                         51                     50  02 78 00144      ASHL    #2, I, R1                                       ; 2978
                53  04   AC                   04  51 EF 00148      EXTZV   R1, #4, VALUE, CLUDGE_VAL                       ;
                                                04  12 0014E          BNEQ    11$                                             ;
                                                50  D7 00150          DECL    I                                               ; 2974
                                                EC  11 00152          BRB     10$                                             ;
                                           0A       53  D1 00154 11$: CMPL    CLUDGE_VAL, #10                                 ; 2983
                                                06  1F 00157          BLSSU   12$                                             ;
                                           08   A5  9F 00159          PUSHAB  P.AAD                                           ; 2985
                                                66  01 FB 0015C       CALLS   #1, PAT$FAO_PUT                                 ;
                                           04   AC  DD 0015F 12$:      PUSHL   VALUE                                           ; 2987
                                                52  DD 00162          PUSHL   CLUDGE_SIZE                                     ;
                                           0C   A5  9F 00164          PUSHAB  P.AAE                                           ; 2986
                                                66  03 FB 00167 13$:   CALLS   #3, PAT$FAO_PUT                                 ;
                                                    04 0016A          RET                                                     ; 2961
                                                01  DD 0016B 14$:      PUSHL   #1                                              ; 2997
                                           7E       08  AC  7D 0016D   MOVQ    NEW_LENGTH, -(SP)                               ;
                                                08  AC  DD 00171      PUSHL   VALUE                                           ;
                              008B  C6              04  FB 00174      CALLS   #4, PAT$OUT_NUM_VAL                             ;
                                                    04 00179 15$:      RET                                                     ; 2998

; Routine Size:  378 bytes,    Routine Base:  _PAT$CODE + 0149
```

```
;  750     2999  1 GLOBAL ROUTINE PAT$SHOW_MODULE : NOVALUE =
;  751     3000  1
;  752     3001  1 !++
;  753     3002  1 ! FUNCTIONAL DESCRIPTION:
;  754     3003  1 !
;  755     3004  1 !       Show which modules from the MC are in the RST.
;  756     3005  1 !
;  757     3006  1 ! FORMAL PARAMETERS:
;  758     3007  1 !
;  759     3008  1 !       NONE.
;  760     3009  1 !
;  761     3010  1 ! IMPLICIT INPUTS:
;  762     3011  1 !
;  763     3012  1 !       PAT$GL_MC_PTR   -I s assumed to point to the beginning
;  764     3013  1 !                         of the MC.
;  765     3014  1 !
;  766     3015  1 ! IMPLICIT OUTPUTS:
;  767     3016  1 !
;  768     3017  1 !       NONE
;  769     3018  1 !
;  770     3019  1 ! COMPLETION CODES:
;  771     3020  1 !
;  772     3021  1 !       NOVALUE.
;  773     3022  1 !
;  774     3023  1 ! SIDE EFFECTS:
;  775     3024  1 !
;  776     3025  1 !       A summary of the current modules which the RST knows
;  777     3026  1 !       about is written to the console.
;  778     3027  1 !
;  779     3028  1 !--
;  780     3029  1
;  781     3030  2 BEGIN
;  782     3031  2 LOCAL
;  783     3032  2         OUTPUT_BUFFER : VECTOR [TTY_OUT_WIDTH, BYTE],           ! Buffer to build report in.
;  784     3033  2         NUM_MC_ENTRIES,                                         ! How many entries there are in the MC.
;  785     3034  2         MC_PTR : REF MC_RECORD;                                 ! Pointer to current MC record.
;  786     3035  2
;  787     3036  2 !++
;  788     3037  2 ! Set up to do standard PATCH I/O.
;  789     3038  2 !--
;  790     3039  2 PAT$CP_OUT_STR = OUTPUT_BUFFER +1;
;  791     3040  2 PAT$GL_BUF_SIZ = 0;
;  792     3041  2
;  793     3042  2 !++
;  794     3043  2 ! Initialize the pointer that we use to 'chain' thru the module chain.
;  795     3044  2 !--
;  796     3045  3 IF ((MC_PTR = .PAT$GL_MC_PTR[MC_NEXT]) EQL 0)
;  797     3046  2 THEN
;  798     3047  3         BEGIN
;  799     3048  3         SIGNAL(PAT$_NOLCL+MSG$K_INFO);
;  800     3049  3         RETURN;
;  801     3050  2         END;
;  802     3051  2
;  803     3052  2 !++
;  804     3053  2 ! Print out the title, and then loop thru the module chain simply giving the
;  805     3054  2 ! relevant information for each module which we know about.
;  806     3055  2 !--
```

```
;  807          3056  2 NUM_MC_ENTRIES = 0;
;  808          3057  2 PAT$FAO_PUT( UPLIT (%ASCIC 'module name  symbols  size!/'));
;  809          3058  2 PAT$OUT_PUT(OUTPUT_BUFFER);
;  810          3059  2 DO
;  811          3060  3         BEGIN
;  812          3061  3         !++
;  813          3062  3         ! Print out the standard information - name and
;  814          3063  3         ! whether or not it has been initialized into the RST.
;  815          3064  3         !--
;  816          3065  3         NUM_MC_ENTRIES = .NUM_MC_ENTRIES +1;
;  817          3066  3         PAT$CP_OUT_STR = OUTPUT_BUFFER +1;
;  818          3067  3         PAT$GL_BUF_SIZ = 0;
;  819          3068  3         PAT$FAO_PUT( UPLIT( %ASCIC '!15AC !3AC    ' ), MC_PTR [MC_NAME_CS],
;  820          3069  4                   (IF (.MC_PTR [MC_IN_RST]) THEN UPLIT( %ASCIC 'yes' )
;  821          3070  3                                             ELSE UPLIT( %ASCIC 'no' )));
;  822          3071  3
;  823          3072  3         !++
;  824          3073  3         ! We have to call a routine to output the size
;  825          3074  3         ! so that it comes out in the right RADIX.
;  826          3075  3         !--
;  827          3076  3         PAT$FAO_PUT(UPLIT(%ASCIC'!UL.'),PAT$MODULE_SIZE(.MC_PTR));
;  828          3077  3         PAT$OUT_PUT(OUTPUT_BUFFER);                    ! Finally output the buffer.
;  829          3078  3         END
;  830          3079  2 WHILE ((MC_PTR = .MC_PTR[MC_NEXT]) NEQ 0);
;  831          3080  2
;  832          3081  2 !++
;  833          3082  2 ! Give final summary information.
;  834          3083  2 !--
;  835          3084  2 PAT$CP_OUT_STR = OUTPUT_BUFFER +1;
;  836          3085  2 PAT$GL_BUF_SIZ = 0;
;  837          3086  2 PAT$FAO_PUT( UPLIT( %ASCIC '!/total modules:  !UL.'), .NUM_MC_ENTRIES);
;  838          3087  2 PAT$FAO_PUT( UPLIT( %ASCIC '!/remaining size: !UL.'), PAT$REPORT_FREE());
;  839          3088  2 PAT$OUT_PUT(OUTPUT_BUFFER);
;  840          3089  1 END;


                                                                .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

73 20 20 65 6D 61 6E 20 65 6C 75 64 6F 6D 1C 00018 P.AAF:   .ASCII  <28>\module name  symbols  size!/\<0>          ;
00 2F 21 65 7A 69 73 20 20 73 6C 6F 62 6D 79 00027                                                            ;
                                          00 00 00036       .ASCII  <0><0>                                    ;
00 20 20 20 43 41 33 21 20 43 41 35 31 21 0D 00038 P.AAG:   .ASCII  <13>\!15AC !3AC   \<0><0>                 ;
                                          00 00047                                                            ;
                                    73 65 79 03 00048 P.AAH:   .ASCII  <3>\yes\                               ;
                                    00 6F 6E 02 0004C P.AAI:   .ASCII  <2>\no\<0>                             ;
                              00 00 00 2E 4C 55 21 04 00050 P.AAJ:   .ASCII  <4>\!UL.\<0><0><0>               ;
65 6C 75 64 6F 6D 20 6C 61 74 6F 74 2F 21 16 00058 P.AAK:   .ASCII  <22>\!/total modules:  !UL.\<0>           ;
                  00 2E 4C 55 21 20 20 3A 73 00067                                                            ;
69 73 20 67 6E 69 6E 69 61 6D 65 72 2F 21 16 00070 P.AAL:   .ASCII  <22>\!/remaining size: !UL.\<0>           ;
                  00 2E 4C 55 21 20 3A 65 7A 0007F                                                            ;


                                                                .PSECT  _PAT$CODE,NOWRT,2

                                                01FC 00000      .ENTRY  PAT$SHOW_MODULE, Save R2,R3,R4,R5,R6,R7,R8  ; 2999
```

```
                58 00000000G   EF   9E 00002         MOVAB   PAT$CP_OUT_STR, R8                                ;
                57 00000000G   EF   9E 00009         MOVAB   PAT$GL_RST_BEGN, R7                              ;
                56 00000000G   EF   9E 00010         MOVAB   PAT$GL_BUF_SIZ, R6                               ;
                55 00000000'   EF   9E 00017         MOVAB   P.AAF, R5                                        ;
                54     FD1B    CF   9E 0001E         MOVAB   PAT$FAO_PUT, R4                                  ;
                5E     FF7C    CE   9E 00023         MOVAB   -132(SP), SP                                     ;
                68       01    AE   9E 00028         MOVAB   OUTPUT_BUFFER+1, PAT$CP_OUT_STR                  ; 3039
                               66   D4 0002C         CLRL    PAT$GL_BUF_SIZ                                   ; 3040
     50 00000000G   EF         67   C1 0002E         ADDL3   PAT$GL_RST_BEGN, PAT$GL_MC_PTR, R0               ; 3045
                    52         60   3C 00036         MOVZWL  (R0), MC_PTR                                     ;
                               0E   12 00039         BNEQ    1$                                               ;
                 006D81CB     8F   DD 0003B         PUSHL   #7176651                                         ; 3048
        00000000G   00         01   FB 00041         CALLS   #1, LIB$SIGNAL                                   ;
                               04   00048         RET                                                      ; 3047
                               53   D4 00049 1$:     CLRL    NUM_MC_ENTRIES                                   ; 3056
                               55   DD 0004B         PUSHL   R5                                               ; 3057
                    64         01   FB 0004D         CALLS   #1, PAT$FAO_PUT                                  ;
                               5E   DD 00050         PUSHL   SP                                               ; 3058
              0128  C4         01   FB 00052         CALLS   #1, PAT$OUT_PUT                                  ;
                               53   D6 00057 2$:     INCL    NUM_MC_ENTRIES                                   ; 3065
                    68       01   AE   9E 00059         MOVAB   OUTPUT_BUFFER+1, PAT$CP_OUT_STR                  ; 3066
                               66   D4 0005D         CLRL    PAT$GL_BUF_SIZ                                   ; 3067
     50                        52   C1 0005F         ADDL3   PAT$GL_RST_BEGN, MC_PTR, R0                      ; 3069
     06       03              A0   01   E1 00063         BBC     #1, 3(R0), 3$                                    ;
                    51       30   A5   9E 00068         MOVAB   P.AAH, R1                                        ;
                               04   11 0006C         BRB     4$                                               ;
                    51       34   A5   9E 0006E 3$:     MOVAB   P.AAI, R1                                        ; 3070
                               51   DD 00072 4$:     PUSHL   R1                                               ;
                             0C     A0   9F 00074         PUSHAB  12(R0)                                           ; 3068
                             20     A5   9F 00077         PUSHAB  P.AAG                                            ;
                    64               03   FB 0007A         CALLS   #3, PAT$FAO_PUT                                  ;
                               52   DD 0007D         PUSHL   MC_PTR                                           ; 3076
                      4B     A4         01   FB 0007F         CALLS   #1, PAT$MODULE_SIZE                              ;
                               50   DD 00083         PUSHL   R0                                               ;
                             38     A5   9F 00085         PUSHAB  P.AAJ                                            ;
                    64               02   FB 00088         CALLS   #2, PAT$FAO_PUT                                  ;
                               5E   DD 0008B         PUSHL   SP                                               ; 3077
              0128  C4               01   FB 0008D         CALLS   #1, PAT$OUT_PUT                                  ;
     50                        52   C1 00092         ADDL3   PAT$GL_RST_BEGN, MC_PTR, R0                      ; 3079
                               52   60   3C 00096         MOVZWL  (R0), MC_PTR                                     ;
                               BC   12 00099         BNEQ    2$                                               ;
                    68       01   AE   9E 0009B         MOVAB   OUTPUT_BUFFER+1, PAT$CP_OUT_STR                  ; 3084
                               66   D4 0009F         CLRL    PAT$GL_BUF_SIZ                                   ; 3085
                               53   DD 000A1         PUSHL   NUM_MC_ENTRIES                                   ; 3086
                             40     A5   9F 000A3         PUSHAB  P.AAK                                            ;
                    64               02   FB 000A6         CALLS   #2, PAT$FAO_PUT                                  ;
        00000000G   EF               00   FB 000A9         CALLS   #0, PAT$REPORT_FREE                              ; 3087
                               50   DD 000B0         PUSHL   R0                                               ;
                             58     A5   9F 000B2         PUSHAB  P.AAL                                            ;
                    64               02   FB 000B5         CALLS   #2, PAT$FAO_PUT                                  ;
                               5E   DD 000B8         PUSHL   SP                                               ; 3088
                      0128  C4         01   FB 000BA         CALLS   #1, PAT$OUT_PUT                                  ;
                               04   000BF         RET                                                      ; 3089

; Routine Size:  192 bytes,    Routine Base:  _PAT$CODE + 02C3
```

```
;  842          3090  1 GLOBAL ROUTINE PAT$SHOW_SCOPE( PATH_VEC_PTR ) : NOVALUE =
;  843          3091  1
;  844          3092  1 !++
;  845          3093  1 ! Functional Description:
;  846          3094  1 !
;  847          3095  1 !       Print out the current scope position vector (CSP).
;  848          3096  1 !
;  849          3097  1 ! Formal Parameters:
;  850          3098  1 !
;  851          3099  1 !       none.
;  852          3100  1 !
;  853          3101  1 ! Implicit Inputs:
;  854          3102  1 !
;  855          3103  1 !       The CSP is a PATHNAME_VECTOR.
;  856          3104  1 !
;  857          3105  1 ! Return Value:
;  858          3106  1 !
;  859          3107  1 !       NOVALUE
;  860          3108  1 !
;  861          3109  1 !--
;  862          3110  1
;  863          3111  2 BEGIN
;  864          3112  2
;  865          3113  2 LOCAL
;  866          3114  2         OUTPUT_BUFFER : VECTOR [TTY_OUT_WIDTH, BYTE];           ! Build output message here.
;  867          3115  2
;  868          3116  2 !++
;  869          3117  2 ! Set up to use a new output buffer, and encode the standard beginning of
;  870          3118  2 ! the SHOW SCOPE message into it.
;  871          3119  2 !--
;  872          3120  2 PAT$CP_OUT_STR = OUTPUT_BUFFER + 1;
;  873          3121  2 PAT$GL_BUF_SIZ = 0;
;  874          3122  2
;  875          3123  2 PAT$FAO_PUT( UPLIT( %ASCIC 'scope: ') );
;  876          3124  2
;  877          3125  2 !++
;  878          3126  2 ! The actual thing we print out depends on whether or not there is actually
;  879          3127  2 ! a CSP just now.
;  880          3128  2 !--
;  881          3129  3 IF (.PAT$GL_CSP_PTR NEQ 0)
;  882          3130  2 THEN
;  883          3131  2         PAT$PRINT_PATH( .PAT$GL_CSP_PTR )
;  884          3132  2 ELSE
;  885          3133  2         PAT$FAO_PUT( UPLIT( %ASCIC '<null>' ));
;  886          3134  2
;  887          3135  2 !++
;  888          3136  2 ! And finally force out the message.
;  889          3137  2 !--
;  890          3138  2 PAT$OUT_PUT( OUTPUT_BUFFER );
;  891          3139  1 END;


                                                        .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                    20  3A  65  70  6F  63  73  07  00088 P.AAM:  .ASCII  <7>\scope: \                                    ;
                    00  3E  6C  6C  75  6E  3C  06  00090 P.AAN:  .ASCII  <6>\<null>\<0>                                  ;
```

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

                                                0004 00000      .ENTRY  PAT$SHOW_SCOPE, Save R2                 ; 3090
                          52      FC77  CF   9E 00002          MOVAB   PAT$FAO_PUT, R2                         ;
                          5E      FF7C  CE   9E 00007          MOVAB   -132(SP), SP                            ;
                00000000G EF        01  AE   9E 0000C          MOVAB   OUTPUT_BUFFER+1, PAT$CP_OUT_STR         ; 3120
                        00000000G EF      D4 00014          CLRL    PAT$GL_BUF_SIZ                          ; 3121
                        00000000' EF      9F 0001A          PUSHAB  P.AAM                                   ; 3123
                          62              01 FB 00020          CALLS   #1, PAT$FAO_PUT                         ;
                          50 00000000G    EF D0 00023          MOVL    PAT$GL_CSP_PTR, R0                      ; 3129
                                          0B 13 0002A          BEQL    1$                                      ;
                                          50 DD 0002C          PUSHL   R0                                      ; 3131
                00000000V EF              01 FB 0002E          CALLS   #1, PAT$PRINT_PATH                      ;
                                          09 11 00035          BRB     2$                                      ;
                        00000000' EF      9F 00037 1$:         PUSHAB  P.AAN                                   ; 3133
                          62              01 FB 0003D          CALLS   #1, PAT$FAO_PUT                         ;
                                          5E DD 00040 2$:      PUSHL   SP                                      ; 3138
                    0128  C2              01 FB 00042          CALLS   #1, PAT$OUT_PUT                         ;
                                             04 00047          RET                                             ; 3139

; Routine Size:  72 bytes,    Routine Base:  _PAT$CODE + 0383
```

```
;  893          3140  1 GLOBAL ROUTINE PAT$PV_TO_CS( PATH_VEC_PTR, OUTPUT_BUF ) : NOVALUE =
;  894          3141  1
;  895          3142  1 !++
;  896          3143  1 ! Functional Description:
;  897          3144  1 !
;  898          3145  1 !       Encode a pathname vector into a given vector
;  899          3146  1 !       so that what we have built is a counted string.
;  900          3147  1 !
;  901          3148  1 ! Formal Parameters:
;  902          3149  1 !
;  903          3150  1 !       PATH_VEC_PTR    -A pointer to the pathname vector we
;  904          3151  1 !                        want encoded.
;  905          3152  1 !
;  906          3153  1 !       OUTPUT_BUF      -A pointer to where we build the counted string.
;  907          3154  1 !
;  908          3155  1 ! Implicit Inputs:
;  909          3156  1 !
;  910          3157  1 !       We will encode up to TTY_OUTPUT_WIDTH characters.
;  911          3158  1 !       The rest of the pathname vector is ignored.
;  912          3159  1 !
;  913          3160  1 ! Return Value:
;  914          3161  1 !
;  915          3162  1 !       NOVALUE.
;  916          3163  1 !
;  917          3164  1 ! Side Effects:
;  918          3165  1 !
;  919          3166  1 !       The pathname is encoded into the given buffer.
;  920          3167  1 !
;  921          3168  1 !--
;  922          3169  1
;  923          3170  2 BEGIN
;  924          3171  2
;  925          3172  2 MAP
;  926          3173  2         OUTPUT_BUF : REF VECTOR[,BYTE],
;  927          3174  2         PATH_VEC_PTR : REF PATHNAME_VECTOR;
;  928          3175  2
;  929          3176  2 LOCAL
;  930          3177  2         CHAR_COUNT,
;  931          3178  2         OUTPUT_PTR : REF VECTOR[,BYTE];
;  932          3179  2
;  933          3180  2 !++
;  934          3181  2 ! Make sure we are called with valid parameters.
;  935          3182  2 !--
;  936          3183  3 IF (.PATH_VEC_PTR EQL 0) OR (.OUTPUT_BUF EQL 0)
;  937          3184  2 THEN
;  938          3185  2         RETURN;
;  939          3186  2
;  940          3187  2 !++
;  941          3188  2 ! The size of the counted string we build starts off as 0.  The first element
;  942          3189  2 ! in the string starts at the next character.
;  943          3190  2 !--
;  944          3191  2 CHAR_COUNT = 0;
;  945          3192  2 OUTPUT_PTR = OUTPUT_BUF[1];
;  946          3193  2
;  947          3194  2 !++
;  948          3195  2 ! Just encode the characters into the indicated position in the buffer until we
;  949          3196  2 ! have exhausted the CS pointers in the pathname vector.
```

```
;  950          3197  2 !--
;  951          3198  2 INCR I FROM 0 TO MAX_PATH_SIZE
;  952          3199  2         DO
;  953          3200  3         BEGIN
;  954          3201  3         LOCAL
;  955          3202  3                 CS_PTR : CS_POINTER;                    ! Each element of the pathname vector
;  956          3203  3                                                         ! is a pointer to a counted string.
;  957          3204  3
;  958          3205  3         !++
;  959          3206  3         ! Terminate when the path vector ends.
;  960          3207  3         !--
;  961          3208  4         IF ((CS_PTR = .PATH_VEC_PTR[.I]) EQL 0)
;  962          3209  3         THEN
;  963          3210  3                 EXITLOOP;
;  964          3211  3
;  965          3212  3         !++
;  966          3213  3         ! Output the pathname element separation character in all but the first
;  967          3214  3         ! iteration.
;  968          3215  3         !--
;  969          3216  4         IF (.I NEQ 0)
;  970          3217  3         THEN
;  971          3218  4                 BEGIN
;  972          3219  4                 CHAR_COUNT = .CHAR_COUNT +1;
;  973          3220  4                 OUTPUT_PTR[0] = %C'\';
;  974          3221  4                 OUTPUT_PTR = .OUTPUT_PTR + 1;
;  975          3222  3                 END;
;  976          3223  3
;  977          3224  3         !++
;  978          3225  3         ! See if we have enough room left for the next pathname element.
;  979          3226  3         !--
;  980          3227  4         IF (NOT .CHAR_COUNT + .CS_PTR[0] LSS TTY_OUT_WIDTH)
;  981          3228  3         THEN
;  982          3229  3                 EXITLOOP;
;  983          3230  3         CH$MOVE( .CS_PTR[0], CS_PTR[1], .OUTPUT_PTR );
;  984          3231  3         CHAR_COUNT = .CHAR_COUNT +.CS_PTR[0];
;  985          3232  3         OUTPUT_PTR = .OUTPUT_PTR + .CS_PTR[0];
;  986          3233  2         END;
;  987          3234  2
;  988          3235  2 !++
;  989          3236  2 ! Now make the output buffer a real counted string.
;  990          3237  2 !--
;  991          3238  2 OUTPUT_BUF[0] = .CHAR_COUNT;
;  992          3239  1 END;


                                    03FC 00000          .ENTRY  PAT$PV_TO_CS, Save R2,R3,R4,R5,R6,R7,R8,R9 ; 3140
                              04    AC   D5 00002       TSTL    PATH_VEC_PTR                               ; 3183
                                   48   13 00005       BEQL    4$                                         ;
                              08    AC   D5 00007       TSTL    OUTPUT_BUF                                 ;
                                   43   13 0000A       BEQL    4$                                         ;
                        59    08    AC        01   C1 0000C       ADDL3   #1, OUTPUT_BUF, OUTPUT_PTR         ; 3192
                                         56   7C 00011       CLRQ    CHAR_COUNT                         ; 3191
                              58        04 BC47   D0 00013 1$:  MOVL    @PATH_VEC_PTR[I], CS_PTR           ; 3208
                                         31   13 00018       BEQL    3$                                 ;
```

```
                                        57  D5 0001A          TSTL    I                                       ; 3216
                                        06  13 0001C          BEQL    2$                                      ;
                                        56  D6 0001E          INCL    CHAR_COUNT                              ; 3219
                                  89    5C  8F  90 00020          MOVB    #92,(OUTPUT_PTR)+                       ; 3220
                                  50        68  9A 00024 2$:     MOVZBL  (CS_PTR), R0                            ; 3227
                                  50        56  C0 00027          ADDL2   CHAR_COUNT, R0                          ;
                           00000084   8F    50  D1 0002A          CMPL    R0, #132                                ;
                                        18  18 00031          BGEQ    3$                                      ;
                                  50        68  9A 00033          MOVZBL  (CS_PTR), R0                            ; 3230
                        69        01  A8    50  28 00036          MOVC3   R0, 1(CS_PTR), (OUTPUT_PTR)             ;
                                  50        68  9A 0003B          MOVZBL  (CS_PTR), R0                            ; 3231
                                  56        50  C0 0003E          ADDL2   R0, CHAR_COUNT                          ;
                                  50        68  9A 00041          MOVZBL  (CS_PTR), R0                            ; 3232
                                  59        50  C0 00044          ADDL2   R0, OUTPUT_PTR                          ;
                        C8                57  0A  F3 00047          AOBLEQ  #10, I, 1$                      ; 3198
                                  08    BC      56  90 0004B 3$:     MOVB    CHAR_COUNT, @OUTPUT_BUF         ; 3238
                                              04  0004F 4$:     RET                                             ; 3239

; Routine Size:  80 bytes,    Routine Base:  _PAT$CODE + 03CB
```

```
;  994          3240  1 GLOBAL ROUTINE PAT$PRINT_PATH( PATH_VEC_PTR ) : NOVALUE =
;  995          3241  1
;  996          3242  1 !++
;  997          3243  1 ! Functional Description:
;  998          3244  1 !
;  999          3245  1 !       Encode a pathname vector into the output stream
; 1000          3246  1 !
; 1001          3247  1 ! Formal Parameters:
; 1002          3248  1 !
; 1003          3249  1 !       PATH_VEC_PTR    -A pointer to the pathname vector we want printed.
; 1004          3250  1 !
; 1005          3251  1 ! Implicit Inputs:
; 1006          3252  1 !
; 1007          3253  1 !       Some exterior routine has done all the setup for
; 1008          3254  1 !       PATCH standard I/O.  This routine just does 'puts'.
; 1009          3255  1 !
; 1010          3256  1 ! Return Value:
; 1011          3257  1 !
; 1012          3258  1 !       NOVALUE.
; 1013          3259  1 !
; 1014          3260  1 ! Side Effects:
; 1015          3261  1 !
; 1016          3262  1 !       The pathname is encoded into the current output buffer.
; 1017          3263  1 !
; 1018          3264  1 !--
; 1019          3265  1
; 1020          3266  2 BEGIN
; 1021          3267  2 MAP
; 1022          3268  2         PATH_VEC_PTR : REF PATHNAME_VECTOR;
; 1023          3269  2
; 1024          3270  2 LOCAL
; 1025          3271  2         SYM_BUFFER : VECTOR[ TTY_OUT_WIDTH, BYTE];
; 1026          3272  2
; 1027          3273  2 !++
; 1028          3274  2 ! First encode the characters into our own buffer - because we have a routine
; 1029          3275  2 ! that already knows how to do this.
; 1030          3276  2 !--
; 1031          3277  2 PAT$PV_TO_CS( .PATH_VEC_PTR, SYM_BUFFER );
; 1032          3278  2
; 1033          3279  2 !++
; 1034          3280  2 ! Then simply output the pathname.
; 1035          3281  2 !--
; 1036          3282  2 PAT$FAO_PUT(UPLIT(%ASCIC '!AC'), SYM_BUFFER );
; 1037          3283  1 END;


                                                .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                        43  41  21  03  00098 P.AAO:  .ASCII  <3>\!AC\                          ;


                                                .PSECT  _PAT$CODE,NOWRT,2

                                0000 00000      .ENTRY  PAT$PRINT_PATH, Save nothing           ; 3240
                    5E      FF7C    CE  9E 00002        MOVAB   -132(SP), SP                   ;
```

```
                                5E  DD 00007       PUSHL   SP                              ; 3277
                            04  AC  DD 00009       PUSHL   PATH_VEC_PTR                    ;
                        A0  AF      02  FB 0000C   CALLS   #2, PAT$PV_TO_CS                ;
                                5E  DD 00010       PUSHL   SP                              ; 3282
                    00000000'   EF  9F 00012       PUSHAB  P.AAO                           ;
                    FBC8    CF      02  FB 00018   CALLS   #2, PAT$FAO_PUT                 ;
                                    04 0001D       RET                                     ; 3283

; Routine Size:  30 bytes,    Routine Base:  _PAT$CODE + 041B
```

```
; 1039          3284  1 END
; 1040          3285  0 ELUDOM



                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _PAT$CODE                      1081  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  .  ABS  .                         0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)
;  _PAT$OWN                          4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  _PAT$PLIT                       156  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(0)



;                           Library Statistics
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1         18619         4         0      1000          00:01.9



;                                       COMMAND QUALIFIERS

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

; Size:          1081 code + 160 data bytes
; Run Time:         00:35.3
; Elapsed Time:     01:56.3
; Lines/CPU Min:     5581
; Lexemes/CPU-Min: 34161
; Memory Used:  227 pages
; Compilation Complete
```

PATREB
LIS

PATSCA
LIS

PATSIO
LIS

PATRST
LIS

PATSPA
LIS

PATSSV
LIS

0304 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PHONE

PATSYM
LIS

PHONE
MAP

BASICCMDS
LIS

PATSTO
LIS

PATVEC
LIS

PHONEREQ
REQ

PATWRT
LIS

FILECMDS
LIS