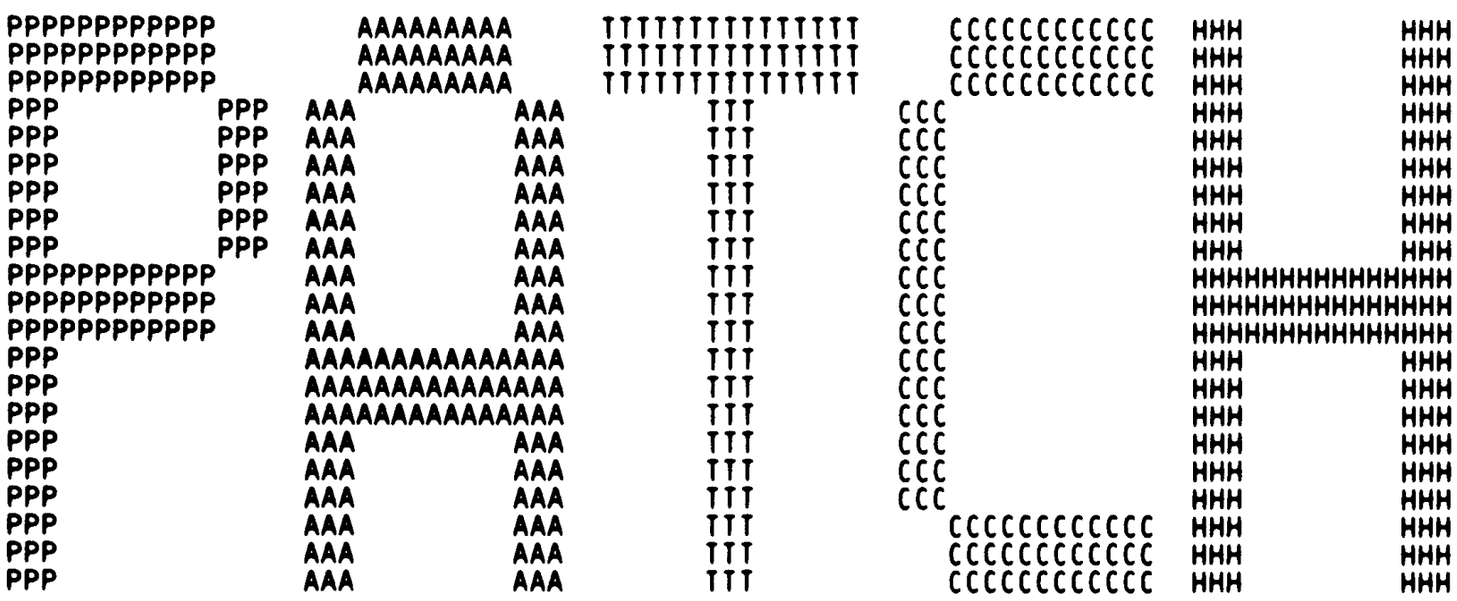
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPPPPPPPPPPP AAAAAAAAAAAAAAAA TTT CCC HHHHHHHHHHHHHHHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCCC HHH HHH

FILEID**PATVEC

```
PPPPPPPP      AAAAAA    TTTTTTTTTT  VV      VV  EEEEEEEEEE    CCCCCCCC
PPPPPPPP      AAAAAA    TTTTTTTTTT  VV      VV  EEEEEEEEEE    CCCCCCCC
PP      PP   AA    AA       TT      VV      VV  EE           CC
PP      PP   AA    AA       TT      VV      VV  EE           CC
PP      PP   AA    AA       TT      VV      VV  EE           CC
PP      PP   AA    AA       TT      VV      VV  EE           CC
PPPPPPPP     AA    AA       TT      VV      VV  EEEEEEEE     CC
PPPPPPPP     AA    AA       TT      VV      VV  EEEEEEEE     CC
PP           AAAAAAAAAA     TT      VV      VV  EE           CC
PP           AAAAAAAAAA     TT      VV      VV  EE           CC
PP           AA    AA       TT        VV  VV    EE           CC                ....
PP           AA    AA       TT        VV  VV    EE           CC                ....
PP           AA    AA       TT          VV      EEEEEEEEEE    CCCCCCCC       ....
PP           AA    AA       TT          VV      EEEEEEEEEE    CCCCCCCC       ....

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 PATVEC ( !
;  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 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                               *
; 13          0013  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
; 14          0014  1 !*  ALL RIGHTS RESERVED.                                                   *
; 15          0015  1 !*                                                                        *
; 16          0016  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 17          0017  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 18          0018  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 19          0019  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 20          0020  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 21          0021  1 !*  TRANSFERRED.                                                           *
; 22          0022  1 !*                                                                        *
; 23          0023  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 24          0024  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 25          0025  1 !*  CORPORATION.                                                           *
; 26          0026  1 !*                                                                        *
; 27          0027  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 28          0028  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
; 29          0029  1 !*                                                                        *
; 30          0030  1 !*                                                                        *
; 31          0031  1 !**************************************************************************
; 32          0032  1 !
; 33          0033  1
; 34          0034  1 !++
; 35          0035  1 ! FACILITY:     PATCH
; 36          0036  1 !
; 37          0037  1 ! ABSTRACT:     Handle so-called 'vector storage' in conjunction
; 38          0038  1 !               with the RST build and rebuild modules.
; 39          0039  1 !
; 40          0040  1 ! ENVIRONMENT:  This module runs on VAX under VAX/VMS, user mode, non-AST level.
; 41          0041  1 !
; 42          0042  1 ! Author:       Kevin Pammett, August 12, 1977.
; 43          0043  1 !
; 44          0044  1 ! Version:      X01.03
; 45          0045  1 !
; 46          0046  1 ! MODIFCATIONS:
; 47          0047  1 !
; 48          0048  1 !       V03-001 MTR0012         Mike Rhodes             16-Aug-1982
; 49          0049  1 !               Modify file names to remove duplicate file name useage
; 50          0050  1 !               between code and require files.
; 51          0051  1 !
; 52          0052  1 !  NO   DATE            PROGRAMMER              PURPOSE
; 53          0053  1 !  --   ----            ----------              -------
; 54          0054  1 !
; 55          0055  1 !  00   21-DEC-77       K.D. MORSE              ADAPT VERSION 35 FOR PATCH.
; 56          0056  1 !  01   25-APR-78       K.D. MORSE              CONVERT TO NATIVE COMPILER.
; 57          0057  1 !  02   18-MAY-78       K.D. MORSE              NO CHANGES FOR VERS 36.
```

```
;   58          0058  1 !  03   13-JUN-78       K.D. MORSE          ADD FAO COUNTS TO SIGNALS.
;   59          0059  1 !
;   60          0060  1 !--
```

**F

```
;    62          0061  1 !
;    63          0062  1 ! TABLE OF CONTENTS:
;    64          0063  1 !
;    65          0064  1
;    66          0065  1 FORWARD ROUTINE
;    67          0066  1         PAT$VS_INIT : NOVALUE,                  ! Initialize a vector storage area.
;    68          0067  1         PAT$VS_GET,                             ! Allocate records from a so-called
;    69          0068  1                                                 !  "vector storage" vector.
;    70          0069  1         PAT$VS_FREE : NOVALUE,                  ! Release vector storage.
;    71          0070  1         PAT$VS_SHRINK : NOVALUE;                ! Free up unused vector storage.
;    72          0071  1
;    73          0072  1 !
;    74          0073  1 ! INCLUDE FILES:
;    75          0074  1 !
;    76          0075  1
;    77          0076  1 LIBRARY 'SYS$LIBRARY:STARLET.L32';
;    78          0077  1 REQUIRE 'SRC$:PATPCT.REQ';
;    79          0117  1 REQUIRE 'SRC$:VXSMAC.REQ';
;    80          0182  1 REQUIRE 'SRC$:PATRTS.REQ';
;    81          1278  1 REQUIRE 'SRC$:SYSSER.REQ';
```

```
;  R1310  1      SWITCHES LIST (SOURCE);
;  R1311  1
;  R1312  1      EXTERNAL ROUTINE
;  R1313  1              PAT$fao_out;            ! formats a line and outputs to the terminal
;  R1314  1
```

```
;    82          1360  1
;    83          1361  1 !
;    84          1362  1 ! MACROS:
;    85          1363  1 !
;    86          1364  1
;    87          1365  1 !
;    88          1366  1 ! EQUATED SYMBOLS:
;    89          1367  1 !
;    90          1368  1
;    91          1369  1
;    92          1370  1 !
;    93          1371  1 ! EXTERNAL REFERENCES:
;    94          1372  1 !
;    95          1373  1
;    96          1374  1 EXTERNAL ROUTINE
;    97          1375  1         PAT$RST_FREEZ,                  ! Allocate and zero RST-pointer storage.
;    98          1376  1         PAT$RST_RELEASE;                ! Free up RST-pointer storage.
;    99          1377  1
;   100          1378  1 EXTERNAL
;   101          1379  1         PAT$GL_RST_BEGN;                ! STARTING ADDRESS OF RST
```

```
;  103     1380  1 GLOBAL ROUTINE PAT$VS_INIT( STORE_DESC_ADDR, NUM_ELEMENTS, ELEMENT_SIZE ) : NOVALUE =
;  104     1381  1
;  105     1382  1 !++
;  106     1383  1 ! Functional Description:
;  107     1384  1 !
;  108     1385  1 !       Initialize storage for a so-called "storage vector".
;  109     1386  1 !       See DBGRST.REQ for a description of these structures.
;  110     1387  1 !
;  111     1388  1 ! Formal Parameters:
;  112     1389  1 !
;  113     1390  1 !       STORE_DESC_ADDR -The address of the storage vector descriptor
;  114     1391  1 !                        which this routine is to initialize.
;  115     1392  1 !       NUM_ELEMENTS    -The maximum number of elements which will
;  116     1393  1 !                        ever be allocated from this vector.
;  117     1394  1 !       ELEMENT_SIZE    -The maximum size, in bytes, of each element.
;  118     1395  1 !
;  119     1396  1 ! Implicit Inputs:
;  120     1397  1 !
;  121     1398  1 !       Since we assume that records are fixed-length,
;  122     1399  1 !       we actually allocate more storage than
;  123     1400  1 !       we will really need.  This will be fixed up when we
;  124     1401  1 !       can 'shrink' RST storage.
;  125     1402  1 !
;  126     1403  1 ! Routine Value:
;  127     1404  1 !
;  128     1405  1 !       NOVALUE - because if we run out of storage (the only
;  129     1406  1 !                 thing which can go wrong), we don't get
;  130     1407  1 !                 control back anyways.
;  131     1408  1 !
;  132     1409  1 ! Side Effects:
;  133     1410  1 !
;  134     1411  1 !       The storage descriptor is initialized in such
;  135     1412  1 !       a way that GET_VEC_STORAGE can do its thing.
;  136     1413  1 !
;  137     1414  1 !       RST storage is allocated.
;  138     1415  1 !--
;  139     1416  1
;  140     1417  2 BEGIN
;  141     1418  2
;  142     1419  2 MAP
;  143     1420  2         STORE_DESC_ADDR : REF VECT_STORE_DESC;                  ! We get passed a pointer to the storage
;  144     1421  2                                                                ! descriptor for the module we are
;  145     1422  2                                                                ! initializing vector storage for.
;  146     1423  2
;  147     1424  2 LOCAL
;  148     1425  2         STORAGE : REF MC_RECORD;                                ! Pointer to the actual storage
;  149     1426  2                                                                ! we allocate.
;  150     1427  2
;  151     1428  2 !++
;  152     1429  2 ! Ask for the necessary RST storage.   Note that we don't get control back
;  153     1430  2 ! if the requisition is denied.
;  154     1431  2 !--
;  155     1432  2 STORAGE = PAT$RST_FREEZ( RST_UNITS(.ELEMENT_SIZE * .NUM_ELEMENTS));
;  156     1433  2
;  157     1434  2 !++
;  158     1435  2 ! Initialize the vector storage descriptor. The 'marker' starts out being
;  159     1436  2 ! the same as the beginning address since initially the entire vector is
```

```
;  160          1437  2 ! available for re-allocation.
;  161          1438  2 !--
;  162          1439  2 STORE_DESC_ADDR[ STOR_BEGIN_RST ] = .STORAGE;
;  163          1440  2 STORE_DESC_ADDR[ STOR_MARKER ] = .STORAGE,
;  164          1441  2
;  165          1442  2 !++
;  166          1443  2 ! The end RST address is calculated by taking into account that the standard
;  167          1444  2 ! PATCH storage manager works in LONGWORDs and rounds up.
;  168          1445  2 !--
;  169          1446  2 STORE_DESC_ADDR[ STOR_END_RST ] = .STORAGE
;  170          1447  2                 + (RST_UNITS(.ELEMENT_SIZE*.NUM_ELEMENTS) * %UPVAL);
;  171          1448  1 END;


                                                        .TITLE  PATVEC
                                                        .IDENT  \V04-000\

                                                        .EXTRN  PAT$FAO_OUT, PAT$RST_FREEZ
                                                        .EXTRN  PAT$RST_RELEASE
                                                        .EXTRN  PAT$GL_RST_BEGN

                                                        .PSECT  _PAT$CODE,NOWRT,2

                                      000C 00000        .ENTRY  PAT$VS_INIT, Save R2,R3                 ; 1380
                52       0C   AC     08   AC  C5 00002  MULL3   NUM_ELEMENTS, ELEMENT_SIZE, R2          ; 1432
                              52           03  C0 00008  ADDL2   #3, R2                                 ;
                              52           04  C6 0000B  DIVL2   #4, R2                                 ;
                                           52  DD 0000E  PUSHL   R2                                     ;
                00000000G     EF           01  FB 00010  CALLS   #1, PAT$RST_FREEZ                     ;
                              51           04  AC  D0 00017  MOVL    STORE_DESC_ADDR, R1                 ; 1439
                         01   A1           50  B0 0001B  MOVW    STORAGE, 1(R1)                      ;
                         05   A1           50  B0 0001F  MOVW    STORAGE, 5(R1)                      ; 1440
                              53         6042  DE 00023  MOVAL   (STORAGE)[R2], R3                   ; 1447
                         03   A1           53  B0 00027  MOVW    R3, 3(R1)                           ;
                                             04 0002B    RET                                         ; 1448

; Routine Size:  44 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
;  173     1449  1 GLOBAL ROUTINE PAT$VS_FREE( STORE_DESC_ADDR ) : NOVALUE =
;  174     1450  1
;  175     1451  1 !++
;  176     1452  1 ! Functional Description:
;  177     1453  1 !
;  178     1454  1 !       Release storage for a so-called "storage vector".
;  179     1455  1 !       See PATRST.REQ for a description of these structures.
;  180     1456  1 !
;  181     1457  1 ! Formal Parameters:
;  182     1458  1 !
;  183     1459  1 !       STORE_DESC_ADDR -The address of the storage vector descriptor
;  184     1460  1 !                        which completely describes this storage.
;  185     1461  1 !
;  186     1462  1 ! Implicit Inputs:
;  187     1463  1 !
;  188     1464  1 !       We assume that subtracting two RST pointers
;  189     1465  1 !       ("end-begin") gives the number of bytes between them.
;  190     1466  1 !
;  191     1467  1 ! Routine Value:
;  192     1468  1 !
;  193     1469  1 !       NOVALUE - because if there is a free storage error (the only
;  194     1470  1 !                 thing which can go wrong), we don't get
;  195     1471  1 !                 control back anyways.
;  196     1472  1 !
;  197     1473  1 ! Side Effects:
;  198     1474  1 !
;  199     1475  1 !       RST storage is released and the 'begin' pointer field of
;  200     1476  1 !       the corresponding storage descriptor is zeroed out.
;  201     1477  1 !--
;  202     1478  1
;  203     1479  2 BEGIN
;  204     1480  2
;  205     1481  2 MAP
;  206     1482  2         STORE_DESC_ADDR : REF VECT_STORE_DESC;                  ! We get passed a pointer to the storage
;  207     1483  2                                                                ! descriptor for the module we are
;  208     1484  2                                                                ! releasing vector storage for.
;  209     1485  2
;  210     1486  2 LOCAL
;  211     1487  2         VECT_SIZE;                                             ! Pointer to the actual storage
;  212     1488  2                                                                ! we allocate.
;  213     1489  2
;  214     1490  2 !++
;  215     1491  2 ! Don't try to free storage which is not allocated.
;  216     1492  2 !--
;  217     1493  3 IF( .STORE_DESC_ADDR[ STOR_BEGIN_RST ] EQL 0 )
;  218     1494  2 THEN
;  219     1495  2         RETURN;
;  220     1496  2
;  221     1497  2 !++
;  222     1498  2 ! Pick up the size of the vector, in bytes.
;  223     1499  2 !--
;  224     1500  2 VECT_SIZE = .STORE_DESC_ADDR[ STOR_END_RST ] - .STORE_DESC_ADDR[ STOR_BEGIN_RST ];
;  225     1501  2
;  226     1502  2 !++
;  227     1503  2 ! Release the storage.
;  228     1504  2 !--
;  229     1505  2 PAT$RST_RELEASE( .STORE_DESC_ADDR[ STOR_BEGIN_RST ], RST_UNITS(.VECT_SIZE));
```

```
;  230          1506  2
;  231          1507  2 !++
;  232          1508  2 ! Zero out the vector storage descriptor's 'begin' field.
;  233          1509  2 ! Routines OK_TO_ADD (etc.) check this field and assume that
;  234          1510  2 ! if it is non-zero then storage is still allocated for this module.
;  235          1511  2 !--
;  236          1512  2 STORE_DESC_ADDR[ STOR_BEGIN_RST ] = 0;
;  237          1513  1 END;


                                                0004 00000         .ENTRY  PAT$VS_FREE, Save R2                                    ; 1449
                                52      04  AC  D0 00002         MOVL    STORE_DESC_ADDR, R2                                     ; 1493
                                51      01  A2  3C 00006         MOVZWL  1(R2), R1                                               ;
                                            1A  13 0000A         BEQL    1$                                                      ;
                                50      03  A2  3C 0000C         MOVZWL  3(R2), VECT_SIZE                                        ; 1500
                                50          51  C2 00010         SUBL2   R1, VECT_SIZE                                           ;
                                50          03  C0 00013         ADDL2   #3, R0                                                  ; 1505
                            7E  50          04  C7 00016         DIVL3   #4, R0, -(SP)                                           ;
                                            51  DD 0001A         PUSHL   R1                                                      ;
                    00000000G   EF          02  FB 0001C         CALLS   #2, PAT$RST_RELEASE                                     ;
                                01      A2  B4 00023             CLRW    1(R2)                                                   ; 1512
                                            04 00026 1$:         RET                                                             ; 1513

; Routine Size:  39 bytes,    Routine Base:  _PAT$CODE + 002C
```

```
;  239     1514  1 GLOBAL ROUTINE PAT$VS_SHRINK( STORE_DESC_ADDR ) : NOVALUE =
;  240     1515  1
;  241     1516  1 !++
;  242     1517  1 ! Functional Description:
;  243     1518  1 !
;  244     1519  1 !       Free up the unused portion of the indicated vector storage.
;  245     1520  1 !
;  246     1521  1 ! Formal Parameters:
;  247     1522  1 !
;  248     1523  1 !       STORE_DESC_ADDR -The address of the storage vector descriptor
;  249     1524  1 !                        which completely describes this storage.
;  250     1525  1 !
;  251     1526  1 ! Implicit Inputs:
;  252     1527  1 !
;  253     1528  1 !       If the STOR_BEGIN_RST is the same as STOR_MARKER then
;  254     1529  1 !       no vector storage was actually allocated.  So far this
;  255     1530  1 !       is the only case we handle - we free up the entire
;  256     1531  1 !       vector.
;  257     1532  1 !
;  258     1533  1 ! Routine Value:
;  259     1534  1 !
;  260     1535  1 !       NOVALUE - because if there is a free storage error (the only
;  261     1536  1 !                 thing which can go wrong), we don't get
;  262     1537  1 !                 control back anyways.
;  263     1538  1 !
;  264     1539  1 ! Side Effects:
;  265     1540  1 !
;  266     1541  1 !       RST storage may be released and the 'begin' pointer field of
;  267     1542  1 !       the corresponding storage descriptor is zeroed out.
;  268     1543  1 !--
;  269     1544  1
;  270     1545  2 BEGIN
;  271     1546  2
;  272     1547  2 MAP
;  273     1548  2         STORE_DESC_ADDR : REF VECT_STORE_DESC;                  ! We get passed a pointer to the storage
;  274     1549  2                                                                ! descriptor for the module we are
;  275     1550  2                                                                ! releasing vector storage for.
;  276     1551  2
;  277     1552  2 LOCAL
;  278     1553  2         VECT_SIZE;                                             ! Pointer to the actual storage
;  279     1554  2                                                                ! we allocate.
;  280     1555  2
;  281     1556  2 !++
;  282     1557  2 ! Don't try to free storage which is not allocated.
;  283     1558  2 !--
;  284     1559  3 IF( .STORE_DESC_ADDR[ STOR_BEGIN_RST ] EQL 0 )
;  285     1560  2 THEN
;  286     1561  2         RETURN;
;  287     1562  2
;  288     1563  2 !++
;  289     1564  2 ! If storage has been allocated but no records therein, we can free up the
;  290     1565  2 ! whole thing.
;  291     1566  2 !--
;  292     1567  3 IF( .STORE_DESC_ADDR[ STOR_BEGIN_RST ] EQL .STORE_DESC_ADDR[ STOR_MARKER ] )
;  293     1568  2 THEN
;  294     1569  2         PAT$VS_FREE(.STORE_DESC_ADDR);
;  295     1570  1 END;
```

```
                                        0000 00000          .ENTRY  PAT$VS_SHRINK, Save nothing                     ; 1514
                          50    04  AC  D0 00002          MOVL    STORE_DESC_ADDR, R0                             ; 1559
                                01  A0  B5 00006          TSTW    1(R0)                                           ;
                                    0D  13 00009          BEQL    1$                                              ;
                          05    A0  01  A0  B1 0000B      CMPW    1(R0), 5(R0)                                    ; 1567
                                    06  12 00010          BNEQ    1$                                              ;
                                    50  DD 00012          PUSHL   R0                                              ; 1569
                          C1    AF      01  FB 00014      CALLS   #1, PAT$VS_FREE                                 ;
                                        04 00018 1$:      RET                                                     ; 1570

; Routine Size:  25 bytes,    Routine Base:  _PAT$CODE + 0053
```

```
;  297     1571  1 GLOBAL ROUTINE PAT$VS_GET( STORE_DESC_ADDR, RECORD_SIZE ) =
;  298     1572  1
;  299     1573  1 !++
;  300     1574  1 ! Functional Description:
;  301     1575  1 !
;  302     1576  1 !       Allocate a given-length record from so-called "vector storage"
;  303     1577  1 !       given a pointer to the associated vector storage descriptor.
;  304     1578  1 !
;  305     1579  1 ! Formal Parameters:
;  306     1580  1 !
;  307     1581  1 !       STORE_DESC_ADDR -The address of the storage vector descriptor
;  308     1582  1 !                        which this routine works from.
;  309     1583  1 !       RECORD_SIZE     -The number of bytes required for
;  310     1584  1 !                        the new record.
;  311     1585  1 !
;  312     1586  1 ! Implicit Inputs:
;  313     1587  1 !
;  314     1588  1 !       This routine builds in how to translate RST-pointers
;  315     1589  1 !       to longword pointers.
;  316     1590  1 !
;  317     1591  1 !       We also build in how one deals with storage vector descriptors.
;  318     1592  1 !       This includes the fact that we look at the pointer-type
;  319     1593  1 !       field in the storage descriptor to determine whether we
;  320     1594  1 !       should return an RST-pointer or a 'longword' pointer.
;  321     1595  1 !       (See PATRST.REQ)
;  322     1596  1 !
;  323     1597  1 ! Routine Value:
;  324     1598  1 !
;  325     1599  1 !       A pointer to the allocated storage, 0 when there is none left.
;  326     1600  1 !
;  327     1601  1 !       The pointer type (RST or 'longword') is determined
;  328     1602  1 !       by looking at the STOR_LONG_PTRS field of the given
;  329     1603  1 !       storage descriptor.
;  330     1604  1 !
;  331     1605  1 ! Side Effects:
;  332     1606  1 !
;  333     1607  1 !       The storage vector descriptor fields
;  334     1608  1 ! are altered to reflect the allocation
;  335     1609  1 ! of another record.
;  336     1610  1 !--
;  337     1611  1
;  338     1612  2 BEGIN
;  339     1613  2
;  340     1614  2 MAP
;  341     1615  2         STORE_DESC_ADDR : REF VECT_STORE_DESC;                  ! We get passed a pointer to the descriptor
;  342     1616  2                                                                ! for the storage we are allocating from.
;  343     1617  2
;  344     1618  2 LOCAL
;  345     1619  2         STORAGE : REF RST_POINTER,                             ! An RST-pointer to the actual storage
;  346     1620  2                                                                ! we will allocate.
;  347     1621  2         NEW_MARKER : REF RST_POINTER;                          ! How far along this allocation will
;  348     1622  2                                                                ! move the current marker.
;  349     1623  2
;  350     1624  2 !++
;  351     1625  2 ! If we get to allocate the requested storage, it will be from where the
;  352     1626  2 ! current marker says we should get it.
;  353     1627  2 !--
```

```
;  354          1628  2 STORAGE = .STORE_DESC_ADDR[ STOR_MARKER ];
;  355          1629  2
;  356          1630  2 !++
;  357          1631  2 ! See if taking the necessary storage from the indicated vector would overflow it.
;  358          1632  2 !--
;  359          1633  2 NEW_MARKER = .STORAGE + .RECORD_SIZE;
;  360          1634  3 IF( .NEW_MARKER GTRA .STORE_DESC_ADDR[ STOR_END_RST ] )
;  361          1635  2 THEN
;  362          1636  2         RETURN(FALSE);                                  ! No more storage left.
;  363          1637  2
;  364          1638  2 !++
;  365          1639  2 ! Update the storage descriptor to reflect the allocation of this new record.
;  366          1640  2 !--
;  367          1641  2 STORE_DESC_ADDR[ STOR_MARKER ] = .NEW_MARKER;
;  368          1642  2
;  369          1643  2 !++
;  370          1644  2 ! We return an RST-pointer or a longword pointer, depending on a field in the
;  371          1645  2 ! storage descriptor.  This field is initialized in build_module().
;  372          1646  2 !--
;  373          1647  3 IF( .STORE_DESC_ADDR[ STOR_LONG_PTRS ] )
;  374          1648  2 THEN
;  375          1649  2         STORAGE = .STORAGE + .PAT$GL_RST_BEGN;          ! Make an RST-pointer into a longword pointe
;  376          1650  2
;  377          1651  2 RETURN( .STORAGE );
;  378          1652  1 END;

                                                0004 00000          .ENTRY  PAT$VS_GET, Save R2                             ; 1571
                                  50        04    AC   D0 00002     MOVL    STORE_DESC_ADDR, R0                             ; 1628
                                  52        05    A0   3C 00006     MOVZWL  5(R0), STORAGE
                              51  52        08    AC   C1 0000A     ADDL3   RECORD_SIZE, STORAGE, NEW_MARKER                ; 1633
                51    03    A0  10              00   ED 0000F     CMPZV   #0, #16, 3(R0), NEW_MARKER                      ; 1634
                                                12   1F 00015     BLSSU   2$
                                  05    A0            51   B0 00017     MOVW    NEW_MARKER, 5(R0)                       ; 1641
                                        07            60   E9 0001B     BLBC    (R0), 1$                                ; 1647
                                  52 00000000G        EF   C0 0001E     ADDL2   PAT$GL_RST_BEGN, STORAGE                ; 1649
                                  50                  52   D0 00025 1$:  MOVL    STORAGE, R0                             ; 1651
                                                      04   00028        RET
                                                50    D4   00029 2$:    CLRL    R0                                      ; 1652
                                                      04   0002B        RET

; Routine Size:  44 bytes,    Routine Base:  _PAT$CODE + 006C
```

```
;  380          1653  1 END                          !End of module
;  381          1654  0 ELUDOM
```

```
;                                  PSECT SUMMARY
;
;       Name                      Bytes                          Attributes
;
;  _PAT$CODE                        152  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                               Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1      9776         0         0       581          00:00.9
```

```
;                                  COMMAND QUALIFIERS

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

; Size:         152 code + 0 data bytes
; Run Time:       00:09.2
; Elapsed Time:   00:29.6
; Lines/CPU Min:  10751
; Lexemes/CPU-Min: 20944
; Memory Used:  71 pages
; Compilation Complete
```

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PHONE

PATSYM
LIS

PHONE
MAP

BASICCMDS
LIS

PATSTO
LIS

PATVEC
LIS

PHONEREQ
REQ

PATWRT
LIS

FILECMDS
LIS