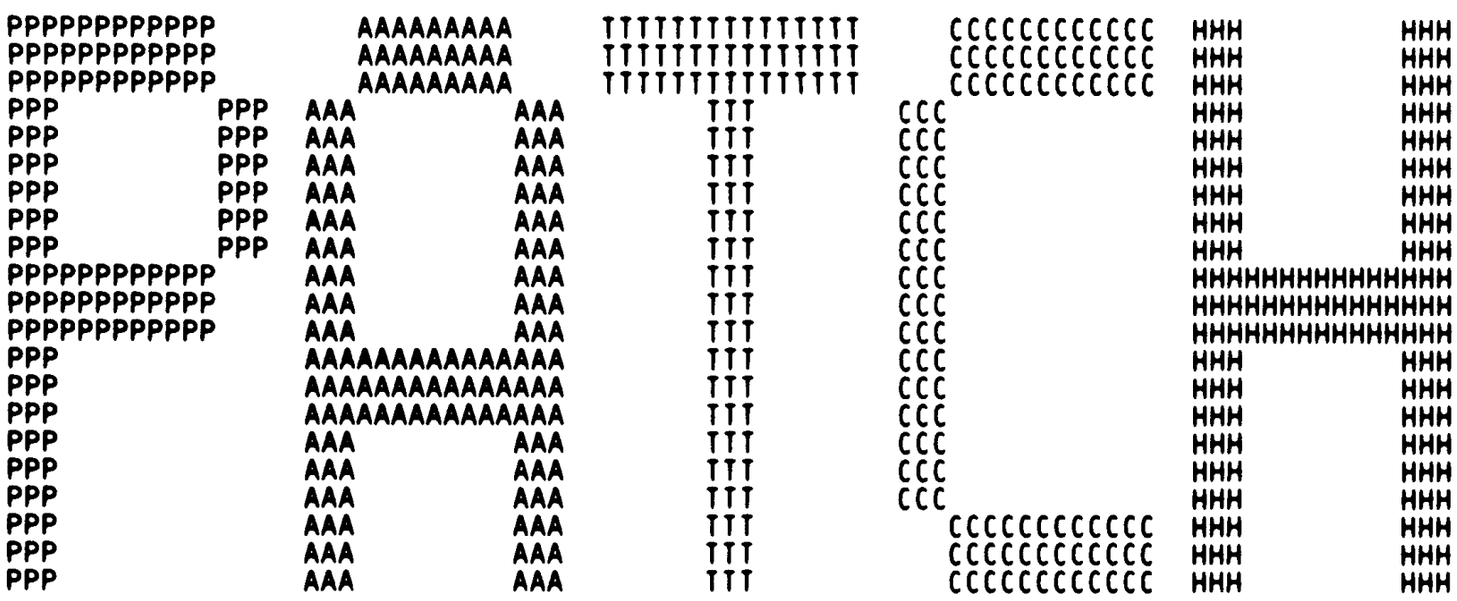
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPPPPPPPPPPP AAAAAAAAAAAAAAAA TTT CCC HHHHHHHHHHHHHHHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCCC HHH HHH

```
DDDDDDDD    YY       YY  NN       NN  MM       MM  EEEEEEEEEE  MM       MM
DDDDDDDD    YY       YY  NN       NN  MM       MM  EEEEEEEEEE  MM       MM
DD      DD  YY       YY  NN       NN  MMMM   MMMM  EE          MMMM   MMMM
DD      DD  YY       YY  NN       NN  MMMM   MMMM  EE          MMMM   MMMM
DD      DD    YY   YY    NNNN     NN  MM  MM  MM  EE          MM  MM  MM
DD      DD    YY   YY    NNNN     NN  MM  MM  MM  EE          MM  MM  MM
DD      DD      YY       NN  NN   NN  MM      MM  EEEEEEEE    MM      MM
DD      DD      YY       NN  NN   NN  MM      MM  EEEEEEEE    MM      MM
DD      DD      YY       NN    NNNN  MM      MM  EE          MM      MM
DD      DD      YY       NN    NNNN  MM      MM  EE          MM      MM
DD      DD      YY       NN      NN  MM      MM  EE          MM      MM
DD      DD      YY       NN      NN  MM      MM  EE          MM      MM
DDDDDDDD        YY       NN      NN  MM      MM  EEEEEEEEEE  MM      MM
DDDDDDDD        YY       NN      NN  MM      MM  EEEEEEEEEE  MM      MM

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        L 0001  0 MODULE DYNMEM (%IF %VARIANT EQL 1
;     2          0002  0                 %THEN
;     3          0003  0                         ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE,
;     4          0004  0                                          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 !
;    37          0037  1 !
;    38          0038  1 !++
;    39          0039  1 !
;    40          0040  1 ! MODULE: DYNMEM
;    41          0041  1 !
;    42          0042  1 ! FACILITY: PATCH
;    43          0043  1 !
;    44          0044  1 ! ABSTRACT:  DYNAMIC MEMORY ALLOCATION AND DEALLOCATION
;    45          0045  1 !
;    46          0046  1 ! HISTORY:
;    47          0047  1 !
;    48          0048  1 !
;    49          0049  1 !       AUTHOR: T.J. PORTER 14-JAN-77
;    50          0050  1 !
;    51          0051  1 ! MODIFIED BY:
;    52          0052  1 !
;    53          0053  1 !
;    54          0054  1 !       V03-001 MTR0007         Mike Rhodes             14-Jun-1982
;    55          0055  1 !               Use shared system messages.  Affected modules include:
;    56          0056  1 !               DYNMEM.B32, PATBAS.B32, PATCMD.B32, PATIHD.B32, PATINT.B32,
;    57          0057  1 !               PATIO.B32, PATMAI.B32, PATMSG.MSG, PATWRT.B32, and PATSPA.B32.
```

```
;    58          0058  1 !
;    59          0059  1 !       V02-001 PCG0001         Peter George            02-FEB-1981
;    60          0060  1 !               Add require statement for LIB$:PATDEF.REQ
;    61          0061  1 !
;    62          0062  1 !       MODIFICATIONS:
;    63          0063  1 !
;    64          0064  1 ! NO.   DATE            PROGRAMMER      PURPOSE
;    65          0065  1 ! ---   ----            ----------      -------
;    66          0066  1 ! 001   19-SEP-77       T.J. PORTER     REMEMBER THE HIGHEST ADDRESS ALLOCATED
;    67          0067  1 ! 002   12-OCT-77       K.D. MORSE      ADAPT TO PAICH
;    68          0068  1 ! 003   19-OCT-77       K.D. MORSE      ADD REQUIRE FILE VXSMAC.REQ
;    69          0069  1 ! 004   18-NOV-77       K.D. MORSE      ADD REQUIRE FILE PATPCT.REQ
;    70          0070  1 ! 005   25-APR-78       K.D. MORSE      CONVERT TO NATIVE COMPILER.
;    71          0071  1 ! 006   13-JUN-78       K.D. MORSE      ADD FAO COUNTS TO SIGNALS.
;    72          0072  1 !
;    73          0073  1 !--
```

```
;    75          0074  1 !++
;    76          0075  1 !
;    77          0076  1 ! FUNCTIONAL DESCRIPTION:
;    78          0077  1 !
;    79          0078  1 !       This module contains all the dynamic memory allocation
;    80          0079  1 !       and de-allocation logic for patch.  A singly
;    81          0080  1 !       linked list of free blocks of memory is maintained
;    82          0081  1 !       (listhead is pat$gl_memlhd) and memory is allocated
;    83          0082  1 !       by first fit.  Should there be no available memory block
;    84          0083  1 !       of required size, the allocation routine expands
;    85          0084  1 !       the program region by the number of pages equal to
;    86          0085  1 !       PAT$K_MEMEXP, links this on the end of the free
;    87          0086  1 !       list and allocates the required memory from that new
;    88          0087  1 !       block.  The free memory list is therefore initialized
;    89          0088  1 !       on first allocation call.  Memory is always allocated
;    90          0089  1 !       in eight byte quanta, with a maximum of PAT$K_MEMEXP*512
;    91          0090  1 !       bytes.  Deallocation effects compaction whenever possible.
;    92          0091  1 !
;    93          0092  1 ! CALLING SEQUENCES:
;    94          0093  1 !       PAT$ALLOBLK(BLOCKSIZE,BLOCKADDR)
;    95          0094  1 !       PAT$DEALBLK(BLOCKSIZE,BLOCKADDR)
;    96          0095  1 !               where:
;    97          0096  1 !                 BLOCKSIZE = number of bytes to be (de)allocated.
;    98          0097  1 !                 BLOCKADDR = address of cell for the address of
;    99          0098  1 !                             the block allocated or to be deallocated.
;   100          0099  1 ! ERROR CONDITIONS:
;   101          0100  1 !
;   102          0101  1 !       1.  BLOCKSIZE < OR = 0 OR > PAT$K_MEMEXP*512 (CODE = 0,10)
;   103          0102  1 !
;   104          0103  1 !       2.  Failure to expand the program region issues a message
;   105          0104  1 !           that memory is full and the linker aborts.
;   106          0105  1 !
;   107          0106  1 !       3.  Any part of a block to be deallocated is:
;   108          0107  1 !               (I) Within a free block (code = 2,13)
;   109          0108  1 !               (II) Beyond top of program region (code = 11)
;   110          0109  1 !               (III) Lower than than lowest block ever allocated (code = 12)
;   111          0110  1 !
;   112          0111  1 !       In cases 1 and 3 a fatal ("bug") message is issued and
;   113          0112  1 !               the patch terminates.
;   114          0113  1 !
;   115          0114  1 !--
;   116          0115  1
```

```
;  118     0116  1 !
;  119     0117  1 ! INCLUDE FILES:
;  120     0118  1 !
;  121     0119  1
;  122     0120  1 LIBRARY 'SYS$LIBRARY:LIB.L32';                  ! Defines system structures and services
;  123     0121  1 REQUIRE 'SRC$:PREFIX.REQ';                      ! Defines useful macros
;  124     0309  1 REQUIRE 'LIB$:PATDEF.REQ';                      ! Defines literals
;  125     0363  1 REQUIRE 'LIB$:PATMSG.REQ';                      ! Define error codes
;  126     0537  1 REQUIRE 'SRC$:VXSMAC.REQ';                      ! Define TRUE and FALSE
;  127     0602  1 REQUIRE 'SRC$:PATPCT.REQ';                      ! Define PSECTS
;  128     0642  1
;  129     0643  1 !
;  130     0644  1 ! TABLE OF CONTENTS:
;  131     0645  1 !
;  132     0646  1
;  133     0647  1 FORWARD ROUTINE
;  134     0648  1         PAT$ALLOBLK : NOVALUE,                  ! Allocation driver
;  135     0649  1         ALLOCATE,                               ! Allocation routine
;  136     0650  1         PAT$DEALBLK : NOVALUE,                  ! Deallocation driver
;  137     0651  1         DEALLOCATE;                             ! Deallocation routine
;  138     0652  1
;  139     0653  1 !
;  140     0654  1 ! EXTERNAL DATA
;  141     0655  1 !
;  142     0656  1
;  143     0657  1 EXTERNAL
;  144     0658  1         PAT$GL_ISVADDR : VECTOR[,LONG],         ! Contains starting and ending virtual addrs of last
;  145     0659  1         PAT$GL_ERRCODE,                         ! Global error code
;  146     0660  1         PAT$GL_MEMLHD : VECTOR[,LONG],          ! Free memory listhead
;  147     0661  1         PAT$GL_MINADDR;                         ! Lowest address ever allocated
;  148     0662  1
;  149     0663  1 !
;  150     0664  1 ! EQUATED SYMBOLS
;  151     0665  1 !
;  152     0666  1
;  153     0667  1 EXTERNAL LITERAL
;  154     0668  1         PAT$K_MEMEXP,                           ! # of pages to extend program region
;  155     0669  1         PAT$K_MAXBLKSIZ,                        ! Maximum allocation size
;  156     0670  1 !
;  157     0671  1 ! Define shared message references.  (resolved @ link time)
;  158     0672  1 !
;  159     0673  1         PAT$_CLOSEIN,                   ! Error closing input file.
;  160     0674  1         PAT$_CLOSEOUT,                  ! Error closing output file.
;  161     0675  1         PAT$_OPENIN,                    ! Error opening input file.
;  162     0676  1         PAT$_OPENOUT,                   ! Error opening output file.
;  163     0677  1         PAT$_READERR,                   ! Error reading from file.
;  164     0678  1         PAT$_SYSERROR,                  ! System Service error.
;  165     0679  1         PAT$_WRITEERR;                  ! Error writing to file.
;  166     0680  1 !
;  167     0681  1 ! OWN STORAGE
;  168     0682  1 !
;  169     0683  1
;  170     0684  1 OWN
;  171     0685  1         ERRORCODE : BYTE,                       ! Error code for failure message
;  172     0686  1         NEWBLOCK : REF VECTOR[2],               ! Current block pointer
;  173     0687  1         NEXTBLOCK : REF VECTOR[2],              ! Next block pointer
;  174     0688  1         LASTBLOCK : REF VECTOR[2];              ! Previous block pointer
```

```
;  176          0689  1 GLOBAL ROUTINE PAT$ALLOBLK(SIZE,BLOCKADDR):NOVALUE =
;  177          0690  1 !++
;  178          0691  1 !       Allocate a block from the free memory list.
;  179          0692  1 !--
;  180          0693  1
;  181          0694  2 BEGIN
;  182          0695  2
;  183          0696  2 LOCAL
;  184          0697  2         BLOCKSIZE;
;  185          0698  2
;  186          0699  2 !++
;  187          0700  2 ! Initialize local error code.
;  188          0701  2 !--
;  189          0702  2 ERRORCODE = 0;                                  ! Initialize error code
;  190          0703  2
;  191          0704  2 !++
;  192          0705  2 ! Set size of block to allocate (round up to 8 bytes).
;  193          0706  2 !--
;  194          0707  2 BLOCKSIZE = (.SIZE + 7) AND ( NOT 7);           ! Round up to multiple of 8 bytes
;  195          0708  2
;  196          0709  2 !++
;  197          0710  2 ! Now check for zero size block then allocate it.
;  198          0711  2 !--
;  199          0712  3 IF (.BLOCKSIZE EQL 0)                          ! Check legal block
;  200          0713  2 OR                                             ! Size was requested
;  201          0714  2         .BLOCKSIZE GTRU PAT$K_MAXBLKSIZ
;  202          0715  3 OR NOT ( ERRORCODE = .ERRORCODE + 1;           ! Set new error code
;  203          0716  3         ALLOCATE (.BLOCKSIZE, .BLOCKADDR))    ! Go allocate
;  204          0717  2 THEN
;  205          0718  2         SIGNAL(PAT$_MEMBUG,1,.BLOCKSIZE,.BLOCKADDR,.ERRORCODE); ! Print error message and TERMINATE IF FAILU
;  206          0719  2 RETURN;                                         ! Otherwise return
;  207          0720  1 END;                                           ! Of PAT$ALLOBLK routine


                                .TITLE  DYNMEM
                                .IDENT  \V04-000\

                                .PSECT  _PAT$OWN,NOEXE,2

                        00000 ERRORCODE:
                                .BLKB   1
                        00001           .BLKB   3
                        00004 NEWBLOCK:
                                .BLKB   4
                        00008 NEXTBLOCK:
                                .BLKB   4
                        0000C LASTBLOCK:
                                .BLKB   4

                                .EXTRN  PAT$GL_ISVADDR, PAT$GL_ERRCODE
                                .EXTRN  PAT$GL_MEMLHD, PAT$GL_MINADDR
                                .EXTRN  PAT$K_MEMEXP, PAT$K_MAXBLKSIZ
                                .EXTRN  PAT$_CLOSEIN, PAT$_CLOSEOUT
                                .EXTRN  PAT$_OPENIN, PAT$_OPENOUT
                                .EXTRN  PAT$_READERR, PAT$_SYSERROR
                                .EXTRN  PAT$_WRITEERR
```

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

                                                 000C 00000     .ENTRY  PAT$ALLOBLK, Save R2,R3                 ; 0689
                          53 00000000'   EF      9E 00002     MOVAB   ERRORCODE, R3
                                         63      94 00009     CLRB    ERRORCODE                               ; 0702
                    50        04   AC      07      C1 0000B     ADDL3   #7, SIZE, R0                           ; 0707
                    52             50      07      CB 00010     BICL3   #7, R0, BLOCKSIZE
                                           1A      13 00014     BEQL    1$                                      ; 0712
                       00000000G   8F      52      D1 00016     CMPL    BLOCKSIZE, #PAT$K_MAXBLKSIZ             ; 0714
                                           11      1A 0001D     BGTRU   1$
                                           63      96 0001F     INCB    ERRORCODE                               ; 0715
                                    08     AC      DD 00021     PUSHL   BLOCKADDR                               ; 0716
                                           52      DD 00024     PUSHL   BLOCKSIZE
                       00000000V   EF              02      FB 00026     CALLS   #2, ALLOCATE
                                   17              50      E8 0002D     BLBS    R0, 2$
                                   7E              63      9A 00030 1$: MOVZBL  ERRORCODE, -(SP)                ; 0718
                                    08     AC      DD 00033     PUSHL   BLOCKADDR
                                           52      DD 00036     PUSHL   BLOCKSIZE
                                           01      DD 00038     PUSHL   #1
                                 006D8194  8F      DD 0003A     PUSHL   #7176596
                       00000000G   00              05      FB 00040     CALLS   #5, LIB$SIGNAL
                                                       04 00047 2$: RET                                         ; 0720

; Routine Size:  72 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
;  209          0721  1 ROUTINE ALLOCATE (SIZE,ADDRESS) =
;  210          0722  1 !++
;  211          0723  1 !       Routine to do actual allocation and program
;  212          0724  1 !       region expansion
;  213          0725  1 !--
;  214          0726  1
;  215          0727  2 BEGIN
;  216          0728  2
;  217          0729  2 LASTBLOCK = PAT$GL_MEMLHD[0];                    ! Initially at top of free list
;  218          0730  2
;  219          0731  2 !++
;  220          0732  2 ! Check down free list for first block of equal or larger size.
;  221          0733  2 !--
;  222          0734  2 WHILE (NEWBLOCK = .LASTBLOCK[0])NEQ 0 DO         ! Follow down free list
;  223          0735  3         BEGIN
;  224          0736  4         IF (.NEWBLOCK[1] EQL .SIZE)              ! Look for suitable free block
;  225          0737  4         THEN BEGIN                               ! Exact size match
;  226          0738  4                 LASTBLOCK[0] = .NEWBLOCK[0];     ! so last points where this one pointed
;  227          0739  4                 IF (.ADDRESS = NEWBLOCK[0]) LSSU .PAT$GL_MINADDR ! Now record lowest
;  228          0740  4                 THEN PAT$GL_MINADDR = NEWBLOCK[0];       ! Allocated address
;  229          0741  4                 RETURN TRUE;                     ! And we are done
;  230          0742  4                 END
;  231          0743  4         ELSE IF (.NEWBLOCK[1] GTRU .SIZE)                ! Or one larger than requested
;  232          0744  4         THEN BEGIN
;  233          0745  4                 NEXTBLOCK = NEWBLOCK[0]+.SIZE;   ! In which case there is a new
;  234          0746  4                 NEXTBLOCK[0] = .NEWBLOCK[0];
;  235          0747  4                 NEXTBLOCK[1] = .NEWBLOCK[1]-.SIZE;       ! Next block (the part remain-
;  236          0748  4                 LASTBLOCK[0] = NEXTBLOCK[0];     ! ing) after taking requested block off
;  237          0749  4                 IF (.ADDRESS = NEWBLOCK[0]) LSSU .PAT$GL_MINADDR ! Now record lowest
;  238          0750  4                 THEN PAT$GL_MINADDR = NEWBLOCK[0];       ! Allocated address
;  239          0751  4                 RETURN TRUE;                     ! And we are done
;  240          0752  4              END
;  241          0753  3         ELSE LASTBLOCK = NEWBLOCK[0];            ! When not suitable this block becomes previous bloc
;  242          0754  2         END;                                     ! Of while loop
;  243          0755  2
;  244          0756  2 !++
;  245          0757  2 !       At this point we have reached the end of the free
;  246          0758  2 !       memory list without finding a block of required size.
;  247          0759  2 !       Thus, we expand the address space and attempt to
;  248          0760  2 !       allocate from additional virtual memory.
;  249          0761  2 !--
;  250        P 0762  2 IF PAT$GL_ERRCODE=$EXPREG(PAGCNT=PAT$K_MEMEXP
;  251          0763  3                           , RETADR=PAT$GL_ISVADDR)
;  252          0764  3                                                  ! Successfully expanded program region
;  253          0765  3 THEN BEGIN                                       ! Deallocate new space to end of
;  254          0766  3         IF NOT DEALLOCATE(PAT$K_MAXBLKSIZ,.PAT$GL_ISVADDR[0],LASTBLOCK[0])
;  255          0767  3                 THEN RETURN FALSE;
;  256          0768  3         IF NOT ALLOCATE(.SIZE,.ADDRESS)          ! Free list then allocate from it
;  257          0769  3                 THEN RETURN FALSE;
;  258          0770  3         RETURN TRUE;
;  259          0771  3      END
;  260          0772  3                                                  ! Failure to expand program region
;  261          0773  2 ELSE SIGNAL(PAT$_SYSERROR,0,.PAT$GL_ERRCODE);    ! Is fatal
;  262          0774  1 END;                                             ! Of allocate routine
; INFO#212                L1:0773
; Null expression appears in value-required context
```

```
                                                        .EXTRN  SYS$EXPREG

                                                00FC 00000 ALLOCATE:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7                  ; 0721
                                    57 00000000G    EF  9E 00002        MOVAB   PAT$GL_ERRCODE, R7
                                    56 00000000G    EF  9E 00009        MOVAB   PAT$GL_ISVADDR, R6
                                    55 00000000G    EF  9E 00010        MOVAB   PAT$GL_MINADDR, R5
                                    54 00000000'    EF  9E 00017        MOVAB   LASTBLOCK, R4
                                    64 00000000G    EF  9E 0001E        MOVAB   PAT$GL_MEMLHD, LASTBLOCK    ; 0729
                                    53      04      AC  D0 00025        MOVL    SIZE, R3                    ; 0736
                                    52              64  D0 00029 1$:    MOVL    LASTBLOCK, R2               ; 0734
                                F8  A4              62  D0 0002C        MOVL    (R2), NEWBLOCK
                                                    39  13 00030        BEQL    5$
                                    51      F8      A4  D0 00032        MOVL    NEWBLOCK, R1                ; 0736
                                    53      04      A1  D1 00036        CMPL    4(R1), R3
                                                    05  12 0003A        BNEQ    2$
                                    62              61  D0 0003C        MOVL    (R1), (R2)                  ; 0738
                                                    17  11 0003F        BRB     3$                          ; 0739
                                                    23  1B 00041 2$:    BLEQU   4$                          ; 0743
                                FC  A4              51  C1 00043        ADDL3   R3, R1, NEXTBLOCK           ; 0745
                                    50      FC      A4  D0 00048        MOVL    NEXTBLOCK, R0               ; 0746
                                    60              61  D0 0004C        MOVL    (R1), (R0)
                                04  A0      04      A1  C3 0004F        SUBL3   R3, 4(R1), 4(R0)            ; 0747
                                    62              50  D0 00055        MOVL    R0, (R2)                    ; 0748
                                08  BC              51  D0 00058 3$:    MOVL    R1, @ADDRESS                ; 0749
                                    65              51  D1 0005C        CMPL    R1, PAT$GL_MINADDR
                                                    42  1E 0005F        BGEQU   6$
                                    65              51  D0 00061        MOVL    R1, PAT$GL_MINADDR          ; 0750
                                                    3D  11 00064        BRB     6$                          ; 0751
                                    64              51  D0 00066 4$:    MOVL    R1, LASTBLOCK               ; 0753
                                                    BE  11 00069        BRB     1$                          ; 0734
                                                    7E  7C 0006B 5$:    CLRQ    -(SP)                       ; 0763
                                                    56  DD 0006D        PUSHL   R6
                                        00000000G   8F  DD 0006F        PUSHL   #PAT$K_MEMEXP
                                00000000G   00      04  FB 00075        CALLS   #4, SYS$EXPREG
                                            67      50  D0 0007C        MOVL    R0, PAT$GL_ERRCODE
                                            25      50  E9 0007F        BLBC    R0, 7$
                                                    64  DD 00082        PUSHL   LASTBLOCK                   ; 0766
                                                    66  DD 00084        PUSHL   PAT$GL_ISVADDR
                                        00000000G   8F  DD 00086        PUSHL   #PAT$K_MAXBLKSIZ
                                00000000V   EF          03  FB 0008C    CALLS   #3, DEALLOCATE
                                            22          50  E9 00093    BLBC    R0, 8$
                                                08      AC  DD 00096    PUSHL   ADDRESS                     ; 0768
                                                    53  DD 00099        PUSHL   R3
                                    FF60    CF          02  FB 0009B    CALLS   #2, ALLOCATE
                                            15          50  E9 000A0    BLBC    R0, 8$
                                            50          01  D0 000A3 6$:    MOVL    #1, R0                  ; 0770
                                                        04  000A6        RET
                                                    67  DD 000A7 7$:    PUSHL   PAT$GL_ERRCODE              ; 0773
                                                    7E  D4 000A9        CLRL    -(SP)
                                        00000000G   8F  DD 000AB        PUSHL   #PAT$_SYSERROR
                                00000000G   00          03  FB 000B1    CALLS   #3, LIB$SIGNAL
                                                    50  D4 000B8 8$:    CLRL    R0                          ; 0774
                                                        04  000BA        RET
```

; Routine Size: 187 bytes,   Routine Base: _PAT$CODE + 0048

```
;  264          0775  1 GLOBAL ROUTINE PAT$DEALBLK(SIZE,BLOCKADDR):NOVALUE =
;  265          0776  1 !++
;  266          0777  1 !       Routine to deallocate a block to the free
;  267          0778  1 !       memory list after checking its size
;  268          0779  1 !--
;  269          0780  1
;  270          0781  2 BEGIN
;  271          0782  2
;  272          0783  2 LOCAL
;  273          0784  2         BLOCKSIZE;
;  274          0785  2
;  275          0786  2 !++
;  276          0787  2 ! Initialize error code to deallocate routine.
;  277          0788  2 !--
;  278          0789  2 ERRORCODE = 10;                                         ! Initialize error code
;  279          0790  2
;  280          0791  2 !++
;  281          0792  2 ! Round up the block size to the nearest quadword.
;  282          0793  2 !--
;  283          0794  2 BLOCKSIZE = (.SIZE + 7) AND ( NOT 7);                   ! Round up to a multiple of 8 bytes
;  284          0795  2
;  285          0796  2 !++
;  286          0797  2 ! Now search for place in free list to insert block.
;  287          0798  2 !--
;  288          0799  3 IF (.BLOCKSIZE EQL 0)                                  ! Check block size is
;  289          0800  2 OR
;  290          0801  3 (.BLOCKSIZE GTRU PAT$K_MAXBLKSIZ)                            ! Legal and that it lies
;  291          0802  3 OR (ERRORCODE = .ERRORCODE + 1;                         !
;  292          0803  2 (.BLOCKADDR + .BLOCKSIZE) ) GTRU CONTROL_REGION         ! Completely within program region
;  293          0804  2                                                              ! and if not...
;  294          0805  3 OR (ERRORCODE = .ERRORCODE + 1;                         ! Issue fatal error message
;  295          0806  3 .BLOCKADDR LSSU .PAT$GL_MINADDR)                       ! Also if below minimum allocated address
;  296          0807  2 OR NOT DEALLOCATE(.BLOCKSIZE,.BLOCKADDR,PAT$GL_MEMLHD)  ! Attempt deallocation
;  297          0808  2 THEN SIGNAL(PAT$_MEMBUG,1,.BLOCKSIZE,.BLOCKADDR,.ERRORCODE); ! Issuing fatal error if failure
;  298          0809  2
;  299          0810  2 RETURN;                                                 ! Otherwise just return
;  300          0811  1 END;


                                        000C 00000      .ENTRY  PAT$DEALBLK, Save R2,R3                         ; 0775
                    53 00000000'    EF  9E 00002        MOVAB   ERRORCODE, R3                                   ;
                    63              0A  90 00009        MOVB    #10, ERRORCODE                                  ; 0789
          50  04    AC              07  C1 0000C        ADDL3   #7, SIZE, R0                                    ; 0794
          52        50              07  CB 00011        BICL3   #7, R0, BLOCKSIZE                               ;
                                    3A  13 00015        BEQL    1$                                              ; 0799
    00000000G   8F                  52  D1 00017        CMPL    BLOCKSIZE, #PAT$K_MAXBLKSIZ                     ; 0801
                                    31  1A 0001E        BGTRU   1$                                              ;
                                    63  96 00020        INCB    ERRORCODE                                       ; 0802
          50        52        08    AC  C1 00022        ADDL3   BLOCKADDR, BLOCKSIZE, R0                        ; 0803
    40000000    8F                  50  D1 00027        CMPL    R0, #1073741824                                 ;
                                    21  1A 0002E        BGTRU   1$                                              ;
                                    63  96 00030        INCB    ERRORCODE                                       ; 0805
    00000000G   EF        08    AC  D1 00032        CMPL    BLOCKADDR, PAT$GL_MINADDR                       ; 0806
                                    15  1F 0003A        BLSSU   1$                                              ;
```

```
                          00000000G   EF  9F 0003C          PUSHAB  PAT$GL_MEMLHD                    ; 0807
                                08    AC  DD 00042          PUSHL   BLOCKADDR                        ;
                                      52  DD 0 045          PUSHL   BLOCKSIZE                        ;
                00000000V     EF      03  FB 0 047          CALLS   #3, DEALLOCATE                   ;
                              17      50  E8 0 04E          BLBS    R0, 2$                           ;
                              7E      63  9A 00051 1$:      MOVZBL  ERRORCODE, -(SP)                 ; 0808
                                08    AC  DD 00054          PUSHL   BLOCKADDR                        ;
                                      52  DD 00057          PUSHL   BLOCKSIZE                        ;
                                      01  DD 00059          PUSHL   #1                               ;
                                006D8194  8F  DD 0005B          PUSHL   #7176596                     ;
                00000000G     00      05  FB 00061          CALLS   #5, LIB$SIGNAL                   ;
                                          04 00068 2$:      RET                                      ; 0811

; Routine Size:  105 bytes,    Routine Base:  _PAT$CODE + 0103
```

```
;  302     0812  1 ROUTINE DEALLOCATE(SIZE,ADDRESS,LISTHEAD) =
;  303     0813  1 !++
;  304     0814  1 !       Routine to put a block onto a list of free blocks,
;  305     0815  1 !       with maximal compaction
;  306     0816  1 !--
;  307     0817  1
;  308     0818  2 BEGIN
;  309     0819  2 !++
;  310     0820  2 ! Initialize search down the free list.
;  311     0821  2 !--
;  312     0822  2 LASTBLOCK = .LISTHEAD;                          ! Previous block initially the listhead
;  313     0823  2 NEWBLOCK = .ADDRESS;                            ! Current block is to be inserted
;  314     0824  2
;  315     0825  2 !++
;  316     0826  2 ! Now search the list for the place to insert the free block.
;  317     0827  2 !--
;  318     0828  2 WHILE (NEXTBLOCK = .LASTBLOCK[0])NEQ 0 DO       ! Follow down free list till
;  319     0829  3 BEGIN                                           ! The end, or till we reach
;  320     0830  3 IF NEWBLOCK[0] LEQU NEXTBLOCK[0]
;  321     0831  3 THEN
;  322     0832  4         BEGIN                                   ! The position for insertion.
;  323     0833  4         IF NEWBLOCK[0]+.SIZE EQL NEXTBLOCK[0]
;  324     0834  4         THEN
;  325     0835  5                 BEGIN                           ! Here we compact with next block
;  326     0836  5                 NEWBLOCK[0] = .NEXTBLOCK[0];
;  327     0837  5                 NEWBLOCK[1] = .NEXTBLOCK[1]+.SIZE;
;  328     0838  5                 END
;  329     0839  4         ELSE
;  330     0840  5                 BEGIN
;  331     0841  5                 IF NEWBLOCK[0] + .SIZE GTRU NEXTBLOCK[0] ! If the block to deallocate
;  332     0842  6                 THEN (ERRORCODE = .ERRORCODE + 1;       ! Extends into next free block
;  333     0843  5                         RETURN FALSE);                  ! And return failure
;  334     0844  5                 NEWBLOCK[0] = NEXTBLOCK[0];             ! Else set pointer and size since no
;  335     0845  5                 NEWBLOCK[1] = .SIZE;                    ! Forward compaction needed
;  336     0846  4                 END;
;  337     0847  4         IF NEWBLOCK[0] EQL LASTBLOCK[0]+.LASTBLOCK[1]
;  338     0848  4         THEN
;  339     0849  5                 BEGIN                           ! Here we compact with previous
;  340     0850  5                 LASTBLOCK[0] = .NEWBLOCK[0];    ! Block
;  341     0851  5                 LASTBLOCK[1] = .NEWBLOCK[1]+.LASTBLOCK[1];
;  342     0852  5                 END
;  343     0853  4         ELSE                                            ! No backward compaction but...
;  344     0854  5                 BEGIN                                   ! Must check that block to
;  345     0855  5                 IF NEWBLOCK[0] LSSU LASTBLOCK[0] + .LASTBLOCK[1]
;  346     0856  5                                                         ! Deallocate is not partially in
;  347     0857  6                         THEN (ERRORCODE = .ERRORCODE + 1; ! Previous hole -- failure if so
;  348     0858  5                                 RETURN FALSE);
;  349     0859  5                 LASTBLOCK[0] = NEWBLOCK[0];             ! If ok previous points to new one.
;  350     0860  4                 END;                                    ! And we are done compacting
;  351     0861  4         RETURN TRUE;                                    ! So return success.
;  352     0862  4         END
;  353     0863  3 ELSE
;  354     0864  3         LASTBLOCK = NEXTBLOCK[0];                       ! Not there yet so last block is one just tested
;  355     0865  2 END;                                                    ! Of while loop
;  356     0866  2
;  357     0867  2 !++
;  358     0868  2 ! The block to deallocate is beyond last hole
```

```
;  359          0869  2 !--
;  360          0870  2 IF NEWBLOCK[0] LSSU LASTBLOCK[0] + .LASTBLOCK[1]          ! But if it starts within
;  361          0871  3 THEN (ERRORCODE = .ERRORCODE + 1;                       ! The last hole - fail it
;  362          0872  3                 RETURN FALSE)
;  363          0873  2 ELSE
;  364          0874  3         BEGIN                                           ! Otherwise check for compaction
;  365          0875  3         IF NEWBLOCK[0] EQL LASTBLOCK[0] + .LASTBLOCK[1] ! With last hole
;  366          0876  3         THEN LASTBLOCK[1] = .LASTBLOCK[1] + .SIZE       ! And add in size if required
;  367          0877  3         ELSE
;  368          0878  4                 BEGIN                                   ! Otherwise just
;  369          0879  4                 NEWBLOCK[0] = 0;                        ! Put on end of free list.
;  370          0880  4                 NEWBLOCK[1] =.SIZE;
;  371          0881  4                 LASTBLOCK[0] = NEWBLOCK[0];
;  372          0882  3                 END;                                    ! And all done
;  373          0883  3         RETURN TRUE;                                    ! So return success
;  374          0884  2         END;
;  375          0885  1 END;                                                    ! Of routine


                                001C 00000 DEALLOCATE:
                                            .WORD   Save R2,R3,R4                   ; 0812
                    54 00000000'  EF  9E 00002  MOVAB   LASTBLOCK, R4               ;
                    64        0C  AC  D0 00009  MOVL    LISTHEAD, LASTBLOCK       ; 0822
                 F8 A4        08  AC  D0 0000D  MOVL    ADDRESS, NEWBLOCK         ; 0823
                    51        F8  A4  D0 00012  MOVL    NEWBLOCK, R1              ; 0830
                    50            64  D0 00016 1$:    MOVL    LASTBLOCK, R0       ; 0828
                 FC A4            60  D0 00019        MOVL    (R0), NEXTBLOCK     ;
                                  46  13 0001D        BEQL    6$                  ;
                    52        FC  A4  D0 0001F        MOVL    NEXTBLOCK, R2       ; 0830
                    52            51  D1 00023        CMPL    R1, R2              ;
                                  38  1A 00026        BGTRU   5$                  ;
                 53               51  04  AC  C1 00028        ADDL3   SIZE, R1, R3    ; 0833
                    52            53  D1 0002D        CMPL    R3, R2              ;
                                  0C  12 00030        BNEQ    2$                  ;
                                  61  62  D0 00032        MOVL    (R2), (R1)          ; 0836
04 A1 04 A2 04 AC C1 00035        ADDL3   SIZE, 4(R2), 4(R1)  ; 0837
                                  0A  11 0003C        BRB     3$                  ; 0833
                                  32  1A 0003E 2$:    BGTRU   7$                  ; 0841
                                  61  52  D0 00040        MOVL    R2, (R1)            ; 0844
                    04 A1         04  AC  D0 00043        MOVL    SIZE, 4(R1)         ; 0845
                 52               50  04  A0  C1 00048 3$:    ADDL3   4(R0), R0, R2   ; 0847
                                  52  51  D1 0004D        CMPL    R1, R2              ;
                                  0A  12 00050        BNEQ    4$                  ;
                                  60  61  D0 00052        MOVL    (R1), (R0)          ; 0850
                    04 A0         04  A1  C0 00055        ADDL2   4(R1), 4(R0)        ; 0851
                                  2E  11 0005A        BRB     11$                 ; 0847
                                  29  1E 0005C 4$:    BGEQU   10$                 ; 0855
                                  12  11 0005E        BRB     7$                  ; 0857
                                  64  52  D0 00060 5$:    MOVL    R2, LASTBLOCK       ; 0864
                                  B1  11 00063        BRB     1$                  ; 0828
                                  50  64  D0 00065 6$:    MOVL    LASTBLOCK, R0       ; 0870
                 52               50  04  A0  C1 00068        ADDL3   4(R0), R0, R2   ;
                                  52  51  D1 0006D        CMPL    R1, R2              ;
                                  05  1E 00070        BGEQU   8$                  ;
```

```
                                        F4    A4  96 00072 7$:     INCB    ERRORCODE                                       ; 0871
                                              17  11 00075        BRB     12$                                             ; 0874
                                              07  12 00077 8$:     BNEQ    9$                                              ; 0875
                                 04   A0      04   A0  C0 00079        ADDL2   SIZE, 4(R0)                                     ; 0876
                                              0A  11 0007E        BRB     11$                                             ;
                                              61  D4 00080 9$:     CLRL    (R1)                                            ; 0879
                                 04   A1      04   AC  D0 00082        MOVL    SIZE, 4(R1)                                     ; 0880
                                      60           51  D0 00087 10$:    MOVL    R1, (R0)                                        ; 0881
                                      50           01  D0 0008A 11$:    MOVL    #1, R0                                          ; 0883
                                                   04  RET                                                                     ; 0874
                                              50  D4 0008E 12$:     CLRL    R0                                              ; 0885
                                                  04 00090        RET                                                         ;

; Routine Size:  145 bytes,    Routine Base:  _PAT$CODE + 016C

;  376          0886  0 END ELUDOM                                          ! Of module


                                                                    .EXTRN  LIB$SIGNAL

;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _PAT$OWN                       16  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  _PAT$CODE                     509  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]LIB.L32;1          18619         7         0      1000          00:01.8


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


;                                       COMMAND QUALIFIERS

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

; Size:         509 code + 16 data bytes
; Run Time:       00:19.0
; Elapsed Time:   01:09.9
; Lines/CPU Min:   2797
```

```
; Lexemes/CPU-Min: 40553
; Memory Used:  136 pages
; Compilation Complete
```