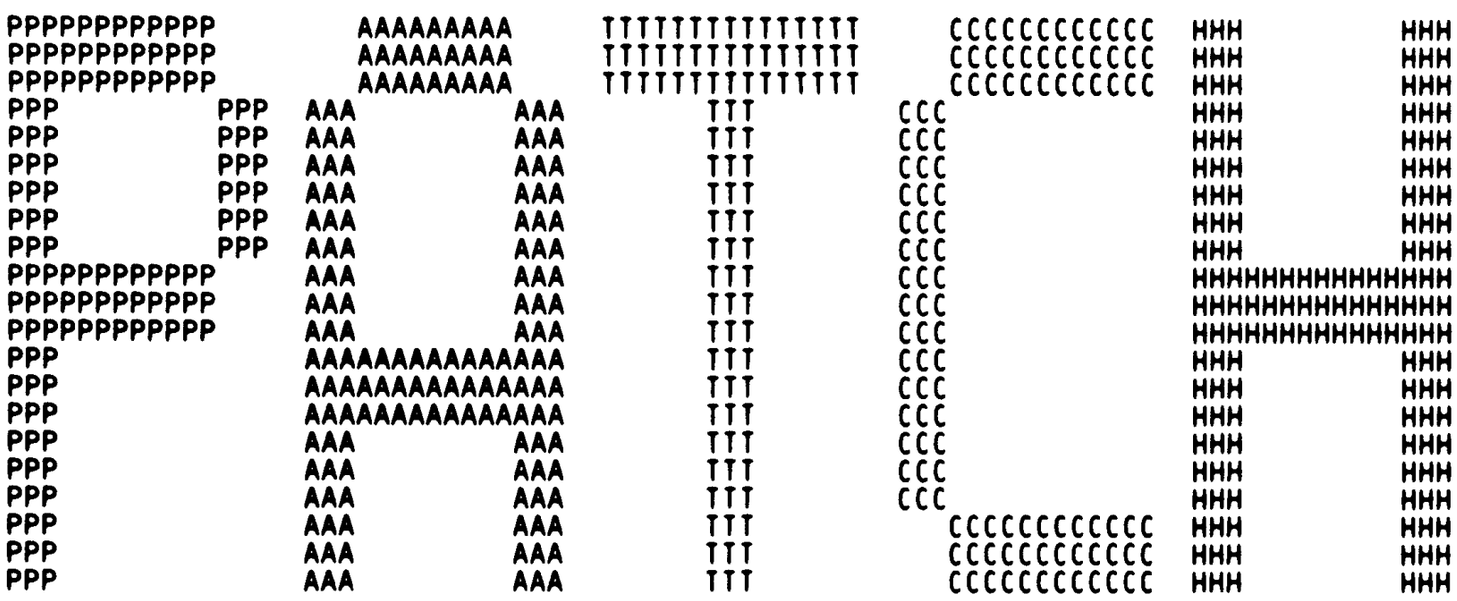
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPPPPPPPPPPP AAAAAAAAAAAAAAAA TTT CCC HHHHHHHHHHHHHHHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCCC HHH HHH

```
PPPPPPPP      AAAAAA    TTTTTTTTTT    IIIIII    NN      NN  TTTTTTTTTT
PPPPPPPP      AAAAAA    TTTTTTTTTT    IIIIII    NN      NN  TTTTTTTTTT
PP      PP   AA      AA      TT        II      NN      NN      TT
PP      PP   AA      AA      TT        II      NN      NN      TT
PP      PP   AA      AA      TT        II      NNNN    NN      TT
PP      PP   AA      AA      TT        II      NNNN    NN      TT
PPPPPPPP     AA      AA      TT        II      NN  NN  NN      TT
PPPPPPPP     AA      AA      TT        II      NN  NN  NN      TT
PP           AAAAAAAAAA      TT        II      NN    NNNN      TT
PP           AAAAAAAAAA      TT        II      NN    NNNN      TT
PP           AA      AA      TT        II      NN      NN      TT          ....
PP           AA      AA      TT        II      NN      NN      TT          ....
PP           AA      AA      TT    IIIIII      NN      NN      TT          ....
PP           AA      AA      TT    IIIIII      NN      NN      TT          ....

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 PATINT (                                                ! RST interface routines for PATCH
;     2        L 0002  0                 %IF %VARIANT EQL 1
;     3          0003  0                 %THEN
;     4          0004  0                         ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE),
;     5          0005  0                 %FI
;     6          0006  0                 IDENT = 'V04-000'
;     7          0007  0                 ) =
;     8          0008  1 BEGIN
;     9          0009  1
;    10          0010  1 !
;    11          0011  1 !*****************************************************************************
;    12          0012  1 !*                                                                           *
;    13          0013  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    14          0014  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    15          0015  1 !*  ALL RIGHTS RESERVED.                                                      *
;    16          0016  1 !*                                                                           *
;    17          0017  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    18          0018  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    19          0019  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    20          0020  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    21          0021  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    22          0022  1 !*  TRANSFERRED.                                                              *
;    23          0023  1 !*                                                                           *
;    24          0024  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    25          0025  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    26          0026  1 !*  CORPORATION.                                                              *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    29          0029  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
;    30          0030  1 !*                                                                           *
;    31          0031  1 !*                                                                           *
;    32          0032  1 !*****************************************************************************
;    33          0033  1 !
;    34          0034  1
;    35          0035  1 !++
;    36          0036  1 ! FACILITY:     PATCH
;    37          0037  1 !
;    38          0038  1 ! ABSTRACT:     This is  the RST/DST/PATCH interface module.
;    39          0039  1 !                This module exists because the DST/RST
;    40          0040  1 !                module simply declares how it wants to see
;    41          0041  1 !                the world, and leaves it up to this module
;    42          0042  1 !                to interface to PATCH to make things look
;    43          0043  1 !                that way.
;    44          0044  1 !
;    45          0045  1 !       This module defines the interface between the
;    46          0046  1 !       PATCH RST builder/manipulator and the LINKER-produced
;    47          0047  1 !       DST.  The former would like to show as little
;    48          0048  1 !       concern for where DST records are actually stored as
;    49          0049  1 !       possible.  The latter would like to provide this facility,
;    50          0050  1 !       but it must do so simply, (because we don't want to
;    51          0051  1 !       re-invent the world), efficiently, and in such
;    52          0052  1 !       as way as to allow us to do radically different
;    53          0053  1 !       things about where the DST actually exists.
;    54          0054  1 !
;    55          0055  1 !       Essentially what we do to solve this is to restrict the
;    56          0056  1 !       DST user to requesting records before he uses them,
;    57          0057  1 !       (probably) saying something about how long he wants
```

```
;    58          0058  1        to use them (or, equivalently, when he is willing to give
;    59          0059  1        them up), and using them given that they exist at the
;    60          0060  1        address he is told they are currently at.  This means that
;    61          0061  1        he can never make any assumptions about where a record is at.
;    62          0062  1        To get around this we introduce the concept of "Record Ids",
;    63          0063  1        which are simply identifiers by which the two sides of the
;    64          0064  1        interface agree to call records.  The first time you
;    65          0065  1        get a record, the interface tells you how you must
;    66          0066  1        henceforth refer to it.
;    67          0067  1
;    68          0068  1        The other aspect of the interface concerns so-called
;    69          0069  1        RST-pointers.  These pointers are used throught the
;    70          0070  1        RST module to access various (all) records.  The code
;    71          0071  1        uses these pointers implicitly, knowing nothing
;    72          0072  1        about what they actually are, and leaves it up to this
;    73          0073  1        interface to define them.  This is done by
;    74          0074  1        having a special storage allocator for the RST
;    75          0075  1        module.  It uses whatever kind of pointer this
;    76          0076  1        allocator returns, and leaves it up to
;    77          0077  1        the definition of the RST structures (RST_NT,
;    78          0078  1        RST_MC, etc.  see PATRST.REQ) to make
;    79          0079  1        sure that these RST-pointers do the job.
;    80          0080  1
;    81          0081  1
;    82          0082  1    ENVIRONMENT:  This module runs on VAX under STARLET, user mode, non-AST level.
;    83          0083  1
;    84          0084  1    AUTHOR:       Kevin Pammett, CREATION DATE: 12 JULY 77
;    85          0085  1
;    86          0086  1    MODIFIED BY:
;    87          0087  1
;    88          0088  1        V03-005 MCN0157         Maria del C. Nasr       20-Mar-1984
;    89          0089  1                Remove any references to OLDRAB since it is not used.
;    90          0090  1
;    91          0091  1        V03-004 MCN0151         Maria del C. Nasr       13-Feb-1984
;    92          0092  1                Add qualifier VOLATILE to local variable GL_SYM_COUNT to
;    93          0093  1                informational messages from the compiler.
;    94          0094  1
;    95          0095  1        V03-003 MTR0017         Mike Rhodes             15-Nov-1982
;    96          0096  1                Correct the 'next entry point' address computations for
;    97          0097  1                GSD$C_EPM and GSD$C_PRO type symbol definitions in routine
;    98          0098  1                PAT$GET_NXT_GST.
;    99          0099  1
;   100          0100  1        V03-002 MTR0012         Mike Rhodes             16-Aug-1982
;   101          0101  1                Modify file names to remove duplicate file name useage
;   102          0102  1                between code and require files.
;   103          0103  1
;   104          0104  1        V03-001 MTR0007         Mike Rhodes             14-Jun-1982
;   105          0105  1                Use shared system messages.  Affected modules include:
;   106          0106  1                DYNMEM.B32, PATBAS.B32, PATCMD.B32, PATIHD.B32, PATINT.B32,
;   107          0107  1                PATIO.B32, PATMAI.B32, PATMSG.MSG, PATWRT.B32, and PATSPA.B32.
;   108          0108  1
;   109          0109  1                The shared messages are defined by DYNMEM.B32's invocation of
;   110          0110  1                SHRMSG.REQ and we simply link against these symbols.  They are
;   111          0111  1                declared as external literals below.
;   112          0112  1
;   113          0113  1        V02-017 MTR0002         Mike Rhodes             30-Nov-1981
;   114          0114  1                Modify routine PAT$GET_NXT_GST to skip global symbol
```

```
;  115     0115  1 !                 definitions for PSECT definition in shareable images.
;  116     0116  1 !
;  117     0117  1 !           V02-016 MTR0001         Mike Rhodes             14-Oct-1981
;  118     0118  1 !                   Modify routine PAT$FIND_DST to allow the create and map
;  119     0119  1 !                   section system service to do expand region calls, instead
;  120     0120  1 !                   of trying to remember the last mapped address in P0 space.
;  121     0121  1 !                   The last mapped address array is updated within the calls.
;  122     0122  1 !
;  123     0123  1 !           V02-015 PCG0001         Peter George            02-FEB-1981
;  124     0124  1 !                   Add require statement for LIB$:PATDEF.REQ
;  125     0125  1 !
;  126     0126  1 !   NO  DATE            PROGRAMMER          PURPOSE
;  127     0127  1 !   --  ----            ----------          -------
;  128     0128  1 !
;  129     0129  1 !   00  13-DEC-77       K.D. MORSE          ADAPT VERSION 19 FOR PATCH.
;  130     0130  1 !   01  2-JAN-78        K.D. MORSE          ALLOW NO GST IN IMAGE.
;  131     0131  1 !   02  23-JAN-78       K.D. MORSE          ADD CODE FOR MORE SPECIFIC
;  132     0132  1 !                                           ERROR MESSAGES. (20)
;  133     0133  1 !   03  28-FEB-78       K.D. MORSE          SAVE_SCOPE NOW DOES A SET
;  134     0134  1 !                                           MODULE ON THE SCOPE'S MODULE.
;  135     0135  1 !   04  06-APR-78       K.D. MORSE          PAT$FIND_DST now maps the GST
;  136     0136  1 !                                           instead of reading it.(22)  Added
;  137     0137  1 !                                           routine POSITION_GST to chain
;  138     0138  1 !                                           through the mapped GST.  Also
;  139     0139  1 !                                           the logic in DBG$GET_NXT_GST
;  140     0140  1 !                                           now calls POSITION_GST. (23)
;  141     0141  1 !                                           Bug fix in FIND_DST to skip the
;  142     0142  1 !                                           first 2 GST records OK. (24)
;  143     0143  1 !                                           Bug fix in POSITION_GST - round
;  144     0144  1 !                                           up a record byte count. (24)
;  145     0145  1 !                                           GSR_NEXT_ADDR is now a REF VECTOR[,byte]. (24)
;  146     0146  1 !                                           Added code to BUILD_PATH to check
;  147     0147  1 !                                           for DEFine symbols before
;  148     0148  1 !                                           consulting the RST.  BUILD_PATH
;  149     0149  1 !                                           has the final word on whether
;  150     0150  1 !                                           a symbol has a value or not. (25)
;  151     0151  1 !                                           None for vers 26.
;  152     0152  1 !   05  25-APR-78       K.D. MORSE          CONVERT TO NATIVE COMPILER.
;  153     0153  1 !   06  17-MAY-78       K.D. MORSE          ERROR MESSGES FROM GST/DST_INIT.
;  154     0154  1 !                                           ARE NOW INFOR SEVERITY. (27)
;  155     0155  1 !                                           POSITION_GST CHECKS FOR NO GST (27).
;  156     0156  1 !                                           NO CHANGES FOR VERS 28.
;  157     0157  1 !                                           DELETE_PATH IS GLOBAL AND HAS NO
;  158     0158  1 !                                           FORMAL_INPUT AND ALWAYS ZEROS
;  159     0159  1 !                                           THE PATH_VEC_PTR. (29)
;  160     0160  1 !                                           BETTER ERROR_MSG IN SAVE_SCOPE (30).
;  161     0161  1 !   07  18-MAY-78       K.D. MORSE          CANCEL THE SCOPE IF THE MODULE
;  162     0162  1 !                                           IT POINTS TO IS CANCELLED. (31)
;  163     0163  1 !                                           POSITION_GST NOW SEES GST AS 3
;  164     0164  1 !                                           RECORDS LESS THAN HEADER SAYS
;  165     0165  1 !                                           NOT 2. (32)
;  166     0166  1 !                                           NOTE THE "ROUND UP" IN GET_NXT_GST
;  167     0167  1 !                                           TO RECOGNIZE END OF GST RECORD. (32)
;  168     0168  1 !                                           NO CHANGES FOR VERS 33.
;  169     0169  1 !   08  24-MAY-78       K.D. MORSE          ADD GSD TYPE 3 - PROCEDURE
;  170     0170  1 !                                           DEFINITION WITH FORMAL ARGUMENT
;  171     0171  1 !                                           DESCRIPTIONS.
```

```
;  172          0172  1 !  09   25-MAY-78        K.D. MORSE          ADD SIGNAL FLAG PARAMETER TO
;  173          0173  1 !                                           PAT$BUILD_PATH FOR FORWARD
;  174          0174  1 !                                           REFERENCED SYMBOLS.
;  175          0175  1 !  10   13-JUN-78        K.D. MORSE          ADD FAO COUNT TO SIGNALS.
;  176          0176  1 !  11   20-JUN-78        K.D. MORSE          NO CHANGES FOR VERS 34-36.
;  177          0177  1 !  12   28-JUN-78        K.D. MORSE          NO CHANGES FOR 37-38.
;  178          0178  1 !                                           PAT$FIND_MODULE HAS NEW ARG
;  179          0179  1 !                                           INDICATING WHETHER OR NOT TO
;  180          0180  1 !                                           SIGNAL IF MODULE IS NOT FOUND (39).
;  181          0181  1 !  13   29-JUN-78        K.D. MORSE          NO CHANGES FOR VERS 40.
;  182          0182  1 !  14   07-JUL-78        K.D. MORSE          NO CHANGES FOR VERS 41.
;  183          0183  1 !
;  184          0184  1 !--
```

```
;  186     0185  1 !
;  187     0186  1 ! TABLE OF CONTENTS:
;  188     0187  1 !
;  189     0188  1
;  190     0189  1 FORWARD ROUTINE
;  191     0190  1         PAT$SAVE_SCOPE,                         ! Store away the current scope position
;  192     0191  1                                                 ! (CSP) vector.
;  193     0192  1         PAT$BUILD_PATH,                         ! Collect symbol pathnames and eventually
;  194     0193  1                                                 !  try to evaluate them.
;  195     0194  1         PAT$DELETE_PATH : NOVALUE,              ! Throw away a pathvector.
;  196     0195  1         PAT$FIND_MODULE,                        ! Scan MC for a given module name.
;  197     0196  1         PAT$RST_FREEZ,                          ! Storage allocator for anything which
;  198     0197  1                                                 !  which is accessed via RST-pointers.
;  199     0198  1         PAT$RST_RELEASE : NOVALUE,              ! Storage deallocator for anything which
;  200     0199  1                                                 !  which is allocated by PAT$RST_FREEZ.
;  201     0200  1         PAT$FIND_DST : NOVALUE,                 ! Find the DST and make it available.
;  202     0201  1         PAT$GET_DST_REC,                        ! Make a certain DST record available.
;  203     0202  1         PAT$POSITON_DST,                        ! Make a certain DST record available
;  204     0203  1                                                 ! and set up for PAT$GET_NXT_DST
;  205     0204  1         POSITION_GST,                           ! Make a certain GST record available
;  206     0205  1         PAT$GET_NXT_DST,                        ! Make the next DST record available.
;  207     0206  1         PAT$GET_NXT_GST;                        ! Make the next GST record available
;  208     0207  1
;  209     0208  1
;  210     0209  1 !
;  211     0210  1 ! INCLUDE FILES:
;  212     0211  1 !
;  213     0212  1
;  214     0213  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;  215     0214  1 REQUIRE 'SRC$:PATPCT.REQ';
;  216     0254  1 REQUIRE 'LIB$:PATDEF.REQ';                      ! Defines literals
;  217     0308  1 REQUIRE 'LIB$:PATMSG.REQ';
;  218     0482  1 REQUIRE 'SRC$:IMGDEF.REQ';
;  219     0549  1 REQUIRE 'SRC$:PATGEN.REQ';
;  220     0771  1 REQUIRE 'SRC$:BSTRUC.REQ';
;  221     0847  1 REQUIRE 'SRC$:LISTEL.REQ';
;  222     0889  1 REQUIRE 'SRC$:DLLNAM.REQ';
;  223     0947  1 REQUIRE 'SRC$:PATRTS.REQ';
;  224     2043  1 REQUIRE 'SRC$:VXSMAC.REQ';
;  225     2108  1 REQUIRE 'SRC$:SYSSER.REQ';
```

```
;  R2140  1     SWITCHES LIST (SOURCE);
;  R2141  1
;  R2142  1     EXTERNAL ROUTINE
;  R2143  1              PAT$fao_out:            ! formats a line and outputs to the terminal
;  R2144  1
```

```
;  226     2190  1 REQUIRE 'SRC$:SYSLIT.REQ';
;  227     2240  1 REQUIRE 'SRC$:PREFIX.REQ';
;  228     2428  1 REQUIRE 'SRC$:PATPRE.REQ';
;  229     2591  1
;  230     2592  1 !
;  231     2593  1 ! MACROS:
;  232     2594  1 !
;  233     2595  1
;  234     2596  1 !
;  235     2597  1 ! EQUATED SYMBOLS·
;  236     2598  1 !
;  237     2599  1 !
;  238     2600  1 ! OWN STORAGE:
;  239     2601  1 !
;  240     2602  1
;  241     2603  1 OWN
;  242     2604  1         PATH_VEC_PTR : REF PATHNAME_VECTOR INITIAL( 0 ),        ! Pointer to the pathname vector we are
;  243     2605  1                                                                 ! currently building.  If 0, no such vector
;  244     2606  1                                                                 ! is under construction.
;  245     2607  1         DST_BEGIN_ADDR,                                        ! virtual address where DST begins.
;  246     2608  1                                                                 ! 0 => no DST.  Initially we do not
;  247     2609  1                                                                 ! want to assume this.
;  248     2610  1         DST_END_ADDR,                                          ! Virtual address of last byte in DST.
;  249     2611  1         DST_NEXT_ADDR,                                         ! Virtual address where 'next' DST record be
;  250     2612  1
;  251     2613  1         !++
;  252     2614  1         ! Now GST symbols corresponding to the above DST symbols.
;  253     2615  1         !--
;  254     2616  1         GSR_BEGIN_ADDR,                                        ! Virtual address where GST begins (0=no GST
;  255     2617  1         GSR_NEXT_ADDR : REF VECTOR[,WORD],                     ! Virtual address where 'next' GST record be
;  256     2618  1         GST_BEGIN_ADDR : REF GST_RECORD,                       ! Virtual address of current GST record (use
;  257     2619  1         GSD_REC_COUNT;                                         ! Count-down of GSD records.
;  258     2620  1
;  259     2621  1 !
;  260     2622  1 ! EXTERNAL REFERENCES:
;  261     2623  1 !
;  262     2624  1 EXTERNAL ROUTINE
;  263     2625  1         PAT$PV_TO_CS,                                          ! Encode pathname vectors for printing.
;  264     2626  1         PAT$FIND_SYM,                                          ! Lookup DEFine symbols
;  265     2627  1         PAT$SET_MODULE : NOVALUE,                              ! Adds module to the RST
;  266     2628  1         PAT$SYM_TO_VAL,                                        ! Correspong pathnames and values.
;  267     2629  1         PAT$SYM_TO_VALU,                                       ! Sym_to_val + goodies.
;  268     2630  1         PAT$INIT_RST : NOVALUE,                                ! Build all RST data structures.
;  269     2631  1         PAT$FREEZ,                                             ! Standard PATCH storage allocator.
;  270     2632  1         PAT$FREERELEASE : NOVALUE,                             ! Standard PATCH storage deallocator.
;  271     2633  1         LIB$_CREMAPSEC;                                        ! Creates and maps a global section
;  272     2634  1
;  273     2635  1 EXTERNAL
;  274     2636  1         PAT$GB_SYMBOLS,                                        ! Indicator if image contains symbols
;  275     2637  1         PAT$GL_IMGHDR : REF BLOCK[,BYTE],                      ! Pointer to image header
;  276     2638  1         PAT$GL_OLDNBK : BLOCK[,BYTE],                          ! Name block for input image file
;  277     2639  1         PAT$GB_OLDNAME,                                        ! Ascii name of input image file
;  278     2640  1         PAT$GL_ISVADDR : VECTOR[,LONG],                        ! Last pair of virtual addresses used
;  279     2641  1         PAT$GL_CSP_PTR : REF PATHNAME_VECTOR,                  ! The Current Scope Position (CSP)
;  280     2642  1                                                                 ! is defined by a pointer to the
;  281     2643  1                                                                 ! pathname vector which is the CSP.
;  282     2644  1         PAT$GL_MC_PTR : REF MC_RECORD,                         ! The module chain
```

```
;  283          2645  1          PAT$GL_RST_BEGN,                    ! Address of start of RST
;  284          2646  1          PAT$GL_HEAD_LST,                    ! Head of PATCH argument list
;  285          2647  1          PAT$GL_SYMTBPTR,                    ! Pointer to current symbol table
;  286          2648  1          PAT$GL_SYMHEAD;                     ! Pointer to user-defined symbol table l'sth
;  287          2649  1
;  288          2650  1  EXTERNAL LITERAL
;  289          2651  1  !
;  290          2652  1  ! Define shared message references.  (resolved @ link time)
;  291          2653  1  !
;  292          2654  1          PAT$_CLOSEIN,                       ! Error closing input file.
;  293          2655  1          PAT$_CLOSEOUT,                      ! Error closing output file.
;  294          2656  1          PAT$_OPENIN,                        ! Error opening input file.
;  295          2657  1          PAT$_OPENOUT,                       ! Error opening output file.
;  296          2658  1          PAT$_READERR,                       ! Error reading from file.
;  297          2659  1          PAT$_SYSERROR,                      ! System Service error.
;  298          2660  1          PAT$_WRITEERR;                      ! Error writing to file.
;  299          2661  1
;  300          2662  1
```

```
;  302     2663  1 GLOBAL ROUTINE PAT$BUILD_PATH( SYMBOL_DESC, PASS_BACK_VALUE, SIGNAL_FLAG ) =
;  303     2664  1
;  304     2665  1 !++
;  305     2666  1 ! Functional Description:
;  306     2667  1 !
;  307     2668  1 !    This routine serves two fairly distinct purposes.
;  308     2669  1 !
;  309     2670  1 ! 1. If SYMBOL_DESC is a valid string descriptor, (ie not = 0),
;  310     2671  1 !    then the call was made to BUILD_PATH so that it could
;  311     2672  1 !    accumulate the elements of a pathname in order to
;  312     2673  1 !    build up a pathname vector.
;  313     2674  1 !
;  314     2675  1 ! 2. Otherwise, the 0 SYMBOL_DESC is a flag which signals that
;  315     2676  1 !    no more elements are to come and what we have accumulated
;  316     2677  1 !    is supposedly a complete pathname.  What we are to do then
;  317     2678  1 !    is to simply look up this pathname in the RST data base and
;  318     2679  1 !    return the corresponding value via the PASS_BACK_VALUE pointer.
;  319     2680  1 !
;  320     2681  1 !       When a lookup is done, the following priority is observed:
;  321     2682  1 !
;  322     2683  1 !       1) a pathname consisting of 1 element may first be:
;  323     2684  1 !             1) a permanent symbol name (e.g. "R0")
;  324     2685  1 !             2) a DEFine symbol
;  325     2686  1 !       2) if 1), above, is not the case, or if the pathname
;  326     2687  1 !          is longer than 1 element, then the symbol must
;  327     2688  1 !          be found in the RST or an error occurs.
;  328     2689  1 !
;  329     2690  1 ! Calling Sequence:
;  330     2691  1 !
;  331     2692  1 !       PAT$BUILD_PATH ( SYMBOL_DESC, PASS_BACK_VALUE, SIGNAL_FLAG)
;  332     2693  1 !
;  333     2694  1 ! Inputs:
;  334     2695  1 !
;  335     2696  1 !       SYMBOL_DESC     - String descriptor for next peice of pathname or
;  336     2697  1 !                         zero indicating accumulated pathname is to be
;  337     2698  1 !                         evaluated.
;  338     2699  1 !       PASS_BACK_VALUE - Address of where to return the symbol's value
;  339     2700  1 !       SIGNAL_FLAG     - Flag indicating whether to signal error message
;  340     2701  1 !                         if symbol is undefined. (TRUE=yes, FALSE=no)
;  341     2702  1 !
;  342     2703  1 ! Implicit Inputs:
;  343     2704  1 !
;  344     2705  1 !       This routine works from the OWN that is local to this
;  345     2706  1 !       module, PATH_VEC_PTR, which points to the current pathname vector
;  346     2707  1 !       we are building.  The reason why this is not local to BUILD_PATH
;  347     2708  1 !       is so that it can be shared by SAVE_SCOPE.
;  348     2709  1 !
;  349     2710  1 ! Return Value:
;  350     2711  1 !
;  351     2712  1 !       On pathname accumulation, we return TRUE unless some error
;  352     2713  1 !       like PATCH running out of free storage occurs; then an error is SIGNALed.
;  353     2714  1 !
;  354     2715  1 !       On symbol evaluation, we return TRUE if the symbol was found
;  355     2716  1 !       in the image symbol tables and PAT$K_USER_DEF if the symbol was found
;  356     2717  1 !       in the user-defined symbol table.  If the symbol is undefined,
;  357     2718  1 !       then depending upon SIGNAL_FLAG either an error message is SIGNALed
;  358     2719  1 !       and an UNWIND is done, or PAT$BUILD_PATH returns FALSE.  This is to
```

```
;  359     2720  1 !       handle forward references inside symbolic instructions.
;  360     2721  1 !--
;  361     2722  1
;  362     2723  2 BEGIN
;  363     2724  2
;  364     2725  2 MAP
;  365     2726  2         SYMBOL_DESC : REF BLOCK[,BYTE],                 ! This describes the element of the
;  366     2727  2                                                         ! pathname which we are to add or
;  367     2728  2                                                         ! to our list.
;  368     2729  2         PASS_BACK_VALUE : REF VECTOR[,LONG];            ! This is where we are to pass back
;  369     2730  2                                                         ! the pathname value to.
;  370     27 1  2
;  371     2732  2 OWN
;  372     2733  2         PV_INDEX;                                       ! We use an OWN index into the OWN
;  373     2734  2                                                         ! pathname vector so that on each call
;  374     2735  2                                                         ! we know where we're at.
;  375     2736  2
;  376     2737  2 LOCAL
;  377     2738  2         CS_PTR : CS_POINTER,                            ! Temp counted string pointer.
;  378     2739  2         STATUS;                                         ! Success/failure indication that we return.
;  379     2740  2
;  380     2741  2 !++
;  381     2742  2 ! Now see whether a pathname translation to symbolic value
;  382     2743  2 ! is to occur. This is signaled by the flag SYMBOL_DESC being
;  383     2744  2 ! equal to 0.
;  384     2745  2 !--
;  385     2746  3 IF (.SYMBOL_DESC EQL 0)
;  386     2747  2 THEN
;  387     2748  3         BEGIN
;  388     2749  3         !++
;  389     2750  3         ! Evaluate the symbol.  First, for single-element pathnames we give
;  390     2751  3         ! priority to the so-called PATCH permanent symbols, and to the symbols
;  391     2752  3         ! DEFined by the user at PATCH-time.  No longer pathname could be such
;  392     2753  3         ! a thing.
;  393     2754  3         !--
;  394     2755  3         STATUS = 0;
;  395     2756  4         IF (.PATH_VEC_PTR[1] EQL 0)
;  396     2757  3         THEN
;  397     2758  4                 BEGIN
;  398     2759  4                 LOCAL
;  399     2760  4                         TEMP_SYM_TBL,
;  400     2761  4                         DEF_SYM_DESC : BLOCK[8,BYTE];
;  401     2762  4
;  402     2763  4                 !++
;  403     2764  4                 ! A 1-element pathname may be or a DEFine symbol.  First build
;  404     2765  4                 ! a string descriptor for the name (since this is what
;  405     2766  4                 ! PAT$FIND_SYM wants).
;  406     2767  4                 !--
;  407     2768  4                 CS_PTR = .PATH_VEC_PTR[0];
;  408     2769  4                 DEF_SYM_DESC[DSC$W_LENGTH] = .CS_PTR[0];
;  409     2770  4                 DEF_SYM_DESC[DSC$A_POINTER] = CS_PTR[1];
;  410     2771  4
;  411     2772  4                 !++
;  412     2773  4                 ! The symbol is not a permanent one.  Now lookup it up in the
;  413     2774  4                 ! linked list reserved for DEFine symbols.
;  414     2775  4                 !--
;  415     2776  4                 TEMP_SYM_TBL = .PAT$GL_SYMTBPTR;                ! Remember curren symbol table
```

```
;   416     2777  4                 PAT$GL_SYMTBPTR = .PAT$GL_SYMHEAD;                  ! Use user-defined symbol table
;   417     2778  4                 STATUS = PAT$FIND_SYM(DEF_SYM_DESC);
;   418     2779  4                 PAT$GL_SYMTBPTR = .TEMP_SYM_TBL;                    ! Restore current symbol table
;   419     2780  4
;   420     2781  4                 !++
;   421     2782  4                 ! If we found something, pass back the associated value
;   422     2783  4                 ! and set STATUS to the appropriate success code.
;   423     2784  4                 !--
;   424     2785  5                 IF (.STATUS NEQ 0)
;   425     2786  4                 THEN
;   426     2787  5                         BEGIN
;   427     2788  5                         PASS_BACK_VALUE[0] = .SYM_VALUE(.STATUS);
;   428     2789  5                         STATUS = PAT$K_USER_DEF;
;   429     2790  4                         END;
;   430     2791  3                 END;
;   431     2792  3
;   432     2793  3         !++
;   433     2794  3         ! Now, if we didn't get something from the DEFine
;   434     2795  3         ! or permanent symbol data bases, try the RST.
;   435     2796  3         !--
;   436     2797  4         IF (NOT .STATUS)
;   437     2798  3         THEN
;   438     2799  3                 STATUS = PAT$SYM_TO_VAL( .PATH_VEC_PTR, .PASS_BACK_VALUE);
;   439     2800  3
;   440     2801  3         !++
;   441     2802  3         ! If no translation can be fourd, Check whether to SIGNAL an error
;   442     2803  3         ! and cause an UNWIND or return FALSE.
;   443     2804  3         !--
;   444     2805  4         IF (NOT .STATUS)
;   445     2806  3         THEN
;   446     2807  4                 BEGIN
;   447     2808  4                 LOCAL MESSAGE_BUF : VECTOR[TTY_OUT_WIDTH,BYTE];
;   448     2809  4
;   449     2810  4                 !++
;   450     2811  4                 ! Encode the pathname into a counted
;   451     2812  4                 ! string, and output the associated message.
;   452     2813  4                 !--
;   453     2814  4                 PAT$PV_TO_CS( .PATH_VEC_PTR, MESSAGE_BUF );
;   454     2815  4                 PAT$DELETE_PATH();
;   455     2816  4                 PATH_VEC_PTR = 0;
;   456     2817  4
;   457     2818  4                 !++
;   458     2819  4                 ! Check if this might be a forward reference and therefore
;   459     2820  4                 ! should not be signaled as an error.
;   460     2821  4                 !--
;   461     2822  5                 IF (NOT .SIGNAL_FLAG)
;   462     2823  4                 THEN
;   463     2824  5                         RETURN(FALSE)
;   464     2825  4                 ELSE
;   465     2826  4                         SIGNAL(PAT$_NOSYMBOL, 1, MESSAGE_BUF ); ! no return
;   466     2827  3                 END;
;   467     2828  3
;   468     2829  3         !++
;   469     2830  3         ! If the evaluation succeeded, discard the pathname vector and
;   470     2831  3         ! return success.
;   471     2832  3         !--
;   472     2833  3         PAT$DELETE_PATH();
```

```
;  473          2834  3          RETURN(.STATUS);
;  474          2835  2          END;
;  475          2836  2
;  476          2837  2 !++
;  477          2838  2 ! A real string descriptor is supposed to pass on to us another pathname
;  478          2839  2 ! element to accumulate.
;  479          2840  2 !
;  480          2841  2 ! If this is the first call for a new pathname, we must allocate the storage
;  481          2842  2 ! we will need for the vector of pointers to the element strings.
;  482          2843  2 !--
;  483          2844  3 IF (.PATH_VEC_PTR EQL 0)
;  484          2845  2 THEN
;  485          2846  3          BEGIN
;  486          2847  4          IF ((PATH_VEC_PTR = PAT$freez(RST_UNITS(%SIZE(PATHNAME_VECTOR)))) EQL 0)
;  487          2848  3          THEN
;  488          2849  3                  SIGNAL(PAT$_NOFREE);                                    ! No more storage.
;  489          2850  3
;  490          2851  3          !++
;  491          2852  3          ! The storage manager zeros out the pathname vector for us, so we only
;  492          2853  3          ! have to set up the right pathname vector index.
;  493          2854  3          !--
;  494          2855  3          PV_INDEX = 0;
;  495          2856  2          END;
;  496          2857  2
;  497          2858  2 !++
;  498          2859  2 ! Now we need space for the element name itself, (including the count! ).
;  499          2860  2 !--
;  500          2861  3 IF ((CS_PTR = PAT$freez(RST_UNITS(.SYMBOL_DESC[DSC$W_LENGTH]+1))) EQL 0)
;  501          2862  2 THEN
;  502          2863  2          SIGNAL(PAT$_NOFREE);                                    ! No more storage.
;  503          2864  2
;  504          2865  2 !++
;  505          2866  2 ! Copy the string into the allocated storage. Note that we must make up a counted
;  506          2867  2 ! string because this is what pathname vector pointers are defined to point to.
;  507          2868  2 !--
;  508          2869  2 CS_PTR[0] = .SYMBOL_DESC[DSC$W_LENGTH];
;  509          2870  2 CH$MOVE( .SYMBOL_DESC[DSC$W_LENGTH], .SYMBOL_DESC[DSC$A_POINTER], CS_PTR[1] );
;  510          2871  2
;  511          2872  2 !++
;  512          2873  2 ! Now store the address of this counted string in  the 'next' slot in the
;  513          2874  2 ! pathname vector.
;  514          2875  2 !--
;  515          2876  2 PATH_VEC_PTR[.PV_INDEX] = .CS_PTR;
;  516          2877  2
;  517          2878  2 !++
;  518          2879  2 ! And set up so that the next call to this routine stores the CS pointer into the
;  519          2880  2 ! next slot.
;  520          2881  2 !--
;  521          2882  3 IF ((PV_INDEX = .PV_INDEX +1) GTR MAX_PATH_SIZE)
;  522          2883  2 THEN
;  523          2884  3          BEGIN
;  524          2885  3          SIGNAL (PAT$_PATHTLONG);
;  525          2886  3          RETURN(FALSE);
;  526          2887  2          END;
;  527          2888  2 RETURN(TRUE);
;  528          2889  1 END;
```

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

                                .PSECT  _PAT$OWN,NOEXE,2

00000000  00000 PATH_VEC_PTR:
                                .LONG   0                                       ;
          00004 DST_BEGIN_ADDR:
                                .BLKB   4
          00008 DST_END_ADDR:
                                .BLKB   4
          0000C DST_NEXT_ADDR:
                                .BLKB   4
          00010 GSR_BEGIN_ADDR:
                                .BLKB   4
          00014 GSR_NEXT_ADDR:
                                .BLKB   4
          00018 GST_BEGIN_ADDR:
                                .BLKB   4
          0001C GSD_PEC_COUNT:
                                .BLKB   4
          00020 PV_INDEX:
                                .BLKB   4

                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$PV_TO_CS
                                .EXTRN  PAT$FIND_SYM, PAT$SET_MODULE
                                .EXTRN  PAT$SYM_TO_VAL, PAT$SYM_TO_VALU
                                .EXTRN  PAT$INIT_RST, PAT$FREEZ
                                .EXTRN  PAT$FREERELEASE
                                .EXTRN  LIB$_CREMAPSEC, PAT$GB_SYMBOLS
                                .EXTRN  PAT$GL_IMGHDR, PAT$GL_OLDNBK
                                .EXTRN  PAT$GB_OLDNAME, PAT$GL_ISVADDR
                                .EXTRN  PAT$GL_CSP_PTR, PAT$GL_MC_PTR
                                .EXTRN  PAT$GL_RST_BEGN
                                .EXTRN  PAT$GL_HEAD_LST
                                .EXTRN  PAT$GL_SYMTBPTR
                                .EXTRN  PAT$GL_SYMHEAD, PAT$_CLOSEIN
                                .EXTRN  PAT$_CLOSEOUT, PAT$_OPENIN
                                .EXTRN  PAT$_OPENOUT, PAT$_READERR
                                .EXTRN  PAT$_SYSERROR, PAT$_WRITEERR
                                .WEAK   ACCESS_CHECK

                                .PSECT  _PAT$CODE,NOWRT,2

                    0FFC 00000          .ENTRY  PAT$BUILD_PATH, Save R2,R3,R4,R5,R6,R7,R8,- ; 2663
                                                R9,R10,R11                                  ;
5B 00000000G   EF   9E 00002            MOVAB   PAT$FREEZ, R11                          ;
5A 00000000V   EF   9E 00009            MOVAB   PAT$DELETE_PATH, R10                    ;
59 00000000G   EF   9E 00010            MOVAB   PAT$GL_SYMTBPTR, R9                     ;
58 00000000G   00   9E 00017            MOVAB   LIB$SIGNAL, R8                          ;
57 00000000'   EF   9E 0001E            MOVAB   PATH_VEC_PTR, R7                        ;
```

```
                5E      FF7C    CE  9E 00025          MOVAB   -132(SP), SP
                52        04    AC  D0 0002A          MOVL    SYMBOL_DESC, R2                 ; 2746
                                79  12 0002E          BNEQ    3$
                                54  D4 00030          CLRL    STATUS                          ; 2755
                50              67  D0 00032          MOVL    PATH_VEC_PTR, R0                ; 2756
                          04    A0  D5 00035          TSTL    4(R0)
                                32  12 00038          BNEQ    1$
                56              60  D0 0003A          MOVL    (R0), CS_PTR                    ; 2768
             7C AE              66  9B 0003D          MOVZBW  (CS_PTR), DEF_SYM_DESC          ; 2769
             FC AD        01    A6  9E 00041          MOVAB   1(R6), DEF_SYM_DESC+4           ; 2770
                53              69  D0 00046          MOVL    PAT$GL_SYMTBPTR, TEMP_SYM_TBL   ; 2776
                69 00000000G    EF  D0 00049          MOVL    PAT$GL_SYMHEAD, PAT$GL_SYMTBPTR ; 2777
                          7C    AE  9F 00050          PUSHAB  DEF_SYM_DESC                    ; 2778
00000000G       EF              01  FB 00053          CALLS   #1, PAT$FIND_SYM
                54              50  D0 0005A          MOVL    R0, STATUS
                69              53  D0 0005D          MOVL    TEMP_SYM_TBL, PAT$GL_SYMTBPTR   ; 2779
                                54  D5 00060          TSTL    STATUS                          ; 2785
                                08  13 00062          BEQL    1$
             08 BC        08    A4  D0 00064          MOVL    8(STATUS), @PASS_BACK_VALUE     ; 2788
                54              03  D0 00069          MOVL    #3, STATUS                      ; 2789
                33              54  E8 0006C 1$:      BLBS    STATUS, 2$                      ; 2797
                          08    AC  DD 0006F          PUSHL   PASS_BACK_VALUE                 ; 2799
                                67  DD 00072          PUSHL   PATH_VEC_PTR
00000000G       EF              02  FB 00074          CALLS   #2, PAT$SYM_TO_VAL
                54              50  D0 0007B          MOVL    R0, STATUS
                21              54  E8 0007E          BLBS    STATUS, 2$                      ; 2805
                                5E  DD 00081          PUSHL   SP                              ; 2814
                                67  DD 00083          PUSHL   PATH_VEC_PTR
00000000G       EF              02  FB 00085          CALLS   #2, PAT$PV_TO_CS
                6A              00  FB 0008C          CALLS   #0, PAT$DELETE_PATH             ; 2815
                                67  D4 0008F          CLRL    PATH_VEC_PTR                    ; 2816
                78        0C    AC  E9 00091          BLBC    SIGNAL_FLAG, 8$                 ; 2822
                                5E  DD 00095          PUSHL   SP                              ; 2826
                                01  DD 00097          PUSHL   #1
                   006D8090     8F  DD 00099          PUSHL   #7176336
                68              03  FB 0009F          CALLS   #3, LIB$SIGNAL
                6A              00  FB 000A2 2$:      CALLS   #0, PAT$DELETE_PATH             ; 2833
                50              54  D0 000A5          MOVL    STATUS, R0                      ; 2834
                                04  000A8             RET
                                67  D5 000A9 3$:      TSTL    PATH_VEC_PTR                    ; 2844
                                16  12 000AB          BNEQ    5$
                                0B  DD 000AD          PUSHL   #11                             ; 2847
                6B              01  FB 000AF          CALLS   #1, PAT$FREEZ
                67              50  D0 000B2          MOVL    R0, PATH_VEC_PTR
                                09  12 000B5          BNEQ    4$
                   006D8112     8F  DD 000B7          PUSHL   #7176466                        ; 2849
                68              01  FB 000BD          CALLS   #1, LIB$SIGNAL
                          20    A7  D4 000C0 4$:      CLRL    PV_INDEX                        ; 2855
                50              62  3C 000C3 5$:      MOVZWL  (R2), R0                        ; 2861
                50              04  C0 000C6          ADDL2   #4, R0
          7E    50              04  C7 000C9          DIVL3   #4, R0, -(SP)
                6B              01  FB 000CD          CALLS   #1, PAT$FREEZ
                56              50  D0 000D0          MOVL    R0, CS_PTR
                                09  12 000D3          BNEQ    6$
                   006D8112     8F  DD 000D5          PUSHL   #7176466                        ; 2863
                68              01  FB 000DB          CALLS   #1, LIB$SIGNAL
                66              62  90 000DE 6$:      MOVB    (R2), (CS_PTR)                  ; 2869
```

```
                01    A6        04    B2                62  28 000E1          MOVC3   (R2), @4(R2), 1(CS_PTR)                ; 2870
                                  50          20        A7  D0 000E7          MOVL    PV_INDEX, R0                           ; 2876
                                00 B740                 56  D0 000EB          MOVL    CS_PTR, @PATH_VEC_PTR[R0]              ;
                      50        20    A7                C1  C1 000F0          ADDL3   #1, PV_INDEX, R0                       ; 2882
                                20    A7                50  D0 000F5          MOVL    R0, PV_INDEX                           ;
                                      0A                50  D1 000F9          CMPL    R0, #10                                ;
                                                        0B  15 000FC          BLEQ    7$                                     ;
                                          006D8152      8F  DD 000FE          PUSHL   #7176530                               ; 2885
                                      68                01  FB 00104          CALLS   #1, LIB$SIGNAL                         ;
                                                        04  11 00107          BRB     8$                                     ; 2886
                                      50                01  D0 00109 7$:      MOVL    #1, R0                                 ; 2888
                                                            04 0010C          RET                                            ;
                                                        50  D4 0010D 8$:      CLRL    R0                                     ; 2889
                                                            04 0010F          RET                                            ;

; Routine Size:  272 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
;  530     2890  1 GLOBAL ROUTINE PAT$DELETE_PATH : NOVALUE =
;  531     2891  1
;  532     2892  1 !++
;  533     2893  1 ! Functional Description:
;  534     2894  1 !
;  535     2895  1 !       Delete the pathname vector we are passed a pointer to by the OWN,
;  536     2896  1 !       PATH_VEC_PTR, which several routines in this module work from.  Also,
;  537     2897  1 !       zero out this pointer so that the next call to BUILD_PATH knows
;  538     2898  1 !       there is no 'current' pathname vector being built.
;  539     2899  1 !
;  540     2900  1 ! Formal Parameters:
;  541     2901  1 !
;  542     2902  1 !       none
;  543     2903  1 !
;  544     2904  1 ! Implicit Inputs:
;  545     2905  1 !
;  546     2906  1 !       PATH_VEC_PTR - See above.
;  547     2907  1 !
;  548     2908  1 ! Return Value:
;  549     2909  1 !
;  550     2910  1 !       NOVALUE - because the only thing which can go wrong
;  551     2911  1 !                 is a free storage error and in that
;  552     2912  1 !                 case the manager itself SIGNALs its way out.
;  553     2913  1 !
;  554     2914  1 !--
;  555     2915  1
;  556     2916  2 BEGIN
;  557     2917  2
;  558     2918  2 LOCAL
;  559     2919  2         CS_PTR : CS_POINTER;                                    ! Each element of the pathname vector
;  560     2920  2                                                                 ! is a pointer to a counted string.
;  561     2921  2
;  562     2922  2 !++
;  563     2923  2 ! Now see if there is really a pathname vector currently pointed to by the
;  564     2924  2 ! pointer, PATH_VEC_PTR.
;  565     2925  2 !--
;  566     2926  3 IF (.PATH_VEC_PTR EQLA 0)
;  567     2927  2 THEN
;  568     2928  2         RETURN;
;  569     2929  2
;  570     2930  2 !++
;  571     2931  2 ! Simply throw away the storage which we allocated
;  572     2932  2 ! for each element of the vector.
;  573     2933  2 !-
;  574     2934  2 INCR I FROM 0 TO MAX_PATH_SIZE
;  575     2935  2 DO
;  576     2936  2         !++
;  577     2937  2         ! The first 0 entry ends the vector.
;  578     2938  2         !--
;  579     2939  3         IF ((CS_PTR = .PATH_VEC_PTR[.I]) EQL 0)
;  580     2940  2         THEN
;  581     2941  2                 EXITLOOP
;  582     2942  2         ELSE
;  583     2943  2                 PAT$FREERELEASE( .CS_PTR, RST_UNITS(.CS_PTR[0]+1) );
;  584     2944  2
;  585     2945  2 !++
;  586     2946  2 ! Then throw away the vector itself.
```

```
;  587          2947  2 !--
;  588          2948  2 PAT$FREERELEASE( .PATH_VEC_PTR, RST_UNITS( %SIZE(PATHNAME_VECTOR) ));
;  589          2949  2
;  590          2950  2 !++
;  591          2951  2 ! Zero out the pointer so that subsequent re-uses know there is no longer
;  592          2952  2 ! one there.
;  593          2953  2 !--
;  594          2954  2 PATH_VEC_PTR = 0;
;  595          2955  2
;  596          2956  1 END;


                                    003C 00000          .ENTRY  PAT$DELETE_PATH, Save R2,R3,R4,R5                ; 2890
                 55 00000000G   EF  9E 00002          MOVAB   PAT$FREERELEASE, R5                              ;
                 54 00000000'   EF  9E 00009          MOVAB   PATH_VEC_PTR, R4                                 ;
                                64  D5 00010          TSTL    PATH_VEC_PTR                                     ; 2926
                                25  13 00012          BEQL    3$                                               ;
                                52  D4 00014          CLRL    I                                                ; 2934
                 53          00 B442  D0 00016 1$:    MOVL    @PATH_VEC_PTR[I], CS_PTR                         ; 2939
                                13  13 0001B          BEQL    2$                                               ;
                 50             63  9A 0001D          MOVZBL  (CS_PTR), R0                                     ; 2943
                 50             04  C0 00020          ADDL2   #4, R0                                           ;
              7E 50             04  C7 00023          DIVL3   #4, R0, -(SP)                                    ;
                                53  DD 00027          PUSHL   CS_PTR                                           ;
                 65             02  FB 00029          CALLS   #2, PAT$FREERELEASE                              ;
              E6 52             0A  F3 0002C          AOBLEQ  #10, I, 1$                                       ; 2939
                                0B  DD 00030 2$:      PUSHL   #11                                              ; 2948
                                64  DD 00032          PUSHL   PATH_VEC_PTR                                     ;
                             65     02  FB 00034      CALLS   #2, PAT$FREERELEASE                              ;
                                64  D4 00037          CLRL    PATH_VEC_PTR                                     ; 2954
                                    04 00039 3$:      RET                                                      ; 2956

; Routine Size:  58 bytes,    Routine Base:  _PAT$CODE + 0110
```

```
;  598          2957  1 GLOBAL ROUTINE PAT$_  _SCOPE( SET_SCOPE_FLAG ) =
;  599          2958  1
;  600          2959  1 !++
;  601          2960  1 ! Functional Description:
;  602          2961  1 !
;  603          2962  1 !    This routine serves two fairly distinct purposes.
;  604          2963  1 !
;  605          2964  1 ! 1. IF SET_SCOPE_FLAG is ON, then this routine was
;  606          2965  1 !    called to SET the new current scope position (CSP).
;  607          2966  1 !    In this case we delete the storage taken by the old
;  608          2967  1 !    CSP, if there was any, and install the new CSP
;  609          2968  1 !    having checked its validity.
;  610          2969  1 !       SET SCOPE also implies SET MODULE.
;  611          2970  1 !
;  612          2971  1 ! 2. If SET_SCOPE_FLAG is OFF, then the call was made to simply
;  613          2972  1 !    install a null CSP vector.  This happens as a result of the user
;  614          2973  1 !    cancelling scope, or cancelling a module whose name is the same as what the
;  615          2974  1 !    CSP pathname begins with.  The latter avoids the 'dangling scope' problem.
;  616          2975  1 !
;  617          2976  1 ! Implicit Inputs:
;  618          2977  1 !
;  619          2978  1 !       This routine works from the OWN that is local to this
;  620          2979  1 !       module, PATH_VEC_PTR, which points to the current pathname vector
;  621          2980  1 !       which was (presumably) built by BUILD_PATH.  We store
;  622          2981  1 !       away this pathname vector pointer, and then zero out the
;  623          2982  1 !       one that BUILD_PATH uses so that it 'forgets' completely
;  624          2983  1 !       about having built it.
;  625          2984  1 !
;  626          2985  1 ! Return Value:
;  627          2986  1 !
;  628          2987  1 ! TRUE, if we are simply throwing away the old CSP,
;  629          2988  1 !       or if we installed a new one which was acceptable,
;  630          2989  1 ! FALSE, otherwise.  (we were asked to install one which was invalid).
;  631          2990  1 !
;  632          2991  1 !--
;  633          2992  1
;  634          2993  2 BEGIN
;  635          2994  2
;  636          2995  2 LOCAL
;  637          2996  2         NEW_CSP_PTR : REF PATHNAME_VECTOR,
;  638          2997  2         MC_PTR : REF MC_RECORD,                          ! Used to chain along the MC.
;  639          2998  2         CS_PTR : CS_POINTER,                             ! Temp counted string pointer.
;  640          2999  2         STATUS;                                          ! Success/failure indication that we return.
;  641          3000  2
;  642          3001  2 !++
;  643          3002  2 ! The gross structure of this routine just implements the two-function logic.
;  644          3003  2 !--
;  645          3004  3 IF (.SET_SCOPE_FLAG)
;  646          3005  2 THEN
;  647          3006  3         BEGIN
;  648          3007  3         !++
;  649          3008  3         ! Install a new CSP vector. Check that the CSP we were given is valid.
;  650          3009  3         ! First, see if the beginning element of the pathvector (which must be
;  651          3010  3         ! MODULE) is in the MC.  Note that we don't consider the first entry in
;  652          3011  3         ! the MC since it is used for globals only and hence is nameless.
;  653          3012  3         !--
;  654          3013  3         CS_PTR = .PATH_VEC_PTR[0];
```

```
;  655     3014  3          MC_PTR = .PAT$GL_MC_PTR;
;  656     3015  4          WHILE ((MC_PTR = .MC_PTR [MC_NEXT]) NEQ 0)
;  657     3016  3          DO
;  658     3017  4                  BEGIN
;  659     3018  5                  IF (CH$EQL(.MC_PTR[MC_NAME_CS], MC_PTR[MC_NAME_ADDR],
;  660     3019  5                                  .CS_PTR[0], CS_PTR[1]))
;  661     3020  4                  THEN
;  662     3021  4                          EXITLOOP                                        ! Found.  Continue on to do further checking
;  663     3022  3                  END;
;  664     3023  3
;  665     3024  3          !++
;  666     3025  3          ! If the module name was not found, we must not accept the CSP.
;  667     3026  3          !--
;  668     3027  4          IF (.MC_PTR EQL 0)
;  669     3028  3          THEN
;  670     3029  4                  BEGIN
;  671     3030  4                  !++
;  672     3031  4                  ! This is an error.  Note that if there was previous to this
;  673     3032  4                  ! call a valid CSP, it is not affected by this error.  ALso note
;  674     3033  4                  ! that the storage for the CSP we just found to be invalid is
;  675     3034  4                  ! discarded by the end-of-line processing AFTER the SIGNAL
;  676     3035  4                  ! produces the message.
;  677     3036  4                  !--
;  678     3037  4                  SIGNAL(PAT$_NOSUCHMODU,1,.CS_PTR);
;  679     3038  4                  RETURN(FALSE);
;  680     3039  3                  END;
;  681     3040  3
;  682     3041  3          !++
;  683     3042  3          ! Make sure that the indicated module is in the RST so that
;  684     3043  3          ! further checking can be done and because a "set scope" implies a
;  685     3044  3          ! "SET MODULE" command.
;  686     3045  3          !--
;  687     3046  3          IF NOT .MC_PTR[MC_IN_RST]
;  688     3047  3          THEN
;  689     3048  3                  PAT$SET_MODULE(.MC_PTR);                                ! IF THIS FAILS, THERE IS NOT RETURN FROM TH
;  690     3049  3
;  691     3050  3          !++
;  692     3051  3          ! The module name is valid and in the RST.  Any further checking depends
;  693     3052  3          ! on whether the given CSP is any longer than simply "module".  If this
;  694     3053  3          ! is the case, we've done all the validating we can.
;  695     3054  3          !--
;  696     3055  4          IF (.PATH_VEC_PTR[1] NEQ 0)
;  697     3056  3          THEN
;  698     3057  4                  BEGIN
;  699     3058  4                  !++
;  700     3059  4                  ! Further checking is RST-dependent.
;  701     3060  4                  !--
;  702     3061  4                  LOCAL
;  703     3062  4                          VAL_DESC : VALU_DESCRIPTOR,
;  704     3063  4                          NT_PTR : REF NT_RECORD;
;  705     3064  4
;  706     3065  4                  !++
;  707     3066  4                  ! For initialized modules, we can do a complete check.
;  708     3067  4                  ! This means that we effectively do a lookup, and then
;  709     3068  4                  ! make sure that the path leads to a symbol of type
;  710     3069  4                  ! ROUTINE.
;  711     3070  4                  !--
```

```
;  712          3071  5          IF (NOT PAT$SYM_TO_VALU( .PATH_VEC_PTR, VAL_DESC))
;  713          3072  4          THEN
;  714          3073  5                  BEGIN
;  715          3074  5                  !++
;  716          3075  5                  ! Encode the pathname into a counted string and output
;  717          3076  5                  ! the associated message.
;  718          3077  5                  !--
;  719          3078  5                  LOCAL MESSAGE_BUF : VECTOR[TTY_OUT_WIDTH, BYTE];
;  720          3079  5
;  721          3080  5                  PAT$PV_TO_CS(.PATH_VEC_PTR, MESSAGE_BUF);
;  722          3081  5                  SIGNAL(PAT$_NOSYMBOL, T, MESSAGE_BUF);  ! No return
;  723          3082  5                  RETURN(FALSE);                  !****** THIS SHOULDN'T BE NEEDED
;  724          3083  4                  END;
;  725          3084  4
;  726          3085  4          !++
;  727          3086  4          ! Now we simply have to see that the valid path leads
;  728          3087  4          ! to ROUTINE.  First we pick up the pointer to this
;  729          3088  4          ! symbol's name table record.
;  730          3089  4          !--
;  731          3090  4          NT_PTR = .VAL_DESC [VALU_NT_PTR];
;  732          3091  5          IF (NOT .NT_PTR[NT_TYPE] EQL DSC$K_DTYPE_RTN)
;  733          3092  4          THEN
;  734          3093  5                  BEGIN
;  735          3094  5                  !++
;  736          3095  5                  ! A valid path, but we can't accept it as a CSP
;  737          3096  5                  ! because perpending it to any symbol would
;  738          3097  5                  ! never result in a valid path.
;  739          3098  5                  !--
;  740          3099  5                  SIGNAL(PAT$_BADCSP);
;  741          3100  5                  RETURN(FALSE);
;  742          3101  4                  END;
;  743          3102  3                  END;
;  744          3103  3          !++
;  745          3104  3          ! The CSP we are to SET has been checked out OK.
;  746          3105  3          !--
;  747          3106  3          NEW_CSP_PTR = .PATH_VEC_PTR;
;  748          3107  2          END;
;  749          3108  2
;  750          3109  2  !++
;  751          3110  2  ! If we get this far, the new CSP will be accepted. First, we have to release
;  752          3111  2  ! the storage we used up in accumulating the pathname elements of the  old CSP,
;  753          3112  2  ! if there was one.
;  754          3113  2  !--
;  755          3114  3  IF ((PATH_VEC_PTR = .PAT$GL_CSP_PTR) NEQ 0)
;  756          3115  2  THEN
;  757          3116  3          BEGIN
;  758          3117  3          PAT$DELETE_PATH();
;  759          3118  2          END;
;  760          3119  2  !++
;  761          3120  2  ! If we were only throwing away the old vector, then we must be done.
;  762          3121  2  !--
;  763          3122  3  IF (NOT .SET_SCOPE_FLAG)
;  764          3123  2  THEN
;  765          3124  3          BEGIN
;  766          3125  3          PAT$GL_CSP_PTR = 0;
;  767          3126  3          RETURN(TRUE);
;  768          3127  2          END;
```

```
;  769          3128  2
;  770          3129  2 !++
;  771          3130  2 ! Installing a new CSP is simply a matter of saving away the pointer to the
;  772          3131  2 ! PATHNAME_VECTOR.  We must also zero out the pointer to the vector which
;  773          3132  2 ! BUILD_PATH uses to deal with these vectors, since we have effectively taken
;  774          3133  2 ! this one away.
;  775          3134  2 !--
;  776          3135  2 PAT$GL_CSP_PTR = .NEW_CSP_PTR;
;  777          3136  2 PATH_VEC_PTR = 0;
;  778          3137  2
;  779          3138  2 RETURN(TRUE);
;  780          3139  1 END;


                                                03FC 00000          .ENTRY  PAT$SAVE_SCOPE, Save R2,R3,R4,R5,R6,R7,R8,-      ; 2957
                                                                            R9                                                   :
                                59 00000000G   00  9E 00002          MOVAB   LIB$SIGNAL, R9                                       :
                                58 00000000G   EF  9E 00009          MOVAB   PAT$GL_CSP_PTR, R8                                   :
                                57 00000000G   EF  9E 00010          MOVAB   PAT$GL_RST_BEGN, R7                                  :
                                56 00000000'   EF  9E 00017          MOVAB   PATH_VEC_PTR, R6                                     :
                                5E      FF74   CE  9E 0001E          MOVAB   -140(SP), SP                                         :
                                03        04   AC  E8 00023          BLBS    SET_SCOPE_FLAG, 1$                                   ; 3004
                                           0098  31  00027          BRW     9$                                                   :
                                55        00   B6  D0 0002A 1$:      MOVL    @PATH_VEC_PTR, CS_PTR                                ; 3013
                                54 00000000G   EF  D0 0002E          MOVL    PAT$GL_MC_PTR, MC_PTR                                ; 3014
                     50         54             67  C1 00035 2$:      ADDL3   PAT$GL_RST_BEGN, MC_PTR, R0                          ; 3015
                                54             60  3C 00039          MOVZWL  (R0), MC_PTR                                         :
                                               15  13 0003C          BEQL    3$                                                   :
                     50         54             67  C1 0003E          ADDL3   PAT$GL_RST_BEGN, MC_PTR, R0                          ; 3018
                                52         0C  A0  9A 00042          MOVZBL  12(R0), R2                                           :
                                51             65  9A 00046          MOVZBL  (CS_PTR), R1                                         ; 3019
51                   00        0D  A0             52  2D 00049          CMPC5   R2, 13(R0), #0, R1, 1(CS_PTR)                        :
                                               01  A5      0004F                                                                    :
                                               E2  12 00051          BNEQ    2$                                                   :
                                               54  D5 00053 3$:       TSTL    MC_PTR                                               ; 3027
                                               0C  12 00055          BNEQ    4$                                                   :
                                               55  DD 00057          PUSHL   CS_PTR                                               ; 3037
                                               01  DD 00059          PUSHL   #1                                                   :
                                        006D8080   8F  DD 0005B          PUSHL   #7176320                                             :
                                               3E  11 00061          BRB     6$                                                   :
                     50                    54   67  C1 00063 4$:      ADDL3   PAT$GL_RST_BEGN, MC_PTR, R0                          ; 3046
                     09        03          A0   01  E0 00067          BBS     #1, 3(R0), 5$                                        :
                                               54  DD 0006C          PUSHL   MC_PTR                                               ; 3048
                          00000000G        EF   01  FB 0006E          CALLS   #1, PAT$SET_MODULE                                   :
                                               50  66  D0 00075 5$:      MOVL    PATH_VEC_PTR, R0                                     ; 3055
                                           04  A0  D5 00078          TSTL    4(R0)                                                :
                                               42  13 0007B          BEQL    8$                                                   :
                                           F8  AD  9F 0007D          PUSHAB  VAL_DESC                                             ; 3071
                                               50  DD 00080          PUSHL   R0                                                   :
                          00000000G        EF   02  FB 00082          CALLS   #2, PAT$SYM_TO_VALU                                  :
                                               1A  50  E8 00089          BLBS    R0, 7$                                               :
                                               5E  DD 0008C          PUSHL   SP                                                   ; 3080
                                               66  DD 0008E          PUSHL   PATH_VEC_PTR                                         :
                          00000000G        EF   02  FB 00090          CALLS   #2, PAT$PV_TO_CS                                     :
```

```
                                5E  DD 00097          PUSHL   SP                                      ; 3081
                                01  DD 00099          PUSHL   #1                                      ;
                    006D8090    8F  DD 0009B          PUSHL   #7176336                                ;
                69              03  FB 000A1 6$:      CALLS   #3, LIB$SIGNAL                          ;
                                37  11 000A4          BRB     13$                                     ; 3082
                50      F8      AD  3C 000A6 7$:      MOVZWL  VAL_DESC, NT_PTR                        ; 3090
                50              67  C0 000AA          ADDL2   PAT$GL_RST_BEGN, R0                     ; 3091
             BE 8F      02      A0  91 000AD          CMPB    2(R0), #190                             ;
                                0B  13 000B2          BEQL    8$                                      ;
                    006D8060    8F  DD 000B4          PUSHL   #7176288                                ; 3099
                69              01  FB 000BA          CALLS   #1, LIB$SIGNAL                          ;
                                1E  11 000BD          BRB     13$                                     ; 3100
                52              66  D0 000BF 8$:      MOVL    PATH_VEC_PTR, NEW_CSP_PTR               ; 3106
                66              68  D0 000C2 9$:      MOVL    PAT$GL_CSP_PTR, PATH_VEC_PTR            ; 3114
                                05  13 000C5          BEQL    10$                                     ;
             FEFA CF            00  FB 000C7          CALLS   #0, PAT$DELETE_PATH                     ; 3117
                04      04      AC  E8 000CC 10$:     BLBS    SET_SCOPE_FLAG, 11$                     ; 3122
                                68  D4 000D0          CLRL    PAT$GL_CSP_PTR                          ; 3125
                                05  11 000D2          BRB     12$                                     ; 3126
                68              52  D0 000D4 11$:     MOVL    NEW_CSP_PTR, PAT$GL_CSP_PTR             ; 3135
                                66  D4 000D7          CLRL    PATH_VEC_PTR                            ; 3136
                50              01  D0 000D9 12$:     MOVL    #1, R0                                  ; 3138
                                    04 000DC          RET                                             ;
                                50  D4 000DD 13$:     CLRL    R0                                      ; 3139
                                    04 000DF          RET                                             ;

; Routine Size:  224 bytes,    Routine Base:  _PAT$CODE + 014A
```

```
;  782          3140  1 GLOBAL ROUTINE PAT$FIND_MODULE( MOD_NAME_DESC, SIGNAL_FLAG ) =
;  783          3141  1
;  784          3142  1 !++
;  785          3143  1 ! Functional Description:
;  786          3144  1 !
;  787          3145  1 !       Search the MC to see if the given module is there.
;  788          3146  1 !
;  789          3147  1 ! Formal Parameters:
;  790          3148  1 !
;  791          3149  1 !       MOD_NAME_DESC   -a string descriptor for the supposed
;  792          3150  1 !                        module name.
;  793          3151  1 !       SIGNAL_FLAG     -indicator whether or not this routine should
;  794          3152  1 !                        SIGNAL if the module is not found
;  795          3153  1 !
;  796          3154  1 ! Implicit Inputs:
;  797          3155  1 !
;  798          3156  1 !       none.
;  799          3157  1 !
;  800          3158  1 ! Implicit Outputs:
;  801          3159  1 !
;  802          3160  1 !       none
;  803          3161  1 !
;  804          3162  1 ! Returned Value:
;  805          3163  1 !
;  806          3164  1 !       0 - if the module is not found,
;  807          3165  1 !       an MC_PTR (non-zero) to the indicated MC record, otherwise.
;  808          3166  1 !
;  809          3167  1 ! Side Effects:
;  810          3168  1 !
;  811          3169  1 !       none
;  812          3170  1 !--
;  813          3171  1
;  814          3172  2 BEGIN
;  815          3173  2 MAP
;  816          3174  2         MOD_NAME_DESC : REF BLOCK[,BYTE];                       ! The supposed module name is
;  817          3175  2                                                                 ! described via an SRM string descriptor.
;  818          3176  2
;  819          3177  2 LOCAL
;  820          3178  2         MODU_CS_NAME : VECTOR[SYM_MAX_LENGTH+1, BYTE],          ! COPY OF MODULE NAME FOR NOSUCHMODU ERROR M
;  821          3179  2         MC_PTR : REF MC_RECORD;                                 ! We chain along the MC via this temp pointe
;  822          3180  2
;  823          3181  2 !++
;  824          3182  2 ! Scan along the MC comparing the given string with the module name stored
;  825          3183  2 ! therein.  Note that we skip the first MC record because it is reserved for
;  826          3184  2 ! globals and is therefore nameless.
;  827          3185  2 !--
;  828          3186  2 MC_PTR = .PAT$GL_MC_PTR;
;  829          3187  3 WHILE ((MC_PTR = .MC_PTR [MC_NEXT]) NEQ 0)
;  830          3188  2 DO
;  831          3189  3         BEGIN
;  832          3190  4         IF (CH$EQL(.MC_PTR[MC_NAME_CS],MC_PTR[MC_NAME_ADDR],
;  833          3191  4                         .MOD_NAME_DESC[DSC$W_LENGTH],.MOD_NAME_DESC[DSC$A_POINTER] ))
;  834          3192  3         THEN
;  835          3193  4                 BEGIN
;  836          3194  4                 !++
;  837          3195  4                 ! Found.  Internally in PATCH we agree that the 'value' of a
;  838          3196  4                 ! module string will be the RST address of its MC record.
```

```
;  839          3197  4                    !--
;  840          3198  4                    RETURN(.MC_PTR);
;  841          3199  3                    END;
;  842          3200  2            END;
;  843          3201  2
;  844          3202  2 !++
;  845          3203  2 ! If we fall out of the above loop, then the given module name was not found.
;  846          3204  2 ! Therefore if a SIGNAL is allowed, then construct a COUNTED_STRING pointer and
;  847          3205  2 ! pass it as the error message argument.
;  848          3206  2 !--
;  849          3207  2 IF .SIGNAL_FLAG
;  850          3208  2 THEN
;  851          3209  3            BEGIN
;  852          3210  3            MODU_CS_NAME[0] = .MOD_NAME_DESC[DSC$W_LENGTH];
;  853          3211  3            CH$MOVE (.MODU_CS_NAME[0], .MOD_NAME_DESC[DSC$A_POINTER], MODU_CS_NAME[1]);
;  854          3212  3            SIGNAL(PAT$_NOSUCHMODU, 1, MODU_CS_NAME[0]);            ! No return
;  855          3213  2            END;
;  856          3214  2
;  857          3215  2 RETURN (0);
;  858          3216  1 END;


                                                007C 00000          .ENTRY  PAT$FIND_MODULE, Save R2,R3,R4,R5,R6        ; 3140
                                    56 00000000G    EF   9E 00002   MOVAB   PAT$GL_RST_BEGN, R6
                                    5E              20   C2 00009   SUBL2   #32, SP
                                    54 00000000G    EF   D0 0000C   MOVL    PAT$GL_MC_PTR, MC_PTR                       ; 3186
                                    55        04    AC   D0 00013   MOVL    MOD_NAME_DESC, R5                           ; 3191
                                50  54              66   C1 00017 1$:   ADDL3   PAT$GL_RST_BEGN, MC_PTR, R0            ; 3187
                                    54              60   3C 0001B   MOVZWL  (R0), MC_PTR
                                                    17   13 0001E   BEQL    2$
                                50  54              66   C1 00020   ADDL3   PAT$GL_RST_BEGN, MC_PTR, R0               ; 3190
                                    51        0C    A0   9A 00024   MOVZBL  12(R0), R1
                04  BC          00        0D    A0   51   2D 00028   CMPC5   R1, 13(R0), #0, @MOD_NAME_DESC, @4(R5)
                                                04   B5     0002F
                                                     E4   12 00031   BNEQ    1$
                                    50              54   D0 00033   MOVL    MC_PTR, R0                                  ; 3198
                                                    04   00036   RET
                                    1E        08    AC   E9 00037 2$:   BLBC    SIGNAL_FLAG, 3$                         ; 3207
                                    6E        04    BC   90 0003B   MOVB    @MOD_NAME_DESC, MODU_CS_NAME                ; 3210
                                    50              6E   9A 0003F   MOVZBL  MODU_CS_NAME, R0                           ; 3211
                          01    AE        04    B5        50   28 00042   MOVC3   R0, @4(R5), MODU_CS_NAME+1
                                                     5E   DD 00048   PUSHL   SP                                          ; 3212
                                                     01   DD 0004A   PUSHL   #1
                                              006D8080    8F   DD 0004C   PUSHL   #7176320
                                00000000G     00               03   FB 00052   CALLS   #3, LIB$SIGNAL
                                                     50   D4 00059 3$:   CLRL    R0                                   ; 3215
                                                     04   0005B   RET                                                 ; 3216

; Routine Size:  92 bytes,    Routine Base:  _PAT$CODE + 022A
```

```
;  860          3217  1 GLOBAL ROUTINE PAT$FIND_DST : NOVALUE =
;  861          3218  1
;  862          3219  1 !++
;  863          3220  1 ! FUNCTIONAL DESCRIPTION:
;  864          3221  1 !
;  865          3222  1 !       Find out where the DST begins and make it available for
;  866          3223  1 !       PAT$GET_NXT_DST and PAT$GET_DST_REC.
;  867          3224  1 !       (or make it so that these routines return EOF if no DST exists).
;  868          3225  1 !       Then do the same for the GST.
;  869          3226  1 !
;  870          3227  1 ! Calling Sequence:
;  871          3228  1 !
;  872          3229  1 !       PAT$FIND_DST()
;  873          3230  1 !
;  874          3231  1 ! FORMAL PARAMETERS:
;  875          3232  1 !
;  876          3233  1 !       none
;  877          3234  1 !
;  878          3235  1 ! IMPLICIT INPUTS:
;  879          3236  1 !
;  880          3237  1 !       The image header has been read and PAT$GL_IMGHDR points to it.
;  881          3238  1 !       The old image file is open and ready to read the DST and GST.
;  882          3239  1 !       The variables pointing to the file are:
;  883          3240  1 !               PAT$GL_OLDFAB, AND PAT$GL_OLDNAME.
;  884          3241  1 !
;  885          3242  1 ! IMPLICIT OUTPUTS:
;  886          3243  1 !
;  887          3244  1 !       none
;  888          3245  1 !
;  889          3246  1 ! COMPLETION CODES:
;  890          3247  1 !       none
;  891          3248  1 !
;  892          3249  1 ! SIDE EFFECTS:
;  893          3250  1 !
;  894          3251  1 !       The notion of 'next' DST record is initialized
;  895          3252  1 !       here so that a call to PAT$GET_NXT_DST made after
;  896          3253  1 !       a call to this routine will fetch the first record.
;  897          3254  1 !
;  898          3255  1 !       The begin and end address of the DST are also established,
;  899          3256  1 !       but only for the purposes of the interface routines.
;  900          3257  1 !       There is no explicit requirement for this from the RST's
;  901          3258  1 !       viewpoint - so long as the interface can somehow
;  902          3259  1 !       know when the last record has been passed on.
;  903          3260  1 !
;  904          3261  1 !       If anything goes wrong during the GST/DST initializations,
;  905          3262  1 !       (can't EXPREG, etc.), we output the corresponding message forcing
;  906          3263  1 !       the severity to -I-, and then continue on without the GST or DST.
;  907          3264  1 !       The exceptions to this are that there must be symbol table info in
;  908          3265  1 !       the header (even if what's there is simply a pointer to say that
;  909          3266  1 !       there is no DST or GST).
;  910          3267  1 !--
;  911          3268  1
;  912          3269  2 BEGIN
;  913          3270  2
;  914          3271  2 BIND
;  915          3272  2         SYM_TBL_DATA = .PAT$GL_IMGHDR + .PAT$GL_IMGHDR [IHD$W_SYMDBGOFF]
;  916          3273  2                                       : BLOCK [, BYTE],
```

```
;  917          3274  2          EXESECNAM = UPLIT BYTE (%ASCIC 'DST'),
;  918          3275  2          GSTSECNAM = UPLIT BYTE (%ASCIC 'GST');
;  919          3276  2
;  920          3277  2 LITERAL
;  921          3278  2          GL_OVERHEAD_REC = 2,                                    ! GST overhead records from Linker
;  922          3279  2          SYMS_PER_GLREC  = 28,                                   ! Average GSTs per GST record.
;  923          3280  2          START_ADDRESS   = 0,                                    ! Starting address offset
;  924          3281  2          END_ADDRESS     = 1;                                    ! Ending address offset
;  925          3282  2
;  926          3283  2 LOCAL
;  927          3284  2          STATUS : BLOCK[%UPVAL, BYTE],
;  928          3285  2          GLOBAL_RECORD : BLOCK [A_PAGE, BYTE],
;  929          3286  2          EXESECNAM_DESC : VECTOR [2, LONG],
;  930          3287  2          EXEFILNAM_DESC : VECTOR [2, LONG],
;  931          3288  2          GL_SYM_COUNT : VOLATILE;
;  932          3289  2
;  933          3290  2 !++
;  934          3291  2 ! Check if this .EXE file has symbols at all.  There are two kinds of checks
;  935          3292  2 ! which we make.  First, we see if the image header is consistent.
;  936          3293  2 ! There are two checks for this - one which is always relevant, and one which
;  937          3294  2 ! is relevant only if we have already determined that there will be DSTs.
;  938          3295  2 !--
;  939          3296  3 IF (.PAT$GL_IMGHDR [IHD$W_SYMDBGOFF] EQL 0)
;  940          3297  2 THEN
;  941          3298  3          BEGIN
;  942          3299  3          GST_BEGIN_ADDR = 0;
;  943          3300  3          DST_BEGIN_ADDR = 0;
;  944          3301  3          PAT$GB_SYMBOLS = FALSE;                                 ! Indicate image has no symbols
;  945          3302  3          SIGNAL(PAT$_NOGBL+MSG$K_INFO);
;  946          3303  3          SIGNAL(PAT$_NOLCL+MSG$K_INFO);
;  947          3304  3          RETURN;
;  948          3305  3          END
;  949          3306  2 ELSE
;  950          3307  2          PAT$GB_SYMBOLS = TRUE;                                  ! Indicate image has symbols
;  951          3308  2
;  952          3309  2 !++
;  953          3310  2 ! Then we see if this is a simple case of there legitimately not being a DST.
;  954          3311  2 ! (i.e. the modules were simply not compiled with /DEBUG on).
;  955          3312  2 !--
;  956          3313  3 IF ((DST_BEGIN_ADDR = .SYM_TBL_DATA[IHS$W_DSTBLKS]) EQL 0)
;  957          3314  2 THEN
;  958          3315  3          BEGIN
;  959          3316  3          !++
;  960          3317  3          ! Check that the VBN of the DST is also zero.  If it is not,
;  961          3318  3          ! then the image header is contradictory.  Therefore, inform the
;  962          3319  3          ! user and fix the header by setting the DST fields to zero.
;  963          3320  3          ! This should only be an informational message.
;  964          3321  3          !--
;  965          3322  4          IF (.SYM_TBL_DATA[IHS$L_DSTVBN] NEQ 0)
;  966          3323  3          THEN
;  967          3324  3                    SIGNAL(PAT$_INVIMGHDR+MSG$K_INFO);
;  968          3325  3          SIGNAL(PAT$_NOLCL+MSG$K_INFO);
;  969          3326  3          DST_BEGIN_ADDR = 0;
;  970          3327  3          SYM_TBL_DATA[IHS$L_DSTVBN] = 0;
;  971          3328  3          SYM_TBL_DATA[IHS$W_DSTBLKS] = 0;
;  972          3329  3          END
;  973          3330  2 ELSE
```

```
;  974          3331  2          !++
;  975          3332  2          ! Check that the VBN is legal.  If not, then this is an inconsistent
;  976          3333  2          ! header.  Inform the user that it is invalid and
;  977          3334  2          ! fix up the header, ignoring the symbols that might be there.
;  978          3335  2          !--
;  979          3336  2          IF (.SYM_TBL_DATA[IHS$L_DSTVBN] LEQ 2) OR
;  980          3337  3             (.SYM_TBL_DATA[IHS$W_DSTBLKS] LSS 0)
;  981          3338  2          THEN
;  982          3339  3                  BEGIN
;  983          3340  3                  SIGNAL(PAT$_INVIMGHDR+MSG$K_INFO);
;  984          3341  3                  SIGNAL(PAT$_NOLCL+MSG$K_INFO);
;  985          3342  3                  DST_BEGIN_ADDR = 0;
;  986          3343  3                  SYM_TBL_DATA[IHS$L_DSTVBN] = 0;
;  987          3344  3                  SYM_TBL_DATA[IHS$W_DSTBLKS] = 0;
;  988          3345  2                  END;
;  989          3346  2
;  990          3347  2  !++
;  991          3348  2  ! Check that a GST exists.  If not, set an indicator.  Also make a valid image
;  992          3349  2  ! header.  This insures PAT$WRTIMG will work correctly.
;  993          3350  2  !--
;  994          3351  3  IF ((GST_BEGIN_ADDR = .SYM_TBL_DATA[IHS$W_GSTRECS]) EQL 0)
;  995          3352  2  THEN
;  996          3353  3          BEGIN
;  997          3354  3          !++
;  998          3355  3          ! Check that the VBN of the GST is also zero.  If it is not,
;  999          3356  3          ! then the image header is contradictory.  Therefore, inform the
; 1000          3357  3          ! user and fix the header by setting the GST fields to zero.
; 1001          3358  3          ! This should only be an informational message.
; 1002          3359  3          !--
; 1003          3360  4          IF (.SYM_TBL_DATA[IHS$L_GSTVBN] NEQ 0)
; 1004          3361  3          THEN
; 1005          3362  3                  SIGNAL(PAT$_INVIMGHDR+MSG$K_INFO);
; 1006          3363  3          SIGNAL(PAT$_NOGBL+MSG$K_INFO);
; 1007          3364  3          GST_BEGIN_ADDR = 0;
; 1008          3365  3          SYM_TBL_DATA[IHS$L_GSTVBN] = 0;
; 1009          3366  3          SYM_TBL_DATA[IHS$W_GSTRECS] = 0;
; 1010          3367  3          END
; 1011          3368  2  ELSE
; 1012          3369  2          !++
; 1013          3370  2          ! Check that the VBN is legal.  If not, then this is an inconsistent
; 1014          3371  2          ! header.  Inform the user that it is invalid and
; 1015          3372  2          ! fix up the header, ignoring the symbols that might be there.
; 1016          3373  2          !--
; 1017          3374  2          IF (.SYM_TBL_DATA[IHS$L_GSTVBN] LEQ 2) OR
; 1018          3375  3             (.SYM_TBL_DATA[IHS$W_GSTRECS] LSS 0)
; 1019          3376  2          THEN
; 1020          3377  3                  BEGIN
; 1021          3378  3                  SIGNAL(PAT$_INVIMGHDR+MSG$K_INFO);
; 1022          3379  3                  SIGNAL(PAT$_NOGBL+MSG$K_INFO);
; 1023          3380  3                  GST_BEGIN_ADDR = 0;
; 1024          3381  3                  SYM_TBL_DATA[IHS$L_GSTVBN] = 0;
; 1025          3382  3                  SYM_TBL_DATA[IHS$W_GSTRECS] = 0;
; 1026          3383  2                  END;
; 1027          3384  2
; 1028          3385  2  !++
; 1029          3386  2  ! Don't try to create and map the DST if there is not one in the .EXE file to map in.
; 1030          3387  2  !--
```

```
;  1031     3388  3 IF (.DST_BEGIN_ADDR NEQ 0)
;  1032     3389  2 THEN
;  1033     3390  3         BEGIN
;  1034     3391  3         PAT$GL_ISVADDR [START_ADDRESS] = 200;                   ! Set the address vectors to point to the
;  1035     3392  3         PAT$GL_ISVADDR [END_ADDRESS] = 200;                     !  first available addresses in P0 space.
;  1036     3393  3         EXESECNAM_DESC [0] = 3;                                                                          ;
;  1037     3394  3         EXESECNAM_DESC [1] = EXESECNAM;
;  1038     3395  3         EXEFILNAM_DESC [0] = .PAT$GL_OLDNBK[NAM$B_RSL];
;  1039     3396  3         EXEFILNAM_DESC [1] = PAT$GB_OLDNAME;
;  1040     3397  3
;  1041     3398  4         IF NOT (STATUS = LIB$_CREMAPSEC (PAT$GL_ISVADDR
;  1042     3399  4                                        , PAT$GL_ISVADDR
;  1043     3400  4                                        , SEC$M_EXPREG
;  1044     3401  4                                        , EXESECNAM_DESC
;  1045     3402  4                                        , 0
;  1046     3403  4                                        , EXEFILNAM_DESC
;  1047     3404  4                                        , .SYM_TBL_DATA [IHS$W_DSTBLKS]
;  1048     3405  4                                        , .SYM_TBL_DATA [IHS$L_DSTVBN]))
;  1049     3406  3         THEN
;  1050     3407  4                 BEGIN
;  1051     3408  4                 !++
;  1052     3409  4                 ! Unconditionally make the severity level informational so
;  1053     3410  4                 ! that the message will be produced with no side effects.
;  1054     3411  4                 !--
;  1055     3412  4 !               STATUS[STS$V_SEVERITY] = SYS$K_INFO;
;  1056     3413  4                 STATUS[STS$V_SEVERITY] = 3;
;  1057     3414  4                 DST_BEGIN_ADDR =0;
;  1058     3415  4                 SIGNAL(PAT$_SYSERROR-MSG$K_FATAL+MSG$K_INFO, 0, .STATUS);
;  1059     3416  4                 SIGNAL(.STATUS);
;  1060     3417  4                 END
;  1061     3418  3         ELSE
;  1062     3419  3                 !++
;  1063     3420  3                 ! Now load up the addresses of the beginning
;  1064     3421  3                 ! and end of the DST.
;  1065     3422  3                 !--
;  1066     3423  4                 BEGIN
;  1067     3424  4                 DST_BEGIN_ADDR = .PAT$GL_ISVADDR [START_ADDRESS];
;  1068     3425  4                 DST_END_ADDR = .PAT$GL_ISVADDR [END_ADDRESS];
;  1069     3426  4                 DST_NEXT_ADDR = .DST_BEGIN_ADDR;
;  1070     3427  3                 END;
;  1071     3428  2         END;                                                    ! For no DSTs.
;  1072     3429  2
;  1073     3430  2 !++
;  1074     3431  2 ! Now map in the GST in the same way we did the DST.  Don't try to create and
;  1075     3432  2 ! map the GST if there is not one in the .exe file to map in.
;  1076     3433  2 !--
;  1077     3434  3 IF (.GST_BEGIN_ADDR NEQ 0)
;  1078     3435  2 THEN
;  1079     3436  3         BEGIN
;  1080     3437  3         LOCAL
;  1081     3438  3                 GST_REC_PTR : REF VECTOR[,WORD];
;  1082     3439  3
;  1083     3440  3         !++
;  1084     3441  3         ! Find the last mapped address used and compute the addresses into
;  1085     3442  3         ! which the GST will be mapped.
;  1086     3443  3         !--
;  1087     3444  3         PAT$GL_ISVADDR[START_ADDRESS] = 200;                    ! Set the address vectors to point to the
```

```
;  1088     3445  3          PAT$GL_ISVADDR[END_ADDRESS] = 200;                      !  first available addresses in P0 space.
;  1089     3446  3          EXESECNAM_DESC [0] = 3;
;  1090     3447  3          EXESECNAM_DESC [1] = GSTSECNAM;
;  1091     3448  3          EXEFILNAM_DESC [0] = .PAT$GL_OLDNBK[NAM$B_RSL];
;  1092     3449  3          EXEFILNAM_DESC [1] = PAT$GB_OLDNAME;
;  1093     3450  3
;  1094     3451  4          IF NOT (STATUS = LIB$_CREMAPSEC (PAT$GL_ISVADDR
;  1095     3452  4                                          , PAT$GL_ISVADDR
;  1096     3453  4                                          , SEC$M_EXPREG
;  1097     3454  4                                          , EXESECNAM_DESC
;  1098     3455  4                                          , 0
;  1099     3456  4                                          , EXEFILNAM_DESC
;  1100     3457  4                                          , .SYM_TBL_DATA [IHS$W_GSTRECS]
;  1101     3458  4                                          , .SYM_TBL_DATA [IHS$L_GSTVBN]
;  1102     3459  4                                          ))
;  1103     3460  3          THEN
;  1104     3461  4                  BEGIN
;  1105     3462  4                  !++
;  1106     3463  4                  ! Unconditionally make the severity level informational so
;  1107     3464  4                  ! that the message will be produced with no side effects.
;  1108     3465  4                  !--
;  1109     3466  4 !                STATUS[STS$V_SEVERITY] = SYS$K_INFO;
;  1110     3467  4                  STATUS[STS$V_SEVERITY] = 3;
;  1111     3468  4                  GST_BEGIN_ADDR =0;
;  1112     3469  4                  GSR_BEGIN_ADDR =0;
;  1113     3470  4                  SIGNAL (PAT$_SYSERROR-MSG$K_FATAL+MSG$K_INFO, 0, .STATUS);
;  1114     3471  4                  SIGNAL(.STATUS);
;  1115     3472  4                  END
;  1116     3473  3          ELSE
;  1117     3474  4                  BEGIN
;  1118     3475  4                  !++
;  1119     3476  4                  ! Now skip the first two records because they
;  1120     3477  4                  ! are module header and module sub-header, respectively.
;  1121     3478  4                  ! NOTE: this builds in the knowledge of how these
;  1122     3479  4                  ! usually-RMS records are formatted.
;  1123     3480  4                  !--
;  1124     3481  4                  GST_REC_PTR = .PAT$GL_ISVADDR[START_ADDRESS];
;  1125     3482  4
;  1126     3483  4                  !++
;  1127     3484  4                  ! Get to the next record by adding the rounded-up
;  1128     3485  4                  ! record byte count to the previous beginning
;  1129     3486  4                  ! virtual address, then adding on 2 because the count
;  1130     3487  4                  ! field is 2 bytes long.
;  1131     3488  4                  !--
;  1132     3489  4                  GST_REC_PTR = .GST_REC_PTR + 2 + ((.GST_REC_PTR[0] +1)/2)*2;
;  1133     3490  4
;  1134     3491  4                  !++
;  1135     3492  4                  ! Now skip the sub-module header.
;  1136     3493  4                  !--
;  1137     3494  4                  GST_REC_PTR = .GST_REC_PTR + 2 + ((.GST_REC_PTR[0] +1)/2)*2;
;  1138     3495  4
;  1139     3496  4                  !++
;  1140     3497  4                  ! And this is the address we wanted.  Both the first, and, at this
;  1141     3498  4                  ! point, the 'next' records, start at this address.
;  1142     3499  4                  !--
;  1143     3500  4                  GSR_BEGIN_ADDR = .GST_REC_PTR;
;  1144     3501  4                  GSR_NEXT_ADDR = .GSR_BEGIN_ADDR;
```

```
;  1145          3502  4
;  1146          3503  4                    !++
;  1147          3504  4                    ! Tell the inner mechanism how many GST records there will be.
;  1148          3505  4                    ! This number is the number that the LINKer gave us, -3,
;  1149          3506  4                    ! because of the 2 records we just skipped over, PLUS the
;  1150          3507  4                    ! module-end record at the end of the GST.
;  1151          3508  4                    !--
;  1152          3509  4                    POSITION_GST( .SYM_TBL_DATA[IHS$W_GSTRECS] - 3 );
;  1153          3510  3                    END;
;  1154          3511  2            END;                                                    ! For no GSTs.
;  1155          3512  2
;  1156          3513  2 PAT$INIT_RST (.GL_SYM_COUNT);
;  1157          3514  1 END;


                                                        .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                                                54  53  44  03  00000 P.AAA:  .ASCII  <3>\DST\                                    ;
                                                54  53  47  03  00004 P.AAB:  .ASCII  <3>\GST\                                    ;

                                                                    EXESECNAM=          P.AAA
                                                                    GSTSECNAM=          P.AAB

                                                        .PSECT  _PAT$CODE,NOWRT,2

                                                    0FFC 00000          .ENTRY  PAT$FIND_DST, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 3217
                                                                                R10,R11                                         ;
                                            5B 00000000G   EF   9E 00002        MOVAB   LIB$_CREMAPSEC, R11                     ;
                                            5A 00000000G   EF   9E 00009        MOVAB   PAT$GB_OLDNAME, R10                     ;
                                            59 00000000G   EF   9E 00010        MOVAB   PAT$GL_OLDNBK+3, R9                     ;
                                            58 00000000G   EF   9E 00017        MOVAB   PAT$GB_SYMBOLS, R8                      ;
                                            57 00000000G   EF   9E 0001E        MOVAB   PAT$GL_ISVADDR, R7                      ;
                                            56 00000000G   00   9E 00025        MOVAB   LIB$SIGNAL, R6                          ;
                                            55 00000000'   EF   9E 0002C        MOVAB   DST_BEGIN_ADDR, R5                      ;
                                            5E      FDEC   CE   9E 00033        MOVAB   -532(SP), SP                            ;
                                            50 00000000G   EF   D0 00038        MOVL    PAT$GL_IMGHDR, R0                       ; 3272
                                            51        04   A0   3C 0003F        MOVZWL  4(R0), R1                               ;
                                        52  51            50   C1 00043        ADDL3   R0, R1, R2                              ;
                                                      04   A0   B5 00047        TSTW    4(R0)                                   ; 3296
                                                           1A   12 0004A        BNEQ    1$                                      ;
                                                      14   A5   D4 0004C        CLRL    GST_BEGIN_ADDR                          ; 3299
                                                           65   D4 0004F        CLRL    DST_BEGIN_ADDR                          ; 3300
                                                           68   D4 00051        CLRL    PAT$GB_SYMBOLS                          ; 3301
                                                006D81D3   8F   DD 00053        PUSHL   #7176659                                ; 3302
                                                    66         01   FB 00059        CALLS   #1, LIB$SIGNAL                          ;
                                                006D81CB   8F   DD 0005C        PUSHL   #7176651                                ; 3303
                                                    66         01   FB 00062        CALLS   #1, LIB$SIGNAL                          ;
                                                               04   00065        RET                                             ; 3298
                                                    68         01   D0 00066 1$:    MOVL    #1, PAT$GB_SYMBOLS                      ; 3307
                                                    65   08    A2   3C 00069        MOVZWL  8(R2), DST_BEGIN_ADDR                   ; 3313
                                                               06   12 0006D        BNEQ    2$                                      ;
                                                               62   D5 0006F        TSTL    (R2)                                    ; 3322
                                                               07   12 00071        BNEQ    3$                                      ;
                                                               0E   11 00073        BRB     4$                                      ; 3325
                                                    02         62   D1 00075 2$:    CMPL    (R2), #2                                ; 3336
```

```
                                19  14 00078          BGTR    5$                                              ;
                 006D8243       8F  DD 0007A 3$:     PUSHL   #7176771                                        ; 3340
              66                01  FB 00080          CALLS   #1, LIB$SIGNAL                                  ;
                 006D81CB       8F  DD 00083 4$:     PUSHL   #7176651                                        ; 3341
              66                01  FB 00089          CALLS   #1, LIB$SIGNAL                                  ;
                                65  D4 0008C          CLRL    DST_BEGIN_ADDR                                  ; 3342
                                62  D4 0008E          CLRL    (R2)                                            ; 3343
                          08    A2  B4 00090          CLRW    8(R2)                                           ; 3344
14    A5                  0A    A2  3C 00093 5$:      MOVZWL  10(R2), GST_BEGIN_ADDR                          ; 3351
                                0A  12 00098          BNEQ    6$                                              ;
              54          04    A2  9E 0009A          MOVAB   4(R2), R4                                       ; 3360
                                64  D5 0009E          TSTL    (R4)                                            ;
                                0B  12 000A0          BNEQ    7$                                              ;
                                12  11 000A2          BRB     8$                                              ; 3363
              54          04    A2  9E 000A4 6$:      MOVAB   4(R2), R4                                       ; 3374
              02                64  D1 000A8          CMPL    (R4), #2                                        ;
                                1A  14 000AB          BGTR    9$                                              ;
                 006D8243       8F  DD 000AD 7$:      PUSHL   #7176771                                        ; 3378
              66                01  FB 000B3          CALLS   #1, LIB$SIGNAL                                  ;
                 006D81D3       8F  DD 000B6 8$:      PUSHL   #7176659                                        ; 3379
              66                01  FB 000BC          CALLS   #1, LIB$SIGNAL                                  ;
                          14    A5  D4 000BF          CLRL    GST_BEGIN_ADDR                                  ; 3380
                                64  D4 000C2          CLRL    (R4)                                            ; 3381
                          0A    A2  B4 000C4          CLRW    10(R2)                                          ; 3382
                                65  D5 000C7 9$:      TSTL    DST_BEGIN_ADDR                                  ; 3388
                                60  13 000C9          BEQL    11$                                             ;
              67          C8    8F  9A 000CB          MOVZBL  #200, PAT$GL_ISVADDR                            ; 3391
04    A7                  C8    8F  9A 000CF          MOVZBL  #200, PAT$GL_ISVADDR+4                          ; 3392
0C    AE                        03  D0 000D4          MOVL    #3, EXESECNAM_DESC                              ; 3393
10    AE  00000000'             EF  9E 000D8          MOVAB   EXESECNAM, EXESECNAM_DESC+4                     ; 3394
04    AE                        69  9A 000E0          MOVZBL  PAT$GL_OLDNBK+3, EXEFILNAM_DESC                 ; 3395
08    AE                        6A  9E 000E4          MOVAB   PAT$GB_OLDNAME, EXEFILNAM_DESC+4                ; 3396
                                62  DD 000E8          PUSHL   (R2)                                            ; 3405
           7E           08    A2  3C 000EA          MOVZWL  8(R2), -(SP)                                    ; 3404
                        0C    AE  9F 000EE          PUSHAB  EXEFILNAM_DESC                                  ; 3398
                                7E  D4 000F1          CLRL    -(SP)                                           ;
                        1C    AE  9F 000F3          PUSHAB  EXESECNAM_DESC                                  ;
                   00020000       8F  DD 000F6          PUSHL   #131072                                         ;
                                57  DD 000FC          PUSHL   R7                                              ;
                                57  DD 000FE          PUSHL   R7                                              ;
              6B                08  FB 00100          CALLS   #8, LIB$_CREMAPSEC                              ;
              53                50  D0 00103          MOVL    R0, STATUS                                      ;
              1B                53  E8 00106          BLBS    STATUS, 10$                                     ;
53          03          00      03  F0 00109          INSV    #3, #0, #3, STATUS                              ; 3413
                                65  D4 0010E          CLRL    DST_BEGIN_ADDR                                  ; 3414
                                53  DD 00110          PUSHL   STATUS                                          ; 3415
                                7E  D4 00112          CLRL    -(SP)                                           ;
                00000000G       8F  DD 00114          PUSHL   #PAT$_SYSERROR-1                                ;
              66                03  FB 0011A          CALLS   #3, LIB$SIGNAL                                  ;
                                53  DD 0011D          PUSHL   STATUS                                          ; 3416
              66                01  FB 0011F          CALLS   #1, LIB$SIGNAL                                  ;
                                07  11 00122          BRB     11$                                             ; 3398
              65                67  7D 00124 10$:     MOVQ    PAT$GL_ISVADDR, DST_BEGIN_ADDR                  ; 3424
          08  A5                65  D0 00127          MOVL    DST_BEGIN_ADDR, DST_NEXT_ADDR                   ; 3426
                          14    A5  D5 0012B 11$:     TSTL    GST_BEGIN_ADDR                                  ; 3434
                                5B  13 0012E          BEQL    12$                                             ;
              67          C8    8F  9A 00130          MOVZBL  #200, PAT$GL_ISVADDR                            ; 3444
```

```
                04   A7        C8   8F   9A 00134          MOVZBL  #200, PAT$GL_ISVADDR+4                    ; 3445
                0C   AE             03   D0 00139          MOVL    #3, EXESECNAM_DESC                       ; 3446
                10   AE  00000000'  EF   9E 0013D          MOVAB   GSTSECNAM, EXESECNAM_DESC+4              ; 3447
                04   AE             69   9A 00145          MOVZBL  PAT$GL_OLDNBK+3, EXEFILNAM_DESC          ; 3448
                08   AE             6A   9E 00149          MOVAB   PAT$GB_OLDNAME, EXEFILNAM_DESC+4         ; 3449
                                    64   DD 0014D          PUSHL   (R4)                                     ; 3458
                     7E        0A   A2   3C 0014F          MOVZWL  10(R2), -(SP)                            ; 3457
                               0C   AE   9F 00153          PUSHAB  EXEFILNAM_DESC                           ; 3451
                                    7E   D4 00156          CLRL    -(SP)
                               1C   AE   9F 00158          PUSHAB  EXESECNAM_DESC
                          00020000  8F   DD 0015B          PUSHL   #131072
                                    57   DD 00161          PUSHL   R7
                                    57   DD 00163          PUSHL   R7
                     6B             08   FB 00165          CALLS   #8, LIB$_CREMAPSEC
                     53             50   D0 00168          MOVL    R0, STATUS
                     1F             53   E8 0016B          BLBS    STATUS, 13$
          53    03    00             03   F0 0016E          INSV    #3, #0, #3, STATUS                       ; 3467
                               14   A5   D4 00173          CLRL    GST_BEGIN_ADDR                           ; 3468
                               0C   A5   D4 00176          CLRL    GSR_BEGIN_ADDR                           ; 3469
                                    53   DD 00179          PUSHL   STATUS                                   ; 3470
                                    7E   D4 0017B          CLRL    -(SP)
                          00000000G  8F   DD 0017D          PUSHL   #PAT$_SYSERROR-1
                     66             03   FB 00183          CALLS   #3, LIB$SIGNAL
                                    53   DD 00186          PUSHL   STATUS                                   ; 3471
                     66             01   FB 00188          CALLS   #1, LIB$SIGNAL
                                    34   11 0018B 12$:     BRB     14$                                      ; 3451
                     51             67   D0 0018D 13$:     MOVL    PAT$GL_ISVADDR, GST_REC_PTR              ; 3481
                     50             61   3C 00190          MOVZWL  (GST_REC_PTR), R0                        ; 3489
                                    50   D6 00193          INCL    R0
                     50             02   C6 00195          DIVL2   #2, R0
                     51        02 A140   3E 00198          MOVAW   2(GST_REC_PTR)[R0], GST_REC_PTR
                     50             61   3C 0019D          MOVZWL  (GST_REC_PTR), R0                        ; 3494
                                    50   D6 001A0          INCL    R0
                     50             02   C6 001A2          DIVL2   #2, R0
                     51        02 A140   3E 001A5          MOVAW   2(GST_REC_PTR)[R0], GST_REC_PTR
                0C   A5             51   D0 001AA          MOVL    GST_REC_PTR, GSR_BEGIN_ADDR              ; 3500
                10   A5        0C   A5   D0 001AE          MOVL    GSR_BEGIN_ADDR, GSR_NEXT_ADDR            ; 3501
                     7E        0A   A2   3C 001B3          MOVZWL  10(R2), -(SP)                            ; 3509
                     6E             03   C2 001B7          SUBL2   #3, (SP)
          00000000V  EF             01   FB 001BA          CALLS   #1, POSITION_GST
                                    6E   DD 001C1 14$:     PUSHL   GL_SYM_COUNT                             ; 3513
          00000000G  EF             01   FB 001C3          CALLS   #1, PAT$INIT_RST
                                    04   001CA            RET                                              ; 3514

; Routine Size:  459 bytes,    Routine Base:  _PAT$CODE + 0286
```

```
;  1159     3515  1 GLOBAL ROUTINE PAT$GET_DST_REC ( REC_ID ) =
;  1160     3516  1
;  1161     3517  1 !++
;  1162     3518  1 ! FUNCTIONAL DESCRIPTION:
;  1163     3519  1 !
;  1164     3520  1 !       Make the indicated DST record available.
;  1165     3521  1 !
;  1166     3522  1 ! FORMAL PARAMETERS:
;  1167     3523  1 !
;  1168     3524  1 !       REC_ID  - The ID of the record we are to fetch.
;  1169     3525  1 !                 This ID must be one which was previously returned
;  1170     3526  1 !                 by a call to PAT$GET_NXT_DST.
;  1171     3527  1 !
;  1172     3528  1 ! IMPLICIT INPUTS:
;  1173     3529  1 !
;  1174     3530  1 !       NONE
;  1175     3531  1 !
;  1176     3532  1 ! IMPLICIT OUTPUTS:
;  1177     3533  1 !
;  1178     3534  1 !       NONE
;  1179     3535  1 !
;  1180     3536  1 ! COMPLETION CODES:
;  1181     3537  1 !
;  1182     3538  1 !       0, if the indicated record does not exist,
;  1183     3539  1 !       the address of where is can now be referenced, otherwise.
;  1184     3540  1 !
;  1185     3541  1 ! SIDE EFFECTS:
;  1186     3542  1 !
;  1187     3543  1 !       The DST record is made available.
;  1188     3544  1 !
;  1189     3545  1 !--
;  1190     3546  1
;  1191     3547  2 BEGIN
;  1192     3548  2
;  1193     3549  2 BIND
;  1194     3550  2         DST_RECRD = .REC_ID : DST_RECORD;
;  1195     3551  2
;  1196     3552  2 !++
;  1197     3553  2 ! If there is no DST, simply return as though we were asked to read one
;  1198     3554  2 ! past the last one. (The interface's notion of EOF).
;  1199     3555  2 !--
;  1200     3556  3 IF (.DST_BEGIN_ADDR EQL 0)
;  1201     3557  2 THEN
;  1202     3558  2         RETURN(0);
;  1203     3559  2
;  1204     3560  2 !++
;  1205     3561  2 ! The record ID is the same as the virtual address at which it can be
;  1206     3562  2 ! referenced.  The next record, then, is simply the one which is virtually
;  1207     3563  2 ! contiguous to this one, excepting for the case of the last record.
;  1208     3564  2 ! Here we are lenient - we say that the DST ended OK if one asks for a
;  1209     3565  2 ! record which is past the end marker, OR, if the count field
;  1210     3566  2 ! for a supposed 'next' record is 0.
;  1211     3567  2 !--
;  1212     3568  3 IF (.REC_ID EQL .DST_END_ADDR +1)
;  1213     3569  2 THEN
;  1214     3570  2         RETURN(0);
;  1215     3571  2
```

```
;  1216          3572  2 !++
;  1217          3573  2 ! Now that it is safe, check for 0-length records.
;  1218          3574  2 !--
;  1219          3575  3 IF (.DST_RECRD [DSTR_SIZE] EQL 0)
;  1220          3576  2 THEN
;  1221          3577  2         RETURN(0);
;  1222          3578  2
;  1223          3579  2 !++
;  1224          3580  2 ! Then check that the ID is valid.
;  1225          3581  2 !--
;  1226          3582  3 IF (.REC_ID LSSA .dst_begin_addr)   OR    (.REC_ID GTRA .dst_end_addr)
;  1227          3583  2 THEN
;  1228          3584  3         BEGIN
;  1229          3585  3         !++
;  1230          3586  3         ! This should not happen - we check and report
;  1231          3587  3         ! errors here only to help us while debugging.
;  1232          3588  3         !--
;  1233          3589  3         SIGNAL (PAT$_INVDSTREC);                              ! Severe error
;  1234          3590  3         RETURN(0);
;  1235          3591  2         END;
;  1236          3592  2
;  1237          3593  2 RETURN( .REC_ID );
;  1238          3594  1 END;


                                    000C 00000         .ENTRY  PAT$GET_DST_REC, Save R2,R3                     ; 3515
                          53 00000000'  EF   9E 00002  MOVAB   DST_END_ADDR, R3
                          52        04  AC   D0 00009  MOVL    REC_ID, R2                                      ; 3550
                          51        FC  A3   D0 0000D  MOVL    DST_BEGIN_ADDR, R1                              ; 3556
                                        2A   13 00011  BEQL    3$
                    50    63            01   C1 00013  ADDL3   #1, DST_END_ADDR, R0                           ; 3568
                          50            52   D1 00017  CMPL    R2, R0
                                        21   13 0001A  BEQL    3$
                                        62   95 0001C  TSTB    (R2)                                            ; 3575
                                        1D   13 0001E  BEQL    3$
                          51            52   D1 00020  CMPL    R2, R1                                          ; 3582
                                        05   1F 00023  BLSSU   1$
                          63            52   D1 00025  CMPL    R2, DST_END_ADDR
                                        0F   1B 00028  BLEQU   2$
                             006D80E2   8F   DD 0002A  1$:     PUSHL   #7176418                                ; 3589
                00000000G   00          01   FB 00030  CALLS   #1, LIB$SIGNAL
                                        04   11 00037  BRB     3$                                              ; 3590
                          50            52   D0 00039  2$:     MOVL    R2, R0                                  ; 3593
                                              04 0003C  RET
                                        50   D4 0003D  3$:     CLRL    R0                                      ; 3594
                                              04 0003F  RET

; Routine Size:  64 bytes,    Routine Base:  _PAT$CODE + 0451
```

```
;1240     3595  1 GLOBAL ROUTINE PAT$POSITON_DST ( REC_ID ) =
;1241     3596  1
;1242     3597  1 !++
;1243     3598  1 ! FUNCTIONAL DESCRIPTION:
;1244     3599  1 !
;1245     3600  1 !       Make the indicated DST record available in such
;1246     3601  1 !       a way that PAT$GET_NXT_DST's idea of 'next' is
;1247     3602  1 !       defined to be the one after this routine fetches.
;1248     3603  1 !
;1249     3604  1 ! FORMAL PARAMETERS:
;1250     3605  1 !
;1251     3606  1 !       REC_ID  - The ID of the record we are to fetch.
;1252     3607  1 !                 This ID must be one which was previously returned
;1253     3608  1 !                 by a call to PAT$GET_NXT_DST.
;1254     3609  1 !
;1255     3610  1 ! IMPLICIT INPUTS:
;1256     3611  1 !
;1257     3612  1 !       NONE
;1258     3613  1 !
;1259     3614  1 ! IMPLICIT OUTPUTS:
;1260     3615  1 !
;1261     3616  1 !       NONE
;1262     3617  1 !
;1263     3618  1 ! COMPLETION CODES:
;1264     3619  1 !
;1265     3620  1 !       0, if the indicated record does not exist,
;1266     3621  1 !       the address of where is can now be referenced, otherwise.
;1267     3622  1 !
;1268     3623  1 ! SIDE EFFECTS:
;1269     3624  1 !
;1270     3625  1 !       The DST record is made available.
;1271     3626  1 !       The 'next' DST record is henceforth defined to
;1272     3627  1 !       be the one after the one fetched by this call.
;1273     3628  1 !
;1274     3629  1 !--
;1275     3630  1
;1276     3631  2 BEGIN
;1277     3632  2
;1278     3633  2 LOCAL
;1279     3634  2         REC_ADDR : REF DST_RECORD;
;1280     3635  2
;1281     3636  2 !++
;1282     3637  2 ! PAT$GET_DST_REC does most of the work -
;1283     3638  2 ! we just include the above-described side effect.
;1284     3639  2 !--
;1285     3640  3 IF ((REC_ADDR = PAT$GET_DST_REC( .REC_ID )) EQL 0 )
;1286     3641  2 THEN
;1287     3642  2         RETURN(0);
;1288     3643  2
;1289     3644  2 !++
;1290     3645  2 ! RE-initialize INT's notion of 'next' DST record.
;1291     3646  2 !--
;1292     3647  2 DST_NEXT_ADDR = .REC_ADDR + .REC_ADDR [DSTR_SIZE] +1;
;1293     3648  2 RETURN( .REC_ADDR );
;1294     3649  1 END;
```

```
                                        0000 00000          .ENTRY  PAT$POSITON_DST, Save nothing                ; 3595
                                    04  AC  DD 00002        PUSHL   REC_ID                                       ; 3640
                                B7  AF      01  FB 00005    CALLS   #1, PAT$GET_DST_REC                          ;
                                            50  D5 00009    TSTL    REC_ADDR                                     ;
                                            0D  13 0000B    BEQL    1$                                           ;
                                51          60  9A 0000D    MOVZBL  (REC_ADDR), R1                               ; 3647
                  00000000'     EF      01 A140  9E 00010    MOVAB   1(R1)[REC_ADDR], DST_NEXT_ADDR               ;
                                            04  00019        RET                                                  ; 3648
                                            50  D4 0001A 1$:  CLRL    R0                                         ; 3649
                                            04  0001C        RET                                                  ;

; Routine Size:  29 bytes,    Routine Base:  _PAT$CODE + 0491
```

```
;  1296     3650  1 ROUTINE POSITION_GST ( GST_REC_COUNT ) =
;  1297     3651  1
;  1298     3652  1 !++
;  1299     3653  1 ! FUNCTIONAL DESCRIPTION:
;  1300     3654  1 !
;  1301     3655  1 !       This routine, if called with a positive value initializes its OWN
;  1302     3656  1 !       storage to remember the number of RMS-type records in the GST.
;  1303     3657  1 !       If it is called with a negative or zero value, it returns the address
;  1304     3658  1 !       of the next RMS-type record in the GST.  A negative value also causes
;  1305     3659  1 !       the pointers to be positioned at the start of the GST.
;  1306     3660  1 !
;  1307     3661  1 ! FORMAL PARAMETERS:
;  1308     3662  1 !
;  1309     3663  1 !       GST_REC_COUNT   - The number of RMS records in the GST.
;  1310     3664  1 !                         (negative value) re-position to start and return
;  1311     3665  1 !                         address of first GLOBAL.
;  1312     3666  1 !                         (zero) return address of the next GLOBAL.
;  1313     3667  1 !
;  1314     3668  1 ! IMPLICIT INPUTS:
;  1315     3669  1 !
;  1316     3670  1 !       GSR_BEGIN_ADDR  - Holds the starting address of the GST.
;  1317     3671  1 !                         If the value is not GTR 0 or 1, then the GST
;  1318     3672  1 !                         has not been mapped in so this routine returns 0.
;  1319     3673  1 !
;  1320     3674  1 ! IMPLICIT OUTPUTS:
;  1321     3675  1 !
;  1322     3676  1 !       GSR_NEXT_ADDR   - Holds the address of the next RMS record in the GST
;  1323     3677  1 !                         or the GST was not mapped in.
;  1324     3678  1 !
;  1325     3679  1 ! ROUTINE VALUE:
;  1326     3680  1 !
;  1327     3681  1 !       0               - If there are no more records in the GST.
;  1328     3682  1 !       non-zero        - The address of the next GST RMS record.
;  1329     3683  1 !
;  1330     3684  1 ! SIDE EFFECTS:
;  1331     3685  1 !
;  1332     3686  1 !       The next GST record can now be accessed, and an OWN pointer to the next
;  1333     3687  1 !       one is maintained.  The number of GST records yet to go is also updated
;  1334     3688  1 !       so that the end of the GST can be detected.
;  1335     3689  1 !
;  1336     3690  1 !--
;  1337     3691  1
;  1338     3692  2 BEGIN
;  1339     3693  2
;  1340     3694  2 OWN
;  1341     3695  2         TOTAL_RECORDS,
;  1342     3696  2         RECORDS_LEFT;
;  1343     3697  2
;  1344     3698  2 LOCAL
;  1345     3699  2         BLOCK_ADDR;
;  1346     3700  2
;  1347     3701  2 !++
;  1348     3702  2 ! If there is no mapped GST, then return 0, no atter why this routine
;  1349     3703  2 ! was called.
;  1350     3704  2 !--
;  1351     3705  3 IF (NOT .GSR_BEGIN_ADDR GTRA 1)
;  1352     3706  2 THEN
```

```
;1353      3707  2          RETURN(0);
;1354      3708  2
;1355      3709  3 IF (.GST_REC_COUNT GTR 0)
;1356      3710  2 THEN
;1357      3711  3          BEGIN
;1358      3712  3          TOTAL_RECORDS = .GST_REC_COUNT;
;1359      3713  3          RETURN (0);
;1360      3714  2          END;
;1361      3715  2
;1362      3716  3 IF (.GST_REC_COUNT NEQ 0)
;1363      3717  2 THEN
;1364      3718  3          BEGIN
;1365      3719  3          GSR_NEXT_ADDR = .GSR_BEGIN_ADDR;
;1366      3720  3          RECORDS_LEFT  = .TOTAL_RECORDS;
;1367      3721  2          END;
;1368      3722  2
;1369      3723  2 !++
;1370      3724  2 ! Stop the following from faulting if some caller ignores the end condition and
;1371      3725  2 ! effectively causes us to 'run off the end' of the mapped GST.
;1372      3726  2 !--
;1373      3727  3 IF (NOT .RECORDS_LEFT GEQ 1)
;1374      3728  2 THEN
;1375      3729  2          RETURN(0);
;1376      3730  2
;1377      3731  2 !++
;1378      3732  2 ! Pick up the address of the current record, and update the pointer to the
;1379      3733  2 ! subsequent one.
;1380      3734  2 !--
;1381      3735  2 BLOCK_ADDR = .GSR_NEXT_ADDR + 2;
;1382      3736  2 GSR_NEXT_ADDR = .GSR_NEXT_ADDR + 2 + ((.GSR_NEXT_ADDR[0] +1)/2)*2;
;1383      3737  2 RECORDS_LEFT = .RECORDS_LEFT - 1;
;1384      3738  2 RETURN (.BLOCK_ADDR);
;1385      3739  1 END;


                                        .PSECT  _PAT$OWN,NOEXE,2

                                00024 TOTAL_RECORDS:
                                        .BLKB   4
                                00028 RECORDS_LEFT:
                                        .BLKB   4


                                        .PSECT  _PAT$CODE,NOWRT,2

                                000C 00000 POSITION_GST:
                                        .WORD   Save R2,R3                      ;3650
                    53 00000000'  EF  9E 00002          MOVAB   GSR_NEXT_ADDR, R3
                    01        FC  A3  D1 00009          CMPL    GSR_BEGIN_ADDR, #1      ;3705
                                  34  1B 0000D          BLEQU   3$
                    50        04  AC  D0 0000F          MOVL    GST_REC_COUNT, R0       ;3709
                                  06  15 00013          BLEQ    1$
                10  A3            50  D0 00015          MOVL    R0, TOTAL_RECORDS       ;3712
                                  28  11 00019          BRB     3$                      ;3713
                                  09  13 0001B 1$:      BEQL    2$                      ;3716
```

```
                     63       FC    A3   D0 0001D           MOVL    GSR_BEGIN_ADDR, GSR_NEXT_ADDR                   ; 3719
                14   A3       10    A3   D0 00021           MOVL    TOTAL_RECORDS, RECORDS_LEFT                     ; 3720
                     14       14    A3   D5 00026 2$:       TSTL    RECORDS_LEFT                                    ; 3727
                              18          15 00029           BLEQ    3$
                     52               63   D0 0002B           MOVL    GSR_NEXT_ADDR, R2                               ; 3735
                     50       02      A2   9E 0002E           MOVAB   2(R2), BLOCK_ADDR
                     51               62   3C 00032           MOVZWL  (R2), R1                                        ; 3736
                                      51   D6 00035           INCL    R1
                     51               02   C6 00037           DIVL2   #2, R1
                     63       02 A241     3E 0003A           MOVAW   2(R2)[R1], GSR_NEXT_ADDR
                              14   A3     D7 0003F           DECL    RECORDS_LEFT                                    ; 3737
                                          04 00042           RET                                                     ; 3738
                                   50     D4 00043 3$:       CLRL    R0                                              ; 3739
                                          04 00045           RET

; Routine Size:  70 bytes,    Routine Base:  _PAT$CODE + 04AE
```

```
;  1387     3740  1 GLOBAL ROUTINE PAT$GET_NXT_DST ( REC_ID_PTR ) =
;  1388     3741  1
;  1389     3742  1 !++
;  1390     3743  1 ! FUNCTIONAL DESCRIPTION:
;  1391     3744  1 !
;  1392     3745  1 !       Make the next DST record available,
;  1393     3746  1 !       and return both a pointer to where it
;  1394     3747  1 !       can now be referenced, as well as an ID
;  1395     3748  1 !       for it so that we can ask for it later.
;  1396     3749  1 !
;  1397     3750  1 ! FORMAL PARAMETERS:
;  1398     3751  1 !
;  1399     3752  1 !       REC_ID_PTR - the address of where this routine will
;  1400     3753  1 !                    stuff the ID it wants subsequent calls
;  1401     3754  1 !                    to PAT$GET_DST_REC to use to refer
;  1402     3755  1 !                    to the record fetched by this call.
;  1403     3756  1 !
;  1404     3757  1 ! IMPLICIT INPUTS:
;  1405     3758  1 !
;  1406     3759  1 !       To be defined.
;  1407     3760  1 !       (whatever context these routines work from).
;  1408     3761  1 !
;  1409     3762  1 ! IMPLICIT OUTPUTS:
;  1410     3763  1 !
;  1411     3764  1 !       none
;  1412     3765  1 !
;  1413     3766  1 ! COMPLETION CODES:
;  1414     3767  1 !
;  1415     3768  1 !       0, if the indicated record does not exist,
;  1416     3769  1 !       the address of where is can now be referenced, otherwise.
;  1417     3770  1 !
;  1418     3771  1 ! SIDE EFFECTS:
;  1419     3772  1 !
;  1420     3773  1 !       The DST record after the last one fetched is made available.
;  1421     3774  1 !       If no record has yet been fetched, the first record in
;  1422     3775  1 !       the DST is made available.
;  1423     3776  1 !
;  1424     3777  1 !--
;  1425     3778  1
;  1426     3779  2 BEGIN
;  1427     3780  2
;  1428     3781  2 MAP
;  1429     3782  2         REC_ID_PTR : REF VECTOR[,LONG];
;  1430     3783  2
;  1431     3784  2 !++
;  1432     3785  2 ! Since for us record IDs are the same as their virtual addresses, we can get
;  1433     3786  2 ! the next one the same way we can get ANY one.  The only detail to fill in is
;  1434     3787  2 ! passing back the ID for this next one.
;  1435     3788  2 !--
;  1436     3789  2 RETURN(REC_ID_PTR[0] = PAT$POSITON_DST( .DST_NEXT_ADDR ));
;  1437     3790  1 END;

                                0000 00000          .ENTRY  PAT$GET_NXT_DST, Save nothing                ; 3740
```

```
                    00000000'  EF  DD 00002          PUSHL   DST_NEXT_ADDR                          ; 3789
                91  AF             01  FB 00008      CALLS   #1, PAT$POSITON_DST                    ;
                04  BC             50  D0 0000C      MOVL    R0, @REC_ID_PTR                        ;
                                    04 00010         RET                                            ; 3790

; Routine Size:  17 bytes,    Routine Base:  _PAT$CODE + 04F4
```

```
;1439     3791  1 GLOBAL ROUTINE PAT$GET_NXT_GST (ACCESS_FLAG) =
;1440     3792  1
;1441     3793  1 !++
;1442     3794  1 ! Functional description:
;1443     3795  1 !
;1444     3796  1 !       This routine returns the address of a fixed length record that
;1445     3797  1 !       contains a global symbol name and its associated value. This
;1446     3798  1 !       routine expects to be called repeatedly until each global symbol
;1447     3799  1 !       has been returned to the caller.
;1448     3800  1 !
;1449     3801  1 !       Before this routine is ever called, the location of the GST in
;1450     3802  1 !       the image file is found, and it is mapped into PATCH's image.
;1451     3803  1 !       The address of this buffer is held in the OWN variable GST_BEGIN_ADDR.
;1452     3804  1 !       This routine analyzes the GST record, and moves through the buffer,
;1453     3805  1 !       returning the buffer address of each global symbol entry as it is seen.
;1454     3806  1 !       When the buffer is exhausted, this routine reads in the next GST record.
;1455     3807  1 !       It halts at end of file and returns a value of zero to the caller.
;1456     3808  1 !
;1457     3809  1 !       This routine keeps the variable GST_BEGIN_ADDR up to date.
;1458     3810  1 !
;1459     3811  1 !       The format of one of these concatenated records is a single
;1460     3812  1 !       leading byte containing the value 1, indicating that the record
;1461     3813  1 !       is indeed a GSD record. The variable GST_BEGIN_ADDR addresses
;1462     3814  1 !       the byte following this leading byte.
;1463     3815  1 !
;1464     3816  1 !       Each entry in the record has a fixed number of overhead
;1465     3817  1 !       bytes followed by a symbol name that is a variable number of
;1466     3818  1 !       bytes. The entries we are interested in processing are the
;1467     3819  1 !       global symbol definitions, entry point symbol and mask
;1468     3820  1 !       definitions, and procedure definitions with formal argument descriptions.
;1469     3821  1 !       The other defined type, PSECT definition, is noted only because it
;1470     3822  1 !       must be successfully passed over.  The format of each of these types is
;1471     3823  1 !       illustrated below:
;1472     3824  1 !
;1473     3825  1 !
;1474     3826  1 !       Global symbol definition:
;1475     3827  1 !
;1476     3828  1 !               ----------------
;1477     3829  1 !       0       !   GSD type 1   !
;1478     3830  1 !               ----------------
;1479     3831  1 !       1       !   data type    !       ignored for now
;1480     3832  1 !               ----------------
;1481     3833  1 !       2       !      flag      !       bit 1 set means that this is
;1482     3834  1 !       3       !      bytes     !       a definition. ignore bit 0.
;1483     3835  1 !               ----------------
;1484     3836  1 !       4       !  psect index   !       ignored.
;1485     3837  1 !               ----------------
;1486     3838  1 !       5       !                !
;1487     3839  1 !               !     value      !       4 bytes
;1488     3840  1 !               !                !
;1489     3841  1 !               ----------------
;1490     3842  1 !       9       !                !
;1491     3843  1 !               !    symbol      !       stock counted character
;1492     3844  1 !               !     name       !       string.
;1493     3845  1 !               !                !
;1494     3846  1 !               ----------------
;1495     3847  1 !
```

```
;  1496     3848  1 !       The entry point symbol and mask definition entry is identical to
;  1497     3849  1 !       the global symbol definition illustrated above, with the
;  1498     3850  1 !       addition of a two byte field for the procedure's register save
;  1499     3851  1 !       mask. This two byte field is located after the symbol value
;  1500     3852  1 !       field (which is an entry point address).
;  1501     3853  1 !
;  1502     3854  1 !                 ----------------
;  1503     3855  1 !       0         !  GSD type 2  !
;  1504     3856  1 !                 ----------------
;  1505     3857  1 !       1         !  data type   !       ignored for now
;  1506     3858  1 !                 ----------------
;  1507     3859  1 !       2         !     flag     !       not relevant for
;  1508     3860  1 !       3         !     bytes    !       entry point def.
;  1509     3861  1 !                 ----------------
;  1510     3862  1 !       4         ! psect index  !       ignored
;  1511     3863  1 !                 ----------------
;  1512     3864  1 !                 !              !
;  1513     3865  1 !       5         !     value    !       4 bytes
;  1514     3866  1 !                 !              !
;  1515     3867  1 !                 ----------------
;  1516     3868  1 !       9         !  register    !       ignored,
;  1517     3869  1 !       10        !  save mask   !       2 bytes
;  1518     3870  1 !                 ----------------
;  1519     3871  1 !       11        !              !
;  1520     3872  1 !                 !    symbol    !       stock counted character
;  1521     3873  1 !                 !     name     !       string
;  1522     3874  1 !                 !              !
;  1523     3875  1 !                 ----------------
;  1524     3876  1 !
;  1525     3877  1 !       The procedure definition with formal argument descriptions is
;  1526     3878  1 !       identical to the entry point with mask definition above, save
;  1527     3879  1 !       that it has some additional fields.  There is a minimum number
;  1528     3880  1 !       of arguments byte and a maximum number of arguments byte.  These
;  1529     3881  1 !       are followed by a formal argument description for each possible
;  1530     3882  1 !       argument (i.e., the maximum number).  The formal argument descriptions
;  1531     3883  1 !       consist of an argument value control byte and a remaining count byte.
;  1532     3884  1 !       The remaining count byte tells the number of bytes in the detailed
;  1533     3885  1 !       argument description (from 0 to 255).
;  1534     3886  1 !
;  1535     3887  1 !                 ----------------
;  1536     3888  1 !       0         !  GSD type 3  !
;  1537     3889  1 !                 ----------------
;  1538     3890  1 !       1         !  data type   !       ignored for now
;  1539     3891  1 !                 ----------------
;  1540     3892  1 !       2         !     flag     !       bit 1 set means that this is
;  1541     3893  1 !       3         !     bytes    !       a definition. ignore bit 0.
;  1542     3894  1 !                 ----------------
;  1543     3895  1 !       4         ! psect index  !       ignored
;  1544     3896  1 !                 ----------------
;  1545     3897  1 !                 !              !
;  1546     3898  1 !       5         !     value    !       4 bytes
;  1547     3899  1 !                 !              !
;  1548     3900  1 !                 ----------------
;  1549     3901  1 !       9         !  register    !       ignored,
;  1550     3902  1 !       10        !  save mask   !       2 bytes
;  1551     3903  1 !                 ----------------
;  1552     3904  1 !       11        !              !
```

```
                        ----------------
                        !    symbol    !        stock counted character
                        !     name     !        string
                        !              !
                        ----------------
                        ! min # act arg !        1 byte
                        ----------------
                        ! max # act arg !        1 byte
                        ----------------
                        ! formal arg #1 !
                        !  description  !
                        ----------------
                        !       .      !
                        !       .      !
                        !       .      !
                        ----------------
                        ! formal arg #n !
                        !  description  !
                        ----------------

    Each formal argument description has the following format:

                        ----------------
    0                   ! arg. val. ctl.!        1 byte
                        ----------------
    1                   ! rem. byte cnt.!        1 byte
                        ----------------
                        !   detailed     !        anywhere
                        !   argument     !        from 0-255
                        !  description   !        bytes
                        ----------------

    PSECT definition:

                        ----------------
    0                   !   GSD type 0   !
                        ----------------
    1                   !   alignment    !
                        ----------------
    2                   !      flag      !
    3                   !      bytes     !
                        ----------------
    4                   !                !
                        !   allocation   !        4 bytes
                        !                !
                        ----------------
    8                   !                !
                        !    symbol      !        stock counted character
                        !     name       !        string.
                        !                !
                        ----------------

    PSECT definition in a Shareable Image:

                        ----------------
    0                   !   GSD type 0   !
                        ----------------
```

```
;  1610          3962  1 !                 1       !   alignment   .
;  1611          3963  1 !                         ----------------
;  1612          3964  1 !                 2       !     flag      .
;  1613          3965  1 !                 3       !     bytes     .
;  1614          3966  1 !                         ----------------
;  1615          3967  1 !                 4       !               !
;  1616          3968  1 !                         !  allocation   !       4 bytes
;  1617          3969  1 !                         !               !
;  1618          3970  1 !                         ----------------
;  1619          3971  1 !                 8       !  base address !
;  1620          3972  1 !                         ! within Share- !       4 bytes
;  1621          3973  1 !                         !  able Image   !
;  1622          3974  1 !                         ----------------
;  1623          3975  1 !                 12      !               !
;  1624          3976  1 !                         !    symbol     !       stock counted character
;  1625          3977  1 !                         !    name       !       string.
;  1626          3978  1 !                         !               !
;  1627          3979  1 !                         ----------------
;  1628          3980  1 !
;  1629          3981  1 ! Calling sequence:
;  1630          3982  1 !
;  1631          3983  1 !       CALLS #0, PAT$GET_NXT_GST
;  1632          3984  1 !
;  1633          3985  1 ! Inputs:
;  1634          3986  1 !
;  1635          3987  1 !       none
;  1636          3988  1 !
;  1637          3989  1 ! Implicit inputs:
;  1638          3990  1 !
;  1639          3991  1 !       GST_BEGIN_ADDR  - Current address of record buffer
;  1640          3992  1 !
;  1641          3993  1 ! Outputs:
;  1642          3994  1 !
;  1643          3995  1 !       The address of the next global symbol entry, or 0, if EOF.
;  1644          3996  1 !
;  1645          3997  1 ! Implicit outputs:
;  1646          3998  1 !
;  1647          3999  1 !       GST_BEGIN_ADDR is updated to address the next entry.
;  1648          4000  1 !
;  1649          4001  1 ! Routine value:
;  1650          4002  1 !
;  1651          4003  1 !       An address or 0
;  1652          4004  1 !
;  1653          4005  1 ! Side effects:
;  1654          4006  1 !
;  1655          4007  1 !       Another record may be read in.
;  1656          4008  1 !--
;  1657          4009  1
;  1658          4010  2 BEGIN
;  1659          4011  2
;  1660          4012  2 LOCAL
;  1661          4013  2         OLD_ADDRESS : REF BLOCK [, BYTE];
;  1662          4014  2
;  1663          4015  2 LABEL
;  1664          4016  2         GET_RECORD;
;  1665          4017  2
;  1666          4018  3 IF (.ACCESS_FLAG NEQ 0)
```

```
;  1667          4019  2 THEN
;  1668          4020  3         BEGIN
;  1669          4021  4         IF ((GST_BEGIN_ADDR = POSITION_GST(-1)) EQL 0)
;  1670          4022  3         THEN
;  1671          4023  3                 GST_BEGIN_ADDR = %X'FFFFFFFF';
;  1672          4024  3         RETURN(0);
;  1673          4025  2         END;
;  1674          4026  2
;  1675          4027  2 !++
;  1676          4028  2 ! See whether the current buffer address is beyond the
;  1677          4029  2 ! end of the last GST record we looked at.  Note that we
;  1678          4030  2 ! rounded up GSR_NEXT_ADDR when calculating where the next
;  1679          4031  2 ! GST record will begin.  Therefore we must temporarily round
;  1680          4032  2 ! it down again when comparing it with GST_BEGIN_ADDR since it
;  1681          4033  2 ! may point to the last unused byte in a GST record.
;  1682          4034  2 !--
;  1683          4035  2 REPEAT
;  1684          4036  2 GET_RECORD:
;  1685          4037  3         BEGIN
;  1686          4038  3         !++
;  1687          4039  3         ! First check that there is a GST in this image.
;  1688          4040  3         !--
;  1689          4041  4         IF (.GST_BEGIN_ADDR EQL 0)
;  1690          4042  3         THEN
;  1691          4043  3                 RETURN(0);
;  1692          4044  3
;  1693          4045  4         IF (.GST_BEGIN_ADDR GEQA .GSR_NEXT_ADDR-1)
;  1694          4046  3         THEN
;  1695          4047  4                 BEGIN
;  1696          4048  4                 !++
;  1697          4049  4                 ! Record was finished.  Check that there are more records.
;  1698          4050  4                 ! If so, then get another record.
;  1699          4051  4                 !--
;  1700          4052  5                 IF ((GST_BEGIN_ADDR = POSITION_GST(0)) EQL 0)
;  1701          4053  4                 THEN
;  1702          4054  5                         RETURN(0)
;  1703          4055  4                 ELSE
;  1704          4056  5                         BEGIN
;  1705          4057  5                         !++
;  1706          4058  5                         ! If the next record is a GST record, then initialize
;  1707          4059  5                         ! the variable GST_BEGIN_ADDR to point to the first
;  1708          4060  5                         ! global symbol definition block in this record.
;  1709          4061  5                         !--
;  1710          4062  5                         LOCAL
;  1711          4063  5                                 BUFFER_ADDRESS : REF VECTOR [, BYTE];
;  1712          4064  5
;  1713          4065  5                         BUFFER_ADDRESS = .GST_BEGIN_ADDR;
;  1714          4066  5                         IF .BUFFER_ADDRESS [GST_RECORD_TYPE] EQL GST_TYPE
;  1715          4067  5                         THEN
;  1716          4068  5                                 GST_BEGIN_ADDR = .GST_BEGIN_ADDR + 1
;  1717          4069  5                         ELSE
;  1718          4070  6                                 BEGIN
;  1719          4071  6                                 !++
;  1720          4072  6                                 ! This record is not a GST record.
;  1721          4073  6                                 ! Go on to the next.
;  1722          4074  6                                 !--
;  1723          4075  6                                 GST_BEGIN_ADDR = %X'FFFFFFFF';
```

```
;  1724     4076  6                        LEAVE GET_RECORD;
;  1725     4077  5                        END;
;  1726     4078  4                END;
;  1727     4079  4            END
;  1728     4080  3        ELSE
;  1729     4081  4            BEGIN
;  1730     4082  4            !++
;  1731     4083  4            ! This is a global symbol. Save its address.
;  1732     4084  4            ! Then update the variable GST_BEGIN_ADDR to
;  1733     4085  4            ! point to the next symbol.
;  1734     4086  4            !--
;  1735     4087  4            OLD_ADDRESS = .GST_BEGIN_ADDR;
;  1736     4088  4            CASE .OLD_ADDRESS [ENTRY_TYPE] FROM GSD$C_PSC TO GSD$C_SPSC OF
;  1737     4089  4
;  1738     4090  4                    SET
;  1739     4091  4
;  1740     4092  4                    [GSD$C_PSC]:
;  1741     4093  5                        BEGIN
;  1742     4094  5                        GST_BEGIN_ADDR = .OLD_ADDRESS +
;  1743     4095  6                                         (OLD_ADDRESS[GPS$T_NAME] - OLD_ADDRESS[GPS$T_START])
;  1744     4096  5                                         + .OLD_ADDRESS [GPS$B_NAMLNG];
;  1745     4097  4                        END;
;  1746     4098  4
;  1747     4099  4
;  1748     4100  4                    [GSD$C_SYM]:
;  1749     4101  5                        BEGIN
;  1750     4102  5                        GST_BEGIN_ADDR = .OLD_ADDRESS +
;  1751     4103  6                                         (OLD_ADDRESS[SDF$T_NAME] - OLD_ADDRESS[SDF$T_START])
;  1752     4104  5                                         + .OLD_ADDRESS [SDF$B_NAMLNG];
;  1753     4105  5                        RETURN .OLD_ADDRESS
;  1754     4106  4                        END;
;  1755     4107  4
;  1756     4108  4
;  1757     4109  4                    [GSD$C_EPM]:
;  1758     4110  5                        BEGIN
;  1759     4111  5                        GST_BEGIN_ADDR = .OLD_ADDRESS +
;  1760     4112  6                                         (OLD_ADDRESS[EPM$T_NAME] - OLD_ADDRESS[EPM$T_START])
;  1761     4113  5                                         + .OLD_ADDRESS [EPM$B_NAMLNG];
;  1762     4114  5                        RETURN .OLD_ADDRESS
;  1763     4115  4                        END;
;  1764     4116  4
;  1765     4117  4
;  1766     4118  4                    [GSD$C_PRO]:
;  1767     4119  5                        BEGIN
;  1768     4120  5                        LOCAL
;  1769     4121  5                                NUM_ARGS;               ! Max formal args
;  1770     4122  5                        GST_BEGIN_ADDR = .OLD_ADDRESS +
;  1771     4123  6                                         (OLD_ADDRESS[EPM$T_NAME] - OLD_ADDRESS[EPM$T_START])
;  1772     4124  5                                         + .OLD_ADDRESS [EPM$B_NAMLNG];
;  1773     4125  5                        NUM_ARGS = .GST_BEGIN_ADDR[GST_P_MAX_ARG];
;  1774     4126  5                        GST_BEGIN_ADDR = .GST_BEGIN_ADDR + MINMAX_OVERHEAD;
;  1775     4127  6                        WHILE (.NUM_ARGS GTR 0)
;  1776     4128  5                        DO
;  1777     4129  6                                BEGIN
;  1778     4130  6                                GST_BEGIN_ADDR = .GST_BEGIN_ADDR +
;  1779     4131  6                                                 .GST_BEGIN_ADDR[GST_P_REM_CNT] + ARGDSC_OVERHEAD;
;  1780     4132  6                                NUM_ARGS = .NUM_ARGS - 1;
```

```
;  1781          4133  5                                   END;
;  1782          4134  5                           RETURN .OLD_ADDRESS
;  1783          4135  4                           END;
;  1784          4136  4
;  1785          4137  4
;  1786          4138  4                       [GSD$C_SPSC]:
;  1787          4139  5                           BEGIN
;  1788          4140  5                           GST_BEGIN_ADDR = .OLD_ADDRESS +
;  1789          4141  6                                   (OLD_ADDRESS[SGPS$T_NAME] - OLD_ADDRESS[SGPS$T_START])
;  1790          4142  5                                   + .OLD_ADDRESS [SGPS$B_NAMLNG];
;  1791          4143  4                           END;
;  1792          4144  4
;  1793          4145  4
;  1794          4146  4                       [INRANGE]:
;  1795          4147  5                           BEGIN
;  1796          4148  5                           GST_BEGIN_ADDR = %X'FFFFFFFF';
;  1797          4149  4                           END;
;  1798          4150  4
;  1799          4151  4                       [OUTRANGE]:
;  1800          4152  5                           BEGIN
;  1801          4153  5                           GST_BEGIN_ADDR = %X'FFFFFFFF';
;  1802          4154  4                           END;
;  1803          4155  4
;  1804          4156  4                       TES;
;  1805          4157  4
;  1806          4158  3                   END;
;  1807          4159  2           END;
;  1808          4160  1 END;
;  INFO#212                 L1:4025
;  Null expression appears in value-required context
```

```
                                    003C 00000          .ENTRY  PAT$GET_NXT_GST, Save R2,R3,R4,R5                ; 3791
                              55    A4   AF   9E 00002   MOVAB   POSITION_GST, R5                                ;
                              54 00000000'EF   9E 00006   MOVAB   GST_BEGIN_ADDR, R4                            ;
                                    04   AC   D5 0000D   TSTL    ACCESS_FLAG                                     ; 4018
                                         11   13 00010   BEQL    2$                                              ;
                              7E         01   CE 00012   MNEGL   #1, -(SP)                                       ; 4021
                              65         01   FB 00015   CALLS   #1, POSITION_GST                                ;
                              64         50   D0 00018   MOVL    R0, GST_BEGIN_ADDR                              ;
                                         03   12 0001B   BNEQ    1$                                              ;
                              64         01   CE 0001D   MNEGL   #1, GST_BEGIN_ADDR                             ; 4023
                                   00D2  31   00020 1$:  BRW     15$                                             ; 4024
                              50         64   D0 00023 2$:  MOVL    GST_BEGIN_ADDR, R0                          ; 4041
                                         F8   13 00026   BEQL    1$                                              ;
                        51    FC   A4    01   C3 00028   SUBL3   #1, GSR_NEXT_ADDR, R1                         ; 4045
                              51         50   D1 0002D   CMPL    R0, R1                                          ;
                                         16   1F 00030   BLSSU   3$                                              ;
                                         7E   D4 00032   CLRL    -(SP)                                           ; 4052
                              65         01   FB 00034   CALLS   #1, POSITION_GST                                ;
                              64         50   D0 00037   MOVL    R0, GST_BEGIN_ADDR                              ;
                                         E4   13 0003A   BEQL    1$                                              ;
                              50         64   D0 0003C   MOVL    GST_BEGIN_ADDR, BUFFER_ADDRESS                  ; 4065
                              01         60   91 0003F   CMPB    (BUFFER_ADDRESS), #1                            ; 4066
```

```
                                                  25  12 00042          BNEQ    5$                                          ; 4068
                                                  64  D6 00044          INCL    GST_BEGIN_ADDR                              ; 4087
                                                  DB  11 00046          BRB     2$                                          ; 4088
                                      52          50  D0 00048 3$:      MOVL    R0, OLD_ADDRESS                             ;
                                      0C          62  8F 0004B          CASEB   (OLD_ADDRESS), #0, #12                      ;
                0059          0045          0031      0004F 4$:      .WORD   6$-4$,-                                     ;
                00A1          00A1          00A1      00057                  8$-4$,-                                     ;
                00A1          00A1          00A1      0005F                  9$-4$,-                                     ;
                                                      0067                   10$-4$,-                                    ;
                                                                             14$-4$,-                                    ;
                                                                             14$-4$,-                                    ;
                                                                             14$-4$,-                                    ;
                                                                             14$-4$,-                                    ;
                                                                             14$-4$,-                                    ;
                                                                             14$-4$,-                                    ;
                                                                             14$-4$,-                                    ;
                                                                             14$-4$,-                                    ;
                                                                             13$-4$                                      ;
                                                  31  00069 5$:      BRW     14$                                         ; 4153
                        50            52            52  C3 0006C 6$:      SUBL3   OLD_ADDRESS, OLD_ADDRESS, R0                ; 4095
                                      50            52  C0 00070          ADDL2   OLD_ADDRESS, R0                             ; 4094
                                      51      08    A2  9A 00073          MOVZBL  8(OLD_ADDRESS), R1                          ; 4096
                                      50            51  C0 00077          ADDL2   R1, R0                                      ;
                                      64      09    A0  9E 0007A          MOVAB   9(R0), GST_BEGIN_ADDR                       ;
                                                  A3  11 0007E 7$:       BRB     2$                                          ; 4088
                        50            52            52  C3 00080 8$:      SUBL3   OLD_ADDRESS, OLD_ADDRESS, R0                ; 4103
                                      50            52  C0 00084          ADDL2   OLD_ADDRESS, R0                             ; 4102
                                      51      09    A2  9A 00087          MOVZBL  9(OLD_ADDRESS), R1                          ; 4104
                                      50            51  C0 0008B          ADDL2   R1, R0                                      ;
                                      64      0A    A0  9E 0008E          MOVAB   10(R0), GST_BEGIN_ADDR                      ;
                                                  44  11 00092          BRB     12$                                         ; 4105
                        50            52            52  C3 00094 9$:      SUBL3   OLD_ADDRESS, OLD_ADDRESS, R0                ; 4112
                                      50            52  C0 00098          ADDL2   OLD_ADDRESS, R0                             ; 4111
                                      51      0B    A2  9A 0009B          MOVZBL  11(OLD_ADDRESS), R1                         ; 4113
                                      50            51  C0 0009F          ADDL2   R1, R0                                      ;
                                      64      0C    A0  9E 000A2          MOVAB   12(R0), GST_BEGIN_ADDR                      ;
                                                  30  11 000A6          BRB     12$                                         ; 4114
                        50            52            52  C3 000A8 10$:     SUBL3   OLD_ADDRESS, OLD_ADDRESS, R0                ; 4123
                                      50            52  C0 000AC          ADDL2   OLD_ADDRESS, R0                             ; 4122
                                      51      0B    A2  9A 000AF          MOVZBL  11(OLD_ADDRESS), R1                         ; 4124
                                      50            51  C0 000B3          ADDL2   R1, R0                                      ;
                                      64      0C    A0  9E 000B6          MOVAB   12(R0), GST_BEGIN_ADDR                      ;
                                      50            64  D0 000BA          MOVL    GST_BEGIN_ADDR, R0                          ; 4125
                                      53      01    A0  9A 000BD          MOVZBL  1(R0), NUM_ARGS                             ;
                                      64            02  C0 000C1          ADDL2   #2, GST_BEGIN_ADDR                          ; 4126
                                                  53  D5 000C4 11$:      TSTL    NUM_ARGS                                    ; 4127
                                                  10  15 000C6          BLEQ    12$                                         ;
                                      51          64  D0 000C8          MOVL    GST_BEGIN_ADDR, R1                          ; 4130
                                      50      01  A1  9A 000CB          MOVZBL  1(R1), R0                                   ; 4131
                                      64      02 A140 9E 000CF          MOVAB   2(R1)[R0], GST_BEGIN_ADDR                   ;
                                                  53  D7 000D4          DECL    NUM_ARGS                                    ; 4132
                                                  EC  11 000D6          BRB     11$                                         ; 4127
                                      50          52  D0 000D8 12$:      MOVL    OLD_ADDRESS, R0                             ; 4134
                                                  04 000DB              RET                                                 ;
                        50            52            52  C3 000DC 13$:      SUBL3   OLD_ADDRESS, OLD_ADDRESS, R0                ; 4141
                                      50            52  C0 000E0          ADDL2   OLD_ADDRESS, R0                             ; 4140
                                      51      0C    A2  9A 000E3          MOVZBL  12(OLD_ADDRESS), R1                         ; 4142
```

```
                50              51  C0 000E7          ADDL2   R1, R0                                          ;
                64        0D    A0  9E 000EA          MOVAB   13(R0), GST_BEGIN_ADDR                          ;
                                8E  11 000EE          BRB     7$                                              ; 4088
                64              01  CE 000F0 14$:     MNEGL   #1, GST_BEGIN_ADDR                              ; 4148
                                89  11 000F3          BRB     7$                                              ; 4025
                                50  D4 000F5 15$:     CLRL    R0                                              ; 4160
                                    04 000F7          RET                                                     ;

; Routine Size:  248 bytes,    Routine Base:  _PAT$CODE + 0505
```

```
;  18  0          4161  1 GLOBAL ROUTINE PAT$RST_FREEZ ( UNITS ) =
;  1811           4162  1
;  1812           4163  1 !++
;  1813           4164  1 ! FUNCTIONAL DESCRIPTION:
;  1814           4165  1 !
;  1815           4166  1 !       Isolate storage allocation for the RST builder/manipulator.
;  1816           4167  1 !       i.e. Do exactly what PAT$FREEZ does for the rest of
;  1817           4168  1 !       PATCH, but take care of any differences (which may
;  1818           4169  1 !       or may not exits), when it is the RST interface
;  1819           4170  1 !       which wants the storage.
;  1820           4171  1 !
;  1821           4172  1 !       For now, there IS a difference - an RST-pointer is
;  1822           4173  1 !       returned, NOT the usual longword pointer.  RST-pointers
;  1823           4174  1 !       are something internal to the RST builder/manipulator,
;  1824           4175  1 !       and it doesn't want to ever see anything but RST-pointers
;  1825           4176  1 !       (even if someday RST-pointers are the same thing as
;  1826           4177  1 !       virtual addresses).  This is really the motivation for
;  1827           4178  1 !       having PAT$RST_FREEZ.
;  1828           4179  1 !
;  1829           4180  1 ! Formal Parameters:
;  1830           4181  1 !
;  1831           4182  1 !       UNITS - the number of units of storage which are
;  1832           4183  1 !               required.  This unit will remain whatever
;  1833           4184  1 !               unit PAT$FREEZ knows about.
;  1834           4185  1 !
;  1835           4186  1 ! Implicit Inputs:
;  1836           4187  1 !
;  1837           4188  1 !       See PAT$FREEZ
;  1838           4189  1 !
;  1839           4190  1 ! Implicit Outputs:
;  1840           4191  1 !
;  1841           4192  1 !       See PAT$FREEZ
;  1842           4193  1 !
;  1843           4194  1 ! Rc tine Value:
;  1844           4195  1 !
;  1845           4196  1 !       0, if something goes wrong, an RST-pointer to the
;  1846           4197  1 !       allocated storage, otherwise.
;  1847           4198  1 !
;  1848           4199  1 ! Side Effects:
;  1849           4200  1 !
;  1850           4201  1 !       See PAT$FREEZ
;  1851           4202  1 !--
;  1852           4203  1
;  1853           4204  2 BEGIN
;  1854           4205  2 LOCAL
;  1855           4206  2         STORAGE_PTR;
;  1856           4207  2
;  1857           4208  2 STORAGE_PTR = PAT$FREEZ( .UNITS );
;  1858           4209  2
;  1859           4210  2 !++
;  1860           4211  2 ! Currently an RST-pointer is just like a virtual
;  1861           4212  2 ! address except that the top 16 bits are 0 in the
;  1862           4213  2 ! former, and hex 7FFF0000 in the latter.
;  1863           4214  2 ! NOTE:  THIS IS ONLY TRUE IF THE DEBUGGER INDICATOR IS TURNED OFF IN
;  1864           4215  2 ! PAT$FREE_INIT.  IF IT IS TURNED ON, THEN THE STORAGE IS OWN STORAGE, NOT
;  1865           4216  2 ! CONTAINED IN SYSTEM SPACE.
;  1866           4217  2 !--
```

```
;  1867          4218  2 RETURN( .STORAGE_PTR - .PAT$GL_RST_BEGN );
;  1868          4219  2
;  1869          4220  1 END;


                                                0000 00000          .ENTRY  PAT$RST_FREEZ, Save nothing                     ; 4161
                                          04    AC   DD 00002       PUSHL   UNITS                                           ; 4208
                                00000000G   EF         01   FB 00005       CALLS   #1, PAT$FREEZ                                   ;
                                          50 00000000G  EF   C2 0000C       SUBL2   PAT$GL_RST_BEGN, R0                             ; 4218
                                                        04   00013       RET                                                     ; 4220

; Routine Size:  20 bytes,    Routine Base:  _PAT$CODE + 05FD
```

```
;  1871          4221  1 GLOBAL ROUTINE PAT$RST_RELEASE ( RST_PTR, SIZE ) : NOVALUE =
;  1872          4222  1
;  1873          4223  1 !++
;  1874          4224  1 ! FUNCTIONAL DESCRIPTION:
;  1875          4225  1 !
;  1876          4226  1 !       Isolate storage deallocation for all storage which
;  1877          4227  1 !       is accessed via RST-pointers.
;  1878          4228  1 !
;  1879          4229  1 !       i.e. Do exactly what PAT$FREERELEASE does for the rest of
;  1880          4230  1 !       PATCH, but take care of any differences (which may
;  1881          4231  1 !       or may not exits), when it is the RST interface
;  1882          4232  1 !       which wants to free up this special-access storage.
;  1883          4233  1 !
;  1884          4234  1 !       For now, there IS a difference - an RST-pointer is
;  1885          4235  1 !       given to indicate which storage to free up.  This makes
;  1886          4236  1 !       PAT$RST_RELEASE the inverse of PAT$RST_FREEZ, just
;  1887          4237  1 !       as is true for the standard PATCH storage primitives.
;  1888          4238  1 !
;  1889          4239  1 ! Formal Parameters:
;  1890          4240  1 !
;  1891          4241  1 !  RST_PTR - this indicates which storage
;  1892          4242  1 !                 is to be freed.  This must be the same as
;  1893          4243  1 !                 one which was returned by DBG$RST_FREEZ.
;  1894          4244  1 !  SIZE          -The number of units which corresponds
;  1895          4245  1 !                 to the storage to be freed.
;  1896          4246  1 !
;  1897          4247  1 ! Implicit Inputs:
;  1898          4248  1 !
;  1899          4249  1 !       See PAT$FREEZ
;  1900          4250  1 !
;  1901          4251  1 ! Implicit Outputs:
;  1902          4252  1 !
;  1903          4253  1 !       See PAT$FREEZ
;  1904          4254  1 !
;  1905          4255  1 ! Routine Value
;  1906          4256  1 !
;  1907          4257  1 !       NOVALUE
;  1908          4258  1 !
;  1909          4259  1 ! Side Effects:
;  1910          4260  1 !
;  1911          4261  1 !       See PAT$FREEZ
;  1912          4262  1 !--
;  1913          4263  1
;  1914          4264  2 BEGIN
;  1915          4265  2
;  1916          4266  2 !++
;  1917          4267  2 ! Currently an RST-pointer is just like a virtual
;  1918          4268  2 ! address except that the top 16 bits are 0 in
;  1919          4269  2 ! in the former and hex 7FFF0000 in the latter.
;  1920          4270  2 !--
;  1921          4271  2 PAT$FREERELEASE( .RST_PTR + .PAT$GL_RST_BEGN, .SIZE );
;  1922          4272  1 END;
```

```
                                          0000 00000         .ENTRY  PAT$RST_RELEASE, Save nothing                ; 4221
                                    08    AC   DD 00002      PUSHL   SIZE                                         ; 4271
                          7E        04    AC 00000000G  EF   C1 00005      ADDL3   PAT$GL_RST_BEGN, RST_PTR, -(SP)
                             00000000G  EF              02   FB 0000E      CALLS   #2, PAT$FREERELEASE
                                                        04 00015         RET                                         ; 4272

; Routine Size:  22 bytes,    Routine Base:  _PAT$CODE + 0611
```

```
; 1924          4273  1 END
; 1925          4274  0 ELUDOM                                    ! End of module


                                                                .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY

;       Name                    Bytes                           Attributes

;   _PAT$OWN                      44  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   _PAT$CODE                   1575  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$PLIT                      8  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        32         0      1000          00:01.8


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


;                                       COMMAND QUALIFIERS

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

; Size:         1575 code + 52 data bytes
; Run Time:        00:47.7
; Elapsed Time:    02:43.1
; Lines/CPU Min:    5378
; Lexemes/CPU-Min: 30094
; Memory Used:  252 pages
; Compilation Complete
```

PATINT
LIS

PATINS
LIS

PATERR
LIS

PATIHD
LIS

PATEXA
LIS

PATFRE
LIS

PATPAR
LIS

PATMAC
LIS

PATMAI
LIS

PATLST
LIS

PATIO
LIS

PATLEX
LIS

PATMOD
LIS

PATMSG
LIS