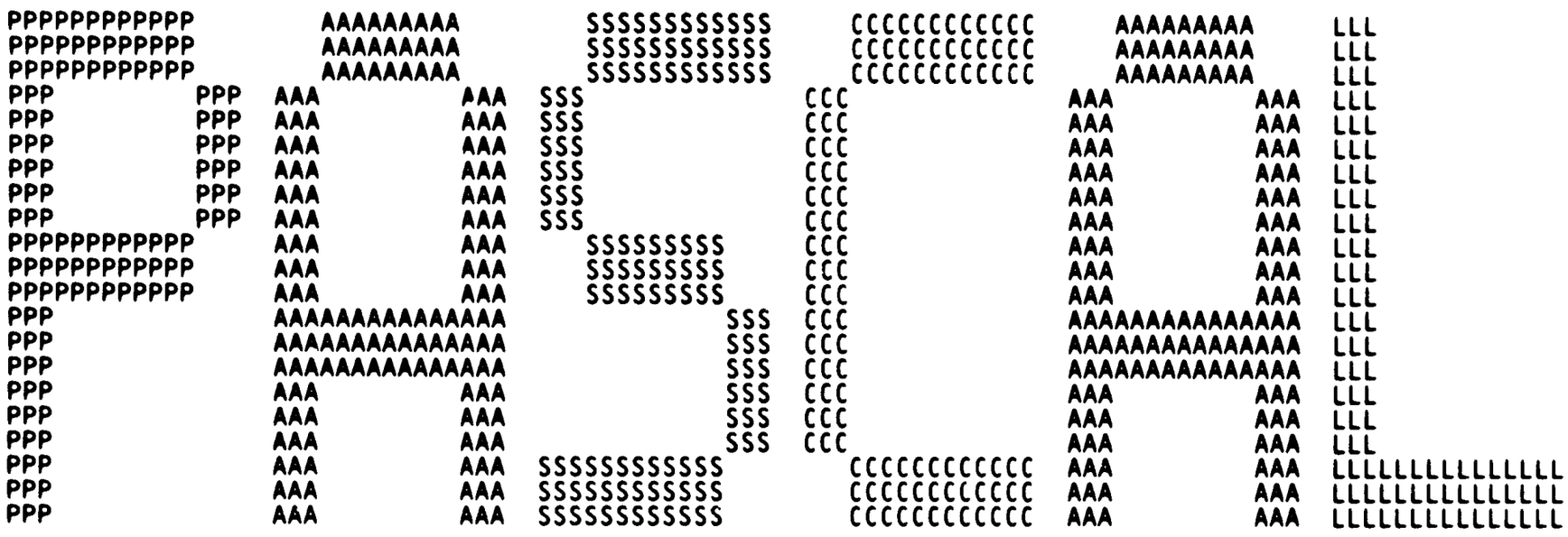
PPPPPPPPPPPP AAAAAAAAA SSSSSSSSSSS CCCCCCCCCCC AAAAAAAAA LLL
PPPPPPPPPPPP AAAAAAAAA SSSSSSSSSSS CCCCCCCCCCC AAAAAAAAA LLL
PPPPPPPPPPPP AAAAAAAAA SSSSSSSSSSS CCCCCCCCCCC AAAAAAAAA LLL
PPP PPP AAA AAA SSS CCC AAA AAA LLL
PPP PPP AAA AAA SSS CCC AAA AAA LLL
PPP PPP AAA AAA SSS CCC AAA AAA LLL
PPP PPP AAA AAA SSS CCC AAA AAA LLL
PPP PPP AAA AAA SSS CCC AAA AAA LLL
PPPPPPPPPPPP AAA AAA SSSSSSSSS CCC AAA AAA LLL
PPPPPPPPPPPP AAA AAA SSSSSSSSS CCC AAA AAA LLL
PPPPPPPPPPPP AAA AAA SSSSSSSSS CCC AAA AAA LLL
PPP AAAAAAAAAAAAAAA SSS CCC AAAAAAAAAAAAAAA LLL
PPP AAAAAAAAAAAAAAA SSS CCC AAAAAAAAAAAAAAA LLL
PPP AAAAAAAAAAAAAAA SSS CCC AAAAAAAAAAAAAAA LLL
PPP AAA AAA SSS CCC AAA AAA LLL
PPP AAA AAA SSS CCC AAA AAA LLL
PPP AAA AAA SSS CCC AAA AAA LLL
PPP AAA AAA SSSSSSSSSSS CCCCCCCCCCC AAA AAA LLLLLLLLLLLLLLL
PPP AAA AAA SSSSSSSSSSS CCCCCCCCCCC AAA AAA LLLLLLLLLLLLLLL
PPP AAA AAA SSSSSSSSSSS CCCCCCCCCCC AAA AAA LLLLLLLLLLLLLLL

```
PPPPPPPP      AAAAAA      SSSSSSSS  RRRRRRRR    TTTTTTTTTT    222222
PPPPPPPP      AAAAAA      SSSSSSSS  RRRRRRRR    TTTTTTTTTT    222222
PP      PP   AA    AA   SS        RR      RR      TT      22      22
PP      PP   AA    AA   SS        RR      RR      TT      22      22
PP      PP   AA    AA   SS        RR      RR      TT              22
PP      PP   AA    AA   SS        RR      RR      TT              22
PPPPPPPP     AA    AA     SSSSSS  RRRRRRRR        TT            22
PPPPPPPP     AA    AA     SSSSSS  RRRRRRRR        TT            22
PP           AAAAAAAAAA         SS RR  RR         TT          22
PP           AAAAAAAAAA         SS RR  RR         TT          22
PP           AA    AA           SS RR    RR       TT        22          ....
PP           AA    AA           SS RR    RR       TT        22          ....
PP           AA    AA  SSSSSSSS   RR      RR      TT      2222222222    ....
PP           AA    AA  SSSSSSSS   RR      RR      TT      2222222222    ....


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
```

```
0000     1 ;
0000     2 ;*****************************************************************************
0000     3 ;*                                                                           *
0000     4 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     5 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     6 ;*  ALL RIGHTS RESERVED.                                                     *
0000     7 ;*                                                                           *
0000     8 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000     9 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    10 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    11 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    12 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    13 ;*  TRANSFERRED.                                                             *
0000    14 ;*                                                                           *
0000    15 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    16 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    17 ;*  CORPORATION.                                                             *
0000    18 ;*                                                                           *
0000    19 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    20 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    21 ;*                                                                           *
0000    22 ;*                                                                           *
0000    23 ;*****************************************************************************
0000    24 ;*                                                                           *
0000    25 ;*                              PAS$RT_HEAP                                  *
0000    26 ;*                RUNTIME SUPPORT MODULE FOR PASCAL -- SECTION 2             *
0000    27 ;*                                                                           *
0000    28 ;*                        VERSION V1.0-1 -- OCTOBER 1979                     *
0000    29 ;*                                                                           *
0000    30 ;*    This module defines the following routines:                            *
0000    31 ;*                                                                           *
0000    32 ;*       pas$new:     routine to implement the Pascal procedure new(p)       *
0000    33 ;*       pas$dispose: routine to implement the Pascal procedure dispose(p)   *
0000    34 ;*       pas$mark:    routine to implement the Pascal procedure mark(p)      *
0000    35 ;*       pas$release: routine to implement the Pascal procedure release(p)   *
0000    36 ;*       pas$snap:    debug routine to examine state of the heap             *
0000    37 ;*                                                                           *
0000    38 ;*    Written by: Jeff Scofield 10-Dec-78                                    *
0000    39 ;*                                                                           *
0000    40 ;*    Edit History :                                                         *
0000    41 ;*          01-002 Paul Hohensee 27feb80. Modified to prevent duplicate      *
0000    42 ;*                 error messages form printing. Eliminate for v1.2          *
0000    43 ;
0000    44 ;       01-003 Susan Azibert 28may80. Changed the printing of error
0000    45 ;              messages so that they are both printed and signaled
0000    46 ;              by calling lib$stop.
0000    47 ;
0000    48 ;       01-004 Susan Azibert 24sep80. Fixed bug introduced by change
0000    49 ;              01-003 in outputting message attdisinv
0000    50 ;
0000    51 ;       01-005 Paul Hohensee 11-Aug-81. Changed references to external routines to u
0000    52 ;*
0000    53 ;*****************************************************************************
0000    54          .title  pas$rt_heap
0000    55          .ident  'V04-000'
0000    56
0000    57          .extrn  pas$_proexchea
```

```
                        0000     58          .extrn  pas$_attdisinv
                        0000     59
                        0000     60          $stsdef
                        0000     61 ;
                        0000     62 ;   ROUTINES  TO  IMPLEMENT  THE  PASCAL  HEAP-MANAGEMENT  PROCEDURES
                        0000     63 ;                       NEW,  DISPOSE,  MARK,  RELEASE
                        0000     64 ;
                        0000     65 ;
                        0000     66 ;   OWN STORAGE USED BY THE ROUTINES
                        0000     67 ;
                    00000000     68          .psect _pas$data,pic,noexe,2
                        0000     69 ;
                        0000     70 ;       lasps: pointers to the lists of available space for each level
                        0000     71 ;       pblbs: pointers to the page block lists for each level
                        0000     72 ;       pools: pointers to the pools of free page block for each level
                        0000     73 ;       marks: pointers to the next block of LAS to be allocated by
                        0000     74 ;                procedure mark
                        0000     75 ;       nestl: the nesting level
                        0000     76 ;
00000000 00000000       0000     77 lasps:   .long   0,0                     ; pointers to LAS's, initialized to nil
00000000 00000000       0008     78 pblps:   .long   0,0                     ; pointers to PBL's, initialized to nil
00000000 00000000       0010     79 pools:   .long   0,0                     ; pointers to pools, initialized to nil
00000000 00000000       0018     80 marks:   .long   0,0                     ; pointers to marks, initialized to nil
              00        0020     81 nestl:   .byte   0                       ; re-entrancy level, initialized to 0
                        0021     82 ;
                    00000000     83          .psect _pas$code,pic,shr,exe,nowrt
                        0000     84 ;
                        0000     85 ;   ROUTINE TO IMPLEMENT THE PASCAL STANDARD PROCEDURE NEW
                        0000     86 ;
                    03FC 0000    87          .entry  pas$new,^m<r2,r3,r4,r5,r6,r7,r8,r9>
                        0002     88 ;
                        0002     89 ;   Move important values to registers
                        0002     90 ;
      00000020'EF     96 0002    91          incb    nestl                   ; increment nest level
  52  00000020'EF     9A 0008    92          movzbl  nestl,r2                ; r2 <- incremented nesting level
53  FFFFFFFC'EF42     D0 000F    93          movl    lasps-4[r2],r3          ; r3 <- LASP for this level
54  00000004'EF42     D0 0017    94          movl    pblps-4[r2],r4          ; r4 <- PBLP for this level
55  0000000C'EF42     D0 001F    95          movl    pools-4[r2],r5          ; r5 <- addr of first block of pool
        56  04 AC     D0 0027    96          movl    4(ap),r6                ; r6 <- # of bytes to allocate
                        002B     97 ;
                        002B     98 ;   Look through LAS for first block large enough to satisfy the request.
                        002B     99 ;   Register 7 finds the predecessor of the block, r8 finds the block.
                        002B    100 ;
            57  D4 002B         101          clrl    r7                      ; initialize the predecessor to nil
        58  53  D0 002D         102          movl    r3,r8                   ; initialize r8 to LASP
            11  13 0030         103          beql    inpool                  ; LAS is empty--find other memory
    56  FC A8  D1 0032         104 10$:     cmpl    -4(r8),r6               ; is this block big enough?
            03  1F 0036         105          blssu   20$                     ; continue looking if not
          00F1  31 0038         106          brw     alloc                   ; else take it
        57  58  D0 003B         107 20$:     movl    r8,r7                   ; no--try next block
        58  68  D0 003E         108          movl    (r8),r8
            EF  12 0041         109          bneq    10$                     ; loop until found or end of LAS
                        0043    110 ;
                        0043    111 ;   Block was not found on LAS.  Try to find block in pool that will do.
                        0043    112 ;   Register 7 has address of last block of LAS; r8 will find predecessor
                        0043    113 ;   of page block, r9 will find page block in pool.
                        0043    114 ;
```

```
                                 0043   115 inpool:                                 ; try to find block in pool
                          58     D4 0043   116          clrl    r8                  ; initialize predecessor to nil
                       59        55 D0 0045   117          movl    r5,r9               ; initialize r9 to pool ptr
                          20     13 0048   118 10$:       beql    expand              ; pool is empty--expand memory
                    50    FC A9  08 C3 004A   119          subl3   #8,-4(r9),r0        ; r0 <- avail space in this block
                       56        50 D1 004F   120          cmpl    r0,r6               ; is there room in this block?
                          08     1E 0052   121          bgequ   20$                 ; yes--link new blk into LAS & PBL
                       58        59 D0 0054   122          movl    r9,r8               ; no--try next page block
                       59        69 D0 0057   123          movl    (r9),r9
                          EC     11 005A   124          brb     10$                 ; loop until found or end of pool
                          58     D5 005C   125 20$:       tstl    r8                  ; is there a predecessor of pg block?
                          05     13 005E   126          beql    30$                 ; no--remove first block of PBL
                       68        69 D0 0060   127          movl    (r9),(r8)           ; yes--link predecessor to successor
                          48     11 0063   128          brb     linkpb              ; link page block into LAS & PBL
                       55        69 D0 0065   129 30$:       movl    (r9),r5             ; remove first block of PBL
                          43     11 0068   130          brb     linkpb              ; link page block into LAS & PBL
                                 006A   131 ;
                                 006A   132 ;   No block was found in pool.  Expand program region to get new block.
                                 006A   133 ;
                                 006A   134 expand:                                 ; expand program region for memory
                58 56   00000207 8F C1 006A   135          addl3   #519,r6,r8          ; round (r6 + 8) up to next 512 bytes
                58      000001FF 8F CA 0072   136          bicl2   #511,r8             ; r8 <- # of bytes to ask for
                                 58 DD 0079   137          pushl   r8                  ; -4(fp) <- # of bytes to ask for
                             59     7E DE 007B   138          moval   -(sp),r9            ; r9 <- addr of free longw on stack
                                 59 DD 007E   139          pushl   r9                  ; push arg2: addr to get addr
                              FC    AD DF 0080   140          pushal  -4(fp)              ; push arg1: addr of # bytes asked for
                00000000'GF      02 FB 0083   141          calls   #2,G^lib$get_vm     ; get memory; (r9) <- addr of mem
                           1A    50 E8 008A   142          blbs    r0,10$              ; continue if no errors
                 7E    10 AD     07 C3 008D   143          subl3   #7,16(fp),-(sp)     ; third FAO argument (PC of call)
                                 00 DD 0092   144          pushl   #0                  ; second FAO argument (null)
                                 00 DD 0094   145          pushl   #0                  ; first FAO argument (null)
                                 03 DD 0096   146          pushl   #3                  ; number of FAO arguments preceding
             7E 00000000'8F      04 C1 0098   147          addl3   #4,#pas$_proexchea,-(sp)        ; error message #8110
                00000000'GF      05 FB 00A0   148          calls   #5,G^lib$stop       ; signal error and stop execution
                          59     69 D0 00A7   149 10$:       movl    (r9),r9             ; r9 <- addr of new memory
                          89     58 D0 00AA   150          movl    r8,(r9)+            ; set size of new block
                                 00AD   151 ;
                                 00AD   152 ;   Link new page block to end of PBL.  If adjacent to previous page
                                 00AD   153 ;   block, merge the two and set flag so that LAS is also merged if
                                 00AD   154 ;   possible.   Register 4 has addr of last page block, r9 has addr of
                                 00AD   155 ;   new page block.
                                 00AD   156 ;
                                 00AD   157 linkpb:                                 ; link new page block to PBL
                                 51 D4 00AD   158          clrl    r1                  ; r1 is flag: init to 0
                                 54 D5 00AF   159          tstl    r4                  ; is PBL empty?
                                 13 13 00B1   160          beql    10$                 ; yes--link new block to PBL
                50    54    FC A4  C1 00B3   161          addl3   -4(r4),r4,r0        ; no--r0 <- addr of end of last blk
                          50     59 D1 00B8   162          cmpl    r9,r0               ; is new block adjacent?
                                 09 12 00BB   163          bneq    10$                 ; no--link new block to PBL
                  FC A4    FC A9  C0 00BD   164          addl2   -4(r9),-4(r4)       ; yes--lengthen old block
                                 51 D6 00C2   165          incl    r1                  ; set flag to show merge done
                                 0F 11 00C4   166          brb     linkla              ; go link block into LAS
                          69     54 D0 00C6   167 10$:       movl    r4,(r9)             ; link new block to PBL
                          54     59 D0 00C9   168          movl    r9,r4               ; set new head of PBL
                          59     08 C0 00CC   169          addl2   #8,r9               ; r9 <- addr of new LAS block
             FC A9    F4 A9     08 C3 00CF   170          subl3   #8,-12(r9),-4(r9)   ; set length of new LAS block
                                 00D5   171 ;
```

```
                                00D5    172 ;   Link new block into LAS, kept sorted in memory order.  If r1 = 1,
                                00D5    173 ;   try to merge new block with predecessor.   Register 7 has addr of
                                00D5    174 ;   last block of LAS, r9 has address of new block.
                                00D5    175 ;
                                00D5    176 linkla:                                     ; link new block into LAS
                          58    D4    00D5    177             clrl    r8                ; set successor to nil
                      57  59    D1    00D7    178             cmpl    r9,r7             ; new block goes at end of LAS?
                          12    1A    00DA    179             bgtru   20$               ; yes--go ahead and insert
                          57    D4    00DC    180             clrl    r7                ; r7 finds predecessor of block
                      58  53    D0    00DE    181             movl    r3,r8             ; r8 finds successor of block
                      58  59    D1    00E1    182 10$:        cmpl    r9,r8             ; does block belong here?
                          08    1F    00E4    183             blssu   20$               ; yes--insert
                      57  58    D0    00E6    184             movl    r8,r7             ; no--try after next block
                      58  68    D0    00E9    185             movl    (r8),r8
                          F3    12    00EC    186             bneq    10$               ; loop until found or end of list
                          57    D5    00EE    187 20$:        tstl    r7                ; is there a predecessor?
                          29    13    00F0    188             beql    50$               ; no--add to front of LAS
                       21 51    E9    00F2    189             blbc    r1,40$            ; do not merge if flag clear
                50    57   FC  A7    C1    00F5    190             addl3   -4(r7),r7,r0      ; r0 <- addr of end of predecessor
                      50  59    D1    00FA    191             cmpl    r9,r0             ; is new block adjacent?
                          17    12    00FD    192             bneq    40$               ; no--link predecessor to block
                FC A7     FC  A9    C0    00FF    193             addl2   -4(r9),-4(r7)     ; yes--lengthen predecessor
                      59  57    D0    0104    194             movl    r7,r9             ; use lengthened block
                      57  53    D0    0107    195             movl    r3,r7             ; init r7 to LASP, find pred of (r9)
                      59  67    D1    010A    196 30$:        cmpl    (r7),r9           ; is this predecessor of (r9)?
                          12    13    010D    197             beql    setmk             ; yes--go set mark location
                      57  67    D0    010F    198             movl    (r7),r7           ; no--try next block
                          F6    12    0112    199             bneq    30$               ; loop until found or end of list
                          08    11    0114    200             brb     setmk             ; go set mark location
                      67  59    D0    0116    201 40$:        movl    r9,(r7)           ; link predecessor to block
                          03    11    0119    202             brb     60$
                      53  59    D0    011B    203 50$:        movl    r9,r3             ; add new block to front of LAS
                      69  58    D0    011E    204 60$:        movl    r8,(r9)           ; link new block to successor
                                0121    205 ;
                                0121    206 ;   Set location to be marked by next call of pas$mark
                                0121    207 ;
                          58    59    D0    0121    208 setmk:  movl    r9,r8             ; r8 <- addr of block to allocate
00000014'EF42             58    D0    0124    209             movl     8,marks-4[r2]    ; set mark location to new block
                                012C    210 ;
                                012C    211 ;   Allocate part or all of the block whose address is in r8.  Register 6
                                012C    212 ;   has the number of bytes to allocate, r7 has addr of the predecessor
                                012C    213 ;   of the block.
                                012C    214 ;
                50    FC  A8    56    C3    012C    215 alloc:  subl3   r6,-4(r8),r0      ; compare sizes
                          04    50    D1    0131    216             cmpl    r0,#4             ; within 4 bytes?
                                1F    14    0134    217             bgtr    split             ; no--split block
                                      0136    218                                       ; yes--remove whole block from LAS
                                57    D5    0136    219             tstl    r7                ; is there a predecessor?
                                05    13    0138    220             beql    10$               ; no--remove first block from LAS
                          67    68    D0    013A    221             movl    (r8),(r7)         ; yes--link predecessor to successor
                                03    11    013D    222             brb     20$
                          53    68    D0    013F    223 10$:        movl    (r8),r3           ; set new LASP
00000014'EF42             58    D1    0142    224 20$:        cmpl    r8,marks-4[r2]    ; is this marked block?
                                36    12    014A    225             bneq    exitmn            ; no--return
                00000014'EF42       D4    014C    226             clrl    marks-4[r2]       ; yes--set mark to nil
                                2D    11    0153    227             brb     exitmn            ; return
                                      0155    228 ;
```

```
                        0155   229 ;   Split block whose address is in r8.  Register 7 has address of
                        0155   230 ;   predecessor of this block, r0 has number of bytes left over after
                        0155   231 ;   split, r6 has number of bytes to allocate.
                        0155   232 ;
          FC A8   56    D0  0155   233 split:  movl    r6,-4(r8)               ; set size of alloc block
        59   58   56    C1  0159   234         addl3   r6,r8,r9                ; r9 <- addr of remainder
          FC A9   50    D0  015D   235         movl    r0,-4(r9)               ; set size of remainder
             69   68    D0  0161   236         movl    (r8),(r9)               ; set link of remainder
                  57    D5  0164   237         tstl    r7                      ; is there a predecessor?
                  05    13  0166   238         beql    10$                     ; no--link to beginning of LAS
             67   59    D0  0168   239         movl    r9,(r7)                 ; link predecessor to block
                  03    11  016B   240         brb     20$
             53   59    D0  016D   241 10$:    movl    r9,r3                   ; set new head of LAS
00000014'EF42   58    D1  0170   242 20$:    cmpl    r8,marks-4[r2]          ; is this marked block?
                  08    12  0178   243         bneq    exitmn                  ; no--return
00000014'EF42   59    D0  017A   244         movl    r9,marks-4[r2]          ; yes--set mark to remainder
                        0182   245 ;
                        0182   246 ;   Store values that may have been changed, set return value, return.
                        0182   247 ;
                        0182   248 exitmn:                                    ; exit from pas$mark or pas$new
FFFFFFFC'EF42   53    D0  0182   249         movl    r3,lasps-4[r2]          ; store LASP
00000004'EF42   54    D0  018A   250         movl    r4,pblps-4[r2]          ; store PBLP
0000000C'EF42   55    D0  0192   251         movl    r5,pools-4[r2]          ; store pool pointer
          08 BC   58    D0  019A   252         movl    r8,@8(ap)               ; return pointer to allocated block
        00000020'EF   97    019E   253         decb    nestl                   ; reset nesting level
                  04    01A4   254         ret
                        01A5   255 ;
                        01A5   256 ;   ROUTINE TO IMPLEMENT THE PASCAL PROCEDURE MARK
                        01A5   257 ;
                        01A5   258 ;      This routine functions as a second entry point for pas$new, sharing
                        01A5   259 ;      much of the code of that routine.
                        01A5   260 ;
                 03FC   01A5   261         .entry  pas$mark,^m<r2,r3,r4,r5,r6,r7,r8,r9>
                        01A7   262 ;
                        01A7   263 ;   Move important values to registers
                        01A7   264 ;
      00000020'EF   96  01A7   265         incb    nestl                   ; increment nest level
  52  00000020'EF   9A  01AD   266         movzbl  nestl,r2                ; r2 <- incremented nesting level
53  FFFFFFFC'EF42   D0  01B4   267         movl    lasps-4[r2],r3          ; r3 <- LASP for this level
54  00000004'EF42   D0  01BC   268         movl    pblps-4[r2],r4          ; r4 <- PBLP for this level
55  0000000C'EF42   D0  01C4   269         movl    pools-4[r2],r5          ; r5 <- addr of first block of pool
       56   04 AC   D0  01CC   270         movl    4(ap),r6                ; r6 <- # of bytes to allocate
                        01D0   271 ;
                        01D0   272 ;   If current marked block exists & is large enough, take it.
                        01D0   273 ;
          57   53    D0  01D0   274         movl    r3,r7                   ; r7 <- LASP
58  00000014'EF42   D0  01D3   275         movl    marks-4[r2],r8          ; r8 <- addr of marked LAS block
                16    13  01DB   276         beql    locend                  ; no marked block--find end of LAS
        56   FC A8   D1  01DD   277         cmpl    -4(r8),r6               ; is marked block big enough?
                10    19  01E1   278         blss    locend                  ; no--find end of LAS
                        01E3   279                                         ; yes--find predecessor of marked block
          58   67    D1  01E3   280 10$:    cmpl    (r7),r8                 ; is this predecessor of marked block?
                03    12  01E6   281         bneq    20$                     ; no--keep looking
              FF41    31  01E8   282         brw     alloc                   ; yes--go allocate the marked block
          57   67    D0  01EB   283 20$:    movl    (r7),r7                 ; else try next block
                F3    12  01EE   284         bneq    10$                     ; loop until found or end of LAS
              FF39    31  01F0   285         brw     alloc                   ; go allocate the marked block
```

```
                                01F3   286 ;
                                01F3   287 ;   Current marked block was not big enough.  Find end of LAS, go try to
                                01F3   288 ;   get marked block from pool, or else get it by expanding memory.
                                01F3   289 ;
                                01F3   290 locend:                                      ; locate end of LAS
                          58    D4   01F3   291         clrl    r8                      ; r8 <- nil successor
                          57    D5   01F5   292         tstl    r7                      ; LASP is nil?
                          03    12   01F7   293         bneq    10$                     ; no--find end of LAS
                        FE47    31   01F9   294         brw     inpool                  ; yes--go find block to mark
                          67    D5   01FC   295 10$:    tstl    (r7)                    ; r7 is last LAS block?
                          03    12   01FE   296         bneq    20$                     ; no--keep looking
                        FE40    31   0200   297         brw     inpool                  ; yes--go find block to mark
                      57  67    D0   0203   298 20$:    movl    (r7),r7                 ; else try next block
                          F4    11   0206   299         brb     10$                     ; loop until last block found
                                0208   300 ;
                                0208   301 ;   ROUTINE TO IMPLEMENT THE PASCAL PROCEDURE RELEASE
                                0208   302 ;
                             03FC   0208   303         .entry  pas$release,^m<r2,r3,r4,r5,r6,r7,r8,r9>
                                020A   304 ;
                                020A   305 ;   Move important values to registers
                                020A   306 ;
              00000020'EF       96   020A   307         incb    nestl                   ; increment the nesting level
            52  00000020'EF     9A   0210   308         movzbl  nestl,r2                ; r2 <- incremented nesting level
          53  FFFFFFFC'EF42     D0   0217   309         movl    lasps-4[r2],r3          ; r3 <- LASP for this level
          54  00000004'EF42     D0   021F   310         movl    pblps-4[r2],r4          ; r4 <- PBLP for this level
                          07    12   0227   311         bneq    10$                     ; continue if not nil
              00000020'EF       97   0229   312         decb    nestl                   ; PBLP is nil--nothing to do
                                04   022F   313         ret                             ; back to caller
          55  0000000C'EF42     D0   0230   314 10$:    movl    pools-4[r2],r5          ; r5 <- addr of pool for this level
                56  04 BC      D0   0238   315         movl    @4(ap),r6               ; r6 <- addr to be released
                                023C   316 ;
                                023C   317 ;   Make sure that the block containing the addr to be released is on the
                                023C   318 ;   PBL of this nesting level.  If so, set r7 to addr of the page block
                                023C   319 ;   containing the released address.  If not, there is nothing to do.
                                023C   320 ;
                    57  54      D0   023C   321         movl    r4,r7                   ; init r7 to first page block
                    56  57      D1   023F   322 20$:    cmpl    r7,r6                   ; freed addr before this block?
                        0A      1A   0242   323         bgtru   30$                     ; yes--loop to next block
          50        57  FC A7    C1   0244   324         addl3   -4(r7),r7,r0            ; r0 <- addr of end of block
                    56  50      D1   0249   325         cmpl    r0,r6                   ; freed addr inside this block?
                        0C      1A   024C   326         bgtu    pool                    ; yes--go add blocks to pool
                    57  67      D0   024E   327 30$:    movl    (r7),r7                 ; no--try next block
                        EC      12   0251   328         bneq    20$                     ; loop until found or end of list
              00000020'EF       97   0253   329         decb    nestl                   ; block not found--nothing to do
                                04   0259   330         ret                             ; back to caller
                                025A   331 ;
                                025A   332 ;   Add the freed blocks to the pool, kept sorted in memory order.
                                025A   333 ;   Register 4 has address of next block to add to pool, r8 and r9 find
                                025A   334 ;   the spot to insert the block.
                                025A   335 ;
                    57  54      D1   025A   336 pool:   cmpl    r4,r7                   ; this block contains freed addr?
                        2B      13   025D   337         beql    part                    ; yes--done adding to pool
                        58      D4   025F   338         clrl    r8                      ; r8 finds predecessor of added block
                    59  55      D0   0261   339         movl    r5,r9                   ; r9 finds successor of added block
                        0D      13   0264   340 10$:    beql    20$                     ; end of list--add new block
                    59  54      D1   0266   341         cmpl    r4,r9                   ; new block goes before (r9)?
                        08      1F   0269   342         blssu   20$                     ; yes--add new block
```

```
                58    59    D0   026B   343          movl    r9,r8                   ; no--try next block
                59    69    D0   026E   344          movl    (r9),r9
                      F1    11   0271   345          brb     10$                     ; loop until spot found
                      58    D5   0273   346 20$:     tstl    r8                      ; is there a predecessor?
                      05    13   0275   347          beql    30$                     ; no--add to front of pool
                68    54    D0   0277   348          movl    r4,(r8)                 ; else link predecessor to new block
                      03    11   027A   349          brb     40$
                55    54    D0   027C   350 30$:     movl    r4,r5                   ; add new block to front of pool
                50    64    D0   027F   351 40$:     movl    (r4),r0                 ; save addr of next block of PBL
                64    59    D0   0282   352          movl    r9,(r4)                 ; link new block to successor
                54    50    D0   0285   353          movl    r0,r4                   ; move to next block of PBL
                      D0    11   0288   354          brb     pool                    ; loop through all freed blocks
                            028A   355 ;
                            028A   356 ;   Find the spot in LAS for the freed storage in the page block containing
                            028A   357 ;   the freed pointer.   LAS is also kept sorted in memory order.  Register 6
                            028A   358 ;   has the address of the freed storage, r8 and r9 find the correct spot
                            028A   359 ;   in LAS for the new free block.
                            028A   360 ;
                      58    D4   028A   361 part:    clrl    r8                      ; r8 finds predecessor of new block
                59    53    D0   028C   362          movl    r3,r9                   ; r9 finds successor of new block
                      0D    13   028F   363 10$:     beql    20$                     ; end of list--add new block
                59    56    D1   0291   364          cmpl    r6,r9                   ; new block goes before (r9)?
                      08    1F   0294   365          blssu   20$                     ; yes--add new block
                58    59    D0   0296   366          movl    r9,r8                   ; no--try next block
                59    69    D0   0299   367          movl    (r9),r9
                      F1    11   029C   368          brb     10$                     ; loop until spot found
                      58    D5   029E   369 20$:     tstl    r8                      ; is there a predecessor?
                      05    13   02A0   370          beql    30$                     ; no--add to front of LAS
                68    56    D0   02A2   371          movl    r6,(r8)                 ; else link predecessor to new block
                      03    11   02A5   372          brb     40$                     ; go set mark loc
                53    56    D0   02A7   373 30$:     movl    r6,r3                   ; add new block to front of pool
00000014'EF42   56    D0   02AA   374 40$:     movl    r6,marks-4[r2]          ; next spot to mark is (r6)
                            02B2   375 ;
                            02B2   376 ;   Compute size of newly-added block, and remove from LAS all former
                            02B2   377 ;   free blocks falling inside the new block.  Register 6 is address of
                            02B2   378 ;   newly-added block, r9 is successor of block, r7 is addr of page block
                            02B2   379 ;   containing the new free block.
                            02B2   380 ;
50   FC A7    57    C1   02B2   381 size:    addl3   r7,-4(r7),r0            ; r0 <- addr of end of page block
FC A6   50    56    C3   02B7   382          subl3   r6,r0,-4(r6)            ; set size of new block
                59    D5   02BC   383          tstl    r9
                0A    13   02BE   384          beql    20$                     ; end of list--set successor
          50    59    D1   02C0   385 10$:     cmpl    r9,r0                   ; this block inside new free block?
                05    1A   02C3   386          bgtru   20$                     ; no--set successor
          59    69    D0   02C5   387          movl    (r9),r9                 ; yes--try next block
                F6    12   02C8   388          bneq    10$                     ; loop through all possibilities
          66    59    D0   02CA   389 20$:     movl    r9,(r6)                 ; set successor of new block
                            02CD   390 ;
                            02CD   391 ;   Try to merge newly-freed storage with predecessor in LAS. Register 7
                            02CD   392 ;   has the address of the predecessor, r6 has the address of the new block.
                            02CD   393 ;
                58    D5   02CD   394          tstl    r8                      ; is there a predecessor?
                1A    13   02CF   395          beql    removl                  ; no--go remove other garbage
50   FC A8    58    C1   02D1   396          addl3   r8,-4(r8),r0            ; r0 <- addr of end of predecessor
          50    56    D1   02D6   397          cmpl    r6,r0                   ; is new block adjacent?
                10    12   02D9   398          bneq    removl                  ; no--go remove other garbage
  FC A8   FC A6   C0   02DB   399          addl2   -4(r6),-4(r8)           ; yes--lengthen predecessor
```

```
                    68   66   D0  02E0   400          movl    (r6),(r8)                       ; set link of predecessor
          00000014'EF42   58   D0  02E3   401          movl    r8,marks-4[r2]                  ; next spot to mark is (r8)
                          02EB   402 ;
                          02EB   403 ;   Remove from LAS all storage contained in blocks now in the pool.
                          02EB   404 ;   Register 7 has address of next page block to check, r8 has address
                          02EB   405 ;   of predecessor of next LAS block to check, r9 has address of next
                          02EB   406 ;   LAS block to check.
                          02EB   407 ;
                          02EB   408 removl:                                                   ; remove garbage from LAS
                    58   D4  02EB   409          clrl    r8                                  ; initialize r8 to nil predecessor
                59  53   D0  02ED   410          movl    r3,r9                               ; initialize r9
                    31   13  02F0   411          beql    exitr                               ; no LAS--exit
                57  55   D0  02F2   412          movl    r5,r7                               ; initialize r7
                    2C   13  02F5   413          beql    exitr                               ; empty pool-exit
             50 57  FC A7   C1  02F7   414 10$:    addl3   -4(r7),r7,r0                        ; r0 <- addr of end of this page block
                57  59   D1  02FC   415 20$:    cmpl    r9,r7                               ; LAS block starts after page block?
                    0A   1A  02FF   416          bgtru   40$                                 ; yes--look for garbage
                58  59   D0  0301   417          movl    r9,r8                               ; no--try next LAS block
                59  69   D0  0304   418 30$:    movl    (r9),r9                             ; move r9 to successor
                    1A   13  0307   419          beql    exitr                               ; end of LAS--exit
                    F1   11  0309   420          brb     20$                                 ; loop through all LAS blocks
                50  59   D1  030B   421 40$:    cmpl    r9,r0                               ; LAS block inside page block?
                    0E   1A  030E   422          bgtru   60$                                 ; no--try next page block
                            0310   423                                                      ; yes--remove garbage LAS block
                    58   D5  0310   424          tstl    r8                                  ; is there a predecessor?
                    05   13  0312   425          beql    50$                                 ; no--delete first LAS block
                68  69   D0  0314   426          movl    (r9),(r8)                           ; yes--delete successor of r8
                    EB   11  0317   427          brb     30$                                 ; go try next LAS block
                53  69   D0  0319   428 50$:    movl    (r9),r3                             ; delete first LAS block
                    E6   11  031C   429          brb     30$                                 ; go try next LAS block
                57  67   D0  031E   430 60$:    movl    (r7),r7                             ; try next page block
                    D4   12  0321   431          bneq    10$                                 ; loop through all page blocks
                            0323   432 ;
                            0323   433 ;   Store values that may have changed, return to caller
                            0323   434 ;
FFFFFFFC'EF42   53   D0  0323   435 exitr:  movl    r3,lasps-4[r2]                          ; store LASP
00000004'EF42   54   D0  032B   436          movl    r4,pblps-4[r2]                          ; store PBLP
0000000C'EF42   55   D0  0333   437          movl    r5,pools-4[r2]                          ; store pool pointer
          04 BC   D4  033B   438          clrl    @4(ap)                                  ; set released ptr to nil
    00000020'EF   97  033E   439          decb    nestl                                   ; reset nesting level
                04  0344   440          ret
                            0345   441 ;
                            0345   442 ;   ROUTINE TO IMPLEMENT THE PASCAL PROCEDURE DISPOSE
                            0345   443 ;
                    00FC    0345   444          .entry  pas$dispose,^m<r2,r3,r4,r5,r6,r7>
                            0347   445 ;
                            0347   446 ;   Move important values to registers
                            0347   447 ;
    00000020'EF   96  0347   448          incb    nestl                                   ; increment nesting level
  52 00000020'EF   9A  034D   449          movzbl  nestl,r2                                ; r2 <- incremented nesting level
53 FFFFFFFC'EF42   D0  0354   450          movl    lasps-4[r2],r3                          ; r3 <- LASP for this level
54 00000014'EF42   D0  035C   451          movl    marks-4[r2],r4                          ; r4 <- addr of marked blk
      55 04 BC   D0  0364   452          movl    @4(ap),r5                               ; r5 <- addr of block to be disposed
            03   12  0368   453          bneq    10$                                     ; continue if not nil
          0084   31  036A   454          brw     exitd                                   ; nil pointer--exit
  55 C0000000 8F   D3  036D   455 10$:    bitl    #^xc0000000,r5                          ; check for illegal value
            20   13  0374   456          beql    look                                    ; continue if not system or stack addr
```

```
                7E    10 AD    07    C3  0376   457          subl3   #7,16(fp),-(sp)         ; third FAO argument (PC of call)
                               00    DD  037B   458          pushl   #0                      ; second FAO argument (null)
                               55    DD  037D   459          pushl   r5                      ; first FAO argument (invalid pointer value)
                               03    DD  037F   460          pushl   #3                      ; number of FAO arguments preceding
             7E  00000000'8F   04    C1  0381   461          addl3   #4,#pas$_attdisinv,-(sp)        ; error message #8160
                   0000002C'EF   97    BA  0389   462          decb    nestl                   ; reset nesting level
                00000000'GF   05    FB  038F   463          calls   #5,G^lib$stop           ; signal error and stop execution
                                        0396   464 ;
                                        0396   465 ;   Look through LAS for the spot to insert the disposed block.  Register 5
                                        0396   466 ;   has address of disposed storage, r6 finds the predecessor of the new
                                        0396   467 ;   block, r7 finds the successor of the new block.
                                        0396   468 ;
                                        0396   469 look:                                  ; look for spot to insert block
                               56    D4  0396   470          clrl    r6                      ; initialize predecessor to nil
                          57   53    D0  0398   471          movl    r3,r7                   ; initialize successor to LASP
                               0D    13  039B   472          beql    insert                  ; LAS is empty--insert new block
                          57   55    D1  039D   473 10$:     cmpl    r5,r7                   ; new block goes before (r7)?
                               08    1F  03A0   474          blssu   insert                  ; yes--go insert it
                          56   57    D0  03A2   475          movl    r7,r6                   ; no--try next block
                          57   67    D0  03A5   476          movl    (r7),r7
                               F3    12  03A8   477          bneq    10$                     ; loop until found or end of LAS
                                        03AA   478 ;
                                        03AA   479 ;   Insert new block into LAS, merging with predecessor or successor if
                                        03AA   480 ;   possible.  Register 5 has addr of disposed block, r6 has addr of
                                        03AA   481 ;   predecessor in LAS, r7 has addr of successor in LAS.
                                        03AA   482 ;
                                        03AA   483 insert:                                ; insert new block into LAS
                50    55    FC  A5    C1  03AA   484          addl3   -4(r5),r5,r0            ; r0 <- addr of end of disposed block
                          57   50    D1  03AF   485          cmpl    r0,r7                   ; is new block adjacent to successor?
                               12    12  03B2   486          bneq    10$                     ; no--link to successor
                  FC  A5    FC  A7    C0  03B4   487          addl2   -4(r7),-4(r5)           ; yes--merge with successor
                          65   67    D0  03B9   488          movl    (r7),(r5)               ; set link of new block
                          54   57    D1  03BC   489          cmpl    r7,r4                   ; was successor the marked block?
                               08    12  03BF   490          bneq    20$                     ; no--try to merge with predecessor
                          54   55    D0  03C1   491          movl    r5,r4                   ; yes--new block is now marked
                               03    11  03C4   492          brb     20$
                          65   57    D0  03C6   493 10$:     movl    r7,(r5)                 ; link new block to successor
                               56    D5  03C9   494 20$:     tstl    r6                      ; is there a predecessor?
                               05    12  03CB   495          bneq    30$                     ; yes--go try to merge
                          53   55    D0  03CD   496          movl    r5,r3                   ; no--add new block to front of LAS
                               1F    11  03D0   497          brb     exitd                   ; exit
              50    56    FC  A6    C1  03D2   498 30$:     addl3   -4(r6),r6,r0            ; r0 <- addr of end of predecessor
                          50   55    D1  03D7   499          cmpl    r5,r0                   ; new block is adjacent to predecessor?
                               12    12  03DA   500          bneq    40$                     ; no--go link predecessor to new block
                FC  A6    FC  A5    C0  03DC   501          addl2   -4(r5),-4(r6)           ; yes--lengthen predecessor
                          66   65    D0  03E1   502          movl    (r5),(r6)               ; set link of predecessor
                          54   55    D1  03E4   503          cmpl    r5,r4                   ; was new block the marked block?
                               08    12  03E7   504          bneq    exitd                   ; no--exit
                          54   56    D0  03E9   505          movl    r6,r4                   ; yes--predecessor is now marked
                               03    11  03EC   506          brb     exitd                   ; exit
                          66   55    D0  03EE   507 40$:     movl    r5,(r6)                 ; link predecessor to new block
                                        03F1   508 ;
                                        03F1   509 ;   Store values that may have changed, return to caller
                                        03F1   510 ;
FFFFFFFC'EF42                53    D0  03F1   511 exitd:   movl    r3,lasps-4[r2]          ; store LASP
00000014'EF42                54    D0  03F9   512          movl    r4,marks-4[r2]          ; store addr of marked block
                        04    BC    D4  0401   513          clrl    @4(ap)                  ; set disposed ptr to nil
```

```
                00000020'EF    97  0404    514          decb    nestl                   ; reset nesting level
                               04  040A    515          ret
                                   040B    516 ;
                                   040B    517 ;   DEBUGGING ROUTINE TO EXAMINE STATE OF HEAP STORAGE
                                   040B    518 ;
                             0000  040B    519          .entry pas$snap,^m<>
                                   040D    520 ;
                                   040D    521 ;   Return current values of level 1 LASP, PBLP, POOL, MARK
                                   040D    522 ;
      04 BC     00000000'EF    D0  040D    523          movl    lasps,@4(ap)            ; return lasp
      08 BC     00000008'EF    D0  0415    524          movl    pblps,@8(ap)            ; return pblp
      0C BC     00000010'EF    D0  041D    525          movl    pools,@12(ap)           ; return pool
      10 BC     00000018'EF    D0  0425    526          movl    marks,@16(ap)           ; return mark
                               04  042D    527          ret
                                   042E    528          .end
```

```
ALLOC                   0000012C  R      03
EXITD                   000003F1  R      03
EXITMN                  00000182  R      03
EXITR                   00000323  R      03
EXPAND                  0000006A  R      03
INPOOL                  00000043  R      03
INSERT                  000003AA  R      03
LASPS                   00000000  R      02
LIB$GET_VM              ********    X    03
LIB$STOP                ********    X    03
LINKLA                  000000D5  R      03
LINKPB                  000000AD  R      03
LOCEND                  000001F3  R      03
LOOK                    00000396  R      03
MARKS                   0000001C  R      02
NESTL                   00000020  R      02
PART                    0000028A  R      03
PAS$DISPOSE             00000345  RG     03
PAS$MARK                000001A5  RG     03
PAS$NEW                 00000000  RG     03
PAS$RELEASE             00000208  RG     03
PAS$SNAP                0000040B  RG     03
PAS$_ATTDISINV          ********    X    00
PAS$_PROEXCHEA          ********    X    00
PBLPS                   00000008  R      02
POOL                    0000025A  R      03
POOLS                   00000010  R      02
REMOVL                  000002E8  R      03
SETMK                   00000121  R      03
SIZE                    000002B2  R      03
SPLIT                   00000155  R      03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PAS$DATA	00000021 (33.)	02 (2.)	PIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
_PAS$CODE	0000042E (1070.)	03 (3.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.09	00:00:00.33
Command processing	106	00:00:00.48	00:00:02.64
Pass 1	136	00:00:02.17	00:00:05.42
Symbol table sort	0	00:00:00.06	00:00:00.06
Pass 2	112	00:00:01.15	00:00:02.69
Symbol table output	3	00:00:00.04	00:00:00.04
Psect synopsis output	3	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00

Assembler run totals 394 00:00:04.02 00:00:11.22

The working set limit was 1200 pages.
13384 bytes (27 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 63 non-local and 46 local symbols.
528 source lines were read in Pass 1, producing 28 object records in Pass 2.
8 pages of virtual memory were used to define 7 macros.

```
+---------------------------+
! Macro library statistics !
+---------------------------+
```

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	4

91 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

```
MACRO/DISABLF=TRACE/LIS=LIS$:PASRT2/OBJ=OBJ$:PASRT2 MSRC$:PASRT2/UPDATE=(ENH$:PASRT2)
```

PASRT1
LIS

PASMACROS
REQ

PASFCB
SDL

PASPROLOG
REQ

PASOPEDEF
REQ

PASABSL
LIS

PASCARD2
LIS

PASRTL

PASBUGCOD
REQ

PASCLOSE2
LIS

PASRT2
LIS

PASRT3
LIS

PASRTL
MAP

PASIO4
LIS

PASEXTERN
REQ

PASKDB
REQ

PASPFD
REQ

PASCONVER
LIS

PASLIB
REQ

PASPFV
REQ

PASBIN
LIS

PASCLOCK2
LIS

PASRT4
LIS