```
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  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
```

```
  0 MODULE PATVEC ( !
L 0                 %IF %VARIANT EQL 1
  0                 %THEN
  0                         ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE),
  0                 %FI
  0                 IDENT = 'V04-000'
  0                 ) =
  1 BEGIN
  1
  1 !****************************************************************************
  1 !*                                                                          *
  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
  1 !*  ALL RIGHTS RESERVED.                                                    *
  1 !*                                                                          *
  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
  1 !*  TRANSFERRED.                                                            *
  1 !*                                                                          *
  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
  1 !*  CORPORATION.                                                            *
  1 !*                                                                          *
  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
  1 !*                                                                          *
  1 !*                                                                          *
  1 !****************************************************************************
  1 !
  1
  1 !++
  1 ! FACILITY:     PATCH
  1 !
  1 ! ABSTRACT:     Handle so-called 'vector storage' in conjunction
  1 !               with the RST build and rebuild modules.
  1 !
  1 ! ENVIRONMENT:  This module runs on VAX under VAX/VMS, user mode, non-AST level.
  1 !
  1 ! Author:       Kevin Pammett, August 12, 1977.
  1 !
  1 ! Version:      X01.03
  1 !
  1 ! MODIFCATIONS:
  1 !
  1 !       V03-001 MTR0012         Mike Rhodes             16-Aug-1982
  1 !               Modify file names to remove duplicate file name useage
  1 !               between code and require files.
  1 !
  1 !  NO   DATE            PROGRAMMER              PURPOSE
  1 !  --   ----            ----------              -------
  1 !
  1 !  00   21-DEC-77       K.D. MORSE              ADAPT VERSION 35 FOR PATCH.
  1 !  01   25-APR-78       K.D. MORSE              CONVERT TO NATIVE COMPILER.
  1 !  02   18-MAY-78       K.D. MORSE              NO CHANGES FOR VERS 36.
```

```
1 ! 03   13-JUN-78        K.D. MORSE              ADD FAO COUNTS TO SIGNALS.
1 !
1 !--
```

```
0061    1 !
0062    1 ! TABLE OF CONTENTS:
0063    1 !
0064    1
0065    1 FORWARD ROUTINE
0066    1         PAT$VS_INIT : NOVALUE,                          ! Initialize a vector storage area.
0067    1         PAT$VS_GET,                                     ! Allocate records from a so-called
0068    1                                                         !  "vector storage" vector.
0069    1         PAT$VS_FREE : NOVALUE,                          ! Release vector storage.
0070    1         PAT$VS_SHRINK : NOVALUE;                        ! Free up unused vector storage.
0071    1
0072    1 !
0073    1 ! INCLUDE FILES:
0074    1 !
0075    1
0076    1 LIBRARY 'SYS$LIBRARY:STARLET.L32';
0077    1 REQUIRE 'SRC$:PATPCT.REQ';
0117    1 REQUIRE 'SRC$:VXSMAC.REQ';
0182    1 REQUIRE 'SRC$:PATRTS.REQ';
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
```

```
1
1 !
1 ! MACROS:
1 !
1
1 !
1 ! EQUATED SYMBOLS:
1 !
1
1
1 !
1 ! EXTERNAL REFERENCES:
1 !
1
1 EXTERNAL ROUTINE
1         PAT$RST_FREEZ,                          ! Allocate and zero RST-pointer storage.
1         PAT$RST_RELEASE;                        ! Free up RST-pointer storage.
1
1 EXTERNAL
1         PAT$GL_RST_BEGN;                        ! STARTING ADDRESS OF RST
```

```
GLOBAL ROUTINE PAT$VS_INIT( STORE_DESC_ADDR, NUM_ELEMENTS, ELEMENT_SIZE ) : NOVALUE =

!++
! Functional Description:
!
!       Initialize storage for a so-called "storage vector".
!       See DBGRST.REQ for a description of these structures.
!
! Formal Parameters:
!
!       STORE_DESC_ADDR -The address of the storage vector descriptor
!                        which this routine is to initialize.
!       NUM_ELEMENTS    -The maximum number of elements which will
!                        ever be allocated from this vector.
!       ELEMENT_SIZE    -The maximum size, in bytes, of each element.
!
! Implicit Inputs:
!
!       Since we assume that records are fixed-length,
!       we actually allocate more storage than
!       we will really need.  This will be fixed up when we
!       can 'shrink' RST storage.
!
! Routine Value:
!
!       NOVALUE - because if we run out of storage (the only
!                 thing which can go wrong), we don't get
!                 control back anyways.
!
! Side Effects:
!
!       The storage descriptor is initialized in such
!       a way that GET_VEC_STORAGE can do its thing.
!
!       RST storage is allocated.
!--

BEGIN

MAP
        STORE_DESC_ADDR : REF VECT_STORE_DESC;                  ! We get passed a pointer to the storage
                                                                ! descriptor for the module we are
                                                                ! initializing vector storage for.

LOCAL
        STORAGE : REF MC_RECORD;                                ! Pointer to the actual storage
                                                                ! we allocate.

!++
! Ask for the necessary RST storage.   Note that we don't get control back
! if the requisition is denied.
!--
STORAGE = PAT$RST_FREEZ( RST_UNITS(.ELEMENT_SIZE * .NUM_ELEMENTS));

!++
! Initialize the vector storage descriptor. The 'marker' starts out being
! the same as the beginning address since initially the entire vector is
```

```
1437  2 ! available for re-allocation.
1438  2 !--
1439  2 STORE_DESC_ADDR[ STOR_BEGIN_RST ] = .STORAGE;
1440  2 STORE_DESC_ADDR[ STOR_MARKER ] = .STORAGE,
1441  2
1442  2 !++
1443  2 ! The end RST address is calculated by taking into account that the standard
1444  2 ! PATCH storage manager works in LONGWORDS and rounds up.
1445  2 !--
1446  2 STORE_DESC_ADDR[ STOR_END_RST ] = .STORAGE
1447  2                 + (RST_UNITS(.ELEMENT_SIZE*.NUM_ELEMENTS) * %UPVAL);
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
```

```
1 GLOBAL ROUTINE PAT$VS_FREE( STORE_DESC_ADDR ) : NOVALUE =
1
1 !++
1 ! Functional Description:
1 !
1 !       Release storage for a so-called "storage vector".
1 !       See PATRST.REQ for a description of these structures.
1 !
1 ! Formal Parameters:
1 !
1 !       STORE_DESC_ADDR -The address of the storage vector descriptor
1 !                        which completely describes this storage.
1 !
1 ! Implicit Inputs:
1 !
1 !       We assume that subtracting two RST pointers
1 !       ("end-begin") gives the number of bytes between them.
1 !
1 ! Routine Value:
1 !
1 !       NOVALUE - because if there is a free storage error (the only
1 !                 thing which can go wrong), we don't get
1 !                 control back anyways.
1 !
1 ! Side Effects:
1 !
1 !       RST storage is released and the 'begin' pointer field of
1 !       the corresponding storage descriptor is zeroed out.
1 !--
1
2 BEGIN
2
2 MAP
2       STORE_DESC_ADDR : REF VECT_STORE_DESC;          ! We get passed a pointer to the storage
2                                                       ! descriptor for the module we are
2                                                       ! releasing vector storage for.
2
2 LOCAL
2       VECT_SIZE;                                      ! Pointer to the actual storage
2                                                       ! we allocate.
2
2 !++
2 ! Don't try to free storage which is not allocated.
2 !--
3 IF( .STORE_DESC_ADDR[ STOR_BEGIN_RST ] EQL 0 )
2 THEN
2       RETURN;
2
2 !++
2 ! Pick up the size of the vector, in bytes.
2 !--
2 VECT_SIZE = .STORE_DESC_ADDR[ STOR_END_RST ] - .STORE_DESC_ADDR[ STOR_BEGIN_RST ];
2
2 !++
2 ! Release the storage.
2 !--
2 PAT$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
```

```
GLOBAL ROUTINE PAT$VS_SHRINK( STORE_DESC_ADDR ) : NOVALUE =

!++
! Functional Description:
!
!       Free up the unused portion of the indicated vector storage.
!
! Formal Parameters:
!
!       STORE_DESC_ADDR -The address of the storage vector descriptor
!                        which completely describes this storage.
!
! Implicit Inputs:
!
!       If the STOR_BEGIN_RST is the same as STOR_MARKER then
!       no vector storage was actually allocated.  So far this
!       is the only case we handle - we free up the entire
!       vector.
!
! Routine Value:
!
!       NOVALUE - because if there is a free storage error (the only
!                 thing which can go wrong), we don't get
!                 control back anyways.
!
! Side Effects:
!
!       RST storage may be released and the 'begin' pointer field of
!       the corresponding storage descriptor is zeroed out.
!--

BEGIN

MAP
        STORE_DESC_ADDR : REF VECT_STORE_DESC;            ! We get passed a pointer to the storage
                                                          ! descriptor for the module we are
                                                          ! releasing vector storage for.

LOCAL
        VECT_SIZE;                                        ! Pointer to the actual storage
                                                          ! we allocate.

!++
! Don't try to free storage which is not allocated.
!--
IF( .STORE_DESC_ADDR[ STOR_BEGIN_RST ] EQL 0 )
THEN
        RETURN;

!++
! If storage has been allocated but no records therein, we can free up the
! whole thing.
!--
IF( .STORE_DESC_ADDR[ STOR_BEGIN_RST ] EQL .STORE_DESC_ADDR[ STOR_MARKER ] )
THEN
        PAT$VS_FREE(.STORE_DESC_ADDR);
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
```

```
1 GLOBAL ROUTINE PAT$VS_GET( STORE_DESC_ADDR, RECORD_SIZE ) =
1
1 !++
1 ! Functional Description:
1 !
1 !       Allocate a given-length record from so-called "vector storage"
1 !       given a pointer to the associated vector storage descriptor.
1 !
1 ! Formal Parameters:
1 !
1 !       STORE_DESC_ADDR -The address of the storage vector descriptor
1 !                        which this routine works from.
1 !       RECORD_SIZE     -The number of bytes required for
1 !                        the new record.
1 !
1 ! Implicit Inputs:
1 !
1 !       This routine builds in how to translate RST-pointers
1 !       to longword pointers.
1 !
1 !       We also build in how one deals with storage vector descriptors.
1 !       This includes the fact that we look at the pointer-type
1 !       field in the storage descriptor to determine whether we
1 !       should return an RST-pointer or a 'longword' pointer.
1 !       (See PATRST.REQ)
1 !
1 ! Routine Value:
1 !
1 !       A pointer to the allocated storage, 0 when there is none left.
1 !
1 !       The pointer type (RST or 'longword') is determined
1 !       by looking at the STOR_LONG_PTRS field of the given
1 !       storage descriptor.
1 !
1 ! Side Effects:
1 !
1 !       The storage vector descriptor fields
1 ! are altered to reflect the allocation
1 ! of another record.
1 !--
1
2 BEGIN
2
2 MAP
2       STORE_DESC_ADDR : REF VECT_STORE_DESC;          ! We get passed a pointer to the descriptor
2                                                       ! for the storage we are allocating from.
2
2 LOCAL
2       STORAGE : REF RST_POINTER,                      ! An RST-pointer to the actual storage
2                                                       ! we will allocate.
2       NEW_MARKER : REF RST_POINTER;                   ! How far along this allocation will
2                                                       ! move the current marker.
2
2 !++
2 ! If we get to allocate the requested storage, it will be from where the
2 ! current marker says we should get it.
2 !--
```

```
2 STORAGE = .STORE_DESC_ADDR[ STOR_MARKER ];
2
2 !++
2 ! See if taking the necessary storage from the indicated vector would overflow it.
2 !--
2 NEW_MARKER = .STORAGE + .RECORD_SIZE;
3 IF( .NEW_MARKER GTRA .STORE_DESC_ADDR[ STOR_END_RST ] )
2 THEN
2         RETURN(FALSE);                                      ! No more storage left.
2
2 !++
2 ! Update the storage descriptor to reflect the allocation of this new record.
2 !--
2 STORE_DESC_ADDR[ STOR_MARKER ] = .NEW_MARKER;
2
2 !++
2 ! We return an RST-pointer or a longword pointer, depending on a field in the
2 ! storage descriptor.  This field is initialized in build_module().
2 !--
3 IF( .STORE_DESC_ADDR[ STOR_LONG_PTRS ] )
2 THEN
2         STORAGE = .STORAGE + .PAT$GL_RST_BEGN;              ! Make an RST-pointer into a longword pointe
2
2 RETURN( .STORAGE );
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