```
MMM           MMM     000000000     NNN           NNN  TTTTTTTTTTTTTTT     000000000     RRRRRRRRRRR
MMM           MMM     000000000     NNN           NNN  TTTTTTTTTTTTTTT     000000000     RRRRRRRRRRRR
MMM           MMM     000000000     NNN           NNN  TTTTTTTTTTTTTTT     000000000     RRRRRRRRRRRR
MMMMMM     MMMMMM  000           000  NNN           NNN        TTT        000           000  RRR         RRR
MMMMMM     MMMMMM  000           000  NNN           NNN        TTT        000           000  RRR         RRR
MMMMMM     MMMMMM  000           000  NNN           NNN        TTT        000           000  RRR         RRR
MMM  MMM   MMM  000           000  NNNNNN        NNN        TTT        000           000  RRR         RRR
MMM  MMM   MMM  000           000  NNNNNN        NNN        TTT        000           000  RRR         RRR
MMM  MMM   MMM  000           000  NNNNNN        NNN        TTT        000           000  RRR         RRR
MMM           MMM  000           000  NNN  NNN  NNN        TTT        000           000  RRRRRRRRRRR
MMM           MMM  000           000  NNN  NNN  NNN        TTT        000           000  RRRRRRRRRRR
MMM           MMM  000           000  NNN  NNN  NNN        TTT        000           000  RRRRRRRRRRR
MMM           MMM  000           000  NNN       NNNNN        TTT        000           000  RRR   RRR
MMM           MMM  000           000  NNN       NNNNN        TTT        000           000  RRR   RRR
MMM           MMM  000           000  NNN       NNNNN        TTT        000           000  RRR   RRR
MMM           MMM  000           000  NNN           NNN        TTT        000           000  RRR      RRR
MMM           MMM  000           000  NNN           NNN        TTT        000           000  RRR      RRR
MMM           MMM  000           000  NNN           NNN        TTT        000           000  RRR      RRR
MMM           MMM     000000000     NNN           NNN        TTT           000000000     RRR         RRR
MMM           MMM     000000000     NNN           NNN        TTT           000000000     RRR         RRR
MMM           MMM     000000000     NNN           NNN        TTT           000000000     RRR         RRR
```

FILEID**GETBUFF

```
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  BBBBBBBB    UU      UU  FFFFFFFFFF  FFFFFFFFFF
  GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  BBBBBBBB    UU      UU  FFFFFFFFFF  FFFFFFFFFF
GG          EE              TT      BB      BB  UU      UU  FF          FF
GG          EE              TT      BB      BB  UU      UU  FF          FF
GG          EE              TT      BB      BB  UU      UU  FF          FF
GG          EE              TT      BB      BB  UU      UU  FF          FF
GG          EEEEEEEE        TT      BBBBBBBB    UU      UU  FFFFFFFF    FFFFFFFF
GG          EEEEEEEE        TT      BBBBBBBB    UU      UU  FFFFFFFF    FFFFFFFF
GG  GGGGGG  EE              TT      BB      BB  UU      UU  FF          FF
GG  GGGGGG  EE              TT      BB      BB  UU      UU  FF          FF
GG      GG  EE              TT      BB      BB  UU      UU  FF          FF              ....
GG      GG  EE              TT      BB      BB  UU      UU  FF          FF              ....
  GGGGGG    EEEEEEEEEE      TT      BBBBBBBB    UUUUUUUUUU  FF          FF              ....
  GGGGGG    EEEEEEEEEE      TT      BBBBBBBB    UUUUUUUUUU  FF          FF              ....

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
```

Table of contents

(2) 55 DECLARATIONS
(3) 70 GET_BUFFERS - Obtain Collection & Stat Buffers

```
0000     1          .TITLE  GETBUFF - Obtain Collection & Stat Buffers
0000     2          .IDENT  'V04-000'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FUR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26
0000    27 ;++
0000    28 ; FACILITY:  VAX/VMS MONITOR Utility
0000    29 ;
0000    30 ; ABSTRACT:
0000    31 ;       Called at request initialization time to obtain Collection
0000    32 ;       and Stat buffers
0000    33 ;
0000    34 ; ENVIRONMENT: Unprivileged user mode.
0000    35 ;
0000    36 ; AUTHOR: Henry M. Levy          , CREATION DATE: 28-March-1977
0000    37 ;         Thomas L. Cafarella
0000    38 ;
0000    39 ; MODIFIED BY:
0000    40 ;
0000    41 ;       V03-003 TLC1090         Thomas L. Cafarella     02-Aug-1984     15:00
0000    42 ;               Correct ACCVIOs in SYSTEM and PROCESSES classes.
0000    43 ;
0000    44 ;       V03-002 TLC1066         Thomas L. Cafarella     01-Apr-1984     11:00
0000    45 ;               Add SYSTEM class.
0000    46 ;
0000    47 ;       V03-001 PRS1008         Paul R. Senn            17-FEB-1984     14:00
0000    48 ;               Split out GET_BUFFERS and associated subroutines from
0000    49 ;               MONITOR.MAR into separate module.
0000    50 ;
0000    51 ;
0000    52 ;
0000    53 ;--
```

```
                        55          .SBTTL  DECLARATIONS
00000000                56          .PSECT  MONDATA,QUAD,NOEXE
                        57 ;
                        58 ; INCLUDE FILES:
                        59 ;
                        60
                        61          $CDBDEF                         ; Define Class Descriptor Block
                        62          $CDXDEF                         ; Define CDB Extension
                        63          $MRBDEF                         ; Define Monitor Request Block
                        64          $MBPDEF                         ; Define Monitor Buffer Pointe s
                        65          $MONDEF                         ; Monitor Recording File Definitions
                        66          $SCBDEF                         ; Define STATS Control Block
                        67
                        68 ;
```

```
                        .SBTTL  GET_BUFFERS - Obtain Collection & Stat Buffers
00000000                .PSECT  $$MONCODE,NOWRT,EXE
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       Standard classes:
                ;
                ;       This routine obtains a number of collection and statistical buffers
                ;       using the LIB$GET_VM facility. For heterogeneous classes, the number
                ;       of buffers obtained is determined by the 3 symbols COLL_BUFS,
                ;       REG_BUFS and PC_BUFS. The buffers are contiguous, forming a block
                ;       which includes at its beginning, a set of longword pointers to the
                ;       buffers which follow immediately thereafter. The buffer block always
                ;       includes COLL_BUFS collection buffers and REG_BUFS regular stats
                ;       buffers. If percent data is being maintained, PC_BUFS percent stats
                ;       buffers are also included. The buffer block is pointed to by
                ;       CDB$A_BUFFERS.
                ;
                ;       For homogeneous classes, the entire buffer block above is repeated
                ;       once for each item being displayed. A set of contiguous pointers
                ;       to the buffer blocks is stored immediately preceding the blocks,
                ;       and is pointed to by CDB$A_BUFFERS. In addition, following the
                ;       buffer blocks are the SCB (STATS Control Block) and Element ID
                ;       Table.
                ;
                ;       Non-standard class (PROCESSES):
                ;
                ;       For the regular PROCESSES display, only one collection
                ;       buffer, and the display buffer will be obtained.
                ;
                ;       For the TOP PROCESSES displays, one collection buffer
                ;       and the 5 arrays (DATA, DIFF, ORDER, PID, ADDR) will
                ;       be obtained. Space for the FAO control string will also
                ;       be obtained, but will not be part of the buffer block.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       JSB GET_BUFFERS
                ;
                ; INPUTS:
                ;
                ;       None
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       COLL_BUFS global symbol -- number of collection buffers to obtain
                ;       REG_BUFS  global symbol -- number of regular stats buffers to obtain
                ;       PC_BUFS   global symbol -- number of percent stats buffers to obtain
                ;       MAXELTS   global symbol -- maximum number of homogeneous elements
                ;       SPTR -- pointer to SYI (System Information Area)
                ;
                ;       R6 -- pointer to CDB
                ;       R7 -- pointer to MRB
                ;       R11 -- pointer to MCA
                ;
                ; OUTPUTS:
```

```
                          0000   127 ;
                          0000   128 ;       None
                          0000   129 ;
                          0000   130 ; IMPLICIT OUTPUTS:
                          0000   131 ;
                          0000   132 ;       CDB$A_BUFFERS and CDB$L_BUFFERS fields of CDB will contain pointer
                          0000   133 ;       and length, respectively, of entire chunk of memory obtained.
                          0000   134 ;
                          0000   135 ;       SUM, MIN and MAX buffers (and PCSUM buffer, if percent requested)
                          0000   136 ;       are cleared to 0.
                          0000   137 ;
                          0000   138 ;       For TOP PROCESSES class, the DATA array will be cleared to 0.
                          0000   139 ;
                          0000   140 ; ROUTINE VALUE:
                          0000   141 ;
                          0000   142 ;       R0 = NORMAL, or error status from LIB$GET_VM, if any.
                          0000   143 ;
                          0000   144 ; SIDE EFFECTS:
                          0000   145 ;
                          0000   146 ;       Registers R0,R1,R2,R3,R4,R5,R8,R9,R10 altered.
                          0000   147 ;
                          0000   148 ;--
                          0000   149 GET_BUFFERS::
                          0000   150
                0880 8F   BB 0000   151         PUSHR   #^M<R7,R11>                     ; Save regs
                          0004   152
                          0004   153 ;
                          0004   154 ; Get buffers for non-standard class (PROCESSES)
                          0004   155 ;
                          0004   156
          03 4B A6   04   E1 0004   157         BBC     #CDB$V_STD,CDB$L_FLAGS(R6),5$   ; Continue if a non-standard class
                  00BE    31 0009   158         BRW     90$                             ; Otherwise, go process standard
                          000C   159 5$:
       52  00000000'EF   D0 000C   160         MOVL    SPTR,R2                         ; Get pointer to SYI
           52    0B A2   3C 0013   161         MOVZWL  MNR_SYI$W_MAXPRCCT(R2),R2       ; Get max process count
           59    20 A6   3C 0017   162         MOVZWL  CDB$W_BLKLEN(R6),R9             ; Get size of one data block
              59    52   C4 001B   163         MULL2   R2,R9                           ; Compute bytes for data blocks
              59    15   C0 001E   164         ADDL2   #<MNR_PRO$K_PSIZE+MNR_CLS$K_HSIZE>,R9 ; Add prefix and class header
                          0021   165                                                 ; ... to get collection buffer size
                    54    D4 0021   166         CLRL    R4                              ; Clear FAO stack (display buffer) b
           43 A7    05   B3 0023   167         BITW    #<MRB$M_DISPLAY+MRB$M_SUMMARY>,MRB$W_FLAGS(R7) ; Displaying or summa
                    13   13 0027   168         BEQL    10$                             ; No -- just need collection buffers
        0E 43 A7    0E   E1 0029   169         BBC     #MRB$V_PROC_REQ,MRB$W_FLAGS(R7),10$ ; Br if PROCESSES not requested
           42 A6    00   91 002E   170         CMPB    #REG_PROC,CDB$B_ST(R6)          ; Regular PROCESSES display ?
                    32   12 0032   171         BNEQ    30$                             ; No -- go get TOP arrays
                          0034   172
                          0034   173 ;
                          0034   174 ; Regular PROCESSES display -- get display buffer (FAO stack)
                          0034   175 ;
                          0034   176
54  52    00000040 8F   C5 0034   177         MULL3   #MNR_PRO$K_FSIZE,R2,R4          ; Calc FAO stack (display buffer) si
                          003C   178 10$:
              54    59   C0 003C   179         ADDL2   R9,R4                           ; Add size of both collection buffer
              54    59   C0 003F   180         ADDL2   R9,R4                           ; ... to FAO stack size
        2A A6    54    0C   C1 0042   181         ADDL3   #12,R4,CDB$L_BUFFERS(R6)        ; ... add enough for 3 pointers
                    02A3   30 0047   182         BSBW    GET_MEM                         ; Obtain the virtual memory
                 03 50   E8 004A   183         BLBS    R0,20$                          ; Continue if obtained OK
```

```
                     0162   31  004D   184          BRW     GB_RSB                          ; Else, go exit with error
                            0050   185 20$:
           55   2E A6   D0  0050   186          MOVL    CDB$A_BUFFERS(R6),R5            ; Now prepare to load 3 pointers
           65   0C A5   DE  0054   187          MOVAL   12(R5),(R5)                     ; Point first ptr to collection buff
     04 A5    59   65   C1  0058   188          ADDL3   (R5),R9,4(R5)                   ; Point 2nd ptr to 2nd coll buffer
  08 A5   59   04 A5   C1  005D   189          ADDL3   4(R5),R9,8(R5)                  ; Point 3rd ptr to FAO stack
                     0145   31  0063   190          BRW     GB_NRSB                         ; ... and take normal return
                            0066   191
                            0066   192 ;
                            0066   193 ; TOP PROCESSES display -- get 5 arrays consisting of 'MAX PROCESS COUNT"
                            0066   194 ; longwords each.
                            0066   195 ;
                            0066   196
                            0066   197 30$:
        58   52   04   C5  0066   198          MULL3   #4,R2,R8                        ; Compute size of one array
        54   58   05   C5  006A   199          MULL3   #5,R8,R4                        ; Need 5 arrays
             54   59   C0  006E   200          ADDL2   R9,R4                           ; Add in size of 2 coll buffs to
             54   59   C0  0071   201          ADDL2   R9,R4                           ; ... get total bytes required
     2A A6   54   1C   C1  0074   202          ADDL3   #<4*7>,R4,CDB$L_BUFFERS(R6)     ; ... add enough for 7 pointers
                0271   30  0079   203          BSBW    GET_MEM                         ; Obtain the virtual memory
              03 50   E8  007C   204          BLBS    R0,40$                          ; Continue if obtained OK
                0130   31  007F   205          BRW     GB_PSB                          ; Else, go exit with error
                            0082   206 40$:
           55   2E A6   D0  0082   207          MOVL    CDB$A_BUFFERS(R6),R5            ; Now prepare to load the 7 pointers
           85   1C A5   DE  0086   208          MOVAL   <4*7>(R5),(R5)+                 ; Point 1st ptr to 1st coll buffer
     85    FC A5    59   C1  008A   209          ADDL3   R9,-4(R5),(R5)+                 ; Point 2nd ptr to 2nd coll buffer
           59    FC A5   C0  008F   210          ADDL2   -4(R5),R9                       ; Compute addr of first of 5 arrays
              51   05   D0  0093   211          MOVL    #5,R1                           ; Loop counter
```

```
                        0096   213 50$:
              85    59  D0  0096   214         MOVL    R9,(R5)+                        ; S ore array pointer and advance R5
              59    58  C0  0099   215         ADDL2   R8,R9                           ; Compute addr of next array
                F7  51  F5  009C   216         SOBGTR  R1,50$                          ; Loop storing 5 pointers
                        009F   217
           58   52   04  C5  009F   218         MULL3   #4,R2,R8                        ; Compute size of DATA array
             59   2E A6  D0  00A3   219         MOVL    CDB$A_BUFFERS(R6),R9            ; ... and get its address
             59   08 A9  D0  00A7   220         MOVL    MBP$A_DATA(R9),R9               ; ...
                  024D   30  00AB   221         BSBW    CLEAR_DATA                      ; Clear DATA array
                         00AE   222
                         00AE   223 ;
                         00AE   224 ; Obtain an FAO control string for PROCESSES/TOP. This buffer
                         00AE   225 ; will not be part of the buffer block, but, instead, will be
                         00AE   226 ; described by the CDB$A_FAOCTR and CDB$L_FAOCTR fields of the
                         00AE   227 ; CDB. The FAO control string for STANDARD classes is obtained
                         00AE   228 ; in the TEMPLATE (BLISS-32) routine.
                         00AE   229 ;
                         00AE   230
        66   00000000'8F  D0  00AE   231         MOVL    #FAOCTR_SIZE,CDB$L_FAOCTR(R6)   ; Store size of FAO control string
                  04 A6  DF  00B5   232         PUSHAL  CDB$A_FAOCTR(R6)                ; Push addr of longword to hold
                         00B8   233                                                 ; ... FAO control string pointer
                     66  DF  00B8   234         PUSHAL  CDB$L_FAOCTR(R6)                ; Now push addr of # of bytes needed
        00000000'GF   02  FB  00BA   235         CALLS   #2,G^LIB$GET_VM                 ; Allocate space
                  03 50  E9  00C1   236         BLBC    R0,80$                          ; Branch if failed
                   00E4  31  00C4   237         BRW     GB_NRSB                         ; Else take normal return
                         00C7   238 80$:
                   00E8  31  00C7   239         BRW     GB_RSB                          ; Take common error exit
                         00CA   240
                         00CA   241 ;
                         00CA   242 ; Get buffers for standard class
                         00CA   243 ;
                         00CA   244
                         00CA   245 90$:
                         00CA   246
                         00CA   247 ;
                         00CA   248 ; Get DATA arrays for the special SYSTEM class.
                         00CA   249 ;
                         00CA   250
     15 4B A6    08  E1  00CA   251         BBC     #CDB$V_SYSCLS,CDB$L_FLAGS(R6),93$ ; Br if not SYSTEM class
        43 A7    05  B3  00CF   252         BITW    #<MRB$M_DISPLAY+MRB$M_SUMMARY>,MRB$W_FLAGS(R7) ; Displaying or summa
                 0F  13  00D3   253         BEQL    93$                             ; Br if no -- don't need DATA arrays
           00  42 A6  91  00D5   254         CMPB    CDB$B_ST(R6),#ALL_STAT          ; ALL stat requested ?
                 09  13  00D9   255         BEQL    93$                             ; Br if yes -- don't need DATA arrays
                0247 30  00DB   256         BSBW    GET_SYS_DATA_ARRAYS             ; Do what it says
              03 50  E8  00DE   257         BLBS    R0,93$                          ; Br if successful
               00CE  31  00E1   258         BRW     GB_RSB                          ; Else take error exit
                         00E4   259 ;
                         00E4   260 ; Compute number of bytes to allocate for heterogeneous class buffer block.
                         00E4   261 ;
                         00E4   262
                         00E4   263 93$:
     03 4B A6    05  E1  00E4   264         BBC     #CDB$V_HOMOG,CDB$L_FLAGS(R6),95$ ; Br if hetero class
                00CB 31  00E9   265         BRW     HOM_BUFFS                       ; Else go do homogeneous
                         00EC   266 95$:
54     00000000'8F  D0  00EC   267         MOVL    #<COLL_BUFS+REG_BUFS>,R4        ; Number of buffers to obtain
     07 45 A6    00  E1  00F3   268         BBC     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),100$ ; If percent not requested, ski
54     00000000'8F  C0  00F8   269         ADDL2   #PC_BUFS,R4                     ; Include PC_BUFS in count of buffer
```

```
                                  00FF  100$:
        59   01   14 A6     C1    00FF         ADDL3   CDB$L_ICOUNT(R6),#1,R9           ; Compute number of data items per b
                                  0104                                                  ; ... + 1 for buffer pointer
                  59   04   C4    0104         MULL2   #4,R9                            ; ... times 4 since items are longwo
                  59   54   C4    0107         MULL2   R4,R9                            ; ... times # of buffers
  2A A6   59  00000000'8F   C1    010A         ADDL3   #COLL_BUFS*MNR_CLS$K_HSIZE,R9,CDB$L_BUFFERS(R6) ; Collection buffers
                    01D7    30    0113         BSBW    GET_MEM                          ; Obtain the virtual memory
                   03 50    E8    0116         BLBS    R0,T05$                          ; Br if status OK
                    0096    31    0119         BRW     GB_RSB                           ; Else exit with error if failed
```

```
                        011C   280 ;
                        011C   281 ; Store values for the buffer pointers at the beginning of the buffer block
                        011C   282 ; just allocated.
                        011C   283 ;
                        011C   284 ; Register Usage:
                        011C   285 ;
                        011C   286 ;       R2  = size of most recent buffer
                        011C   287 ;       R3  = address of most recent buffer
                        011C   288 ;       R4  = number of buffers; later used as loop control
                        011C   289 ;       R5  = pointer into block of pointers
                        011C   290 ;       R6  = CDB pointer
                        011C   291 ;       R10  = buffer block pointer
                        011C   292 ;
                        011C   293
                        011C   294 105$:
     55    2E A6   D0   011C   295         MOVL    CDB$A_BUFFERS(R6),R5    ; Store address of 1st pointer
        5A    55   D0   0120   296         MOVL    R5,R10                  ; Remember buffer block addr for later MOVC5
        54    04   C4   0123   297         MULL2   #4,R4                   ; Compute address of ...
     53    55    54   C1   0126   298         ADDL3   R4,R5,R3                ; ... 1st buffer
        85    53   D0   012A   299         MOVL    R3,(R5)+                ; Move it into 1st pointer
52   14 A6    04   C5   012D   300         MULL3   #4,CDB$L_ICOUNT(R6),R2  ; Calculate size of next buffer
        52    0D   C0   0132   301         ADDL2   #MNR_CLS$K_HSIZE,R2     ; Add in the header size
     54   00'8F   9A   0135   302         MOVZBL  #COLL_BUFS,R4           ; Loop COLL_BUFS times
                        0139   303 110$:
        53    52   C0   0139   304         ADDL2   R2,R3                   ; Calculate address of next buffer
        85    53   D0   013C   305         MOVL    R3,(R5)+                ; ... and store it into next pointer
          F7  54   F5   013F   306         SOBGTR  R4,110$                 ; ....
                        0142   307
        52    0D   C2   0142   308         SUBL2   #MNR_CLS$K_HSIZE,R2     ; Next group don't have headers
        58    52   D0   0145   309         MOVL    R2,R8                   ; Save size of a buffer for later MOVC5
     54   FF'8F   9A   0148   310         MOVZBL  #REG_BUFS-1,R4          ; Loop REG_BUFS-1 times
                        014C   311 120$:
        53    52   C0   014C   312         ADDL2   R2,R3                   ; Calculate address of next buffer
        85    53   D0   014F   313         MOVL    R3,(R5)+                ; ... and store it into next pointer
          F7  54   F5   0152   314         SOBGTR  R4,120$                 ; ....
  2F 45 A6    00   E1   0155   315         BBC     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),150$ ; If percent not requested, ski
     54   00'8F   9A   015A   316         MOVZBL  #PC_BUFS,R4             ; Loop PC_BUFS times
                        015E   317 130$:
        53    52   C0   015E   318         ADDL2   R2,R3                   ; Calculate address of next buffer
        85    53   D0   0161   319         MOVL    R3,(R5)+                ; ... and store it into next pointer
          F7  54   F5   0164   320         SOBGTR  R4,130$                 ; ....
24 BA   58    00   FE AF   00   2C   0167   321         MOVC5   #0,,#0,R8,@MBP$A_PCSUM(R10) ; Zero out PCSUM buffer
20 BA   58    00   FE AF   00   2C   016F   322         MOVC5   #0,,#0,R8,@MBP$A_PCMAX(R10) ; Zero out PCMAX buffer
                        0177   323
                        0177   324 ;
                        0177   325 ; Store large positive number (suitable for integer or floating)
                        0177   326 ; into each longword of PCMIN.
                        0177   327 ;
                        0177   328
        51   1C AA   D0   0177   329         MOVL    MBP$A_PCMIN(R10),R1     ; Get addr of PCMIN buffer
        50   14 A6   D0   017B   330         MOVL    CDB$L_ICOUNT(R6),R0     ; ... and number of longwords
                        017F   331 140$:
     81   00000000'8F   D0   017F   332         MOVL    #LARGE_NO,(R1)+         ; Move in a large value
          F6 50   F5   0186   333         SOBGTR  R0,140$                 ; Loop back for next one
                        0189   334
                        0189   335 150$:
14 BA   58    00   FE AF   00   2C   0189   336         MOVC5   #0,,#0,R8,@MBP$A_SUM(R10) ; Zero out SUM buffer
```

```
10 BA   58   00   FE AF   00   2C  0191          MOVC5   #0,..,#0,R8,@MBP$A_MAX(R10) ; Zero out MAX buffer
                                   0199
                                   0199   ;
                                   0199   ; Store large positive number (suitable for integer or floating)
                                   0199   ; into each longword of MIN.
                                   0199   ;
                                   0199
              51    0C AA      D0  0199          MOVL    MBP$A_MIN(R10),R1        ; Get addr of MIN buffer
              50    14 A6      D0  019D          MOVL    CDB$L_ICOUNT(R6),R0      ; ... and number of longwords
                                   01A1   160$:
         81   00000000'8F      D0  01A1          MOVL    #LARGE_NO,(R1)+          ; Move in a large value
                    F6 50      F5  01A8          SOBGTR  R0,160$                  ; Loop back for next one
                                   01AB
                                   01AB
                                   01AB   GB_NRSB:                                ; Normal return point
         50   00000000'EF      D0  01AB          MOVL    NORMAL,R0                ; Indicate successful status
                                   01B2
                                   01B2   GB_RSB:                                 ; Error return point
                  0880 8F      BA  01B2          POPR    #^M<R7,R11>              ; Restore regs
                               05  01B6          RSB                              ; Return
```

```
                                01B7    358 HOM_BUFFS:
                                01B7    359
                                01B7    360 ;
                                01B7    361 ; Compute number of bytes to allocate for homog class buffer block
                                01B7    362 ;
                                01B7    363:
        54    20 A6    3C      01B7    364          MOVZWL  CDB$W_BLKLEN(R6),R4                     ; .... Compute
   54   00000000'8F    C4      01BB    365          MULL2   #MAXELTS,R4                             ; .... collection
        54    15       C0      01C2    366          ADDL2   #<MNR_CLS$K_HSIZE+MNR_HOM$K_PSIZE>,R4   ; .... buffers
50 54   00000000'8F    C5      01C5    367          MULL3   #COLL_BUFS,R4,R0                        ; .... size
                                01CD    368
   51   00000004'8F    D0      01CD    369          MOVL    #<<<<PC_BUFS+REG_BUFS> * <MAXELTS+1>> + COLL_BUFS+1> * 4>,R1
                                01D4    370
   07 45 A6    00      E0      01D4    371          BBS     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),10$     ; .... Add in
   51   00000000'8F    C2      01D9    372          SUBL2   #<4 * PC_BUFS * <MAXELTS + 1>>,R1       ; .... MBP
                                01E0    373 10$:                                                    ; .... and
        53    32 A6    D0      01E0    374          MOVL    CDB$A_CDX(R6),R3                        ; ....
        58    06 A3    9A      01E4    375          MOVZBL  CDX$B_IDISCT(R3),R8                     ; .... transformation
           51    58    C4      01E8    376          MULL2   R8,R1                                   ; .... buffers
           50    51    C0      01EB    377          ADDL2   R1,R0                                   ; .... size
                                01EE    378
        51    09 A3    9A      01EE    379          MOVZBL  CDX$B_ELIDLEN(R3),R1                    ; .... Add in Element
           51    03    C0      01F2    380          ADDL2   #SCB$K_SIZE,R1                          ; .... ID Table and
   51   00000000'8F    C4      01F5    381          MULL2   #MAXELTS,R1                             ; .... STATS Control
 2A A6    50    51    C1      01FC    382          ADDL3   R1,R0,CDB$L_BUFFERS(R6)                 ; .... Block size
                                0201    383
              00E9      30      0201    384          BSBW    GET_MEM                 ; Obtain the virtual memory
             AB 50      E9      0204    385          BLBC    R0,GB_RSB               ; Exit with error if failed
                                0207    386
                                0207    387 ;
                                0207    388 ; Now store values for the buffer pointers at the beginning of
                                0207    389 ; the buffer block just allocated, and in each of the MBPs (Monitor
                                0207    390 ; Buffer Pointer blocks).
                                0207    391 ;
                                0207    392
           5B    2E A6    D0      0207    393          MOVL    CDB$A_BUFFERS(R6),R11   ; Store addr of 1st ptr
        51    58    04    C5      020B    394          MULL3   #4,R8,R1                ; Compute addr of ...
  00000000'EF    5B    51    C1      020F    395          ADDL3   R1,R11,CB_ADDRS         ; ... 1st coll buff
00000004'EF   00000000'EF    54    C1      0217    396          ADDL3   R4,CB_ADDRS,CB_ADDRS+4  ; ... 2nd coll buff
      57      00000004'EF    54    C1      0223    397          ADDL3   R4,CB_ADDRS+4,R7        ; ... and 1st MBP
                                022B    398
           59   00000000'8F    D0      022B    399          MOVL    #REG_BUFS,R9                            ; Get number of xform
              07 45 A6    00    E1      0232    400          BBC     #CDB$V_PERCENT,CDB$W_QFLAGS(R6),20$     ; buffers for use in
           59   00000000'8F    C0      0237    401          ADDL2   #PC_BUFS,R9                             ; the MBP_FILL routine
                                023E    402
                                023E    403 20$:
              8B    57    D0      023E    404          MOVL    R7,(R11)+               ; Store away MBP ptr
                    17    10      0241    405          BSBB    MBP_FILL                ; Fill the current MBP block
                F8 58     F5      0243    406          SOBGTR  R8,20$                  ; Loop back to fill next MBP
                                0246    407
                                0246    408 ;
                                0246    409 ; Now store addresses of the Element ID Table and the SCB Table.
                                0246    410 ;
                                0246    411
           50    32 A6    D0      0246    412          MOVL    CDB$A_CDX(R6),R0        ; Get addr of CDB eXtension
         10 A0    57    D0      024A    413          MOVL    R7,CDX$A_SCBTABLE(R0)   ; Store SCB Table address
        00000000'8F    C1      024E    414          ADDL3   #<SCB$K_SIZE*MAXELTS>,- ; ... and Element ID Table address
```

```
                OC A0    57        0254   415                 R7,CDX$A_ELIDTABLE(R0)
                                   0257   416
                      FF51   31    0257   417       BRW       GB_NRSB                 ; All done -- go return
```

```
                                                H  1

                     025A   419 MBP_FILL:
                     025A   420
                     025A   421 ;
                     025A   422 ; Fill an MBP (Monitor Buffer Pointers block) with the addresses
                     025A   423 ; of the transformation buffers immediately following it. There
                     025A   424 ; is one MBP for each item being displayed.
                     025A   425 ;
                     025A   426 ;
                     025A   427 ; Input Registers:
                     025A   428 ;
                     025A   429 ;       R7 = current MBP addr
                     025A   430 ;       R9 = number of transformation buffers
                     025A   431 ;
                     025A   432
        5A   57   D0 025A   433         MOVL    R7,R10                  ; Save MBP address for MOVC5 below
87 00000000'EF   7D 025D   434         MOVQ    CB_ADDRS,(R7)+          ; Store coll buff ptrs in MBP
                     0264   435
    55   59   04   C5 0264   436         MULL3   #4,R9,R5                ; Compute address of buffer ...
         55   57   C0 0268   437         ADDL2   R7,R5                   ; ... portion of MBP
                     026B   438
                     026B   439 ;
                     026B   440 ; Move in xform buffer ptrs for the "regular" buffers
                     026B   441 ;
                     026B   442
      50   00'8F   9A 026B   443         MOVZBL  #REG_BUFS,R0            ; Loop REG_BUFS times
                     026F   444 10$:
         87   55   D0 026F   445         MOVL    R5,(R7)+                ; Store address of buffer into next ptr
55   00000000'8F   C0 0272   446         ADDL2   #<4*MAXELTS>,R5         ; Calculate address of next buffer
            F3 50   F5 0279   447         SOBGTR  R0,10$                  ; ....
                     027C   448
                     027C   449 ;
                     027C   450 ; Move in xform buffer ptrs for the percent buffers if needed
                     027C   451 ;
                     027C   452
        11 45 A6   00   E1 027C   453         BBC     #CDB$V_PERCENT, -
                     0281   454                 CDB$W_QFLAGS(R6),30$    ; If percent not requested, skip pc buffs
                     0281   455
          50   00'8F   9A 0281   456         MOVZBL  #PC_BUFS,R0             ; Loop PC_BUFS times
                     0285   457 20$:
             87   55   D0 0285   458         MOVL    R5,(R7)+                ; Store address of buffer into next ptr
      55   00000000'8F   C0 0288   459         ADDL2   #<4*MAXELTS>,R5         ; Calculate address of next buffer
               F3 50   F5 028F   460         SOBGTR  R0,20$                  ; ....
                     0292   461
                     0292   462 30$:
             57   55   D0 0292   463         MOVL    R5,R7                   ; Save ptr to next MBP for next call
                     0295   464
                     0295   465 ;
                     0295   466 ; Initialize buffers which require it.
                     0295   467 ;
                     0295   468
        29 45 A6   00   E1 0295   469         BBC     #CDB$V_PERCENT, -       ; If percent not requested, skip pc buffs
                     029A   470                 CDB$W_QFLAGS(R6),50$
0000'8F   00   FE AF   00   2C 029A   471         MOVC5   #0,,#0,#<4*MAXELTS>,@MBP$A_PCSUM(R10) ; Zero out PCSUM buffer
            24 BA        02A2
0000'8F   00   FE AF   00   2C 02A4   472         MOVC5   #0,,#0,#<4*MAXELTS>,@MBP$A_PCMAX(R10) ; Zero out PCMAX buffer
            20 BA        02AC
                     02AE   473
```

```
                                02AE    474 ;
                                02AE    475 ; Store large positive number (suitable for integer or floating)
                                02AE    476 ; into each longword of PCMIN.
                                02AE    477 ;
                                02AE    478
          51    1C AA   D0      02AE    479          MOVL    MBP$A_PCMIN(R10),R1     ; Get addr of PCMIN buffer
     50   00000000'8F   D0      02B2    480          MOVL    #MAXELTS,R0             ; ... and number of longwords
                                02B9    481 40$:
     81   00000000'8F   D0      02B9    482          MOVL    #LARGE_NO,(R1)+         ; Move in a large value
                F6 50   F5      02C0    483          SOBGTR  R0,40$                  ; Loop back for next one
                                02C3    484
                                02C3    485 50$:
0000'8F   00    FE AF    00     2C      02C3    486          MOVC5   #0,,#0,#<4*MAXELTS>,@MBP$A_SUM(R10) ; Zero out SUM buffer
                   14 BA        02CB
0000'8F   00    FE AF    00     2C      02CD    487          MOVC5   #0,,#0,#<4*MAXELTS>,@MBP$A_MAX(R10) ; Zero out MAX buffer
                   10 BA        02D5
                                02D7    488
                                02D7    489 ;
                                02D7    490 ; Store large positive number (suitable for integer or floating)
                                02D7    491 ; into each longword of MIN.
                                02D7    492 ;
                                02D7    493
          51    0C AA   D0      02D7    494          MOVL    MBP$A_MIN(R10),R1       ; Get addr of MIN buffer
     50   00000000'8F   D0      02DB    495          MOVL    #MAXELTS,R0             ; ... and number of longwords
                                02E2    496 60$:
     81   00000000'8F   D0      02E2    497          MOVL    #LARGE_NO,(R1)+         ; Move in a large value
                F6 50   F5      02E9    498          SOBGTR  R0,60$                  ; Loop back for next one
                                02EC    499
                           05   02EC    500          RSB
```

```
                                                                ;
                                                                ; GET_MEM:
                                                                ;
                                                                ; Obtain virtual memory for required buffers.
                                                                ;
                                                                ;
                                                                ; Push 2 addresses required by LIB$GET_VM and issue request
                                                                ;
                                                                ;
                2E A6    DF                                     PUSHAL  CDB$A_BUFFERS(R6)               ; Push addr of longword to hold
                                                                ;                                       ; ... buffer block pointer
                2A A6    DF                                     PUSHAL  CDB$L_BUFFERS(R6)               ; Now push addr of # of bytes needed
00000000'GF     02       FB                                     CALLS   #2,G^LIB$GET_VM                 ; Allocate buffers
                         05                                     RSB                                     ; Return
                                                                ;
                                                                ;
                                                                ; CLEAR_DATA::
                                                                ;
                                                                ;
                                                                ; Initialize the DATA array to zero.
                                                                ;
                                                                ; Input Registers:
                                                                ;
                                                                ;       R8 = size of DATA array
                                                                ;       R9 = address of DATA array
                                                                ;
                                                                ; Registers R0-R5 and R8,R9 are destroyed.
                                                                ;
                                                                ; The only output of this subroutine is that the
                                                                ; DATA array is cleared to zeroes.
                                                                ;
                                                                ;
                                                                ; 10$:
                58    00007D00  8F    D1                        CMPL    #32000,R8                       ; Is a large MOVC5 required?
                                19    18                        BGEQ    20$                             ; No -- go do a smaller one
69    7D00 8F     00     FE AF    00    2C                        MOVC5   #0,,#0,#32000,(R9)              ; Yes -- clear 32000 bytes
                58    00007D00  8F    C2                        SUBL2   #32000,R8                       ; Calc bytes left to clear
                59    00007D00  8F    C0                        ADDL2   #32000,R9                       ; ... and starting byte addr
                                DE    11                        BRB     10$                             ; Go check size of next move
                                                                ; 20$:
    69    58     00     FE AF    00    2C                        MOVC5   #0,,#0,R8,(R9)                  ; Clear remainder of DATA array
                                                                ;
                         05                                     RSB                                     ; Return
                                                                ;
                                                                ; GET_SYS_DATA_ARRAYS:
                                                                ;
                52    00000000'EF       D0                      MOVL    SPTR,R2                         ; Get pointer to SYI
                        52    0B A2     3C                      MOVZWL  MNR_SYI$W_MAXPRCCT(R2),R2       ; Get max process count
                     5B    52    04     C5                      MULL3   #4,R2,R11                       ; Compute size of one array
          00000000'EF    5B    10     C5                      MULL3   #16,R11,SYS_DATA_LEN            ; Need 16 arrays
                00000000'EF           DF                      PUSHAL  SYS_DATA_ADDR                   ; Push addr of longword to hold
                                                                ;                                       ; ... SYSTEM DATA arrays ptr
                00000000'EF           DF                      PUSHAL  SYS_DATA_LEN                    ; Now push addr of # of bytes needed
             00000000'GF       02     FB                      CALLS   #2,G^LIB$GET_VM                 ; Allocate space
                          01   50     E8                      BLBS    R0,10$                          ; Branch if successful
```

```
                     05  0352   559          RSB                                 ; Else return with error
                         0353   560 10$:
52   00000000'EF     DE  0353   561          MOVAL   SYS_TOP_VEC,R2              ; Get addr of vector of ptrs
53   00000000'EF     D0  035A   562          MOVL    SYS_DATA_ADDR,R3            ; Get ptr to first array
          54   10    D0  0361   563          MOVL    #16,R4                      ; Number of pointers to save
                         0364   564 20$:
          82   53    D0  0364   565          MOVL    R3,(R2)+                    ; Save ptr to first array
          53   5B    C0  0367   566          ADDL2   R11,R3                      ; Point to next one
            F7 54    F5  036A   567          SOBGTR  R4,20$                      ; Loop back to save next ptr
                         036D   568
                         036D   569 ;
                         036D   570 ; Now clear the four DATA arrays
                         036D   571 ;
                         036D   572
5A   00000000'EF     DE  036D   573          MOVAL   SYS_TOP_VEC,R10             ; Get addr of vector of ptrs
          57   04    D0  0374   574          MOVL    #4,R7                       ; Number of arrays to clear
                         0377   575 30$:
          59   6A    D0  0377   576          MOVL    (R10),R9                    ; R9 must contain array addr
          58   5B    D0  037A   577          MOVL    R11,R8                      ; R8 gets array length
               FF7B    30  037D   578          BSBW    CLEAR_DATA                  ; Clear the data
          5A   10    C0  0380   579          ADDL2   #16,R10                     ; Point to next array
            F1 57    F5  0383   580          SOBGTR  R7,30$                      ; Loop back to process next one
50   00000000'8F     D0  0386   581          MOVL    #SS$_NORMAL,R0              ; Load up normal status
                     05  038D   582          RSB
                         038E   583
                         038E   584 .END
```

```
ALL_STAT                = 00000000
AVE_STAT                = 00000002
CB_ADDRS                  ********    X   02
CDB                     = 00000000
CDB$A_BUFFERS           = 0000002E
CDB$A_CDX               = 00000032
CDB$A_CMDHDR            = 0000004F
CDB$A_FAOCTR            = 00000004
CDB$A_ITMSTR            = 0000001C
CDB$A_POSTCOLL          = 00000026
CDB$A_PRECOLL           = 00000022
CDB$A_SUMBUF            = 0000000C
CDB$A_TITLE             = 00000010
CDB$B_FAOPRELEN         = 00000041
CDB$B_FAOSEGLEN         = 00000040
CDB$B_ST                = 00000042
CDB$B_ST_CUR            = 00000044
CDB$B_ST_DEF            = 00000043
CDB$K_SIZE              = 00000053
CDB$L_BUFFERS           = 0000002A
CDB$L_ECOUNT            = 00000018
CDB$L_FAOCTR            = 00000000
CDB$L_FLAGS             = 0000004B
CDB$L_ICOUNT            = 00000014
CDB$L_MIN               = 00000038
CDB$L_RANGE             = 0000003C
CDB$L_SUMBUF            = 00000008
CDB$M_CPU               = 00000002
CDB$M_CPU_COMB          = 00000008
CDB$M_CTPRES            = 00000001
CDB$M_DISABLE           = 00000200
CDB$M_DISKAC            = 00000040
CDB$M_DISKVN            = 00000080
CDB$M_EXPLIC            = 00001000
CDB$M_HOMOG             = 00000020
CDB$M_KUNITS            = 00000400
CDB$M_PERCENT           = 00000001
CDB$M_STD               = 00000010
CDB$M_SWAPBUF           = 00000002
CDB$M_SYSCLS            = 00000100
CDB$M_UNIFORM           = 00000004
CDB$M_WIDE              = 00000800
CDB$S_CDB               = 00000053
CDB$S_FILLER            = 00000013
CDB$S_FLAGS             = 00000004
CDB$S_QFILLER           = 0000000E
CDB$S_QFLAGS            = 00000002
CDB$V_CPU               = 00000001
CDB$V_CPU_COMB          = 00000003
CDB$V_CTPRES            = 00000000
CDB$V_DISABLE           = 00000009
CDB$V_DISKAC            = 00000006
CDB$V_DISKVN            = 00000007
CDB$V_EXPLIC            = 0000000C
CDB$V_FILLER            = 0000000D
CDB$V_HOMOG             = 00000005
CDB$V_KUNITS            = 0000000A
CDB$V_PERCENT           = 00000000
CDB$V_QFILLER           = 00000002
CDB$V_STD               = 00000004
CDB$V_SWAPBUF           = 00000001
CDB$V_SYSCLS            = 00000008
CDB$V_UNIFORM           = 00000002
CDB$V_WIDE              = 00000008
CDB$W_BLKLEN            = 00000020
CDB$W_DISPCTL           = 00000036
CDB$W_QFLAGS            = 00000045
CDB$W_QFLAGS_CUR        = 00000049
CDB$W_QFLAGS_DEF        = 00000047
CDB_EXT                 = 00000000
CDX$A_DISPFAO           = 0000002C
CDX$A_DISPNAM           = 00000028
CDX$A_ELIDTABLE         = 0000000C
CDX$A_ILOOKTAB          = 00000024
CDX$A_SCBTABLE          = 00000010
CDX$A_SELIDTABLE        = 00000018
CDX$B_ELIDLEN           = 00000009
CDX$B_IDISCONSEC        = 00000007
CDX$B_IDISCT            = 00000006
CDX$B_IDISINDEX         = 00000008
CDX$K_SIZE              = 00000030
CDX$L_DCOUNT            = 0000001C
CDX$L_PREV_DCT          = 00000020
CDX$L_SELIDTABLE        = 00000014
CDX$S_CDB_EXT           = 00000030
CDX$S_IBITS             = 00000010
CDX$W_CUMELCT           = 0000000A
CDX$W_IBITS             = 00000000
CDX$W_IBITS_CUR         = 00000004
CDX$W_IBITS_DEF         = 00000002
CLASS_HDR               = 00000000
CLEAR_DATA                000002FB  RG      02
COLL_BUFS                 ********    X     02
CUR_STAT                = 00000001
DEF$A_DISP              = 0000000C
DEF$A_REC               = 00000004
DEF$A_SUMM              = 00000014
DEF$L_DISP              = 00000008
DEF$L_REC               = 00000000
DEF$L_SUMM              = 00000010
DEF$S_DEF_DESC          = 00000018
DEF_DESC                = 00000000
FAOCTR_SIZE               ********    X     02
FILE_HDR                = 00000000
GB_NRSB                   000001AB  R       02
GB_RSB                    000001B2  R       02
GET_BUFFERS               00000000  RG      02
GET_MEM                   000002ED  R       02
GET_SYS_DATA_ARRAYS       00000325  R       02
HOM_BUFFS                 000001B7  R       02
HOM_CLASS_PRE           = 00000000
LARGE_NO                  ********    X     02
LIB$GET_VM                ********    X     02
MAXELTS                   ********    X     02
```

```
MAX_STAT                        =  00000004
MBP                             =  00000000
MBP$A_ADDR                      =  00000018
MBP$A_BIST                      =  00000004
MBP$A_BA                        =  00000000
MBP$A_BUFF1ST                   =  00000004
MBP$A_BUFFA                     =  00000000
MBP$A_BUFFERA                   =  00000000
MBP$A_BUFFERB                   =  00000004
MBP$A_DATA                      =  00000008
MBP$A_DIFF                      =  0000000C
MBP$A_MAX                       =  00000010
MBP$A_MIN                       =  0000000C
MBP$A_ORDER                     =  00000010
MBP$A_PCMAX                     =  00000020
MBP$A_PCMIN                     =  0000001C
MBP$A_PCSTATS                   =  00000018
MBP$A_PCSUM                     =  00000024
MBP$A_PID                       =  00000014
MBP$A_PR_FAOSTK                 =  00000008
MBP$A_STAT_                     =  00000008
MBP$A_SUM                       =  00000014
MBP$K_SIZE                      =  00000028
MBP$S_MBP                       =  00000028
MBP$S_MBP2                      =  0000001C
MBP$S_MBP3                      =  0000000C
MBP2                            =  00000000
MBP3                            =  00000000
MBP_FILL                          0000025A R      02
MIN_STAT                        =  00000003
MNR_CLS$B_TYPE                  =  00000000
MNR_CLS$K_HSIZE                 =  0000000D
MNR_CLS$Q_STAMP                 =  00000003
MNR_CLS$S_CLASS_HDR             =  0000000D
MNR_CLS$S_FILLER                =  0000000F
MNR_CLS$S_FLAGS                 =  00000002
MNR_CLS$S_STAMP                 =  00000008
MNR_CLS$V_CONT                  =  00000000
MNR_CLS$V_FILLER                =  00000001
MNR_CLS$W_FLAGS                 =  00000001
MNR_CLS$W_RESERVED              =  0000000B
MNR_HDR$B_TYPE                  =  00000000
MNR_HDR$K_CLASSBITS             =  00000073
MNR_HDR$K_MAXCOMLEN             =  0000003C
MNR_HDR$K_REVLEVELS             =  00000083
MNR_HDR$K_SIZE                  =  00000103
MNR_HDR$L_FLAGS                 =  00000001
MNR_HDR$L_INTERVAL              =  00000015
MNR_HDR$L_RECCT                 =  00000029
MNR_HDR$O_CLASSBITS             =  00000073
MNR_HDR$O_REVOCLSBITS           =  00000019
MNR_HDR$Q_BEGINNING             =  00000005
MNR_HDR$Q_ENDING                =  0000000D
MNR_HDR$S_BEGINNING             =  00000008
MNR_HDR$S_CLASSBITS             =  00000010
MNR_HDR$S_COMMENT               =  0000003C
MNR_HDR$S_ENDING                =  00000008
MNR_HDR$S_FILE_HDR              =  00000103
MNR_HDR$S_FILLER                =  00000020
MNR_HDR$S_FLAGS                 =  00000004
MNR_HDR$S_LEVEL                 =  00000008
MNR_HDR$S_REVOCLSBITS           =  00000010
MNR_HDR$S_REVLEVELS             =  00000080
MNR_HDR$S_TYPE                  =  00000008
MNR_HDR$T_COMMENT               =  00000035
MNR_HDR$T_LEVEL                 =  0000002D
MNR_HDR$T_REVLEVELS             =  00000083
MNR_HDR$V_FILLER                =  00000000
MNR_HDR$W_COMLEN                =  00000071
MNR_HOM$K_PSIZE                 =  00000008
MNR_HOM$L_ELTCT                 =  00000000
MNR_HOM$L_RESERVED              =  00000004
MNR_HOM$S_HOM_CLASS_PRE         =  00000008
MNR_PRO$B_PRI                   =  0000000A
MNR_PRO$K_DSIZE                 =  0000003B
MNR_PRO$K_FSIZE                 =  00000040
MNR_PRO$K_PSIZE                 =  00000008
MNR_PRO$K_REVODSIZE             =  00000033
MNR_PRO$K_REV1DSIZE             =  0000003B
MNR_PRO$L_BIOCNT                =  0000002F
MNR_PRO$L_CPUTIM                =  0000002B
MNR_PRO$L_DIOCNT                =  00000023
MNR_PRO$L_EFWM                  =  00000037
MNR_PRO$L_EPID                  =  00000033
MNR_PRO$L_IPID                  =  00000000
MNR_PRO$L_PAGEFLTS              =  00000027
MNR_PRO$L_PCTINT                =  00000004
MNR_PRO$L_PCTREC                =  00000000
MNR_PRO$L_STS                   =  0000001F
MNR_PRO$L_UIC                   =  00000004
MNR_PRO$Q_LNAME                 =  0000000B
MNR_PRO$S_LNAME                 =  00000010
MNR_PRO$S_PROCESS_CLASS         =  0000003B
MNR_PRO$S_PRO_CLASS_PRE         =  00000008
MNR_PRO$W_GPGCNT                =  0000001B
MNR_PRO$W_PPGCNT                =  0000001D
MNR_PRO$W_STATE                 =  00000008
MNR_SYI$B_MPCPUS                =  0000000D
MNR_SYI$B_TYPE                  =  00000000
MNR_SYI$K_BALSETMEM             =  0000001E
MNR_SYI$K_CPUTYPE               =  00000026
MNR_SYI$K_MPWHILIM              =  00000022
MNR_SYI$K_NODENAME              =  0000000E
MNR_SYI$K_SIZE                  =  0000002A
MNR_SYI$L_BALSETMEM             =  0000001E
MNR_SYI$L_CPUTYPE               =  00000026
MNR_SYI$L_MPWHILIM              =  00000022
MNR_SYI$Q_BOOTTIME              =  00000003
MNR_SYI$S_BOOTTIME              =  00000008
MNR_SYI$S_FILLER                =  0000000E
MNR_SYI$S_FLAGS                 =  00000002
MNR_SYI$S_NODENAME              =  00000010
MNR_SYI$S_SYS_INFO              =  0000002A
MNR_SYI$S_TYPE                  =  00000008
```

```
MNR_SYI$T_NODENAME              = 0000000E
MNR_SYI$V_CLUSMEM               = 00000000
MNR_SYI$V_FILLER                = 00000002
MNR_SYI$V_RESERVED1             = 00000001
MNR_SYI$W_FLAGS                 = 00000001
MNR_SYI$W_MAXPRCCT              = 0000000B
MRB                             = 00000000
MRB$A_COMMENT                   = 0000002C
MRB$A_DISPLAY                   = 00000020
MRB$A_INPUT                     = 0000001C
MRB$A_RECORD                    = 00000024
MRB$A_SUMMARY                   = 00000028
MRB$B_INP_FILES                 = 00000042
MRB$K_SIZE                      = 00000045
MRB$L_FLUSH                     = 00000014
MRB$L_INTERVAL                  = 00000010
MRB$L_VIEWING_TIME              = 00000018
MRB$M_ALL_CLASS                 = 00000400
MRB$M_BY_NODE                   = 00001000
MRB$M_DISPLAY                   = 00000001
MRB$M_DISP_TO_FILE              = 00000020
MRB$M_DIS_CL_REQ                = 00000100
MRB$M_INDEFEND                  = 00000010
MRB$M_INP_CL_REQ                = 00000040
MRB$M_MFSOM                     = 00000800
MRB$M_PLAYBACK                  = 00000008
MRB$M_PROC_REQ                  = 00004000
MRB$M_RECORD                    = 00000002
MRB$M_REC_CL_REQ                = 00000080
MRB$M_SUMMARY                   = 00000004
MRB$M_SUM_CL_REQ                = 00000200
MRB$M_SYSCLS                    = 00002000
MRB$O_CLASSBITS                 = 00000032
MRB$Q_BEGINNING                 = 00000000
MRB$Q_ENDING                    = 00000008
MRB$S_BEGINNING                 = 00000008
MRB$S_CLASSBITS                 = 00000010
MRB$S_ENDING                    = 00000008
MRB$S_FLAGS                     = 00000002
MRB$S_MRB                       = 00000045
MRB$V_ALL_CLASS                 = 0000000A
MRB$V_BY_NODE                   = 0000000C
MRB$V_DISPLAY                   = 00000000
MRB$V_DISP_TO_FILE              = 00000005
MRB$V_DIS_CL_REQ                = 00000008
MRB$V_FILLER                    = 0000000F
MRB$V_INDEFEND                  = 00000004
MRB$V_INP_CL_REQ                = 00000006
MRB$V_MFSOM                     = 0000000B
MRB$V_PLAYBACK                  = 00000003
MRB$V_PROC_REQ                  = 0000000E
MRB$V_RECORD                    = 00000001
MRB$V_REC_CL_REQ                = 00000007
MRB$V_SUMMARY                   = 00000002
MRB$V_SUM_CL_REQ                = 00000009
MRB$V_SYSCLS                    = 0000000D
MRB$W_CLASSCT                   = 00000030
MRB$W_FLAGS                     = 00000043
NORMAL                            ********    X   02
PC_BUFS                           ********    X   02
PROCDISPS                       = 00000005
PROCESS_CLASS                   = 00000000
PRO_CLASS_PRE                   = 00000000
QUAL$A_ALL                      = 00000064
QUAL$A_AVE                      = 00000074
QUAL$A_BEG                      = 00000004
QUAL$A_BY_NODE                  = 00000054
QUAL$A_CLASS                    = 0000005C
QUAL$A_COMM                     = 0000004C
QUAL$A_CPU                      = 000000AC
QUAL$A_CUR                      = 0000006C
QUAL$A_DISP                     = 00000034
QUAL$A_END                      = 0000000C
QUAL$A_FLUSH                    = 0000001C
QUAL$A_INP                      = 0000002C
QUAL$A_INT                      = 00000014
QUAL$A_ITEM                     = 000000BC
QUAL$A_MAX                      = 00000084
QUAL$A_MIN                      = 0000007C
QUAL$A_PCENT                    = 000000B4
QUAL$A_REC                      = 0000003C
QUAL$A_SUMM                     = 00000044
QUAL$A_TOPB                     = 0000009C
QUAL$A_TOPC                     = 0000008C
QUAL$A_TOPD                     = 00000094
QUAL$A_TOPF                     = 000000A4
QUAL$A_VIEW                     = 00000024
QUAL$L_ALL                      = 00000060
QUAL$L_AVE                      = 00000070
QUAL$L_BEG                      = 00000000
QUAL$L_BY_NODE                  = 00000050
QUAL$L_CLASS                    = 00000058
QUAL$L_COMM                     = 00000048
QUAL$L_CPU                      = 000000A8
QUAL$L_CUR                      = 00000068
QUAL$L_DISP                     = 00000030
QUAL$L_END                      = 00000008
QUAL$L_FLUSH                    = 00000018
QUAL$L_INP                      = 00000028
QUAL$L_INT                      = 00000010
QUAL$L_ITEM                     = 000000B8
QUAL$L_MAX                      = 00000080
QUAL$L_MIN                      = 00000078
QUAL$L_PCENT                    = 000000B0
QUAL$L_REC                      = 00000038
QUAL$L_SUMM                     = 00000040
QUAL$L_TOPB                     = 00000098
QUAL$L_TOPC                     = 00000088
QUAL$L_TOPD                     = 00000090
QUAL$L_TOPF                     = 000000A0
QUAL$L_VIEW                     = 00000020
QUAL$S_QUALIFIER_DESC           = 000000C0
QUALIFIER_DESC                  = 00000000
REG_BUFS                          ********    X   02
```

```
REG_PROC                        =  00000000
SCB$B_FLAGS                     =  00000002
SCB$K_SIZE                      =  00000003
SCB$S_FILLER                    =  00000006
SCB$S_FLAGS                     =  00000001
SCB$S_STATS_BLOCK               =  00000003
SCB$V_ACTIVE                    =  00000001
SCB$V_CURRENT                   =  00000000
SCB$V_FILLER                    =  00000002
SCB$W_DBIDX                     =  00000000
SPTR                               ********    X   02
SS$_NORMAL                         ********    X   02
STATS                           =  00000005
STATS_BLOCK                     =  00000000
SYS_DATA_ADDR                      ********    X   02
SYS_DATA_LEN                       ********    X   02
SYS_INFO                        =  00000000
SYS_TOP_VEC                        ********    X   02
TOPB_PROC                       =  00000003
TOPC_PROC                       =  00000001
TOPD_PROC                       =  00000002
TOPF_PROC                       =  00000004
```

Psect synopsis

PSECT name	Allocation		PSECT No.		Attributes								
. ABS .	00000000	(0.)	00	(0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT NOVEC BYTE
MONDATA	00000000	(0.)	01	(1.)	NOPIC	USR	CON	REL	LCL	NOSHR	NOEXE	RD	WRT NOVEC QUAD
$$MONCODE	0000038E	(910.)	02	(2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.09	00:00:00.43
Command processing	129	00:00:00.70	00:00:05.16
Pass 1	169	00:00:03.06	00:00:10.19
Symbol table sort	0	00:00:00.54	00:00:01.12
Pass 2	116	00:00:01.39	00:00:05.76
Symbol table output	42	00:00:00.29	00:00:00.62
Psect synopsis output	2	00:00:00.04	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	492	00:00:06.11	00:00:23.32

```
The working set limit was 1350 pages.
20622 bytes (41 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 364 non-local and 31 local symbols.
584 source lines were read in Pass 1, producing 16 object records in Pass 2.
16 pages of virtual memory were used to define 6 macros.
```

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

Macro library name                 Macros defined
------------------                 --------------
_$255$DUA28:[MONTOR.OBJ]MONLIB.MLB;1          6
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                0
_$255$DUA28:[SYSLIB]STARLET.MLB;2             0
TOTALS (all libraries)                        6

355 GETS were required to define 6 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:GETBUFF/OBJ=OBJ$:GETBUFF MSRC$:GETBUFF/UPDATE=(ENH$:GETBUFF)+EXECML$/LIB+LIB$:MONLIB/LIB
```

0239 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

COLLEVT
LIS

MOMTEST
LIS

MONDEF
SDL

GETBUFF
LIS

MONSUB
CLD

NPARSE
LIS

MONTOR

MONITOR
MAP

DSPDEF
MDL

HOMOG
LIS

MONITOR
LIS

MFSUMM
LIS

MONDAT
LIS